

Sheffield Hallam University

Development and evaluation of vision processing algorithms in multi-robotic systems.

AHMED, M. Shuja.

Available from the Sheffield Hallam University Research Archive (SHURA) at:

<http://shura.shu.ac.uk/19223/>

A Sheffield Hallam University thesis

This thesis is protected by copyright which belongs to the author.

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Please visit <http://shura.shu.ac.uk/19223/> and <http://shura.shu.ac.uk/information.html> for further details about copyright and re-use permissions.

Learning and Information Services
Adsetts Centre, City Campus
Sheffield S1 1WD

27432

102 105 850 5



Sheffield Hallam University
Learning and Information Services
Adsetts Centre, City Campus
Sheffield S1 1WD

REFERENCE

ProQuest Number: 10694103

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10694103

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Development and Evaluation of Distributed Vision Processing Algorithms in Multi-Robotic Systems

M Shuja Ahmed

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

May 2013

Abstract

The trend in swarm robotics research is shifting to the design of more complicated systems in which the robots have abilities to form a robotic organism. In such systems, a single robot has very limited memory and processing resources, but the complete system is rich in these resources. As vision sensors provide rich surrounding awareness and vision algorithms also requires intensive processing. Therefore, vision processing tasks are the best candidate for distributed processing in such systems.

To perform distributed vision processing, a number of scenarios are considered in swarm and the robotic organism form. In the swarm form, as the robots use low bandwidth wireless communication medium, so the exchange of simple visual features should be made between robots. This is addressed in a swarm mode scenario, where novel distance vector features are exchanged within a swarm of robots to generate a precise environmental map. The generated map facilitates the robot navigation in the environment. If features require encoding with high density information, then sharing of such features is not possible using the wireless channel with limited bandwidth. So methods were devised which process such features onboard and then share the process outcome to perform vision processing in a distributed fashion. This is shown in another swarm mode scenario in which a number of optimisation stages are followed and novel image pre-processing techniques are developed which enable the robots to perform onboard object recognition, and then share the process outcome in terms of object identity and its distance from the robot, to localise the objects.

In the robotic organism, the use of reliable communication medium facilitates vision processing in distributed fashion, and this is presented in two scenarios. In the first scenario, the robotic organism detect objects in the environment in distributed fashion, but to get detailed surrounding awareness, the organism needs to learn these objects. This leads to a second scenario, which presents a modular approach to object classification and recognition. This approach provides a mechanism to learn newly detected objects and also ensure faster response to object recognition. Using the modular approach, it is also demonstrated that the collective use of 4 distributed processing resources in a robotic organism can provide 5 times the performance of an individual robot module. The overall performance was comparable to an individual less flexible robot (e.g., Pioneer-3AT) with significant higher processing capability.

Publications

Journals:

- (i) Vision Based Object Recognition and Localisation by a Wireless Connected Distributed Robotic Systems. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)*, Vol. 11, Issue. 1, Pages: 54-67, 2012.
- (ii) Distributed Vision Processing in Multi-robotics Organism. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. In *Journal of Robotics and Autonomous Systems*, Elsevier, (In Review).
- (iii) Object Classification and Recognition in Distributed Modular Robotic Systems. M Shuja Ahmed and Reza Saatchi. *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)*, (Re-submitted with Revisions).
- (iv) Performance Comparison of Distributed Modular Robotic System vs High Processing Systems. M Shuja Ahmed and Reza Saatchi. *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)*, (In Review).
- (v) Distributed Vision Processing in Reconfigurable Modular Robotic Systems. M Shuja Ahmed and Reza Saatchi. *Journal of Intelligent and Robotic Systems*, Springer, (In Review).
- (vi) Distributed Mosaic Formation and Object Detection in Modular Robotic Systems. M Shuja Ahmed and Reza Saatchi. *Journal of Intelligent and Robotic Systems*, Springer, (In Review).

Conferences:

- (i) VISION BASED OBSTACLE AVOIDANCE AND ODOMETERY FOR SWARMS OF SMALL SIZE ROBOTS. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. In *Proceedings of 2nd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, Rome, Italy, 24-26 February 2012, SciTePress, Pages: 115-122.
- (ii) SUPPORT FOR ROBOT DOCKING AND ENERGY FORAGING - A Computer Vision Approach. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. In *Proceedings of 2nd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, Rome, Italy, 24-26 February 2012, SciTePress, Pages: 123-128.
- (iii) AN EFFICIENT APPROACH TO OBJECT RECOGNITION FOR MOBILE ROBOTS. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. In *Proceedings of 3rd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, Barcelona, Spain, 19-21 February 2013, SciTePress, Accepted for publication.
- (iv) VISION BASED ENVIRONMENT MAPPING BY NETWORK CONNECTED MULTI-ROBOTIC SYSTEM. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. In *Proceedings of 3rd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS)*, Barcelona, Spain, 19-21 February 2013, SciTePress, Accepted for publication.

- (v) IMPLEMENTATION OF DISTRIBUTED MOSAIC FORMATION AND OBJECT DETECTION IN MODULAR ROBOTIC SYSTEMS. M Shuja Ahmed, Reza Saatchi and Fabio Caparrelli. In Proceedings of 3rd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS), Barcelona, Spain, 19-21 February 2013, SciTePress, Accepted for publication.

EU-FP7 Research Project Replicator: Technical Deliverables

- (i) EU-FP7 Research Project Replicator. Year 3 Technical Deliverable. Title: "Distributed Information Gathering and Appearance Based Places Recognition To Perform A Search Operation By Swarm Of Robots". M Shuja Ahmed and Fabio Caparrelli. Stuttgart University, Germany, 24 March 2011.
- (ii) EU-FP7 Research Project Replicator. Year 4 Technical Deliverable. Title: "Distributed Vision Processing in Multi-Robotic Organism". M Shuja Ahmed and Fabio Caparrelli. Stuttgart University, Germany, 16 March 2012.

EU-FP7 Research Project Replicator: Presentations

- (i) Replicator/Symbion Progress Meeting, INRIA - Paris France, March 2011
- (ii) Replicator/Symbion Review Meeting, Stuttgart University, Germany, May 2011
- (iii) Replicator/Symbion Progress Meeting, York University, United Kingdom, Oct 2011
- (iv) Replicator/Symbion Progress Meeting, Karlsruhe University, Germany, March 2012
- (v) Replicator/Symbion Review Meeting, Stuttgart University, Germany, 3 May 2012
- (vi) Replicator/Symbion Review Meeting, Stuttgart University, Germany, 4 May 2012
- (vii) Material and Engineering Research Institute (MERI) Day, Sheffield Hallam University, Sheffield, United Kingdom, May 2011.

EU-FP7 Research Project Replicator: Technical Workshops

- (i) Replicator Workshop, Sheffield Hallam University, Sheffield, United Kingdom, Dec 2010
- (ii) Replicator Workshop, Stuttgart University, Germany, April 2012

Contents

Abstract	i
Publications	ii
Acronyms	ix
Symbols	xiii
List of Figures	xxiii
List of Tables	xxiv
Acknowledgements	xxv
Declaration	xxvi
Disclaimer	xxvii
1 Introduction	1
1.1 Aims, objectives and background of the study	4
1.2 Contribution of Thesis	6
1.2.1 Object Recognition and Localisation in Distributed Robotic Systems	8
1.2.2 Computer Vision Support for Robot Docking and Energy Foraging	10
1.2.3 Environment Mapping by Distributed Multi-Robotic System	11
1.2.4 Distributed Object Recognition and Visual Information Gathering in a Multi- Robotic Organism	11

1.2.5	Distributed Object Classification and Recognition in a Robotic Organism . . .	11
1.3	Thesis Outline	12
2	Literature Review	15
2.1	Communication	21
2.2	Obstacle Avoidance	23
2.3	Energy Foraging and Vision Support for Docking	24
2.4	Localisation	24
2.5	Appearance Based Object Recognition	28
2.6	Multi-robot Environment Mapping	32
2.7	Distributed Processing in Robotic Organism	33
2.8	Conclusions	35
3	Hardware and Software Requirements	36
3.1	Hardware Requirements for Swarm Mode Scenarios	36
3.1.1	Swarm Robot Units	37
3.1.2	Hardware for Swarm Communication	39
3.1.3	Robot Tracking Cameras	40
3.2	Hardware Requirement for Organism Mode Scenarios	41
3.2.1	Analog Devices - Blackfin Processor	41
3.2.2	Blackfin Evaluation Board EVAL-BF5xx	44
3.2.3	Blackfin Extender Board EXT-BF5xx-Camera	44
3.2.4	Blackfin Experimental Board	45
3.2.5	CMOS Camera Sensor	46
3.2.6	Robot Base and Motor Control Board	46
3.2.7	Onboard Communication	47
3.2.8	Navigation board	48
3.3	Selection of Operating System (Robot Firmware)	49
3.4	Conclusions	51
4	Embedded Vision Processing	52
4.1	Robot Camera Calibration	53

4.2	Embedded Vision Algorithms	62
4.2.1	YUV to Colour Image Conversion	62
4.2.2	Colour to Grey Scale Conversion	63
4.2.3	Grey Scale Gradient	64
4.2.4	Colour Gradient	65
4.2.5	Grey Scale and Colour Image Segmentation	66
4.2.6	Colour Blob Detection	66
4.2.7	Image Erosion	68
4.2.8	Image Dilation	68
4.3	Vision Based Obstacle Avoidance	69
4.3.1	Approaches To Obstacle Avoidance	70
4.3.2	Experimental Results	77
4.4	Energy Foraging	86
4.4.1	Experimental Results	87
4.5	Vision Based Docking Support	88
4.5.1	Blob Detection of Red LEDs in ON State	90
4.5.2	Obtaining the Statistics of Red LED Blobs	94
4.5.3	Classification of Red LED blobs	94
4.5.4	Control Algorithm to Approach the Blobs	95
4.5.5	Experimental Results	97
4.6	Conclusions	102
5	Multi-Robot Localisation and Tracking System	103
5.1	Camera Calibration	104
5.2	Visual Localisation and Tracking System	111
5.2.1	Colour Blob Extraction	112
5.2.2	Extraction of Blobs Statistics	113
5.2.3	Template Matching and Pattern Recognition	113
5.2.4	Multi Camera Based Robot Tracking	116
5.3	Multi-Robot Visual Guidance	120
5.4	Conclusions	126

6	Distributed Vision Processing in Multi-Robotic Swarm	127
6.1	Communication among Swarm of Robots	128
6.2	Vision Based Object Recognition and Localisation by Multi-Robotic Systems	128
6.2.1	Processor Specific Optimisation	133
6.2.2	Image Pre-processing	134
6.2.3	Multi-resolution Analysis	135
6.2.4	Experimental Results	138
6.3	Environment Mapping by Distributed Multi-robotic System	149
6.3.1	Methodology - Environment Mapping	151
6.3.2	Experimental Results	159
6.4	Conclusions	165
7	Distributed Object Recognition and Information Gathering in a Multi-Robotic Organism	166
7.1	Communication in the Robotic Organism	169
7.2	Tasks Distribution for Distributed Object Recognition and Information Gathering	173
7.2.1	Communication Awareness within the Robotic Organism	173
7.2.2	Distributed Object Recognition	176
7.2.3	Distributed Information Gathering	178
7.3	Experiments with Robotic Organism	185
7.4	Conclusions	200
8	Distributed Object Classification and Recognition in a Robotic Organism	201
8.1	Multi-processor Robotic Organism	203
8.2	Communication within the Robotic Organism	204
8.3	Vision Task Distribution for Distributed Object Classification and Recognition	205
8.3.1	Module 1: Information Pre-processing	206
8.3.2	Module 2: Object Classification	207
8.3.3	Module 3: Object Recognition	210
8.3.4	Module 4: Vocabulary of Visual Words	211
8.4	Experimental Results	213

8.5	Performance Comparison of Distributed Modular Robotic System versus High Processing Systems	221
8.5.1	Object Recognition - High Processing System	222
8.5.2	Performance Comparison	223
8.6	Conclusions	231
9	Conclusions & Future Work	233
9.1	Conclusions	233
9.2	Future Work	235

Acronyms

CMOS	Complementary metal oxide semiconductor
YUV	Y luma,U chroma blue,V chroma red
LEDs	Light emitting diodes
WiFi	Wireless fidelity
A8	Adjacency in eight directions
FOE	Focus of expansion
SURF	Speeded up robust features
2D	Two dimensional
3D	Three dimensional
TCP	Transmission control protocol
UDP	User datagram protocol
DLT	Divisible load theory
IUB	Inter-national university Bremen
FPS	Frame per second
MBPS	Mega bits per second
COM	Component object model

CORBA	Common object request broker architecture
TEAR	(Transmission control protocol) emulation at receiver
MANET	Mobile adhoc network
PERA	Probabilistic emergent routing algorithm
RSCA	Robot software communications architecture
MAVs	Micro aerial vehicles
IR	Infra red
RFID	Radio frequency identification
POMDP	Partially observable Markov decision process
SIFT	Scale-invariant feature transform
SVM	Support vector machines
CRF	Conditional random fields
PCA	Principle component analysis
KNN	K nearest neighbour
RANSAC	Random sample consensus
kd-tree	K-dimensional tree algorithm
MIPS	Mega instructions per second
WLAN	Wireless local area network
GNU	A recursive acronym for "GNU's Not Unix!"
JTAG	Joint test action group
SDRAM	Synchronous direct random access memory
DC	Direct current

GPL	General public license
WEP	Wireless encryption protocol
ADSP	Advanced digital signal processor
DSP	Digital signal processor
RISC	Reduced instruction set computing
MMU	Memory management unit
SIMD	Single instruction multiple data
ALU	Arithmetic logic unit
MHz	Mega hertz
I/O	Input and output
DMA	Direct memory access
SPORTs	Synchronous serial ports
PPI	Parallel peripheral interface
CMBF	Core module Blackfin
DAC	Digital to Analog converter
SYMBRION	Symbiotic Evolutionary Robot Organisms
REPLICATOR	Robotic Evolutionary Self-Programming and Self-Assembling Organisms
OV7670	Omni-vision 7670 camera sensor
SPI	Serial peripheral interface bus
MAC	Media access control
GPIO	General purpose input output
SD	Secure digital

UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus
PWM	Pulse width modulation
VGA	Video graphics array
GPS	Global positioning system
OS	Operating system
uCLINUX	Micro controller Linux
FPU	Floating point unit
R	Red channel
G	Green channel
B	Blue channel
TTC	Time to contact
QVGA	Quarter video graphics array
A*	Path search algorithm
OPEN CV	Open source computer vision library
SLAM	Simultaneous localisation and mapping

Symbols

E_x	Partial derivatives in x-axis
E_y	Partial derivatives in y-axis
E_t	Partial derivatives of time
α_x	x component of camera focal length
α_y	y component of camera focal length
γ	Skew coefficient between x and y axis of image
(u_0, v_0)	Principal point of the camera.
r_{11}	Parameter of camera rotation matrix
r_{12}	Parameter of camera rotation matrix
r_{13}	Parameter of camera rotation matrix
r_{21}	Parameter of camera rotation matrix
r_{22}	Parameter of camera rotation matrix
r_{23}	Parameter of camera rotation matrix
r_{31}	Parameter of camera rotation matrix
r_{32}	Parameter of camera rotation matrix
r_{33}	Parameter of camera rotation matrix

t_x	x parameter of camera translation matrix
t_y	y parameter of camera translation matrix
t_z	z parameter of camera translation matrix
θ	Orientation
Δ	Delta
C_w	3D world points
C_i	2D image points
K	Camera matrix
s	Scale factor
x_w	x coordinate of world point
y_w	y coordinate of world point
z_w	z coordinate of world point
A	Camera intrinsic matrix
R	Rotation camera matrix
T	Translation camera matrix
B	Camera extrinsics
1x_i	x-coordinates of the features points from first image
1y_i	y-coordinates of the features points from first image
2x_i	x-coordinates of the features points from second image
2y_i	y-coordinates of the features points from second image
1H_2	Homography matrix from image 1 to image 2

List of Figures

1.1	(a) Ants trying to move heavy prey [5]. (b) Robots search for the big red object in the environment [6]. (c) Swarm of robots moving the object collectively [6].	2
1.2	(a) Ants forming a bridge to facilitate other members for carrying food to the colony [9]. (b) Robots physically joining together and forming a bridge [6].	3
1.3	Multi-Robotic Systems (a) Symbrion [7] (b) Replicator [8].	7
1.4	Scenario for distributed vision processing among swarm of robots.	9
1.5	Chapters organisation.	14
2.1	Swarm of self assembled robots moving over an obstacle [7].	15
2.2	Single robot from a team, detecting and tracking a coloured ball to perform distributed map building [18].	16
2.3	Active vision agents based target detection and tracking [23].	17
2.4	Processing stages for remotely sensed images [28].	18
2.5	Image processing stages for image mining system [27].	19
2.6	An omni-directional vision based formation of mobile robots [29].	20
2.7	Ants following artificial pheromones [112].	21
2.8	Robot moving around 7x7 WiFi grid [36].	22
2.9	RFID tags attached to walls for robot localisation [50].	25
2.10	Negative image of a lamp on ceiling used for robot localisation [53].	26
2.11	Some example markers from cantag [57].	27
2.12	SURF features detected in omni-directional camera image [59].	28
2.13	(a) Input image. (b) First eigen vector.(c) Second eigen vector.(d) Third eigen vector. [70].	30

2.14 (a) Image 1. (b) Image 2. (c) Reconstructed image from 20 dimensional eigen vectors. [70].	31
3.1 (a) REPLICATOR robot [8]. (b) SRV1 blackfin robot by surveyor corporation [89]. (c) Swarm of SRV1 robots	37
3.2 Linksys wireless access point [104].	40
3.3 Logitech webcam Pro 9000 [105].	40
3.4 Blackfin CM-BF537E core [91].	42
3.5 Evaluation board EVAL-BF5xx [100].	44
3.6 Blackfin extender board EXT-BF5xx-Camera [101].	45
3.7 Blackfin experimental board "EXT-BF5xx-EXP" [106].	46
3.8 (a)Ov7670 camera sensor [107]. (b) Flexi cable [108].	46
3.9 (a) Robot base for the organism [109]. (b) Motor control board [109].	47
3.10 (a) Lantronix matchport 802.11b/g WiFi [110]. (b) Ethernet communication cable.	48
3.11 Navigation board M3 [111].	48
3.12 Robot organism.	49
4.1 Image captured by the robot vision system.	53
4.2 Robot images used for calibration: image 1 (left) and image 2 (right)	55
4.3 Robot Images used for calibration: image 3 (left) and image 4 (right)	55
4.4 Robot images used for calibration: image 5 (left) and image 6 (right)	56
4.5 Robot images used for calibration: image 7 (left) and image 8 (right)	56
4.6 Robot images used for calibration: image 9 (left) and image 10 (right)	56
4.7 Robot images used for calibration: image 11 (left) and image 12 (right)	57
4.8 Robot images used for calibration: image 13	57
4.9 Calibration results: Re-projections on image (left) and enhanced image (right)	58
4.10 Calibration results: Re-projections on image (left) and enhanced image (right)	59
4.11 Calibration results: Re-projections on image (left) and enhanced image (right)	60
4.12 Calibration results: Re-projections on image (left) and enhanced image (right)	61
4.13 Calibration results: Re-projections on image (left) and enhanced image (right)	62
4.14 Image obtained after YUV to colour image conversion.	63

4.15 Image obtained after colour image (left) to greyscale image (right) conversion.	63
4.16 Image coordinates and Sobel operator for gradient in x and y direction.	64
4.17 Greyscale image (left) and gradient image (right) obtained from BF537E processor.	64
4.18 Colour image (left) and gradient image (right) obtained from BF537E processor.	65
4.19 A8 adjacency. Directions in which region can grow is shown by arrows.	66
4.20 Colour image (left) and segmented image (right) obtained from BF537E processor.	67
4.21 UV plane providing the chrominance information.	67
4.22 Colour input image (left) and processed image (right) showing detection of red colour blobs.	68
4.23 Input image (left) and processed image (right) obtained after erosion.	68
4.24 Input image (left) and processed image (right) obtained after dilation.	69
4.25 (a) Colour input image. (b) Processed image obtained after applying segmentation. (c) Initial ground map isolated from obstacle's. (d) Ground map visible to the robot vision system.	71
4.26 The estimates of three partial derivatives E_x , E_y and E_t of image brightness at the centre of the cube are each obtained from the average of first differences along four parallel edges of the cube. Here the index ' j ' corresponds to the x direction in the image, the index ' i ' to the y direction and index ' k ' lies in the time direction.	74
4.27 Focus of expansion (FOE).	76
4.28 Path finder approach to obstacle avoidance	77
4.29 Test platform for SRV-1 robot with obstacles placed on it.	78
4.30 In left image robot environment and placement of obstacles are shown. In right image, the path followed by the robot is shown.	79
4.31 Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 1 and 2	80
4.32 Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 3 and 4	80
4.33 Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 5 and 6	81

4.34	Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 7 and 8	81
4.35	Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 9 and 10	82
4.36	Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 11 and 12	82
4.37	Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 13 and 14	83
4.38	Path followed by the robot when segmentation based obstacle avoidance is used. .	83
4.39	Optical flow field obtained between two consecutive images.	84
4.40	Zoom-in version of the optical flow field shown in Figure 4.39.	85
4.41	Path followed by the robot when optical flow based obstacle avoidance is used. . .	85
4.42	Image showing the detection of red colour blobs when colour blob detection algo- rithm is used.	87
4.43	Path followed by the robot to locate the energy resources.	88
4.44	Image showing LEDs used for docking operation	89
4.45	U image of Figure 4.44	91
4.46	V image of Figure 4.44	92
4.47	Output of colour blob detection algorithm	92
4.48	Y image of Figure 4.44	93
4.49	Blobs resulting from red LEDs in ON state.	93
4.50	Image representing blobs statistical information.	94
4.51	Search field for neighbouring blobs.	95
4.52	Flow diagram of control algorithm.	96
4.53	Robot approaching the LEDs for docking from left side with reference to the LEDs location.	97
4.54	Robot approaching the LEDs for docking from right side with reference to the LEDs location.	98
4.55	Initial pose of the robot before performing alignment.	99
4.56	Trajectory followed by the robot in experiment 2.	99

4.57 Trajectory followed by the robot in experiment 4.	100
4.58 Pose of the robot after performing alignment.	101
4.59 (a) Swarm of robots starting collective search for docking port. (b) One robot finds the docking port and the rest quit mission.	102
5.1 Image 1 (left) and image 2 (right)	105
5.2 Image 3 (left) and image 4 (right)	106
5.3 Image 5 (left) and image 6 (right)	106
5.4 Image 7 (left) and image 8 (right)	106
5.5 Image 9 (left) and image 10 (right)	107
5.6 Image 11 (left) and image 12 (right)	107
5.7 After grid points selection: image 1 (left) and image 2 (right)	107
5.8 After grid points selection: image 3 (left) and image 4 (right)	108
5.9 After grid points selection: image 5 (left) and image 6 (right)	109
5.10 After grid points selection: image 7 (left) and image 8 (right)	109
5.11 After grid points selection: image 9 (left) and image 10 (right)	110
5.12 After grid points selection: image 11 (left) and image 12 (right)	110
5.13 Reprojected 2D grid points.	111
5.14 Fiducial markers template.	112
5.15 Color blobs extraction.	113
5.16 Pattern recognition process. (a) Window around the selected blob. (b) Selecting the expected tail blob. (c) Search line from tail to the head blob along which cover blob will be searched. (d) Search line along which cover blob is searched for final validation of the pattern.	114
5.17 Determining the location and orientation of the four robots.	115
5.18 Ceiling mounted camera set-up for robot tracking.	117
5.19 Collective tracking of robots from ceiling mounted cameras.	118
5.20 Collective tracking of robots from ceiling mounted cameras.	119
5.21 Robot arena used for the visual guidance of the robots.	121
5.22 Image showing the shortest path to the target location for all the robots on the grid map.	122

5.23	Test 1: Positions of the robots in the arena before the test started.	122
5.24	Test 1: Robots path to the target location on the grid map.	123
5.25	Robot 1 colliding the obstacle boundary.	123
5.26	Test 1: Robots reached the target location successfully.	124
5.27	Test 2: Positions of the robots before the test begin.	124
5.28	Test 2: Shortest path to the target location on the grid map.	125
5.29	Test 2: Robots reached the target location successfully.	125
6.1	Scenario where group of robots performing search operation to find the object of interest.	131
6.2	Processor specific optimisation to reduce execution time.	134
6.3	Image pre-processing to reduce the amount of data to process.	136
6.4	Scale-space image pyramid.	137
6.5	Resolution switching based on distance to object.	138
6.6	Execution timing - SURF vs Optimised SURF.	139
6.7	Recognition rate - SURF vs Optimised SURF.	141
6.8	Objects used in training to extract SURF features.	141
6.9	Images from 16 poses of the 3D object.	142
6.10	Position of robots and objects of interest before experiment is performed.	143
6.11	Position of the robots when they recognise the object in the environment.	143
6.12	Sequence of Robot 1 positions when it detected and then missed object 3.	144
6.13	Sequence of robot 2 positions when it detected and then missed object 1.	145
6.14	2D and 3D objects used for experiment.	146
6.15	Position of robots before experiment.	147
6.16	All objects are localised by using information from the team of robots and visual tracking system collectively.	148
6.17	Error observed in localising object.	149
6.18	Input image used for boundary detection.	152
6.19	Segmented image.	153
6.20	Boundary vector plotted on the segmented image.	153
6.21	Visual tracking and localisation information (a) Left camera. (b) Right camera. . .	154

6.22	Zoom-in information of robot 1 position.	155
6.23	Profile between distance from object (pixels) and scale factor.	156
6.24	Mapping process.	157
6.25	(a) Distance vector from robot 1 mapped To coordinates of visual tracking system. (b) Robot tracking image from second ceiling mounted camera. (c) Global map generated using the two ceiling mounted cameras.	159
6.26	Zoom-in version of Figure 6.25a.	159
6.27	Experiment 1: Visual tracking system (a) Left camera. (b) Right camera.	160
6.28	Experiment half finished: (a) Left camera. (b) Right camera. (c) Map.	161
6.29	Mapping error caused by the orientation error.	162
6.30	Experiment 2: Visual tracking system (a) Left camera. (b) Right camera.	163
6.31	Experiment 2 half finished: (a) Position of robots in left camera after half of the experiment is finished. (b) Position of robots in right camera after half of the experiment is finished. (c) Progress on map generation after half of the experiment is finished. (d) Experiment 2 final map generated.	164
6.32	Experiment 3: (a) Left camera image from visual tracking system. (b) Right camera image from visual tracking system.	164
6.33	Experiment 3 final map generated.	165
7.1	Robotic organism scenario.	167
7.2	Robotic organism recognising the object in the environment.	168
7.3	Robotic organism gathering the visual information and making mosaic.	169
7.4	Communication set-up between robotic modules in the organism.	170
7.5	Communication network layers.	170
7.6	Application interfaced to TCP/UDP layer.	171
7.7	Application interfaced to ethernet layer.	171
7.8	Ethernet frame (IEEE 802.3 standard).	172
7.9	Example Replicator robots participating to form an organism [8].	174
7.10	Replicator robotic organism from three robots [8].	174
7.11	Communication within the organism.	175
7.12	Distributed object recognition within the organism.	177

7.13 Distributed information gathering - task communication.	179
7.14 Image re-projection on common plane.	181
7.15 Images re-projection using product of homographies.	182
7.16 Distributed information gathering - vision processing phases.	183
7.17 Slave 2 :Four step processing phases - Step 1 and 2.	183
7.18 Slave 2 :Four step processing phases - Step 3 and 4.	184
7.19 Multi-processor robotic organism.	186
7.20 Target 2D objects.	186
7.21 Obstacle avoidance referesh time: Single robot vs robot organism.	187
7.22 Timing analysis of object recognition algorithm with varying object features.	189
7.23 Object recognition: single robot vs distributed processing.	191
7.24 Unknown 2D objects.	192
7.25 Test arena for distributed vision processing in organism mode.	192
7.26 (a) Position of the organism when the test begin. (b) Position of the robot when it has identified the target object 2. (c) Identified location of the target object 2 in the image.	193
7.27 Object 2 surrounding scanned by the organism.	193
7.28 Images grabbed by the organism to form a mosaic.	194
7.29 Mosaic of images scanned around object 2.	195
7.30 Mosaic of images scanned around object 2.	196
7.31 Object detection in complete mosaic.	197
7.32 Trajectory made by the organism and location from which mosaics were generated.	197
7.33 (a) Mosaic from the images scanned around object 1. (b) Segmentation based ground elimination. (c) Objects detection in mosaics.	198
7.34 (a) Mosaic from the images scanned around object 3. (b) Segmentation based ground elimination. (c) Objects detection in mosaics.	198
7.35 (a) Erroneous image stitching. (b) Second last image. (c) Last image.	200
8.1 REPLICATOR robotic organism [2].	204
8.2 Communication within the multi-processor system.	205
8.3 Vision task distribution.	206

8.4	(a) Low resolution image for close object. (b) High resolution image for far object.	208
8.5	Classification probability.	209
8.6	Feature space clustering and histograms generation.	212
8.7	Classification and recognition ID.	214
8.8	Classification probability.	215
8.9	Processor usage.	216
8.10	Memory usage.	217
8.11	Frame per second - Four processors system.	219
8.12	Frame per second - Single processor robot.	220
8.13	Classification probabilities with increasing number of objects in the vocabulary.	221
	221	
	221	
8.16	Pioneer-3AT robot.	222
8.17	Traditional object recognition implementation - high processor system.	224
8.18	Memory usage for a robot with a single blackfin processor.	225
8.19	Processor usage for a robot with a single blackfin processor.	225
8.20	Frame processing time versus total SURF features for single processor systems.	226
8.21	Memory usage for multi-processing system.	227
8.22	Processor usage for multi-processing system.	229
8.23	Frame processing time versus number of SURF features for modular system.	230
8.24	Classification probabilities versus number of SURF features for modular system.	231

List of Tables

6.1	Object Localisation Information	147
6.2	Scale factor: From robot to ceiling camera.	156

Acknowledgments

I would like to express sincere thanks to my Director of Studies and Supervisor Dr. Reza Saatchi who provided me with the necessary support, advice and enthusiasm required to carry on this PhD research work. Many thanks to my parents who allowed me to stay abroad for carrying on with my studies. I am grateful to my wife for all the help she provided me to successfully finish the PhD research. I am grateful to my colleagues, specially Jan Wedekind and Georgios Chliveros for providing me necessary support and advice. Many thanks to Dr Fabio Caparrelli for funding my PhD from EU Project REPLICATOR. At the end, special thanks to Dr Lyuba Alboul for all of her moral support.

Declaration

Sheffield Hallam University

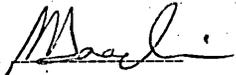
Materials and Engineering Research Institute

The undersigned hereby certify that they have read and recommend to the Faculty of Arts, Computing, Engineering and Sciences for acceptance a thesis entitled “**Development and Evaluation of Distributed Vision Processing Algorithms in Multi-Robotic Systems**” by M Shuja Ahmed in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Date: _____ 2013

M Shuja Ahmed _____

Director of Studies: Dr. Reza Saatchi 

Supervisor: Dr. Reza Saatchi 

Disclaimer

Sheffield Hallam University

Materials and Engineering Research Institute

Author: M Shuja Ahmed

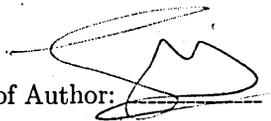
Title: Development and Evaluation of Distributed Vision Processing Algorithms in Multi-Robotic Systems

Department: Material and Engineering Research Institute

Degree: PhD Year: 2013

Permission is herewith granted to Sheffield Hallam University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.

Signature of Author: 

13 June 2013

Chapter 1

Introduction

A major research in the field of robotics is focused on moving the robots and vehicles autonomously in an unknown environment and taking intelligent decisions (e.g., grab object after recognising) to accomplish specific tasks. After achieving excellence in moving individual robots autonomously, the trend in the robotic research has shifted to collaborative achievement of tasks in multi-robotic environment, where multiple small size robots work together to achieve common objectives and goals [1] [2] [3] [4]. This has led to the concept of swarm robotics. The idea of swarm robotics is based on swarm intelligence which is inspired from the social behaviour exhibited by animals, specially insects. One example in the nature showing collective behaviour is the manner ants pick up the prey together, which is many times heavier than their own weight and bring this food back to their colony. This is shown in Figure 1.1a. The same behaviour when exhibited by the swarm of robots, is shown in Figure 1.1b, where a number of small size robots are collectively searching for a big Red coloured object. Once the object is found, then the robots have to move it. The object is too heavy for a single robot to move. Hence, a robot swarm needs to cooperate with each other for moving it. This cooperative behaviour exhibited by the robot swarm for moving the object collectively towards the light source is shown in Figure 1.1c.

Some of the swarm robotic systems can take more complex forms by introducing enhanced robotic features that, apart from contributing to the common objective by working on individual basis, robots can also be physically attached and de-attached from each other to become a bigger structure (i.e. a single robotic organism) for achieving a global objective [7] [8]. This is shown in Figure 1.2b where a group of robots physically join together to cross over a gap. Again, the inspiration comes from

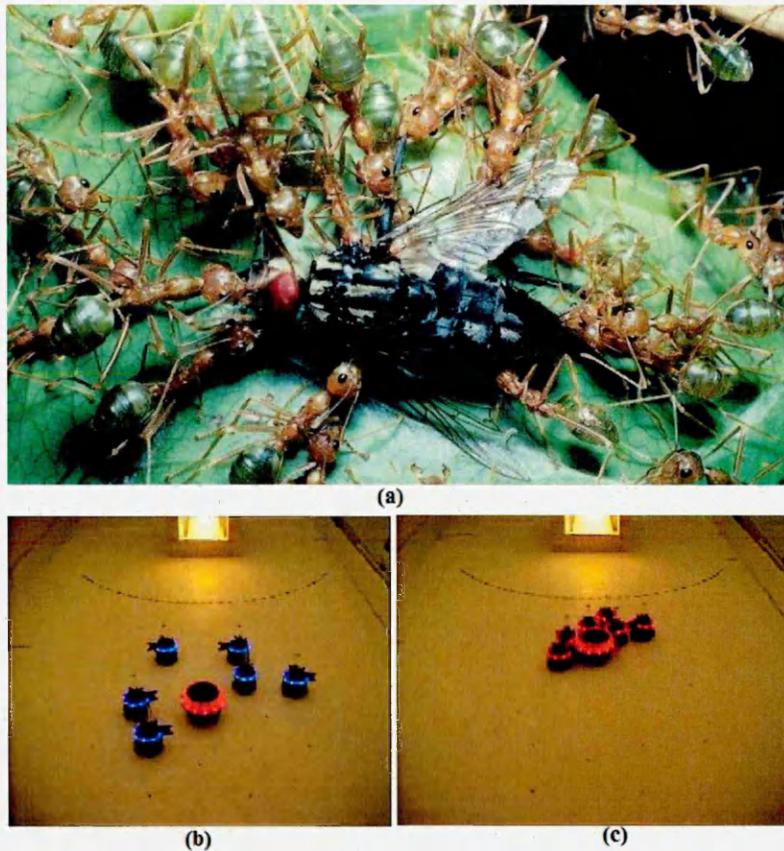


Figure 1.1: (a) Ants trying to move heavy prey [5]. (b) Robots search for the big red object in the environment [6]. (c) Swarm of robots moving the object collectively [6].

the nature as shown in Figure 1.2a. This intelligent practice is being adopted by a type of ants, commonly found in Central and South America. These ants categorise among themselves and select the members having suitable body size to fill the gaps between the stems and eventually form a bridge. This cooperative behaviour by the ants facilitates other ants to carry food back to their colony.

The term “Cooperation” requires “Knowledge Sharing”. In swarm robotics, the cooperative achievement of a common task by multiple robots requires the robot modules to share their knowledge with each other. In some cases, the robotic modules also share the computational resources with other team members. This knowledge and resource sharing lead to the concept of distributed computing [84] [83]. So the swarm robotic system is basically a distributed robotic system, that acquires rich energy and processing resources, but these resources are distributed among many robotic units. The efficient utilisation of these resources introduces new chal-

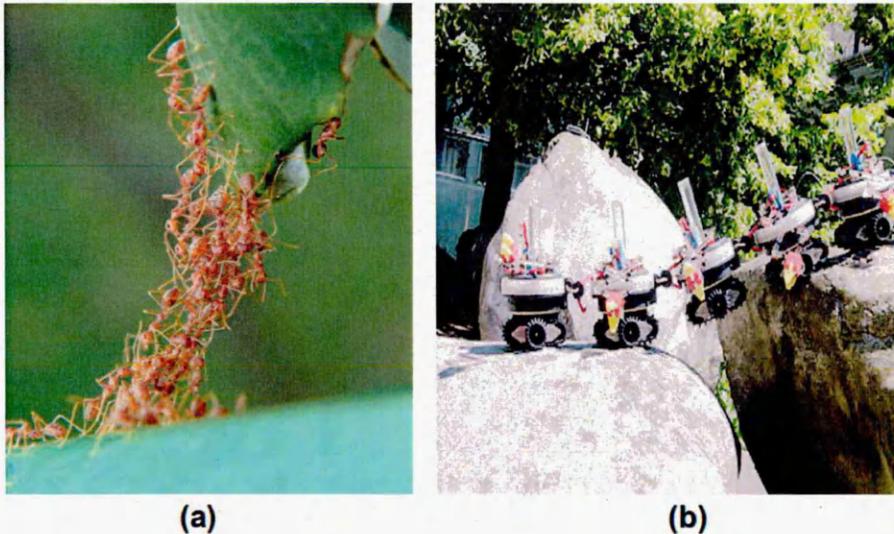


Figure 1.2: (a) Ants forming a bridge to facilitate other members for carrying food to the colony [9]. (b) Robots physically joining together and forming a bridge [6].

lenges of efficient knowledge sharing among multiple robots, so that the collective and cooperative achievement of the task can be made. In the field of robotics, the use of inexpensive vision sensors provides rich surrounding information which can be used effectively to enhance the robot's knowledge about the environment. The use of these sensors alone enables the solution to many problems in robotics (e.g. obstacle avoidance, object detection, scene understanding and pattern recognition). Although integration of these vision sensors in swarm robotics has led researchers to use the power of computer vision technology, but at a cost of increased computational complexity as these sensors generate huge amount of data that require processing. In swarm robotics, this computational complexity problem turns out to be worst as the target robots have very limited memory and processing power. In such systems, a single robot unit alone has limited processing power, but the swarm system as a whole is rich in processing resources, which are distributed among the whole system. So the computationally complex nature of the vision processing algorithms makes the vision processing task the best candidate to be distributed in the rich processing environment of the swarm systems. This distribution of vision information in swarm systems introduces the concept of distributed vision processing among multi-robotic systems [12].

Vision processing in miniature size robots is a difficult task to perform (e.g. scene understanding) and when it comes to a swarm of miniature size robots which are

visually guided and trying to cooperate with each other by sharing the knowledge they gained about the environment, this makes the scenario quite interesting and challenging. So when a swarm of robots is moving in an environment to achieve a certain task collectively, ideally the swarm is expected to achieve the task more quickly as compared with a single robot, of-course subject to how efficient knowledge sharing and task distribution is performed by the robots. The task distribution can be performed on equal level that is, all the robots are given the same task. For example, to perform energy foraging [2], searching and localising important objects in the environment or mapping the environment. Or it can be on demand basis that is, the robots which are performing computationally heavy tasks dynamically determine when they want to distribute the tasks and may ask for help from other robots which are less busy.

Hence, distributed processing in the swarm robotic systems is a very active area of robotic research in which researchers have solved many problems by sharing the knowledge learned from different on-board sensors. But most of these research outcomes are based on computer simulations, with very little work done using real multi-robotic systems. However, distributed vision processing in swarm systems is a new and open research area in which detailed research is required. The use of real multi-robotic system for performing the distributed vision processing task makes this research area interesting and more challenging.

1.1 Aims, objectives and background of the study

The main aim of this research is to address the manner distributed vision processing could be achieved in a multi-robotic system which could work in swarm and organism modes. The multi-robotic system, in either swarm and organism form, is itself a distributed embedded system environment, with processing and energy resources, very limited in a single robot unit, but rich in terms of the whole system. So, for distributed vision processing in swarm and the organism form, a number of distributed vision processing scenarios are developed with the following objectives.

- (i) Development and implementation of a library of vision processing algorithms which can execute on small size robot with limited onboard memory and processing resources. This library of vision processing algorithms will act like basic building blocks for the development of distributed vision processing scenarios

in swarm and the organism form. The vision processing algorithms are required to provide high frame processing rate (i.e., 40 frames per second or higher), so that they can provide real time response in the vision processing scenarios.

- (ii) Development and implementation of light weight vision based obstacle avoidance algorithms, so that when the robots are moving in the environment, they avoid colliding with the obstacles.
- (iii) Development of efficient vision based multi-robot docking support so that transition from the swarm to the organism can be facilitated.
- (iv) Provision of vision based object recognition functionality to the robots. This requires the development and implementation of an efficient object recognition approach suitable for executing on a small size robot.
- (v) Provide vision based surrounding awareness (e.g., by environment mapping and objects localisation) to the robots, which helps the robots to navigate in the environment.
- (vi) Efficient utilisation of memory and processing resources in a robotic organism by distributing the vision processing load among multiple processing modules. This requires the development and implementation of vision based tasks (such as, object classification, recognition and surrounding visual information gathering) in a modular fashion. The modular implementation of vision processing tasks facilitate in distributing the vision processing load, and provides higher frame processing rate, which helps the robotic organism to show quick response to its observations (i.e., information from vision sensor).

This research is carried out as a part of European Commissions FP7 research project "REPLICATOR" [8] [1]. REPLICATOR project is addressing multi robotic systems in which multiple robots have the ability to dock, share information, energy and computational resources with each other and are also able to locate the charging points to charge their batteries, if their battery charge is going low. These swarms of robots can physically join together, self assemble and artificially create a single robotic organism, whenever the need arises. For this purpose, these swarms of robots are required to interact physically with each other and also with their environment. Using a vision system on the robots, information about the surrounding environment as well as other swarms of robots can be gathered in every single robotic

organism. After performing various image processing algorithms, on the gathered information, it can be used in the visual guidance of the swarm of robots so that they can accomplish some specific tasks collectively. As every robot in the swarm gets independent processing resources, and considering that the vision processing tasks can be computationally expensive for miniature size robots, so a need to efficiently utilise the processing resources, in individual robots and also in assembled robotic organism, naturally arises.

This PhD research addresses the different vision based tasks of REPLICATOR project and presents the scenarios which essentially simulate how vision distribution can be achieved when the robots are in the swarm and organism forms.

Apart from the REPLICATOR project, the outcome of this research has applications in many diverse fields. It can be applied to provide robotic support to perform search operations in a hazardous environment, where sending humans to accomplish the tasks may involve risks or dangers to life (e.g a gas leak in a factory). It can be used to get robotic help in searching for survivals after natural disasters such as an earthquake. A very interesting application could be in planetary explorations, e.g a mission to Mars. In planetary explorations, a mission failure is generally not considered affordable and can cause huge sums of money to be lost, so the deployment of swarm of robots reduces the chances of a failure following the fact that, the malfunctioning of a single robot will not put the whole mission at risk.

The fact that, swarm robotics is an emerging field and most of the research performed is still limited to swarm intelligence theories and simulations, hence there is still a long way to go to achieve excellence in this field.

1.2 Contribution of Thesis

In this PhD research work, a swarm robotic system is considered in which robots not only work in a swarm mode, but can also physically join together to work in an organism mode. The swarm system considered in this research is shown in Figure 1.3, where multiple robot units forming a robot organism are shown. The title of this research work is “Development and Evaluation of Distributed Vision Processing Algorithms in Multi-Robotic Systems” and this research deals with many different scenarios, where the collective achievement of the common goal is performed in swarm and organism modes. However, as the title suggests, the main research is focused on how the vision processing is performed in the distributed fashion and

in distributed multi-robotic systems. The main contributions of this research are outline below.

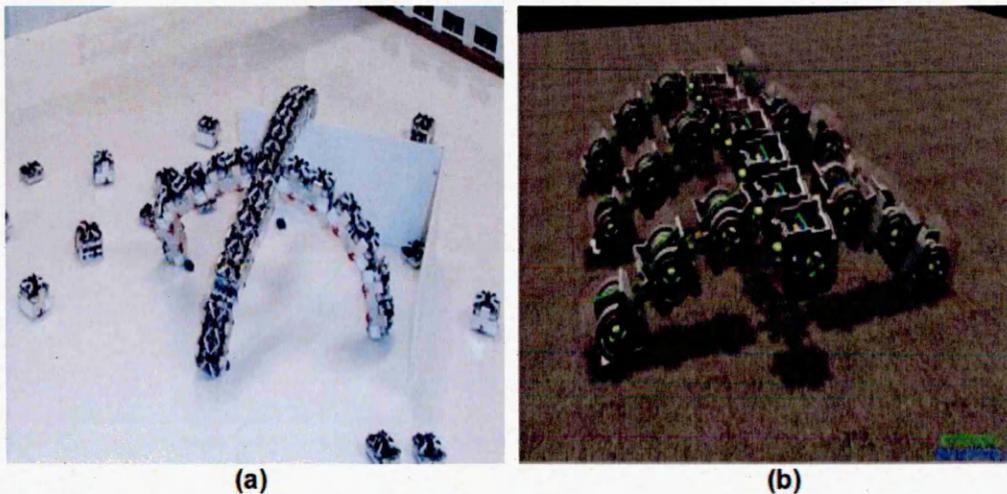


Figure 1.3: Multi-Robotic Systems (a) Symbion [7] (b) Replicator [8].

- (i) Development of multi-processor robotic system which simulates the processing and memory resources of a robotic organism.
- (ii) Development of a library of vision processing algorithms optimised for target embedded system based on Analog Devices Blackfin processor.
- (iii) Distributed object recognition and localisation by wireless connected swarm robotic system. This approach presents novel contributions (i.e., efficient image pre-processing and distance based resolution switching techniques) in the current state of the art appearance based recognition algorithm (i.e., SURF features based recognition). The developed approach makes it possible to achieve robust recognition on small size robots with limited onboard memory and processing resources.
- (iv) A novel distance vector features based environment mapping solution by a swarm of robots. This distance vector feature is easy to compute and share among swarm of robots, without overloading the wireless communication network, and hence, plays a vital role in environment mapping.
- (v) Development and implementation of distributed visual information gathering by making the collective use of processing and memory resources within the

robotic organism. This provides surrounding awareness to the robotic organism and facilitate it in navigating in the environment.

- (vi) A novel approach to object classification and recognition in a robotic organism in modular fashion. A vision processing architecture is developed which implements a modular object classification and recognition approach and shows that, the collective use of the processing and memory resources in a robotic organism can provide performance comparable to bigger robots equipped with high processing systems.

In order to clearly highlight the achievements, the above described contributions of the thesis are demonstrated in terms of distributed vision processing scenarios in swarm and the organism form. These swarm and organism mode scenarios are described one by one below, and shows the effectiveness of the contributions in these scenarios.

1.2.1 Object Recognition and Localisation in Distributed Robotic Systems

The first scenario addressed is “Object Recognition and Localisation in Distributed Multi-Robotic Systems”. For this scenario, as the robots are expected to come across a number of obstacles during the operation, so first of all, a number of vision based obstacle avoidance techniques are researched. A novel and very light weight vision based obstacle avoidance technique is developed which could run in parallel with other vision based tasks, while consuming low processing power (i.e., the execution time of obstacle avoidance algorithm is 22 msec). In this scenario, a set of miniature size robots, with limited on-board processing capabilities and resources, are used to recognise the objects of interest and then localise them in an unknown environment. The overall picture of a targeted scenario is presented in Figure 1.4, where a swarm of network connected robots is gathered near an assembly point and they have to recognise and localise the objects of interest (i.e. Cups images) in the environment.

The purpose of this scenario is the vision based object recognition and localisation in a distributed robotic system. Object recognition and localisation is a big challenge in computer vision and robotics. Although advances in computer vision have resulted in many object recognition techniques, most of them are computationally very heavy and require robot units to have high processing systems. When it

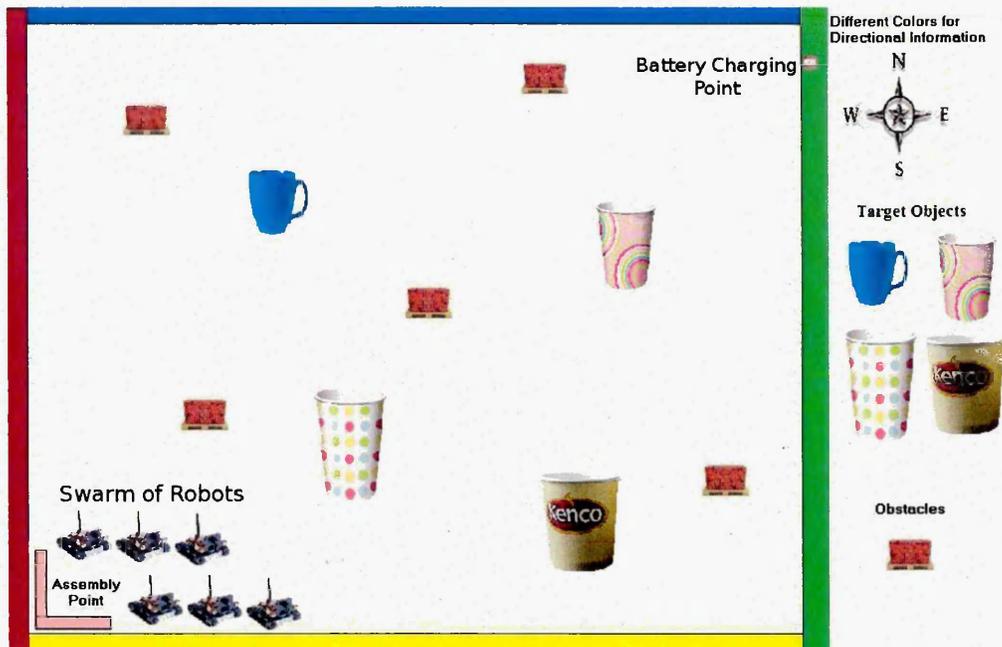


Figure 1.4: Scenario for distributed vision processing among swarm of robots.

comes to small size embedded robotic systems, these techniques can not be applied, because of the constraints of execution time. Here, a popular Speeded Up Robust Features (SURF) based recognition technique is adopted and some of the important changes are addressed which makes it possible to use it on small size robots with limited resources. A team of small robots are used which are given prior training to search for 2D (i.e., images of objects) and 3D objects of interest in the environment. So, unlike other swarm systems which rely on computationally light but inefficient recognition techniques to recognise the objects in the environment, the research work in this scenario bridges the gap between the computationally heavy and efficient recognition techniques and their implementation on miniature size robots. For the localisation of the robots and objects a new template, designed for passive markers based tracking, is developed. These markers are placed on the top of each robot and they are tracked by two ceiling mounted cameras. The information from the ceiling mounted cameras and from the team of robots is used collectively to localise the objects of interest in the environment. *The research outcome of this scenario gives two main contributions. First contribution is achieving reduced execution time (i.e., frame processing time reduced to 800 msec from 33 sec) of the appearance based recognition (Using SURF) algorithm, so that these algorithms can*

be executed on small size robots and can provide frame processing rate of at least 1 FPS (frame per second). The reduced execution time is achieved by performing the system specific optimisation and redundant data elimination from the images. And for increasing the recognition performance with respect to the distance from the target object, a novel resolution switching technique is developed. The second contribution of this scenario is the development of novel approach for the localisation of the objects of interest in the environment by fusing the visual features generated by the distributed multi-robotic systems.

1.2.2 Computer Vision Support for Robot Docking and Energy Foraging

In the second scenario, a computer vision support to facilitate robot docking operation and to search for energy points is addressed. As mentioned, the target swarm robotic system had the ability to physically join or dock together, whenever it became necessary. And for this purpose, their mechanical system required precise alignment before performing the docking operation. This docking operation is different from other swarm robotic systems where simple grippers are used on the robots to hold the other robots, and this can be performed from any direction without requiring the robots to precisely align with each other. The major drawback in the docking operation achieved using grippers is that, they do not have the ability to physically lift the docked robots. In the considered swarm robotic system, physical docking is required to move over a big obstacle or also to perform precise physical connection with certain objectives such as battery charging for guaranteeing long term operations. In this scenario, the information from vision sensors alone is considered to provide support for performing precise physical docking with the other robots. A very simple, but effective, solution based on LEDs, used in a specified pattern on docking station, is adopted. The approach presented in this research is found computationally less expensive and provided real time performance improvement on the small size robots. With the developed approach, a collective search operation for the docking port, performed by the swarm of robots, is addressed. This docking port could be used for multiple purposes, e.g docking to another robot or for an energy point. *The research work in this scenario presents a novel approach for performing alignment between multiple robots. And this alignment is essential for the formation of a robotic organism from the swarm. In more precise words, the approach developed here plays a vital role during the swarm to the organism.*

transition stage.

1.2.3 Environment Mapping by Distributed Multi-Robotic System

In this scenario, a vision based mapping of the unknown environment by the distributed multi-robotic system is addressed, where the robots are provided with the localisation information from a ceiling mounted camera system. Different from other robotic systems that rely mostly on sensors (such as laser range finders) which are very expensive and can not be used with miniature size robots, the technique addressed in this scenario solely relied on the vision information. *The research addressed in this scenario presents a novel method based on very simple visual features, which could be easily shared by the swarm of robots using a wireless communication channel. Using these visual features the robots collectively generated a sufficient environmental map which could be effectively used for the robot navigation.*

1.2.4 Distributed Object Recognition and Visual Information Gathering in a Multi-Robotic Organism

Different from the previous scenarios, in this scenario, the distributed vision processing is performed in the organism mode. *In the field of swarm or modular robotic systems, this is the first time, distributed vision processing research is performed using the distributed memory and processing resources of a robotic organism. In the proposed scenario, a new method is introduced which is adopted by the robotic organism to perform vision based obstacle avoidance, efficient object recognition, and visual information gathering in a distributed fashion.* In the robotic organism, the visual information is processed by multiple processing units at different stages and finally, the information is gathered in the form of image mosaics. These image mosaics provided the necessary awareness about the surrounding of the target objects to the robotic organism.

1.2.5 Distributed Object Classification and Recognition in a Robotic Organism

This scenario presents a novel method which implements a vision based object classification and recognition approach in a modular fashion, using the distributed memory and processing resources of a robotic organism, comprising of four robotic modules. This approach enabled the robotic organism to not only recognise the objects it is

given training for, but also facilitated the organism to learn new objects. This approach established a feedback mechanism between the different computing modules in the organism, and facilitated an efficient utilisation of the distributed resources. In this scenario, a detail comparative analysis is also performed with reference to the high performance processing systems. *It is found that, the efficient utilisation of the processing, memory and communication abilities of the organism, can provide performance which could compete with high processing systems.*

1.3 Thesis Outline

This thesis is organised in nine Chapters which are briefly described below.

- **Chapter 1** provides the fundamental information about the field of research area that is distributed vision processing in multi-robotic systems. It also explains the manner this research technically contributes to the field of computer vision and robotics.
- **Chapter 2** presents the literature review in detail. Initially, different research areas that have implemented distributed vision processing scenarios in swarm and organism mode are identified. Then for each research area, a separate literature review is performed, while keeping the relevance with main objective of vision processing in distributed systems. The different research areas identified in literature review are Communication, Obstacle Avoidance, Energy foraging and Vision Support for Docking, Localisation, Appearance Based Object Recognition, Multi-robot Environment Mapping and Distributed Processing in Robotic Organism.
- **Chapter 3** describes the details of the hardware used in the swarm and organism mode scenarios. For the organism mode scenario, a robotic organism is developed. Different hardware components which are integrated in the robotic organism, are discussed in detail. Finally, the on-board operating system used on each robot in the swarm is described briefly.
- **Chapter 4** describes the basic vision processing algorithms developed in this research. These algorithms provide the basis for the achievement of the swarm and organism mode scenarios. Two small scenarios that are “Vision based

Obstacle Avoidance” and “Vision based Robot Docking Support” are also explained. The experimental results from these scenarios guarantee real time performance of the basic vision algorithms.

- **Chapter 5** explains *Multi-robot Visual Tracking System* developed in this research to provide the localisation support to all the robots in a swarm.
- **Chapter 6** presents the distributed vision processing scenarios in swarm mode. Two main scenarios are identified. These scenarios include *Vision based Distributed Search Operation in Multi-Robotic Systems and Multi-robots Environment Mapping*. For these scenarios to work, vision based obstacle avoidance discussed in Chapter 4 and the Fiducial markers based visual tracking method discussed in Chapter 5 are also utilised.
- **Chapter 7** addresses the distributed vision processing in the organism mode. A scenario is presented which addresses the *Distributed Implementation of Vision Based Object Recognition and Distributed Visual Information Gathering* to provide surrounding awareness in the robotic organism. The robotic organism considered in this scenario comprised of three processing modules.
- **Chapter 8** presents the second scenario, addressing the distributed vision processing in the organism mode. This scenario presents the modular implementation of the object classification and recognition approach using the distributed resources of a robotic organism. The robotic organism used in this scenario comprised of four processing modules.
- **Chapter 9** draws conclusion and presents the future work to the study.

The manner the thesis is organised and Chapters are related to each other, is shown in Figure 1.5. The title of PhD research is *Development and Evaluation of Distributed Vision Processing Algorithms in Multi-Robotic Systems*. The distributed vision processing research is split into two parts; that is, Swarm Mode and the Organism Mode.

The swarm mode research is demonstrated in Chapter 6, where two scenarios are addressed to perform distributed vision processing in swarm systems. The relevant literature review and the hardware/software requirements are presented in Chapters 2 and 3, respectively. To work with the swarm mode scenarios, a multi-robot localisation and tracking system is developed, which is described in Chapter 5. The

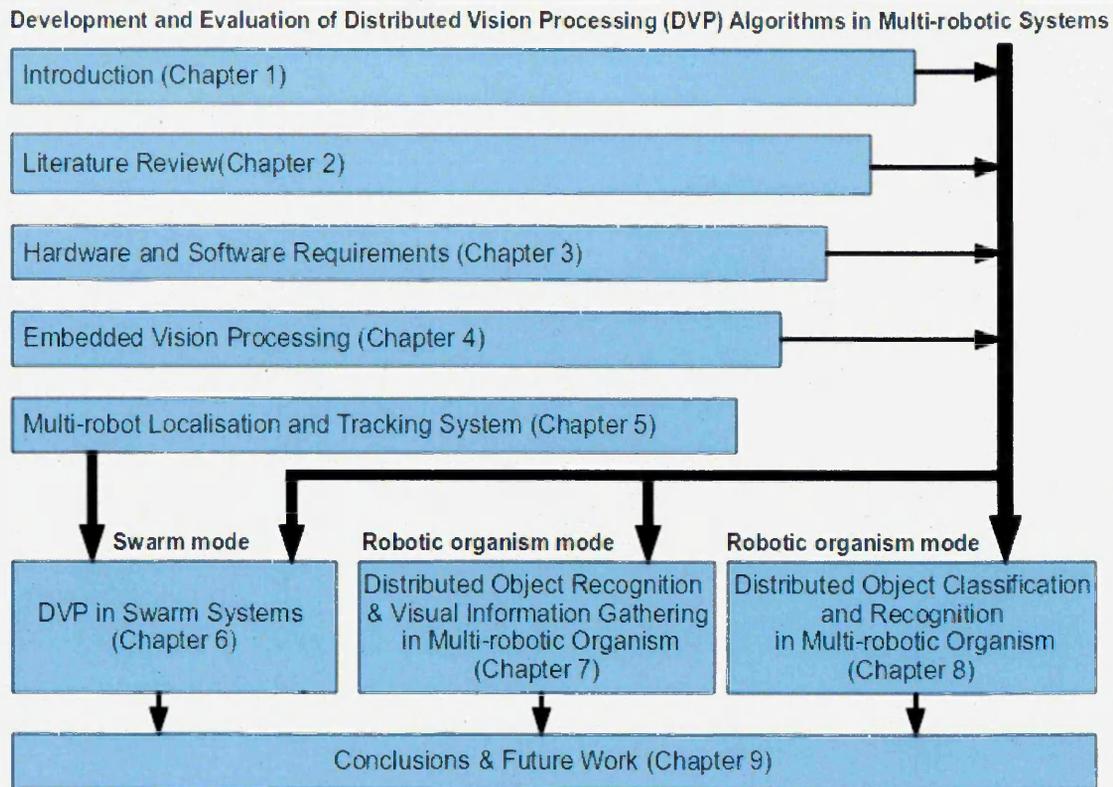


Figure 1.5: Chapters organisation.

swarm mode scenarios also rely on the vision based collision avoidance algorithms and other light weight vision algorithms developed specially to run on embedded platforms. These algorithms are described in detail in Chapter 4.

The organism mode research is addressed in the form of two main scenarios. The first scenario, described in Chapter 7, presents the distributed object recognition and visual information gathering. Whereas, the second scenario is described in Chapter 8 and it presents the modular implementation of object classification and recognition in the robotic organism. The related literature review and hardware/software requirements for the organism mode scenarios are described in Chapters 2 and 3, respectively. Like swarm mode scenarios, the robotic organism used in organism mode scenarios, also relies on vision based collision avoidance algorithms and other vision algorithms optimised specially for the target embedded systems. The implementation of these algorithms is explained in detail in Chapter 4.

Chapter 2

Literature Review

Swarm robotics [7] , multi-robotic systems [8] , evolutionary robotic systems [1] [2], focus on understanding how cooperative achievement of a certain mutual task can be guaranteed in such an efficient manner that the overall time to achieve the goal can be reduced and optimised. The back bone of this efficient achievement is an intelligent task distribution. In [7] [2], a distributed computing scenario is presented, among the swarm of self-assembling robots (with no vision information) to overcome obstacles which would be difficult for a single robot to avoid. This is shown in Figure 2.1 where swarm of robots self assemble and try to move over an obstacle.

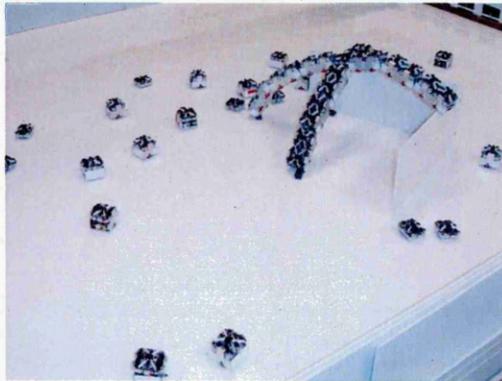


Figure 2.1: Swarm of self assembled robots moving over an obstacle [7].

In [13] [14] [15] a vision based approach addressing the collective behaviour of the robots, in transporting items which are too heavy to be moved by a single robot, is presented. This example shows the distribution of the same task to all the robots to achieve a common goal (i.e. to move the object of interest). In [8] distribution of

vision based task among individual robots and also between robot modules forming a single robotic organism, to efficiently utilise the processing resources is proposed. Most of the research work done in the field of distributed processing in multi-robotic systems, is carried out using on-board high performance processing systems [16]. If miniature size robots are used then online/on-board processing is not performed and major processing is still performed by centralised high performance systems. In [12] [18] [19] the work done in the domain of RoboCup competition is presented where the robots are required to identify and track the ball collectively by distributing the vision information between each other. Using an omni-directional vision sensor, every mobile robot, built a local map of the environment, shared it with other team members and helped all robots to build their own vision of the world. This is clearly a distributed vision processing system. To achieve this cooperative distributed vision processing task, robots with high processing systems were used. One of the robot which cooperates in this task is shown in Figure 2.2.



Figure 2.2: Single robot from a team, detecting and tracking a coloured ball to perform distributed map building [18].

From Figure 2.2, it can be seen that a large size robot is used which is trying to recognise a coloured ball. Detection and tracking of the ball using colour features make the vision processing, in this distributed task, less complicated. In [20] a series of Motorola 68000 microprocessors were configured in master-slave configuration to describe a distributed vision computing architecture. In this master-slave configuration, the master processor extracts the information from the images and passes them to the slave processor for detailed processing. A high speed network communication was used for information sharing between the high performance Motorola processors which make the distribution of vision based tasks possible. In [22], robots were not equipped with vision sensors, but used mobile agents to enquire a

common vision system for required information. This is an example of centralised processing architecture. The use of centralised architecture is necessary, if all the robot modules needed to access common set of data on equal basis, for example, in [22] all the robots need to access a centralised processing system for the processed visual information. The use of centralised architecture is not very famous in swarm robotic systems, because the failure of single central processing sytem will result in failure of complete system. More over, in centralised processing architecture, the processing resources of all the robots in a swarm can not be best utilised.

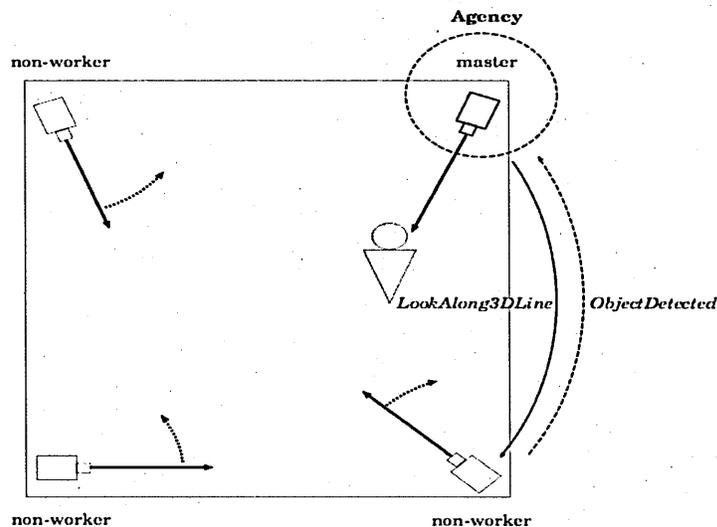


Figure 2.3: Active vision agents based target detection and tracking [23].

The higher utilisation of processing resources is only possible, if the information is processed in distributed fashion. The use of active vision agents is also made in [23] to distribute the vision tasks between the observation stations as shown in Figure 2.3. These active agents were communicated between the observation stations for performing the vision based target detection and tracking. As the observation points were fixed, so simple messages communicated between the stations can be used to focus the attention of all the observation points on a single target. It can be noticed that, in this type of distributed vision processing, the system in whole is not really sharing the visual data between the observation points. Each observation point performs an independent on-board vision processing, and then shares the target detection information in terms of target location in the environment. And as the observation stations are fixed and know the other stations' locations, so it is

easy to translate the target location with reference to the other stations view. This type of distributed vision processing is far less complicated as compared with the distributed vision processing in swarm robotics, where the position of the robots keeps changing and it is also difficult for the robots to keep on tracking each other's location.

In distributed systems, the efficiency of a system is directly related to how data distribution is implemented. In [24] [25] a method called Divisible Load Theory (DLT) is proposed for managing data distribution and its application to vision processing is described. This method is based on the actuality that the data or tasks can be divided arbitrarily into independent modules, where each module can be processed independently from others. Another vision processing work based on DLT is presented in [26] where a simple edge detector called Sobel operator is selected as a distributed application. For distributed image processing, two different task partitioning and scheduling strategies are compared. The first is Partitioning and Scheduling Strategy using DLT (PSSD) and the second is Equal Partitioning Strategy (EQS). The good performance achieved with this system is mainly because of the efficient scheduling and load sharing strategies implemented in communication middleware DLT. As there is no optimised version of DLT which makes it suitable to run on an embedded system environment, so the processing and specially memory requirement of DLT is usually high. Like DLT, another approach to distributed computing called Grid Computing is presented in [28] for performing distributed processing of remotely sensed images. For distributed processing of these remotely sensed images, the images are processed in different stages using a very high performance system. The different stages are shown in Figure 2.4.

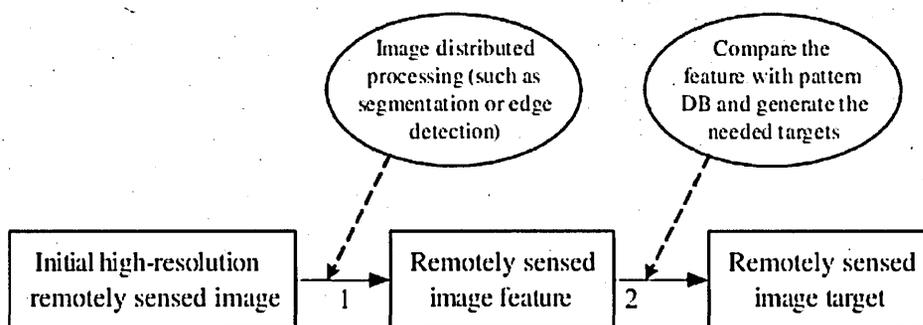


Figure 2.4: Processing stages for remotely sensed images [28].

In the first stage, the high resolution images are transferred to the system through the internet. Then segmentation and edge detection based features are extracted. These features are passed to another system which performs the recognition task. For recognition purposes, the system utilises pattern recognition techniques and makes use of a very big library of target data. For this system to work, a high performance system and high communication bandwidth medium are required. In another work [27] an Image Mining System (IMS) is presented. In IMS, for online image processing, the different parts of the image are processed in a parallel fashion. For creating a parallel processing system, a number of personal computers are connected by a local area network. The connectivity through local area network enables large amount of data to be exchanged between number of processing systems. The different processing stages for this system are shown in Figure 2.5. The first computer performs the image acquisition, the second performs image enhancement, the third performs feature extraction and then finally the mining operation is done. High communication bandwidth is required for transferring images between computers for further processing. Moreover, image mining is itself a difficult task to perform as it requires high processing for matching the extracted image features with the library of features stored in the memory. The library of features usually is also large and essentially have large memory requirements.

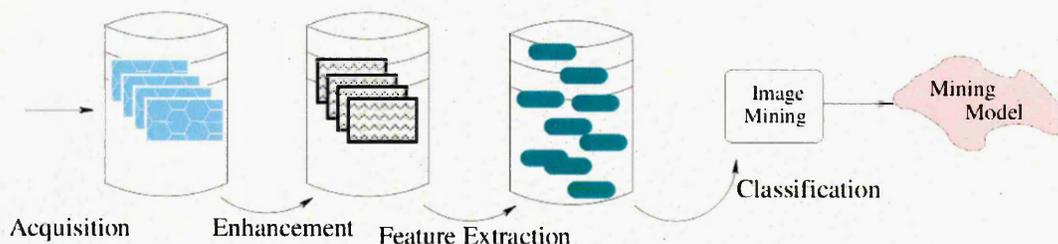


Figure 2.5: Image processing stages for image mining system [27].

In [16], the use of Player/Stage software tools for distributed computing in robotics is presented. In this software, the Player is a robot device server which provides a network transparent control to the target robot, whereas Stage is a robot simulator which helps in simulating the robot and its environment. But the major difference is that, all vision processing is performed on the central computer with basic commands to control the robot may be executed on-board. A relevant distributed vision based formation control research is carried out in [29] in which

a swarm of robots try to visually maintain the formation using motion segmentation approach and also avoid colliding with each other in the absence of any communication medium. For performing the formation control, the robots use an omni-directional camera with a mirror mounted on their top (as shown in Figure 2.6) which provides a panoramic image to each robot. From the help of panoramic image, the robots get the information about the other robots for maintaining the formation. As each robot processes its own panoramic image, so this implementation is more towards vision processing in distributed systems, rather than distributed vision processing in a complete swarm robotic system.

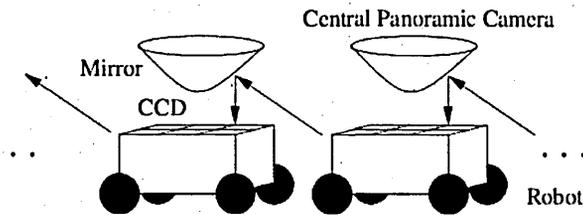


Figure 2.6: An omni-directional vision based formation of mobile robots [29].

In [17] a distributed machine vision application is presented that describes a visual recognition method developed for IUB (International University Bremen) RoboCup team. To simplify the tasks, rather than executing the vision algorithm on the robots, an external camera is placed over the field surface and is connected to a high processing system (system acts as a server). The images captured from this camera are processed at high rate by an external system (i.e. 25fps) and the vision based robot detection performed using different colour markers placed on the robots. Upon determining the location and orientation of the robots, the external system gives commands to the robots using UDP protocol over a 100Mbps network. To establish the communication and defining the transmission packets, a customised library of algorithms was developed rather than relying on other middleware such as COM or CORBA. The approach of not using the communication middleware and relying on the customised developed library for data transmission acknowledge the possibility of achieving communication between swarms of robots in real time, considering the embedded system implementation. All this work, in distributed vision processing field, is performed using high performance systems which are the basic requirement for most computer vision algorithms. Moreover, if low power embedded systems are used then either they are using simple vision based tasks or they are used as still

observation stations set in a predefined configuration (i.e. [20]). To demonstrate the distributed vision processing task for multi-robotic systems operating in swarm and the organism mode, a number of distributed vision processing scenarios were defined. For the implementation of these distributed vision processing scenarios, the research areas identified which needed to be explored are: Communication, Obstacle Avoidance, Energy foraging and Vision Support for Docking, Localisation, Appearance Based Object Recognition, Multi-robot Environment Mapping and Distributed Processing in Robotic Organism.

2.1 Communication

Communication is a backbone of any distributed computing system. It provides a medium through which data or information can be distributed among several processing systems following some communication rules called protocols. This communication medium can be wired or wireless. In swarm robotic systems, there is a more general description of types of communication that can be established among robots, i.e. explicit or implicit communication. In explicit communication, the information is shared between robots directly using a communication medium. While in implicit communication (e.g stigmatic), the information sharing is achieved by interaction with the environment (e.g by modifying the environment) or by observing the actions of the other robots in the surrounding [30]. In [31], simple communication strategies are explored which can be used to perform implicit and explicit communication in a swarm robotic environment. Implicit communication can be observed in nature, where ants use pheromones (a chemical substance) to mark the environment for conveying messages (as shown in Figure 2.7). Hence it led some researchers to use pheromones based communication among robot swarms [32].

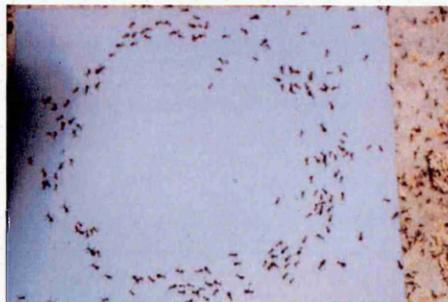


Figure 2.7: Ants following artificial pheromones [112].

If the environment in which robots are required to operate is highly dynamic then implicit communication is not considered suitable as it relies on the changes that are brought into the environment or on the observations of other robots [21]. So, for distributing vision based tasks and knowledge sharing among robot swarms, wireless communication medium can be considered as a sensible choice. A layered protocol structure, based on data transport layers (inspired from computer network system), to define protocols for wireless robot communication is presented in [33], but it assumes the provision of communication network layers in robot which could be possible if some communication middle-ware (a third party software that connects other software together) is used. A cooperative distributed problem solving system is discussed in [34], but it totally relies on a centrally controlled shared memory architecture for data exchange which makes it unsuitable for systems where data management or storing are not centralised, such as swarm of robots. A comparison of TCP, UDP, TEAR and Trinomial protocols for formation control of robot swarms is carried out in [35] with the help of middle-ware and centralised base station. The comparison of MANET routing protocols in mobile robotics is performed in [36], where a 7x7 grid of closely spaced WiFi nodes (equipped with a high performance system to simulate wireless connected robot modules) are used and a mobile robot moves around them. This is shown in Figure 2.8.

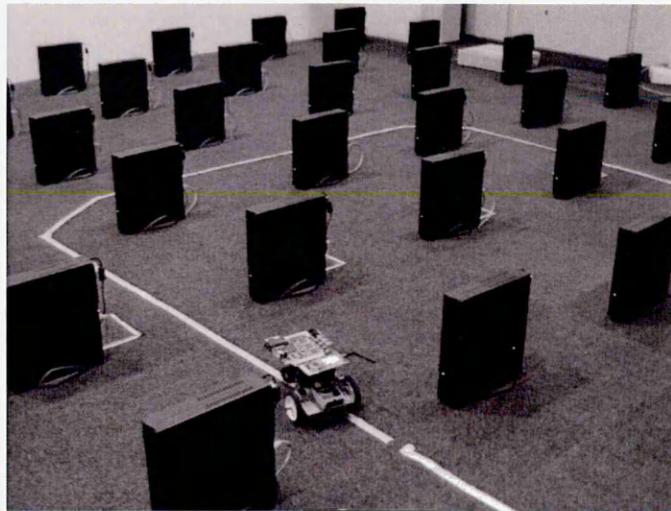


Figure 2.8: Robot moving around 7x7 WiFi grid [36].

The reason for moving the robot around the WiFi nodes is to describe how robot routing table information is updated when a robot moves in different regions of

a wireless grid. This closely simulates a swarm of robots connected using adhoc networks, but in order to keep a record of all the robots in surroundings by every robot in a swarm, the preprocessing demands could be high, and can overload the network. Moreover, all the wireless nodes which are simulating swarm of robots are static. If they are also mobile, then changes in the routing table will be required often.

In [37], the use of PERA (Probabilistic Emergent Routing Algorithm), an adhoc type wireless protocol, for wireless connected robot is described, but again it requires a network communication module in the robot firmware, which basically utilise protocol defined by PERA to establish the communication. Similarly, distributed communication among robots, as presented in [38], using CORBA (Common Object Request Broker Architecture) and in [39] using Robot Software Communications Architecture (RSCA) is also possible, but these communication middle-ware are too processing and memory intensive for miniature size robots which are based on an embedded system with limited memory and processing power.

2.2 Obstacle Avoidance

When the robots are performing distributed vision processing tasks, they are not usually stationary. In the considered scenarios, the robots expect to encounter an obstacle and so need to perform obstacle avoidance based on vision, as vision is the only source of providing surrounding awareness to a robot. Computer vision provides many ways to achieve the obstacle avoidance task. In [40] an obstacle avoidance technique using a monocular vision camera together with laser range finder is addressed. The author has performed testing of the algorithm in an outdoor highly unstructured environment, but the testing system used was not strictly an embedded system, as all the image processings were on a development platform. Therefore, there is possibility that this technique may not be able to meet the real time constraints. Another commonly used method for obstacle avoidance, addressed in [41], is based on edge detection. In this method, the vision algorithm tries to determine the vertical edges of the obstacle and helps robot to move around the edges without colliding with the obstacle. In [42] another approach called Lucas-Canade optical flow based obstacle avoidance is used. This algorithm used for MAVs (Micro Aerial Vehicle) in urban environment. Similar to the approach addressed in [42], Horn and Schunck optical flow based algorithm is also adopted in [43] to perform

obstacle avoidance by the autonomous ground robot. They used optical flow information to determine the image velocity vectors. These vectors could be split into two terms, translational and rotational components. Then using the translational component of the velocity vector, the time to contact information for the obstacle can be calculated. This could then be used to determine when the obstacle was very close to the robot and necessary action was required in order to avoid a collision.

2.3 Energy Foraging and Vision Support for Docking

When robot swarms are performing an operation, there are possibilities that the robots get hungry (i.e. they run out of battery) and need to look for energy resources (can be collective or individually) so that the collective mission can be finished successfully. There are a number of studies in the field of distributed robotics which addresses the collective energy foraging [1]. Search of energy resources can be performed using the information provided by vision sensors [8]. In [7] [8], swarm robotic systems were described which could look for energy resources individually or share energy resources by docking to each other. "Autonomous Docking" is another area of research in swarm robotics community. To perform autonomous docking of robots, Infra-red (IR) or vision based information is used in most of the swarm robotics projects. In [44] [45] IR sensor information is used to perform docking. The use of vision support for docking is presented in [13] [14] [15] [46] [47], in which swarm of robots dock together to drag heavy objects. In [48] [49], a more complicated docking mechanism is presented which requires a precise alignment to successfully attach two robot modules, so help from both vision and IR sensor is utilised.

2.4 Localisation

In pervasive computing systems, determining the location information of the important objects in the environment is given great importance. In multi-robotic systems, this location information may be helpful to gather the where-about awareness to the robots which are working in an unknown environment. Determining the localisation information using radio frequency identification (RFID) data [50] tags is a popular technique (an example of RFID tags attached to the walls is shown in Figure 2.9), but advancements in machine vision field has shifted the research trend towards vision based tracking and localisation systems.



Figure 2.9: RFID tags attached to walls for robot localisation [50].

In machine vision, to localise an object, its detection and tracking in the environment using stream of images are required. The overall localisation performance depends strongly on the robustness of the object detection. A number of vision processing algorithms are developed which locate objects in the images captured in the natural environment (i.e. unconstrained images). The vision processing demands of these algorithms are normally high due to the complexity of the environment. Another reason for complexity is the 3D appearance of the objects in the environment due to which a 3D object recognition algorithm is required. If there is large number of objects to localise, then markers based techniques are adopted [51], especially when working in the indoor robotic environment. Markers used in these techniques can be identified as active or passive markers. In [52] an infrared sensor based active marker technique is presented. In the localisation system installed on the robot, the author has used an image sensor to detect the position of infrared sources, attached to the ceiling and then finally obtained the 2D position and orientation information of the object. For more precise positioning, detection of more than one IR sources is required. Similarly, in [53] ceiling lamps were used as natural landmarks for the robot localisation. The image of a ceiling lamp used for robot localisation is shown in Figure 2.10. The author has intelligently made use of the pre-install lamps as the landmarks, which reduce the overall cost of the solution as no additional landmark was required.

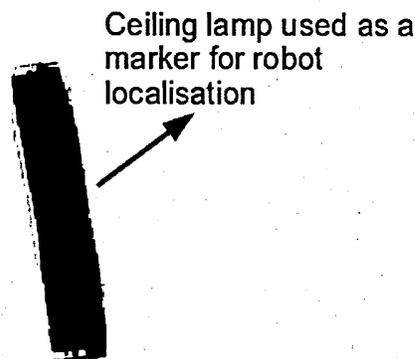


Figure 2.10: Negative image of a lamp on ceiling used for robot localisation [53].

In [54], the author has used a colour histogram based landmark detection algorithm. The landmark used has a symmetric design with repetitive colour patterns. To make the landmark detection less sensitive to illumination changes, a colour similarity based approach is used to update the colour histogram model for the landmark model. It is observed that, the solutions to active markers based approaches are computationally less expensive and can easily run in real time. But the big drawback is the energy consumption by these markers as these markers require power to become functional. The solution to active markers can be more expensive as presented in [52], for detecting infrared sensors attached to a ceiling, where each robot is required to be equipped with a camera and the direction of the camera should be facing vertically upward.

If the robots working in the environment are less in number (e.g. two or three robots), then an active marker solution may be justified as it makes the localisation problem easier. But in case of multi-robotic systems, where the number of robots working in the environment may be large (e.g. ten or more) then an active marker solution does not remain energy efficient as each robot has to provide power to make the markers functional. In such cases, the solution is to use passive markers.

The passive markers solution is normally very cheap to install, but the most challenging stage is their robust detection. For their reliable detection, passive markers are required to be designed very carefully such that their appearance in the environment is very prominent. In [55], the author has done a performance analysis of different shape passive markers and has shown how the markers' size vary depending upon the density of information (i.e. information about marker ID)

encode in it. The author has shown that, square shape markers require a simple algorithm for their reading and detection. But, circular markers provide more robust localisation information as shape fitting algorithms work more efficiently when more points contribute to the shape fitting process. In [56], the author has demonstrated the use of passive markers to identify and track the robots playing in robotic soccer team. To detect the position of robots, the author has used a camera mounted at an oblique angle rather than directly on the top. No colour information is used in designing the markers which reduces the possible number of robots which may be identified. In [57], an open source implementation of a marker-based vision system, called Cantag, is described. Most of the markers used in Cantag are represented in a binary form and their coding scheme provides binary information about the markers' identity. Some examples of markers from Cantag are shown in Figure 2.11.

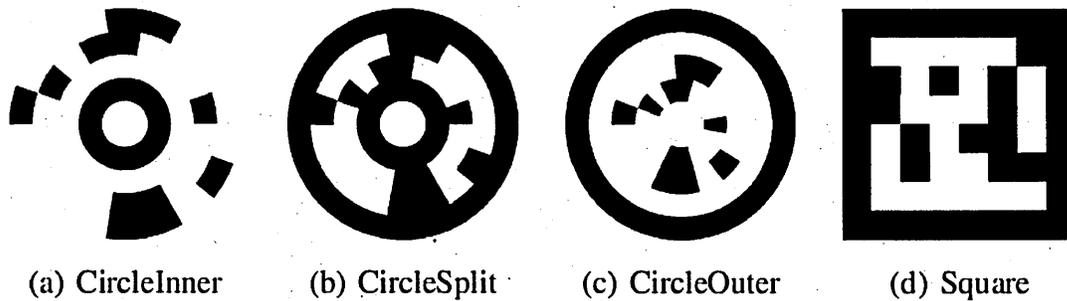


Figure 2.11: Some example markers from cantag [57].

To detect these markers, very complex image processing techniques are adopted in Cantag. As the density of binary information stored in the Cantag binary marker is high, so it is likely that it provides false result if the marker is viewed from a large distance and at sharp angle. To improve the performance, in Cantag, the author has also shown the effect of varying the marker size. But, if it is to be used in an environment where large number of small robots are working and needed to be given an ID, then it is logical to keep the size of the marker at least smaller than the robot top surface.

Similarly, in [58] a low cost localisation system using binary markers, attached to the ceiling, is presented. To identify the markers, a camera (facing vertically up) is mounted on every robot. In this case, high identification accuracy was possible when the robot crossed under the marker. If the marker was viewed from a sharp angle, false identification was possible. So, in multi-robotic environment, a marker

based technique which provides enough information to identify the robots robustly, and also with less complicated algorithm, would be preferred.

2.5 Appearance Based Object Recognition

One of the distributed vision processing scenarios in swarm mode addressed in this research, performs the “Distributed Object Recognition and Localisation in Multi-Robotic Systems”. In this scenario, the major task given to swarm of robots is to perform an appearance based recognition of the set of known objects in an unknown environment. Appearance based recognition is an important research area in autonomous robotics field and has been applied in many applications. A SURF (Speeded Up Robust Features) feature based approach to appearance based recognition for performing robot navigation task is presented in [59] using omni-directional camera images. The detection of SURF features in omni-directional camera image is shown in Figure 2.12. As computing SURF features is processor intensive, so a Core2Duo-2.66GHz high processing system was used to achieve the real time performance which makes the task less challenging.

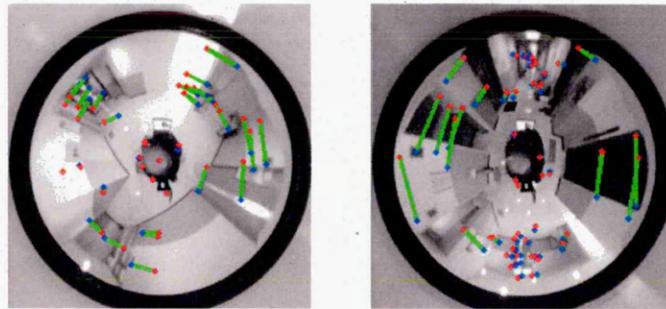


Figure 2.12: SURF features detected in omni-directional camera image [59].

In [61], an application to traffic sign recognition using SURF features in embedded system environment is described. The system is first given a training to extract feature points and create a library of visual words. The system is designed to work in a real life unstructured environment, so measurements are taken to show the system robustness to view point changes and changes in lighting conditions. The advantage of SURF features is that they provide scale and rotation invariant features [62]. For recognition purposes, the system implements the “bag of visual words” approach [61]. The idea is to extract SURF feature points from all the traffic

sign images provided in the library, and then group these feature points into a large number of clusters (i.e. the features with similar SURF descriptors will be assigned or lying in one cluster space). Each cluster will be considered as a visual word representing a pattern shared by all the features lying in that cluster. This way one single image can be represented by the bag of visual words with local features mapped into visual words as a vector containing the weighted count of each visual word. This bag of visual words is used to perform the model description of the sign shown in the image and hence, facilitates the real time recognition of the sign. In some approaches [63], this concept of clustering the feature space [62] is called feature space quantisation. This provides a significant saving in memory compared to the normal approach of saving all features independently during building the visual words vocabulary and thus makes the approach more suited for implementation on embedded systems.

Some researchers have used probabilistic models on the top of SURF feature based bag of visual words to perform robot localisation and mapping [64]. In comparison to this, in some approaches [65], authors have relied on Harris features instead of SURF features to reduce the computation time, and have used Partially Observable Markov Decision Process (POMDP) probabilistic methods to track the probability distribution of the robot's where-about and to essentially localise it. Another example of Harris feature based localisation is described in [66], where Harris extracted features are used together with Scale-Invariant Feature Transform (SIFT) descriptors. And then Support Vector Machines (SVM) is used for classification. But the use of Harris features together with SIFT makes the approach computationally very heavy even for a high processing system. In another work [67], rather than relying on SURF features alone, authors have used advantage of Conditional Random Fields (CRF) to discard those features that do not show geometric consistency. In [68] [69], a detailed comparison of SIFT, PCA-SIFT and SURF feature based approaches to recognition is carried out. SURF was found fastest to compute and still provided SIFT comparable performance. Authors in [69] have used the K-Nearest Neighbour (KNN) approach for feature matching and Random Sample Consensus (RANSAC) to get rid of outliers and have shown that selection of method to perform recognition mainly rely on the target application.

In [70], another probabilistic model based approach is developed, to perform appearance based recognition and robot localisation. Unlike the work presented in [59] [61] [63] [68], linear image features extracted using Principal Component

Analysis (PCA) [71] are used. The appearance model developed is represented as a probability density function of the image feature vectors given the location of the robot. In PCA, the eigenvectors (i.e. image feature vectors) of the image are computed and are used as an orthogonal basis for individual image representation. In Figure 2.13, an example image and the first three eigen vectors computed are shown. The motivation of using eigenvectors is that only few of them are required for visual recognition (although for full image reconstruction all eigenvectors would be required). These eigenvectors represent the eigenspace. PCA algorithm projects the image data onto the eigenspace and resulting in the uncorrelated projections (i.e. eigenvectors).

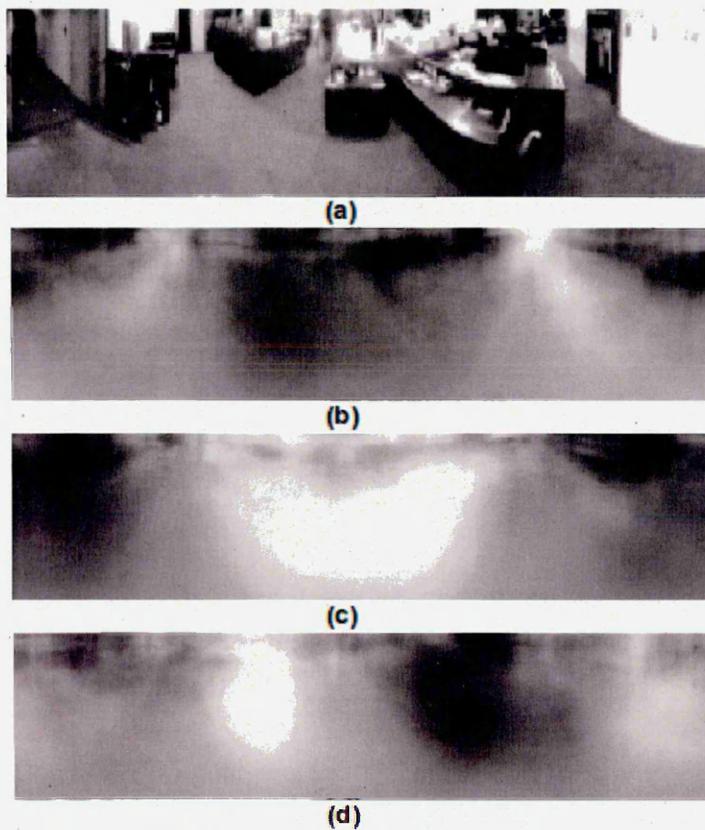


Figure 2.13: (a) Input image. (b) First eigen vector.(c) Second eigen vector.(d) Third eigen vector. [70].

The eigenspace analysis reduces the dimensionality of the feature space tremendously as compared to the SURF or SIFT features space as in [66] [68] [69] but, in general, it is more sensitive to the effect of “perceptual aliasing”. The effect of

perceptual aliasing is shown in Figure 2.14. In Figures 2.14a and 2.14b, two different images are shown. The image reconstructed from 20 dimensional eigen vector, of both of these images, is shown in Figure 2.14c. In other words, the two different appearing images may look similar in eigen space.

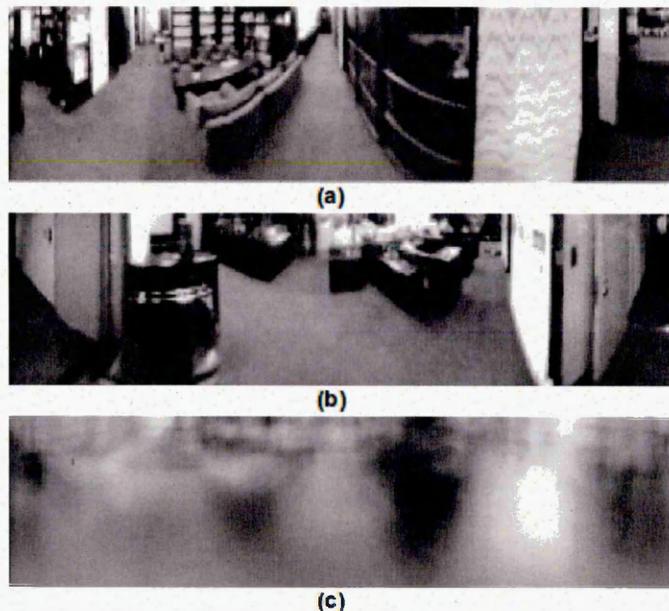


Figure 2.14: (a) Image 1. (b) Image 2. (c) Reconstructed image from 20 dimensional eigen vectors. [70].

In another work [60], fast gradient based feature extraction approach is used together with PCA (in place of SURF or SIFT features) to perform the recognition task. Like the work done in [61] [63], feature space is quantised using k-means algorithm, to make an efficient use of memory. To boost the search process of finding the corresponding features in the recognition phase, kd-tree algorithm is used which stores the centroids of the feature clusters, resulted from k-mean algorithm. The system used for recognition was a 3GHz high processing system which took 350 msec on average to recognise an object. This seems a slow performance considering the use of a high processing system.

Another similar approach using SIFT feature based recognition is presented in [72] where features (feature space quantised using k-mean) in the vocabulary tree are dynamically assigned different weights which are according to the uncertainty (i.e. entropy) associated with the features. The author has claimed that, in most of the appearance based matching approaches which successfully avoids the use of

common features (based on the feature entropies) to perform matching, still use those features which are not common, but at the same time they do not provide strong object recognition clues. It is shown that this problem can be overcome by using the vocabulary tree algorithm which modifies the feature entropies in such a way that it discourages the use of those features which have given poor recognition results.

It is to be noted that, in most studies, a high performance system is used for performing recognition. So in spite of excellent results achieved with these recognition techniques, when it comes to swarm of small size robotic systems, the bottle-neck of the slow rate of visual information processing forces the researchers to make huge compromises with the recognition performance and switch to computationally less expensive algorithms.

2.6 Multi-robot Environment Mapping

In the second distributed vision processing scenario in swarm mode presented in this research, a multi-robot environment mapping problem was addressed. Environment mapping is a process in which a robot senses its surrounding using onboard perception sensors and tries to obtain a global map. The generated global map essentially helps the robot to navigate autonomously in the environment. Environment mapping is a challenging problem to address and in most cases, the robots require human assistance if they are exploring the place first time. Some researchers have worked around this problem in which the robots keep on localising themselves and at the same time, they also build the environment map. This is called Simultaneous localisation And Mapping (SLAM) in robotics field. In [73], a vision based SLAM is presented in which the SIFT (Scale Invariant Feature Transform) feature points are extracted between the images to determine the robots' updated position and hence, it is used to determine the robots' location and to map its trajectory. The use of vision based solution on one hand provides the rich surrounding awareness to the robots but at the same time, it increases the computational time considerably. This leaves most of the vision based solutions limited to be implemented using high performance robots only. In multi-robotics operations, the robots being used have simple design to reduce the cost of overall system. So a single robot unit usually have limited onboard memory and processing resources. For a single robot implementation, researchers have used many different sensors such as laser range finders,

infrared sensors, sonar and vision sensors. But most of the research is focused on using laser range finders. In [74] an approach to outdoor mapping is addressed using 2D laser range finder. Similarly, in [75] a stochastic approach is adopted to an environment mapping using laser range finder. From the approaches adopted by [74] [75], it can be seen that due to the physical size, high power and processing requirement of the laser range finders, they are used with large size robots. Moreover, a single laser range finder can cost from £800 to £3000 which makes it unsuitable to be used in multi-robotic environment where the objective is also to keep the cost of each robot unit low. In [76], the information from the laser scanner is fused with the vision sensor to provide a more accurate map of the environment. But this further increases the computational demands of the approach. In [77] [78] a multi-robot environment mapping problem is addressed. But the results are limited to simulations only. In [79], a grid based mapping solution using multiple robots is addressed, but it also utilised high performance systems. However, when using a group of small size robots, it is preferred to use simple and computationally less expensive algorithms such that, the task can be achieved with limited on-board memory, processing and energy resources.

2.7 Distributed Processing in Robotic Organism

As described in Chapter 1, the latest research in swarm robotics or multi-robotics systems' is shifting to design more complicated systems that are reconfigurable. These reconfigurable modular robotic systems hold a very unique feature that they are not only capable of working in a swarm state, but they are also configurable to produce different forms of three dimensional robotic organisms. A relevant research is addressed in European Commissions Seventh Framework FP7 Replicator [80] and Symbrion [81] projects which presents complicated designs of robot modules. These robot modules are capable of physically joining together and creating a three dimensional robotic organism, whenever the need arises. The reason for robot modules to physically join together is to perform some operation collectively which is not possible to be done by a single robot. For collective achievement of the operation, the robots also need to create certain shape of the organism, when they join together.

Collective achievement of tasks is the basic concept behind the idea of swarm robotic systems [82], which comprises of many simple robotic modules with limited onboard memory and processing resources and basic sensors (e.g., infrared used

onboard). In swarm robotic systems, to show collective effort in achieving a task, the robot modules need to share their knowledge with each other over a wireless medium. This knowledge is processed at multiple stages in different robotic modules to make it meaningful for a complete swarm system. All the robot modules in a swarm share their memory resources, to hold the knowledge, and also processing resources, to process the sensor data in different stages. The sharing of memory and processing resources in swarm robotics introduces the concept of distributed computing [83] [84]. The complete swarm system is actually a distributed robotic system in which each individual robot has limited onboard resources, but the swarm system as a whole is considered very rich in memory and processing resources, which are distributed in the complete system. The sharing of these resources in modular swarm systems requires a reliable communication medium. The limited onboard resources in swarm system, which also includes the energy resources, act as a bottle neck even in establishing a communication medium. Due to limited onboard energy resources, the robots in a swarm are equipped with low power wireless boards which provide low signal to noise ratio and hence, a less reliable communication medium. If the swarm of robots are equipped with basic sensors, such as infrared, and share the information or knowledge learned from these sensors, then the wireless communication medium can be used effectively as these sensors generates less data. But if the robots in a swarm are equipped with vision sensors, then it becomes very difficult to share the visual data between different robots as vision sensors generate huge amount of data. It is to be noted that, the use of vision sensors in robotics is becoming very common because a single vision sensor can be used to achieve multiple objectives. But when these vision sensors are used in swarm systems, then the use of low power wireless communication medium make it difficult to exchange visual data among the robots.

The reconfigurable modular swarm systems are described as Networked Robotics in [85]. They are also known as Modular Robotic Systems in [86] [87] [88], which describe the design of modular robots with the ability to create different forms of locomotion systems. These locomotion systems are inspired from nature, such as snake, round shape and walking systems. In such systems, once the robot modules join together, then they can establish a physical communication medium. The provision of physical communication medium can ensure a reliable communication and hence, it can facilitate the robot organism to effectively utilise its rich distributed processing and memory resources. It is to be noted that, after forming the organ-

ism, if the system still relies on the information learned from the basic sensors, such as infrared, which is easy to process, then the system will not be fully exploring its communication capabilities and its rich processing resources. To fully utilise the communication and distributed resources, the distributed system environment established within the organism, can be used to perform vision based tasks in a distributed fashion. The vision based tasks require intensive processing, huge amount of memory for its operations and high communication bandwidth for information exchange. The provision of rich memory and processing resources and strong communication established within the organism, make it an ideal system for the processing of the vision based tasks. Once the robotic modules form a robotic organism, then the processing and memory resources accumulated within the robotic organism can be efficiently utilised to acquire surrounding awareness using the available onboard sensors. These distributed resources can be used by the organism to gather the surrounding information and to classify and recognise the different objects in the environment.

2.8 Conclusions

In this Chapter the most relevant research areas have been identified which need to be explored to successfully perform distributed vision processing in the swarm and organism modes of the multi-robotic system and a detailed literature review in each of the research areas has been presented. From the literature review, it can be concluded that, most of the distributed processing research in swarm systems is based on computer simulations, and very little work has been performed using real multi-robotic systems. If the use of real multi-robotic systems is made, then in most of the relevant distributed vision processing research, high performance robots or systems have been used. In case, a swarm of small size robot was used, then either the use of basic vision processing algorithms is made or the major vision processing is performed on high performance servers with basic control commands implemented on-board on the swarm of robots. It has been concluded that, distributed vision processing in multi-robotic systems which could work in a swarm or organism form, is an open research area which requires detailed research. In this research, to address distributed vision processing in the robotic swarm or the robotic organism, the use of a real multi-robotic system which comprises of small size robots, is made which makes this research more novel and challenging.

Chapter 3

Hardware and Software

Requirements

Based on a detailed literature review and several experiments, a number of distributed vision processing scenarios are defined as described in Chapter 1. For each of these scenarios, the required hardware and software are identified. In this Chapter, the details of the hardware and software configuration used to implement the swarm and organism mode scenarios, are discussed. The hardware requirements for the swarm and organism mode scenarios are separately identified. As this PhD research is carried out as a part of the European Commission Seventh Framework Programme FP7/2007-2013 research project REPLICATOR, so in order to ensure compatibility, the selection of hardware made is solely based on what are used in the REPLICATOR project. The software in terms of robot operating systems is identical for both swarm and organism mode scenarios. This chapter is divided into the following sections.

- (i) Hardware requirements for swarm mode scenarios.
- (ii) Hardware requirements for organism mode scenarios.
- (iii) Selection of an operating system (robot firmware).

3.1 Hardware Requirements for Swarm Mode Scenarios

The hardware components for the demonstration of the swarm mode scenarios are explained in the following sections.

- (i) Swarm Robot Units.
- (ii) Hardware for Swarm Communication.
- (iii) Robot Tracking Cameras.

3.1.1 Swarm Robot Units

In Figure 3.1a, a single REPLICATOR robot unit is shown. These robot units are not fully functional so for the demonstration of the swarm mode scenarios, Surveyor SRV1 robot was selected as it uses the same processing unit (i.e. Blackfin processor) as the one used in the REPLICATOR robots. A single Surveyor SRV1 robot is shown in Figure 3.1b and a swarm of these robots to be used in the scenarios is shown in Figure 3.1c.

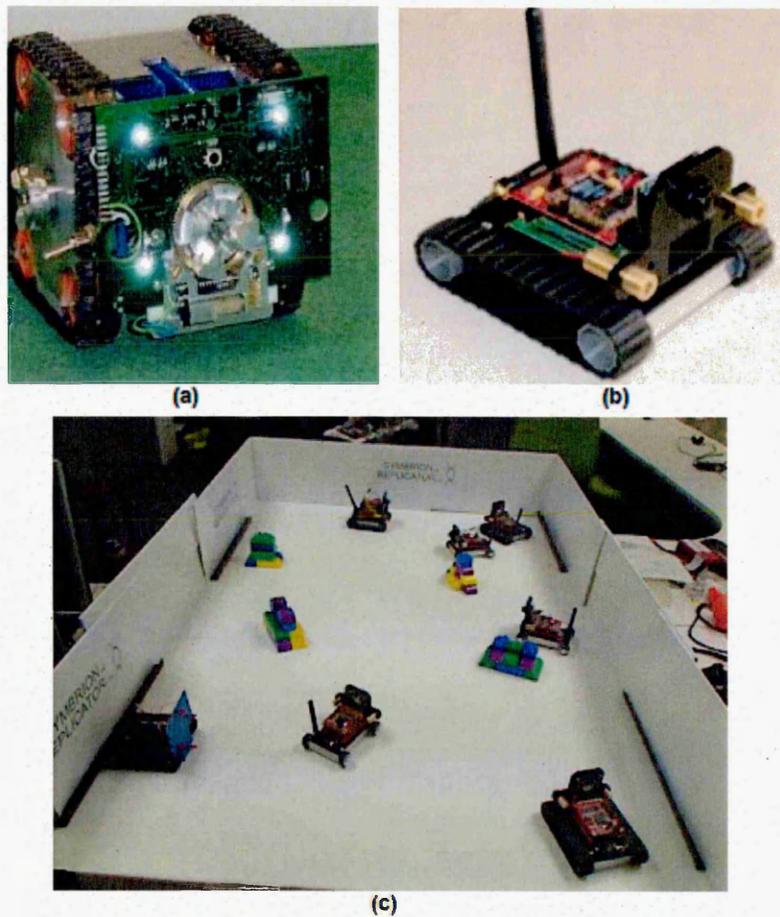


Figure 3.1: (a) REPLICATOR robot [8]. (b) SRV1 blackfin robot by surveyor corporation [89]. (c) Swarm of SRV1 robots

Surveyor SRV-1 [89] is an open Source Wireless Mobile Robot, especially developed for Autonomous and Swarm Operations. It is mainly designed for research purposes. Surveyor SRV-1 robot uses Blackfin camera board with 500MHz Analog Devices Blackfin BF537 processor, a digital video camera with configurable resolution from 160x128 to 1280x1024 pixels, two laser pointers, and WLAN 802.11b/g networking on a quad-motor tracked mobile robotic base. It can be operated remotely and can be programmed as a self-navigating autonomous robot. It can run on-board interpreted C programs, or user-modified firmware. It can be remotely operated from Windows or Linux base station with help of Java-based console software provided with the robot. The Java console software also includes a built-in web server using which SRV-1 can be controlled through a web browser from anywhere in the world. Detailed documentation on SRV-1 Robots can be found in [90]. The SRV1 robot has wireless connection (WLAN 802.11b/g based network) to the code development platform and through this wireless connection, the resultant binary files, obtained after compiling C programs using GNU compiler, can be downloaded and executed on the robot platform. The raw images captured by the robot and also the processed images can be uploaded to development platform through the wireless connection for debugging.

Robot Features

- Open Source: This allows the customers to re-use the already developed software and helps in rapid development.
- Robot is fully programmable for an autonomous operation.
- Extensive software support through third party applications.
- Robot can be tele-operated via web browser.
- Host software has built-in web server.
- Robot can execute programs written in interpreted C and stored in on-board Flash.
- Wireless remote control or viewing up to 100m indoors and 1000m outdoors (line of sight).
- Robot can be controlled from a terminal/console for easy testing.
- Firmware can be programmed with GNU bfin-elf-gcc.

Robot Hardware

- Processor: 1000mips 500MHz Analog Devices Blackfin BF537, 32MB SDRAM, 4MB Flash, JTAG.
- Camera: Omni-vision OV9655 1.3 mega-pixels 160x128 to 1280x1024 resolution.
- Robot Radio: Lantronix Matchport 802.11b/g WiFi .
- Range: 100m indoors, 1000m line-of-site in outdoor environment.
- Sensors: 2 laser pointers for ranging, support for up to 4 Maxbotics ultrasonic ranging modules and various I2C sensors .
- Drive: Tank-style treads with differential drive via four precision DC gear motors.
- Speed: 20cm - 40cm per second.

Robot Software

- Robot Firmware: Easily updated, written in C language under GPL Open Source, compiled with GNU bfin-elf-gcc and bfin-uclinux-gcc toolchains.
- On-board Programming: Interpreter for C language with special robot-specific commands, used to run programs directly from Flash memory.
- Development Tools: GNU toolchains via <http://blackfin.uclinux.org>
- Console Software: Java based application for Windows, MAC and Linux.

3.1.2 Hardware for Swarm Communication

To achieve the tasks collectively, the robot units are required to share their knowledge with each other. To enable this knowledge exchange, the wireless communication medium was established between the swarm of robot units shown in Figure 3.1c and for this purpose, a Linksys Wireless Access Point was used. The wireless access point used in this implementation is shown in Figure 3.2. The Linksys access point supports data rates up-to 54Mbps, which is up to 5 times faster than 802.11b. It can be easily configured from any web-browser and supports 64/128 bit WEP encryption.

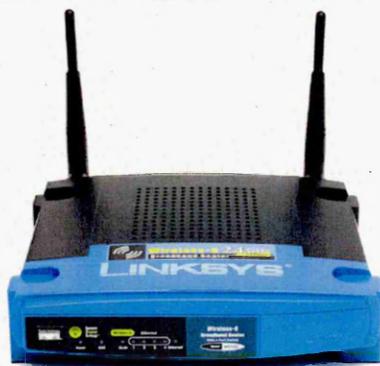


Figure 3.2: Linksys wireless access point [104].

3.1.3 Robot Tracking Cameras

To provide localisation information to the swarm of robots, a fiducial markers based tracking system was developed as described in Chapter 5. Two ceiling mounted cameras were used to track the position of swarm of robots in the environment. The selection of these ceiling mounted cameras was done very carefully as they were mounted higher from the surface of robot arena, so that they can cover more area. At the same time they had to provide a higher resolution image too, so that enough pixels contributed to the fiducial markers. After experimenting with some web-cams, Logitech WebCam Pro 9000 was selected as shown in Figure 3.3. This web-cam provides 2 mega-pixels for video and 8 mega-pixels for picture resolution. It can be configured to provide the resolution of 160x120, 176x144, 320x240, 352x288, 640x480, 800x600, 960x720 and 1600x1200 pixels. In the beginning, 1600x1200 resolution was selected, but then it was found difficult to process such high resolution images at high rate. So finally, 960x720 resolution was configured for grabbing the images from ceiling mounted cameras.



Figure 3.3: Logitech webcam Pro 9000 [105].

3.2 Hardware Requirement for Organism Mode Scenarios

Similar to the swarm mode, the REPLICATOR robot hardware was not fully functional to make a full robot organism. As in the robot organism, there were more than one robot physically docked together, so unlike the case of swarm scenarios, it was difficult to find pre-designed robot hardware for this purpose. To address the distributed vision scenario in organism mode, a multi-processor robotic system was developed in this research. In this multi-processor robotic system, multiple Analog Devices Blackfin processors were used, which were connected through the Ethernet medium. Every processor in the multi-processor robotic system, simulates the processing resources of a single robotic module which contributes in the formation of the robotic organism. In other words, the complete multi-processor robotic system simulates the distributed processing and memory resources which are gathered in a unified robotic organism. The hardware components used to develop this multi-processor robotic system included the following.

- Analog Devices - Blackfin Processor
- Blackfin Evaluation Board EVAL-BF5xx
- Blackfin Extender Board EXT-BF5xx-Camera
- Blackfin Experimental board
- CMOS camera sensor
- Robot base and Motor Control Board
- On-board Communication
- Navigation board

3.2.1 Analog Devices - Blackfin Processor

Analog Devices-Blackfin processors refer to the family of 16 or 32-bit micro-processors with built-in Digital Signal Processor functionalities. These features are traditionally only accompanied by small and power efficient micro-controller. Hence, it results into a low power processor architecture that can run operating system and can simultaneously handle complex numeric tasks such as real time H.264 video encoding.

Because of its relevant features, many hardware development kits have used Blackfin processors. Some of the Blackfin hardware development processors are BF522, BF527, BF533, BF537 and BF561 etc.

Blackfin CM-BF537E Core Module

From the entire family of Blackfin processors, the Blackfin core CM-BF537E was selected to be used on the robot organism. This core module comprises of three main components, i.e. ADSP-BF537 Blackfin processor from Analog Devices, 32 MB SDRAM and 4 MB flash memory. As the Blackfin processor also integrates an Ethernet controller, so a chip dedicated to Ethernet physical layer is also mounted on the core. The provision of two 60-pin expansion connectors is made on the core so that a variety of extension modules, provided by the company, can be connected to the core. The hardware user manual for the core module is available from [91]. Figure 3.4 shows the top side of the core module CMBF537E. Some details about the comprised components of CMBF537 core are given below.



Figure 3.4: Blackfin CM-BF537E core [91].

CM-BF537 PROCESSOR: The Blackfin processor combines DSP capabilities and microprocessor features in a single architecture design. The provision of microprocessor features is made by a 32-bit RISC architecture, a basic Memory Management Unit (MMU) for memory protection, data cache, instruction cache and by providing support for a variety of hardware peripherals. Like other processors, Blackfin processor allows two modes of operation that are Supervisor and User mode. In the supervisor mode, access to all resources can be made whereas in the user mode some sources are reserved. A Single Instruction Multiple Data (SIMD) architecture

is used to integrate DSP functionality in the Blackfin processors. This architecture offers two hardware multiplier accumulators, two Arithmetic Logic Units (ALUs) and a barrel shifter. The architecture allows the execution of three instructions per clock cycle. In the instruction set of its processor, some instructions to facilitate video and image processing are also provided. In general, BF537E processor has maximum speed of 600 MHz, 132 Kilo-Bytes of on-chip memory (64 Kilo-Bytes instruction memory and 16 Kilo-Bytes is used as instruction cache), 64 Kilo-Bytes data memory (32 Kilo-Bytes used as data cache and 4 Kilo-Bytes can be used as scratch-pad memory). An external SDRAM (upto 512 MB) can also be connected through a PC-133 compliant controller. An address space of 32 bits is used for accessing memory and the I/O devices. A Direct Memory Access (DMA) controller is also present for performing fast memory and I/O transfers. More information about BF537E Blackfin processor is provided in the data-sheet [92] [93] and the hardware reference manual [119]. Some of the key features of BF537E processor are given below:

- Due to high execution speed, Blackfin BF537E processors fulfil the camera devices needs. Space for later enhancement is also provided if more processing power is required.
- The processor architecture provides parallel peripheral interface (PPI). The PPI has a dedicated clock input, 16 data pins and 3 frame synchronisation pins. For high speed transfers, DMA can be used, but PPI is made suitable for a variety of applications by several general purpose modes. PPI is also used for transferring data from camera devices.
- Two bidirectional synchronous serial ports (SPORTs) with adjustable word length and support for frame synchronisation are also present. SPORTs are normally used to transfer digital data to DACs.
- The full duplex Serial Peripheral Interface Bus (SPI) is also supported.
- A Fast Ethernet MAC peripheral, supporting operations like 10BASE-T and 100BASE-T modes is also an important feature of Blackfin processor.

MAIN MEMORY: A 32 MB SDRAM is available on the core as a main memory. It supports the maximum clock speed of 133 MHz. The main memory is connected to the processor through data and address buses.

FLASH MEMORY: A Flash memory is also integrated to the Blackfin core. The total size of Flash memory is 4MB, but it is not fully addressable. It is divided into two 2MB memory banks. A General Purpose I/O (GPIO) pin is used to switch between its lower and upper 2 MB banks.

3.2.2 Blackfin Evaluation Board EVAL-BF5xx

Using an expansion slot, the BF537E core module can be connected to the evaluation board EVAL-BF5xx. The basic components of the evaluation board are RJ45 Ethernet plug, an SD card slot, a UART-to-USB converter, JTAG plug, two expansion connectors, voltage regulator and a power connector. The evaluation board EVAL-BF5xx is shown in Figure 3.5 below. The hardware user manual and schematic can be found in [100]. BF537E core together with the EVAL-BF5xx provides a basic embedded environment which can be connected to the development platform. Using a UART-to-USB converter, the UART port of ADSP-BF537 processor can be connected to USB port of the development platform. Using this connection the communication is set-up with the development platform and program can be downloaded and tested on the Blackfin processor.

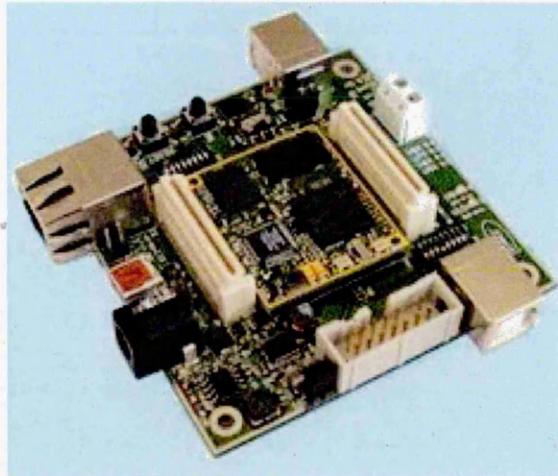


Figure 3.5: Evaluation board EVAL-BF5xx [100].

3.2.3 Blackfin Extender Board EXT-BF5xx-Camera

To connect the evaluation board EVAL-BF5xx with the CMOS camera, the extender board EXT-BF5xx-Camera is used as a bridge. The design of extender board is very

simple and no microchips are installed on it. It simply provides connectors for one LCD and two CMOS cameras. The extender board is used on the robot organism to provide connection to the CMOS camera. The hardware user manual and the schematic can be found in [101]. Blackfin Extender Board EXT-BF5xx-Camera is shown in Figure 3.6 below.

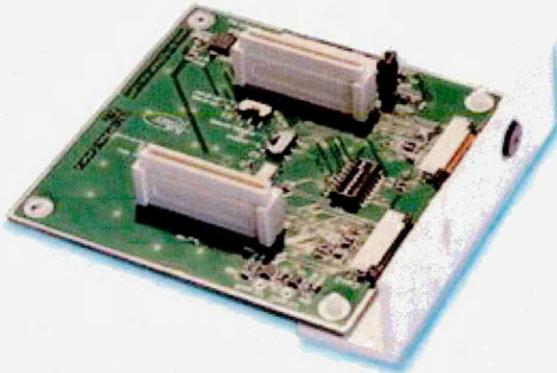


Figure 3.6: Blackfin extender board EXT-BF5xx-Camera [101].

3.2.4 Blackfin Experimental Board

The Blackfin Experimental Board “EXT-BF5xx-EXP”, shown in Figure 3.7, is an extender board which can plug-on the EVAL-BF5xx board. It provides all the connectors from the Blackfin processor (e.g. Digital I/O, PPI signals, Serial clock and data lines, timers outputs, power and ground) on the solder-able pads. The Blackfin Experimental Board was needed for two reasons. First is to provide interface between the Blackfin processor and the Motor Control Board (shown in Figure 3.9b), which is further connected with the four servos used to drive the robot wheels. Required pulse width modulation (PWM) signals were generated from the Blackfin processor using the on-board timers. These PWM signals were used to generate the frequencies which controls the wheel speed through the servos. Similarly, to control the direction of wheels rotation, the digital I/O were also routed from the Blackfin processor to the Motor Control Board through the Experimental Board. The second reason for using Experimental board was to interface the Navigation Board with the Blackfin processor. For this purpose serial data and serial clock signals were required and they could be made available through the Experimental board only.



Figure 3.7: Blackfin experimental board “EXT-BF5xx-EXP” [106].

3.2.5 CMOS Camera Sensor

Two CMOS camera sensors, that is, OV7660 and OV7670, were selected to be used with the robot organism. It was decided to use one of this to provide surrounding awareness to the organism. These are low cost and low power consumption camera sensors and could provide 640x480 (VGA) resolution images. These camera sensors could be installed directly to EXT-BF5xx-Camera extender board, but for the proper placement of the camera sensor on the organism, Flexi cable was used. In Figure 3.8 OV7670 camera sensor and Flexi cable are shown.

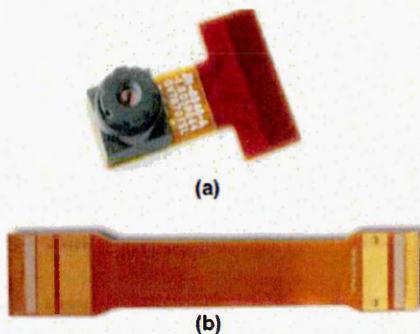


Figure 3.8: (a)OV7670 camera sensor [107]. (b) Flexi cable [108].

3.2.6 Robot Base and Motor Control Board

The robot base from the Surveyor SRV1 robot was used on which all the Blackfin processing boards, comprising the robot organism, were installed. This robot base

includes four motors to drive the robot wheels and 7.2V 2AH Li-poly battery pack. The robot base is shown in Figure 3.9a. To drive the robot motors, the Motor Control Board from Surveyor robot was also integrated with the robot base. It is shown in Figure 3.9b.

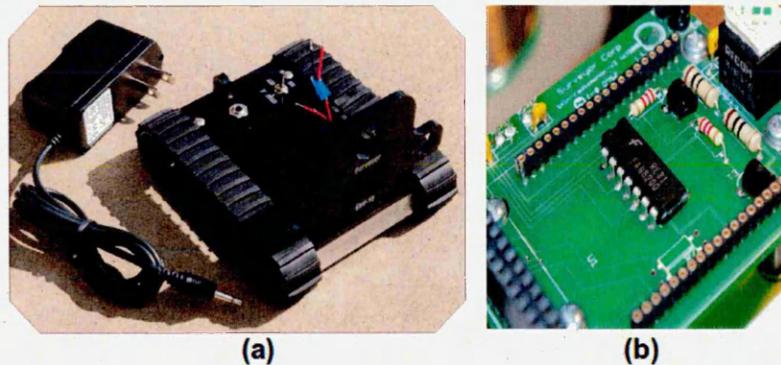


Figure 3.9: (a) Robot base for the organism [109]. (b) Motor control board [109].

3.2.7 Onboard Communication

To establish a wireless communication between the development platform and the robot organism, Lantronix WiFi port from the Surveyor robot was also integrated with the organism. For this purpose, the UART0 signals were routed from the Blackfin processor to the Lantronix wireless board through the Experimental Board “EXT-BF5xx-EXP”. The Lantronix port provides two wireless ports which can be used to connect the two UART ports (i.e. UART0 and UART1) available on the Blackfin processor. In the designed robot organism, only one UART (i.e. UART0) was used with the wireless board. With the Lantronix board on the robot, the wireless baud-rate up-to 2.5Mbps can be achieved. The Lantronix WiFi board is shown in Figure 3.10a and it can be very easily configured through the web browser to work in wireless Adhoc or the Infrastructure mode.

Similarly, to establish a communication backbone in the robot organism, through which the vision information can be distributed between the different processing units, a high speed Ethernet communication medium was established. The Ethernet cable (shown in Figure 3.10b) was used to connect multiple on-board Blackfin processors. The connection to each Blackfin processor was established through the Ethernet port available on the Evaluation Board EVAL-BF5xx.

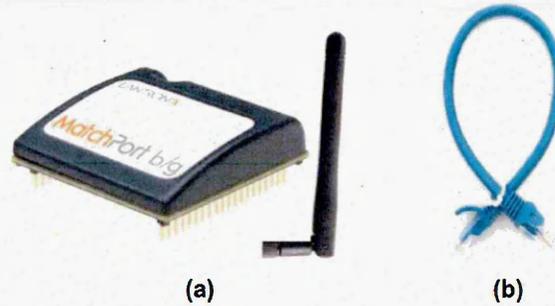


Figure 3.10: (a) Lantronix matchport 802.11b/g WiFi [110]. (b) Ethernet communication cable.

3.2.8 Navigation board

A navigation board, shown in Figure 3.11, was also integrated with the robot organism. The sensors available on the Navigation board were 3 axis accelerometers, 3 axis magnetometer and an on-board GPS unit. It was added to the organism so that some navigational information may be provided to the robot.

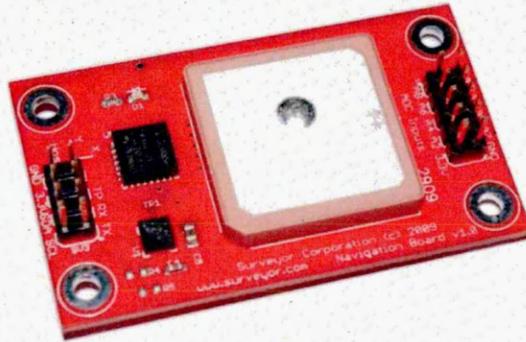


Figure 3.11: Navigation board M3 [111].

Finally, it was decided to make a complete robot organism with multiple on-board distributed processing systems. In the robot organism, the processor connecting to the vision sensor simulates the processing resources of the master robot and the rest of the processors acts as a slave processing units. In the organism mode scenarios, the master processing unit is made responsible for performing the organism locomotion task. At the same time, it also execute light weight vision processing algorithms to pre-process the raw image data and distribute it for further detail processing to the other slave processing units. The complete robot organism hardware would include the following elements:

- 1 robot base with four motors.

- 1 motor control board.
- 4 Blackfin processing units
- 4 evaluation boards (EVAL-BF5xx) to integrate the Blackfin processors.
- 1 extender Board EXT-BF5xx-Camera to attach Omni-vision camera sensor with the master robot.
- 1 Blackfin experimental board with the master robot
- 1 CMOS camera sensor
- 1 wireless board
- 1 navigation board

The developed robot organisms' front and side view is shown in Figures 3.12a and b, respectively.

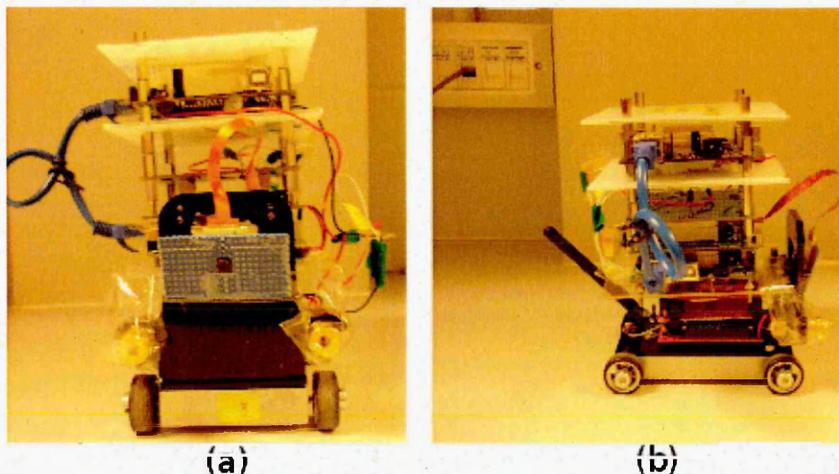


Figure 3.12: Robot organism.

3.3 Selection of Operating System (Robot Firmware)

The selection of an Operating System (OS) plays an important role in the application development for any embedded system. For Blackfin processor, selection of uClinux (micro-controller Linux) was made because of its rapid development and popularity among many embedded software developers. uClinux is a customised version or derived from the Linux 2.0 kernel. It was basically derived to target micro-controllers

or embedded systems without Memory Management Units (MMUs). It requires GNU cross compiler toolchains and C-libraries on the development platform (host system), to compile applications for Blackfin processor. Some of the important features of this OS are the following.

- Open Source
- Provide support for a large number of applications which are in mature stage.
- Provide support for easy system configuration.
- Many developed open source applications can be cross compiled.
- Facilitates the development of applications with real time constraints.
- Provides basic frame-work, customised for Blackfin processors.
- Allows the use of basic Linux commands
- Hardware driver support.

Using the uClinux operating system, the drivers for different on-board sensors were developed. In the distributed vision processing research, the most important sensor which needed the uClinux drivers, was the on-board CMOS camera sensor. As already mentioned, for the swarm mode scenarios, a third party SRV1 robot was used and for the organism mode scenarios a customised robot organism was developed. The CMOS sensor on the SRV1 robot (Omni-vision OV7725) was different from the one used on the robot organism (OV7670), so uClinux camera drivers for both of them were developed. To develop the camera drivers in uClinux, the software was required to access the CMOS sensor registers. For this purpose, I2C interface was setup between the CMOS sensor and the Blackfin processor [119]. The I2C interface uses two bidirectional lines, that is, serial data and serial clock lines. The serial clock line was used to synchronise the peripheral (i.e., CMOS sensor) with the Blackfin processor. And using serial data line, the appropriate registers on the CMOS sensor were selected and required data was written on it. Once all the sensor registers are programmed then the sensor is ready to provide image data to the camera driver, when required.

For robot locomotion, the drivers for driving the robot motors were also developed. For this purpose, the pulse width modulation (PWM) signals were generated from the Blackfin processor using the on-board timers. These PWM signals were

used to generate the frequencies which controls the wheel speed through the servos. To control the direction of wheels rotation, the digital I/O provided by the Blackfin processor were also used. As Navigation board was also integrated in the organism, so the drivers for the different components available on the navigation board (e.g. Compass and Three axis-accelerometer) were developed in uClinux environment.

3.4 Conclusions

In this Chapter the basic hardware components and on-board firmware required to execute the developed distributed vision processing algorithms on the robots, have been described in detail. To perform distributed vision processing in the swarm mode scenarios, off-the-shelf Surveyor SRV1 robots have been selected. These robots will be used in the two swarm mode scenarios presented in Chapter 6. Whereas, for organism mode scenarios, a robotic organism comprising of multiple processing modules is developed. This multi-processor robotic organism is utilised in the two organism mode scenarios presented in Chapters 7 and 8. For the robot on-board firmware, it has been concluded that, an operating system should be selected which facilitates rapid software development and also helps in integrating open source computer vision libraries, such as, OpenCV and OpenSURF. For this purpose, the selection of open source uClinux operating system has been concluded to be the best choice.

Chapter 4

Embedded Vision Processing

In swarm robotic systems, the robots have limited memory and processing resources. In such systems, the use of an on-board vision sensor produces a huge amount of visual information. This information requires intensive processing to become meaningful for the robot control. So, to start working with the distributed vision processing in multi-robotics systems, it is necessary to develop basic vision algorithms for processing the visual information. These algorithms are required to be customised for the underlying embedded vision system based on the Blackfin processor. In other words, it is required to develop vision algorithms which could run in the embedded system environment and at the same time, they are also customised for the Blackfin processor architecture. For this purpose, a small library of vision algorithms is created. These vision algorithms provided the basis for all the scenarios which addressed the vision processing in distributed robotic systems. This Chapter details the developed vision processing algorithms and vision based basic functionalities which the underlying swarm system requires in all the devised scenarios. One basic functionality identified is obstacle avoidance which is essential in a sense that swarm of robots are required to detect the presence of obstacles in the environment and take decision accordingly. The second functionality which facilitates the swarm of robots to become an organism is the ability to detect the docking ports of the robots and also the energy points in the environment to perform energy foraging. Apart from describing these basic functionalities, this chapter also discusses the experimental results which are intended to demonstrate the real time performance of the developed vision algorithms. This chapter is divided into the following sections.

- Robot Camera Calibration.

- Embedded Vision Algorithms.
- Vision Based Obstacle Avoidance.
- Vision Based Robot Docking Support.

4.1 Robot Camera Calibration

In Figure 4.1, an image captured by the robot vision system is shown. It can be clearly seen that the image is too distorted near the boundaries. So, before applying the vision processing algorithms on the images captured by the robot vision system, the enhancement of these images was required. This image enhancement may not be necessary for performing the basic tasks such as obstacle avoidance, but it is essential for achieving reliable performance from the object and pattern recognition algorithms. For performing this image enhancement, the camera calibration of the robot camera sensor is needed.

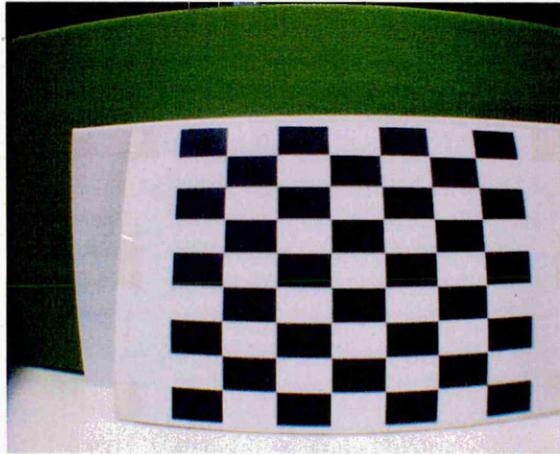


Figure 4.1: Image captured by the robot vision system.

The objective of camera calibration is to find the camera parameters which defines the relation between the 3D world coordinate points, and the 2D points on the camera image. These parameters are represented in the form of a 3x4 matrix which is called camera matrix in computer vision terminology. The relation between 3D world points C_w and 2D image points C_i in terms of camera matrix K is shown as.

$$C_i = KC_w \quad (4.1)$$

This equation when expanded, can be written as

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = AB \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \quad (4.2)$$

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = A[RT] \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \quad (4.3)$$

where, u and v are the 2D camera coordinates and s is the scale factor. The 3D world coordinates are represented as x_w, y_w, z_w . The camera matrix K is represented as $A[RT]$ where A is the camera intrinsic matrix, R is the rotation and T is the translation matrix of camera with respect to the world coordinate points (World coordinate system is the reference coordinate system. It can be used as the starting position of the robot in the test arena, or it can be fixed to the origin of the camera position). R and T together represent camera extrinsic parameters B . The camera intrinsic A and extrinsic B matrix can be expanded as

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.4)$$

$$B = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

where, α_x and α_y represent the focal length of camera in terms of pixels. γ is the skew coefficient between x and y axes of image. And u_0 and v_0 are the principal points of the camera. Similarly, $r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}$ define the parameters of rotation matrix and t_x, t_y, t_z define the parameters of translation matrix. For camera calibration purpose, a calibration grid pattern was used together with MATLAB camera calibration toolbox [115]. Thirteen images of calibration pattern, captured by the robot vision sensor, were used as input to MATLAB camera calibration toolbox. These images are shown in Figures 4.2 to 4.8. It can be noticed that the images were taken while keeping the grid pattern with different orientation and tilt

in front of the robot's camera. In some of the images, the grid pattern was kept very near to the vision sensor. This was done so that the distortion, which is high near the image boundaries, can be removed from the image.

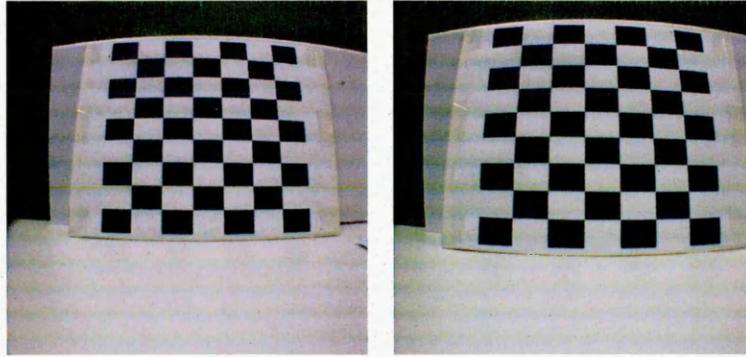


Figure 4.2: Robot images used for calibration: image 1 (left) and image 2 (right)

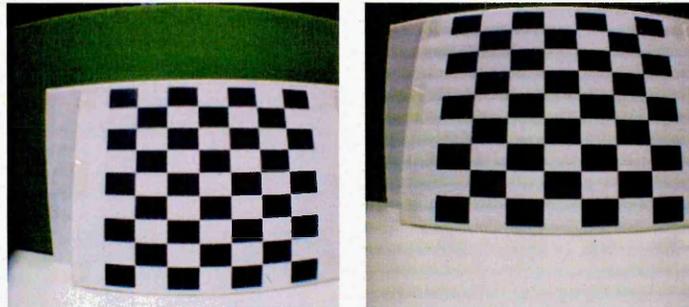


Figure 4.3: Robot Images used for calibration: image 3 (left) and image 4 (right)

The grid points on all the input camera images (shown in Figures 4.2 to 4.8) were selected very carefully using the calibration toolbox. It was necessary to select the grid points accurately as a little error caused poor calibration results. It is a very important feature of the MATLAB Camera Calibration Toolbox that if some grid points on the images are not selected properly, then the error caused by them can be reduced later. For reducing this error, the toolbox enables to consider the specific images again and re-select the grid points. The camera intrinsic parameters generated by the toolbox after complete camera calibration are given as.

$$A = \begin{bmatrix} 1135.98925 & 0 & 647.73695 \\ 0 & 1168.25320 & 532.58715 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.6)$$

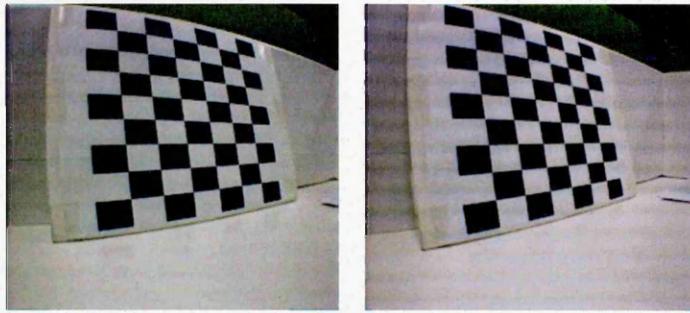


Figure 4.4: Robot images used for calibration: image 5 (left) and image 6 (right)

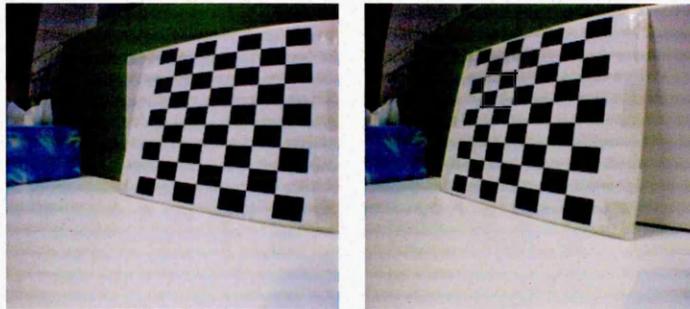


Figure 4.5: Robot images used for calibration: image 7 (left) and image 8 (right)

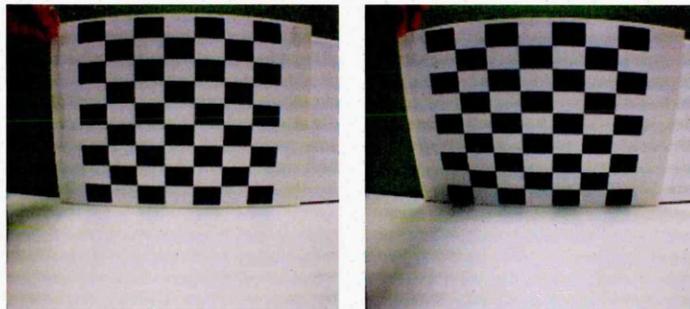


Figure 4.6: Robot images used for calibration: image 9 (left) and image 10 (right)

In Equation 4.6, all measurements are defined in milli-meters. Using these extracted camera parameters, the 2D grid points were re-projected on all the input images. These re-projected 2D points on the images are shown on the left column of Figures 4.9 to 4.13. The enhanced images, after removing the distortion, are also shown on the right column of Figures 4.9 to 4.13. It can be seen on all the enhanced images that the distortion was removed properly. The removal of distortion is very clear from the grid lines. These lines were not appearing straight in the input

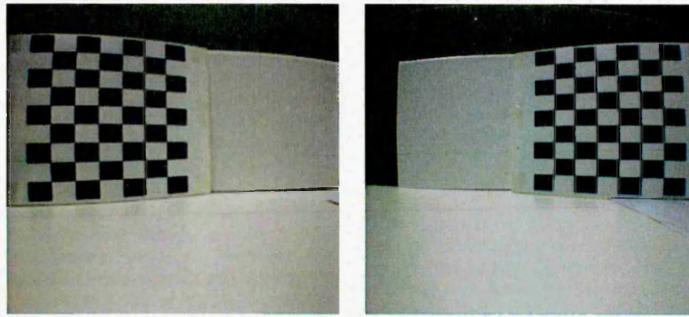


Figure 4.7: Robot images used for calibration: image 11 (left) and image 12 (right)

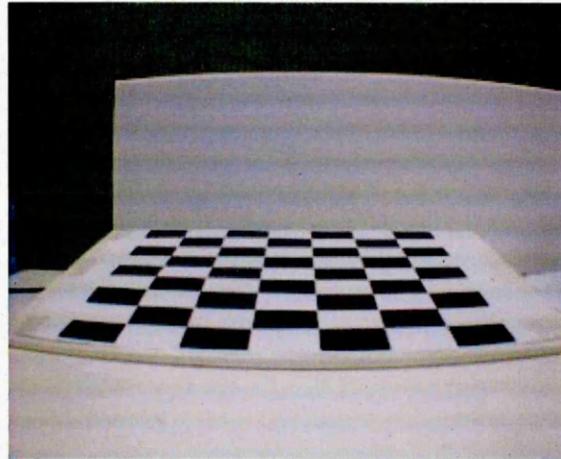


Figure 4.8: Robot images used for calibration: image 13

images, but after enhancement, the grid lines appeared considerably straight even near the image boundaries. The computed intrinsic parameters were used on-board to enhance the images, captured by the robot vision system, before passing them on to the vision algorithms for detailed processing. In most of the vision systems, the effect of camera distortion can be controlled up-to some extent when the camera driver initialises the camera modules and programs the hardware registers of the camera. But for the CMOS camera used on-board in the robot organism, the camera drivers were developed in this research for uClinux operating system and no register settings were found which could help avoiding this calibration. So, the camera calibration step was needed after the image capturing phase.

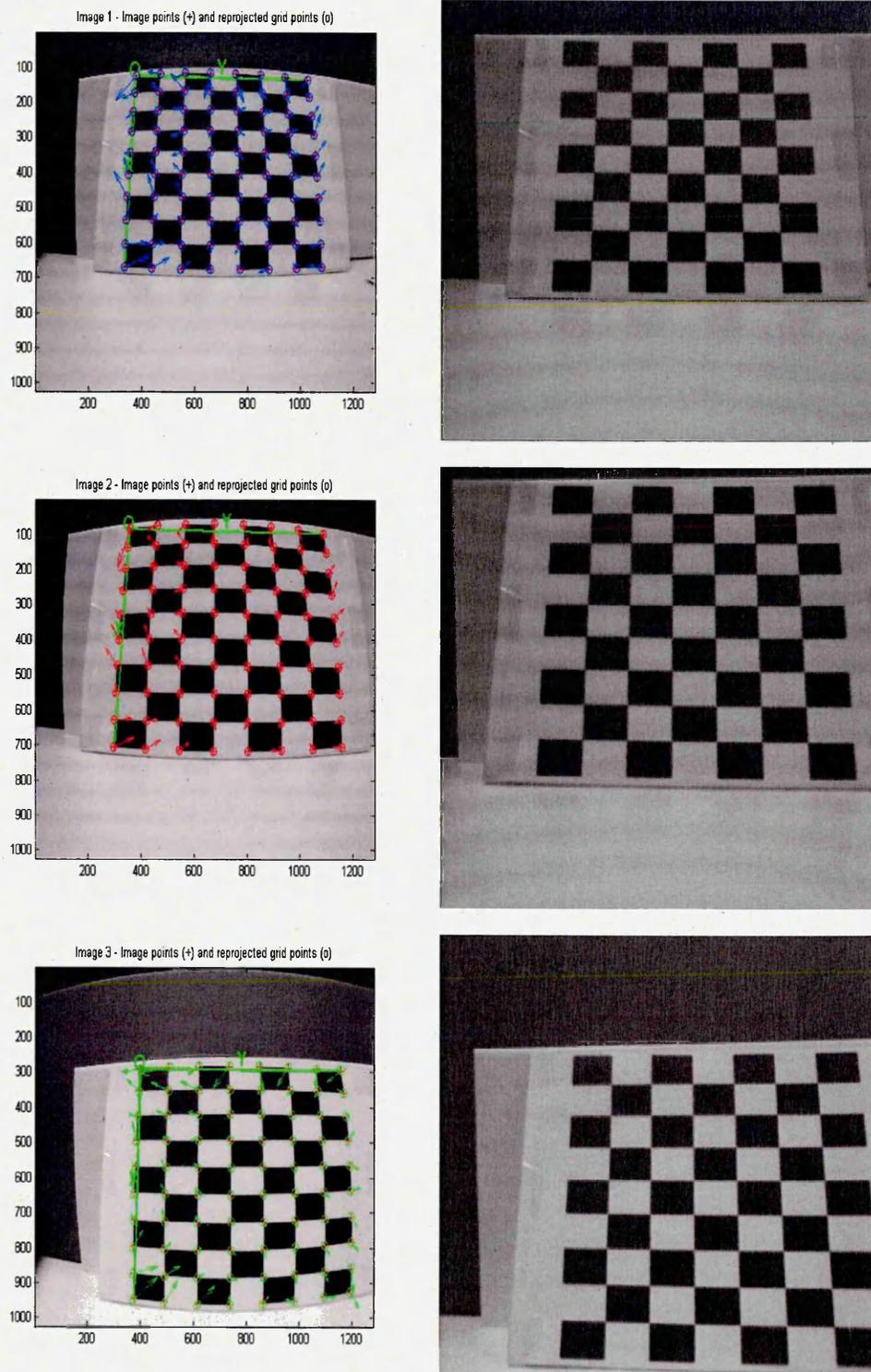


Figure 4.9: Calibration results: Re-projections on image (left) and enhanced image (right)

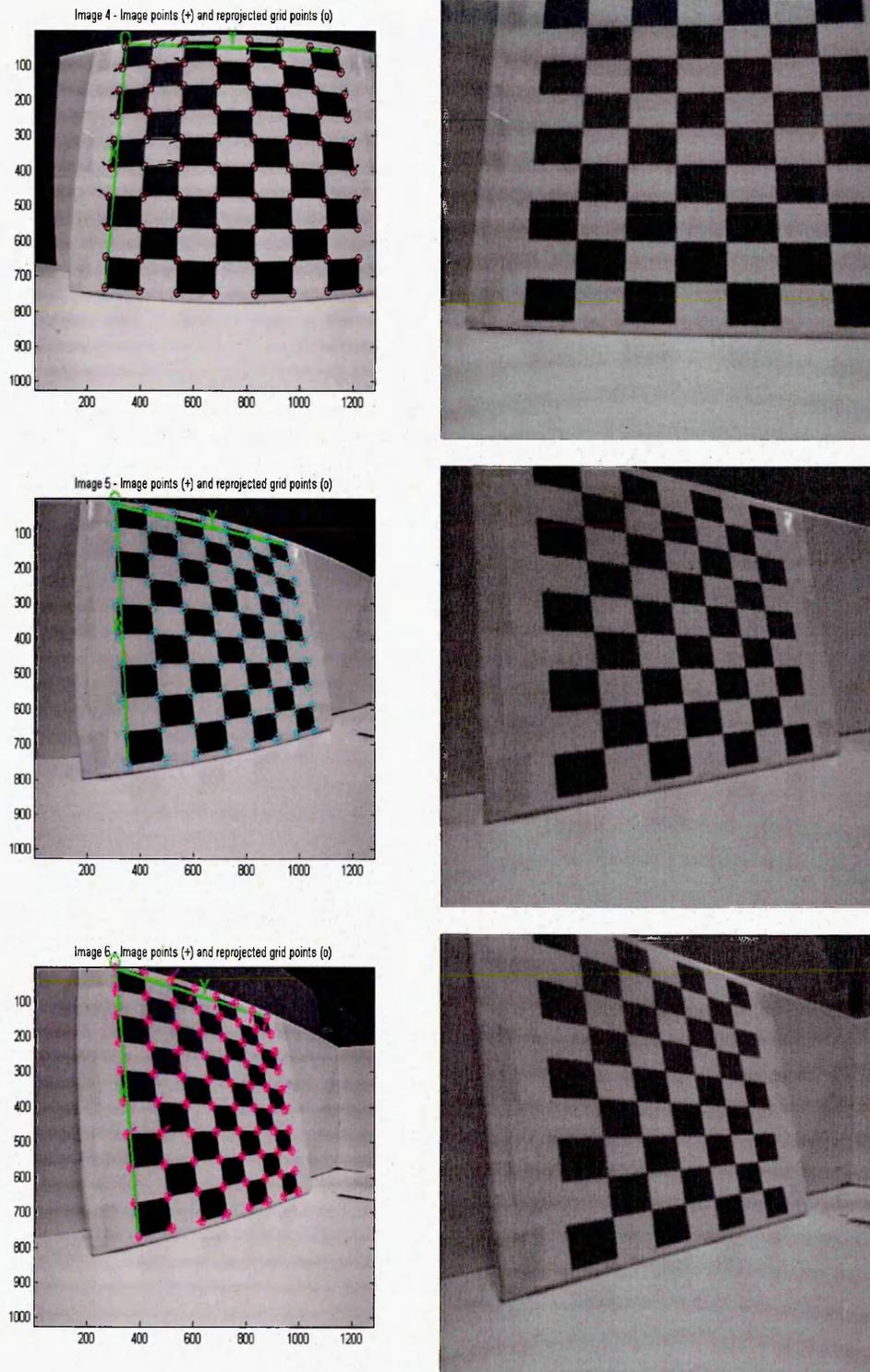


Figure 4.10: Calibration results: Re-projections on image (left) and enhanced image (right)

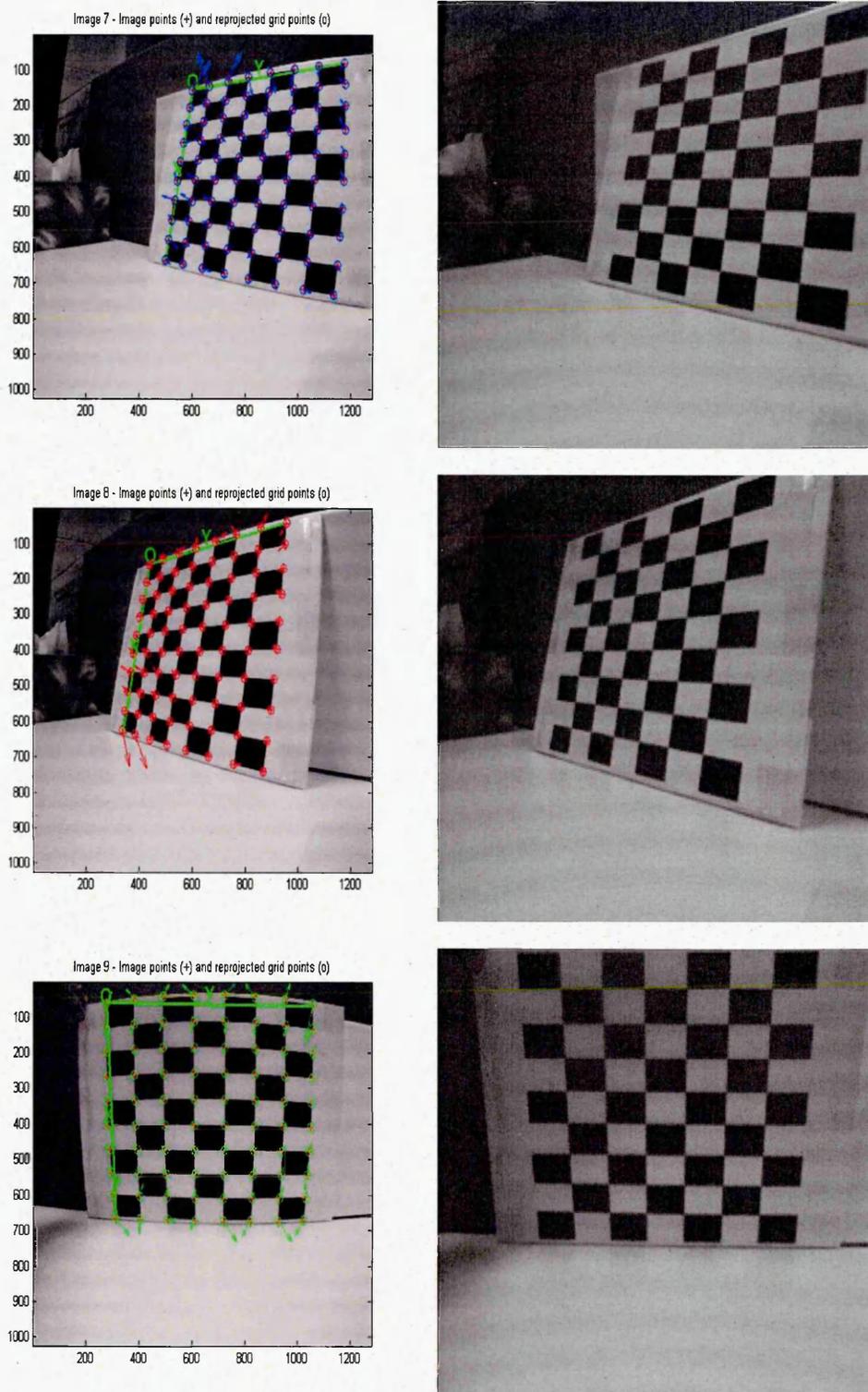


Figure 4.11: Calibration results: Re-projections on image (left) and enhanced image (right)

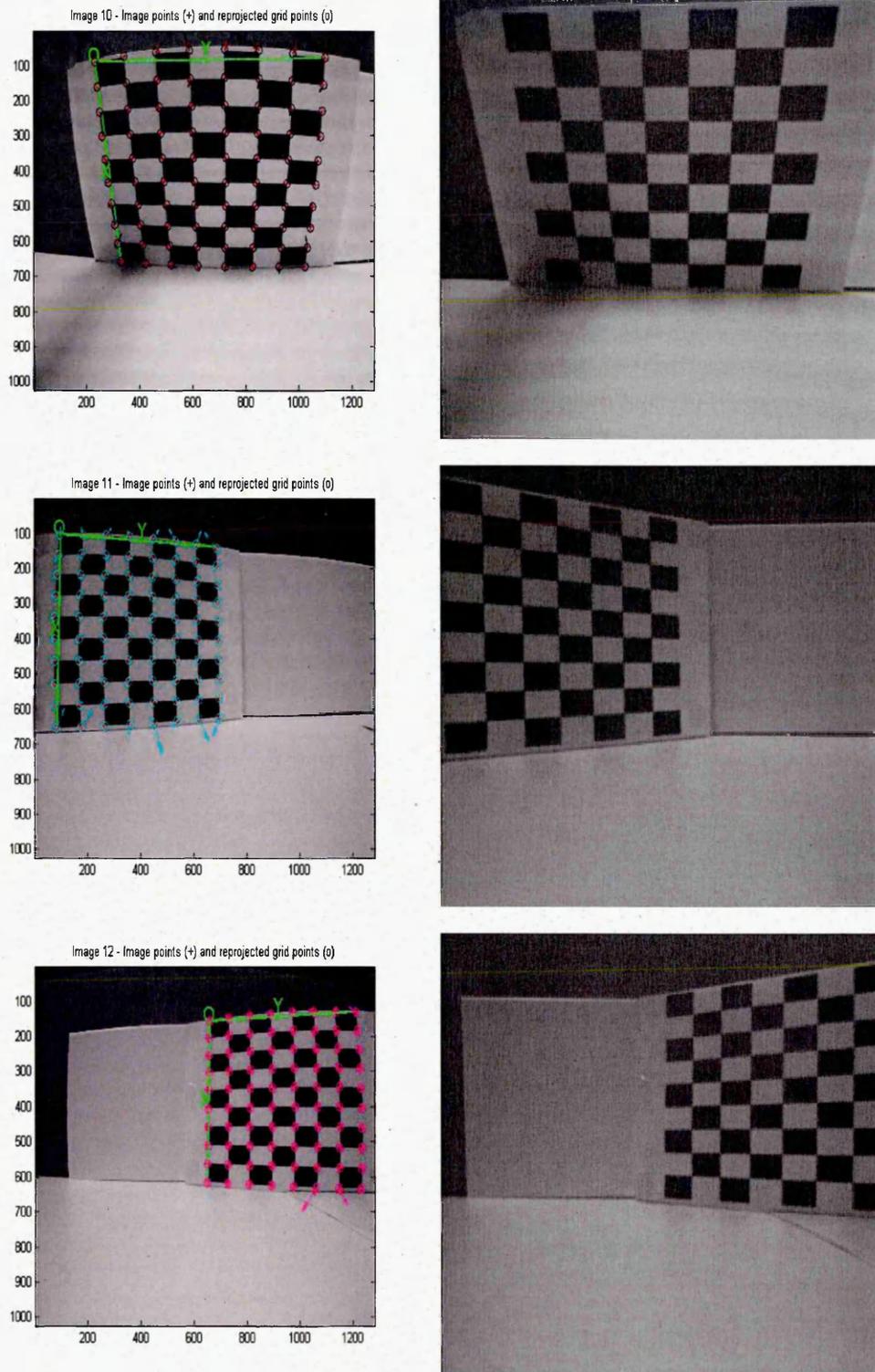


Figure 4.12: Calibration results: Re-projections on image (left) and enhanced image (right)

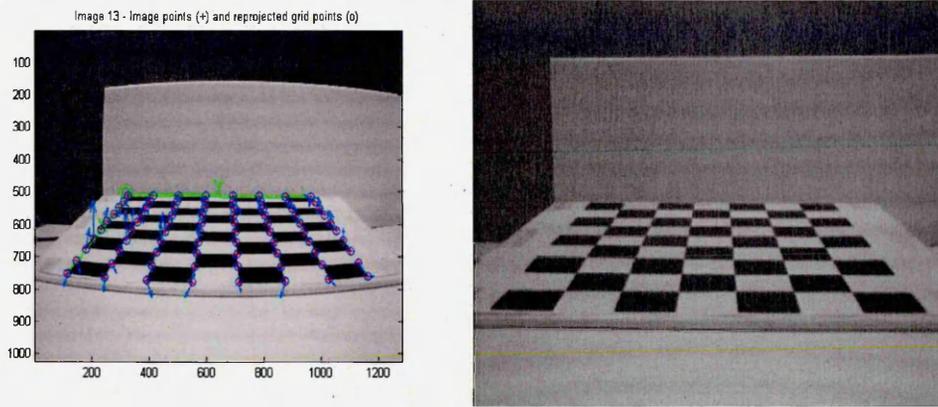


Figure 4.13: Calibration results: Re-projections on image (left) and enhanced image (right)

4.2 Embedded Vision Algorithms

A library of embedded vision algorithms was developed which provided the basic vision processing capabilities in all the distributed vision scenarios. These algorithms were developed keeping in view that the Blackfin processor lacks Floating Point Unit (FPU). So it was preferred to avoid floating point operations and rely only on decimal operations. An effort was made to customise the vision algorithms specifically for the Blackfin architecture. For this purpose, fixed point operations were used in most parts of the vision algorithms. This was done because the Blackfin processor architecture is a fixed point architecture and it provides hardware support for an efficient implementation of fixed point operations. This hardware specific customisation was necessary to achieve real time performance of the vision algorithms. The vision algorithms included in the library are explained in the following sections.

4.2.1 YUV to Colour Image Conversion

By default, the image captured by Blackfin BF537E processor using Omni-vision OV7670 CMOS camera module is in YUV (where Y gives intensity, U and V provide the chrominance information) format. To perform a colour image processing, YUV to colour image conversion is required. For every pixel, YUV values are manipulated to extract Red, Green and Blue colour information. An example colour image obtained after performing YUV to colour image conversion is shown in Figure 4.14. The selection of this picture is made so that the range of different colours can be observed which guarantees proper implementation of algorithm.



Figure 4.14: Image obtained after YUV to colour image conversion.

4.2.2 Colour to Grey Scale Conversion

To perform image processing on grey scale images, colour to grey scale image conversion was required. For every pixel, Red, Green and Blue colour values are weighted by 30%, 59% and 11%, respectively, to obtain the corresponding grey scale values. To achieve this, the following simple equation was implemented.

$$G = 0.3 \times R + 0.59 \times G + 0.11 \times B \quad (4.7)$$

where the Red, Green and Blue values of the pixels are represented by R , G , and B , respectively. And G is the Grey scale value generated.

An example colour to grey scale converted image is shown in Figure 4.15.



Figure 4.15: Image obtained after colour image (left) to greyscale image (right) conversion.

4.2.3 Grey Scale Gradient

To obtain the grey scale gradient, Sobel and Canny operator based gradient computation algorithm was implemented. Here, the process of obtaining gradient using Sobel operator is presented. To obtain gradient in x and y directions, following Sobel operators were used. The assumed x and y directions of the image are also shown in Figure 4.16.

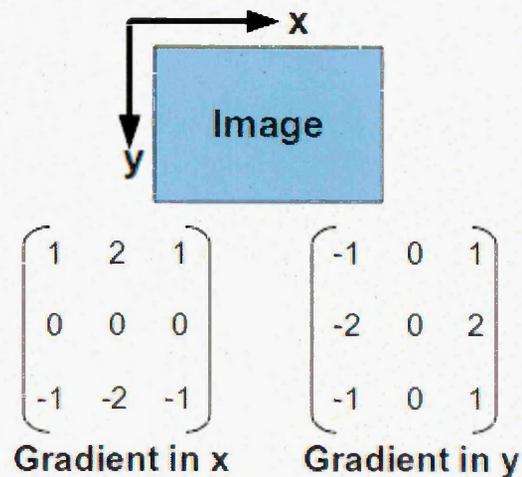


Figure 4.16: Image coordinates and Sobel operator for gradient in x and y direction.

An example grey scale gradient image obtained from the implemented algorithm is shown in Figure 4.17.

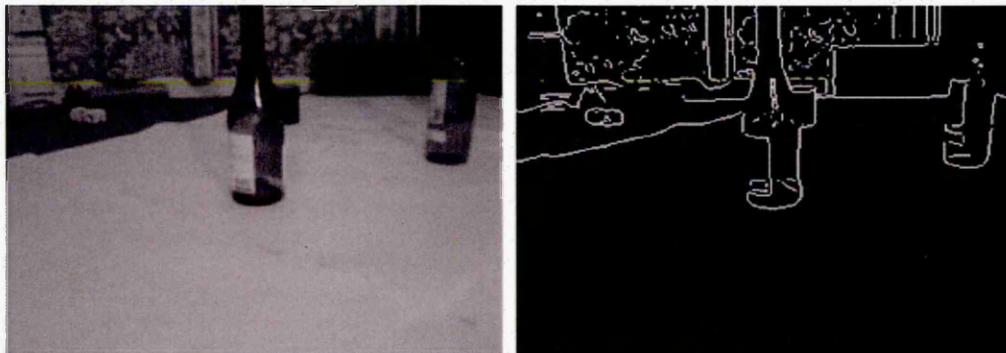


Figure 4.17: Greyscale image (left) and gradient image (right) obtained from BF537E processor.

4.2.4 Colour Gradient

To obtain the colour gradient [102], Sobel operators were applied in x and y direction on Red, Green and Blue channels separately. Let GxR , GyR be the gradient obtained when Sobel operator applied in x and y directions of Red channel, similarly GxG , GyG for Green channel and GxB , GyB for Blue channel. Then the following set of equations were implemented to obtain the final colour gradient magnitude.

$$Gxx = GxR^2 + GxG^2 + GxB^2 \quad (4.8)$$

$$Gyy = GyR^2 + GyG^2 + GyB^2 \quad (4.9)$$

$$Gxy = GxR \times GyR + GxG \times GyG + GxB \times GyB \quad (4.10)$$

Orientation and Magnitude information of the gradient are obtained using the following equations.

$$\theta = \frac{1}{2} \times \arctan \frac{(2 \times Gxy)}{(Gxx - Gyy)} \quad (4.11)$$

$$Magnitude = \sqrt{\frac{1}{2} \times [(Gxx + Gyy) + ((Gxx - Gyy) \times \cos 2\theta) + (2 \times Gxy \times \sin 2\theta)]} \quad (4.12)$$

An example colour gradient image obtained using the implemented algorithm is shown in Figure 4.18. The reason for using this image is due to the presence of range of different colours.

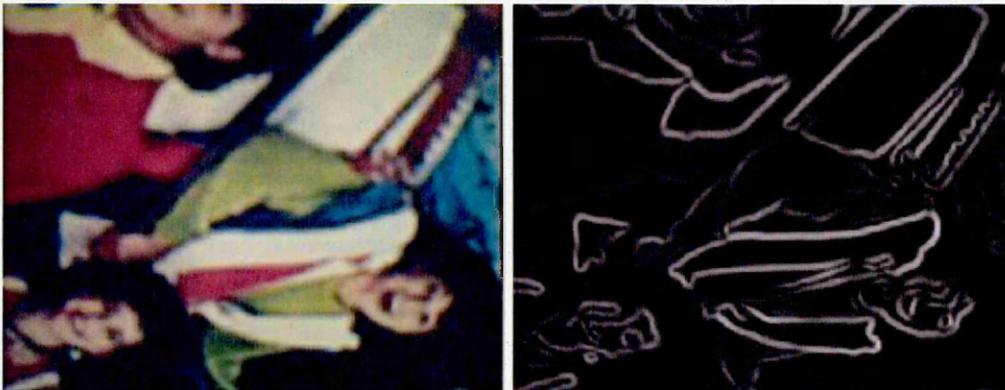


Figure 4.18: Colour image (left) and gradient image (right) obtained from BF537E processor.

4.2.5 Grey Scale and Colour Image Segmentation

A Region Growing based Image segmentation algorithm was implemented. The basic idea of region growing algorithm is that, if intensity difference of the current pixel in the image from the neighbouring pixels lies within some predefined threshold, then the current pixel and its neighbouring pixels are merged into one region. A8 adjacency criteria is implemented so that the region can grow in all possible directions from the current pixel. This criterion states that while processing every pixel, its eight neighbouring pixels are also checked whether they fulfil the criterion or not. It means the region can grow in eight different directions. The concept of region growing and A8 adjacency is also shown in Figure 4.19. While segmenting grey-scale images, only one grey-scale intensity channel is processed, whereas for processing colour images, the Red, Green and Blue colour channels are processed in parallel. In case of colour image segmentation, for a pixel to become a part of a region, the difference of its Red, Green and Blue colour values from the mean Red, Green and Blue values of the region should all be within some predefined threshold. An example colour segmented image obtained from BF537E processor is shown in Figure 4.20.

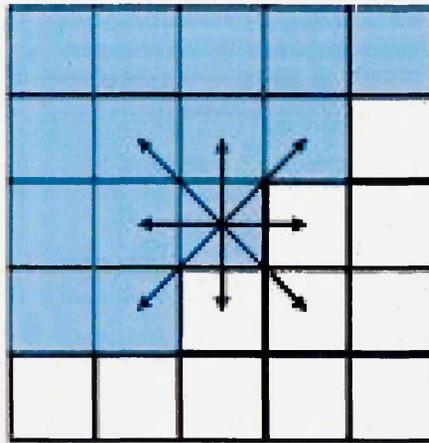


Figure 4.19: A8 adjacency. Directions in which region can grow is shown by arrows.

4.2.6 Colour Blob Detection

A colour blob detection algorithm was implemented to detect different colour blobs in the input images. The algorithm was configurable to detect different colour blobs by adjusting its input parameters. This algorithm directly processed the YUV im-



Figure 4.20: Colour image (left) and segmented image (right) obtained from BF537E processor.

ages. The reason for not using RGB format for input image is that, the RGB format did not separate the luminance information (i.e. the brightness) from the chrominance information (i.e. the colour itself). For example, in RGB format of the image, it is not possible to determine whether the colour is red or not by simply applying a threshold on R, G and B colour values. In YUV format, Y provides luminance, whereas U and V provide the chrominance information. In the current implementation, the chrominance information is utilised to detect the different colour blobs. The UV coordinate plane providing the chrominance information is shown in Figure 4.21.

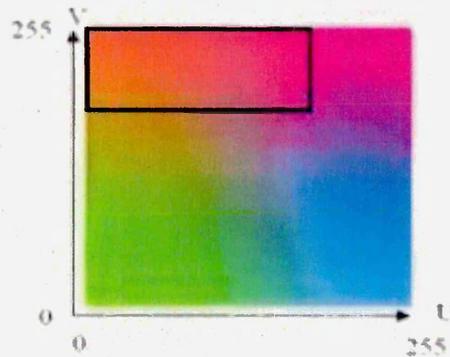


Figure 4.21: UV plane providing the chrominance information.

Now from Figure 4.21, it seems very easy to apply threshold on UV values for the selection of colour blobs. For example, for the detection of Red colour blobs, V values greater than 190 and U less than 200 are selected. This range of UV values is identified by the black boundary line in Figure 4.21. It can be noticed that, the range of U values greater than 130 seems not required. But this range is used to make the algorithm more insensitive to the changes in lighting condition. An

example image showing the detection of red colour blob in input image is shown in Figure 4.22. The reason for using this image is that, it contains the range of different colours and detection of red colour in this image guarantees proper implementation of colour blob detection algorithm.

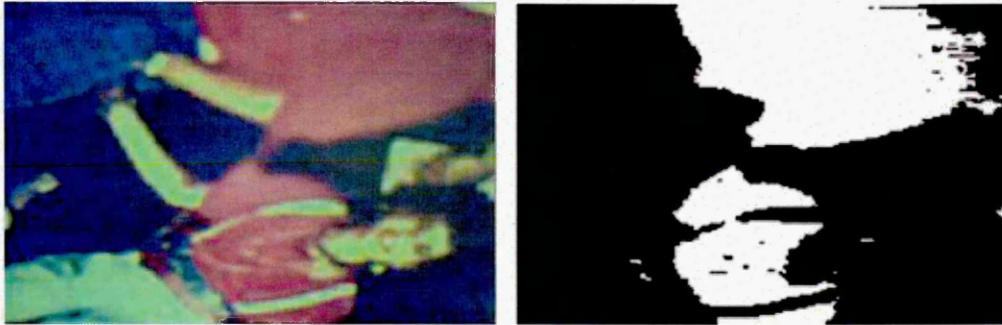


Figure 4.22: Colour input image (left) and processed image (right) showing detection of red colour blobs.

4.2.7 Image Erosion

An image erosion algorithm was implemented for grey scale images. The size of window used for erosion is made configurable. An example image showing proper implementation of erosion algorithm is shown in Figure 4.23. The window size used to obtain this output image is 5x5 pixels.



Figure 4.23: Input image (left) and processed image (right) obtained after erosion.

4.2.8 Image Dilation

Similar to image erosion, an algorithm to dilate grey-scale images is also implemented. The size of window used for dilation is made configurable. An example

image showing proper implementation of dilation algorithm is shown in Figure 4.24. The window size used to obtain this output image is 5x5 pixels.

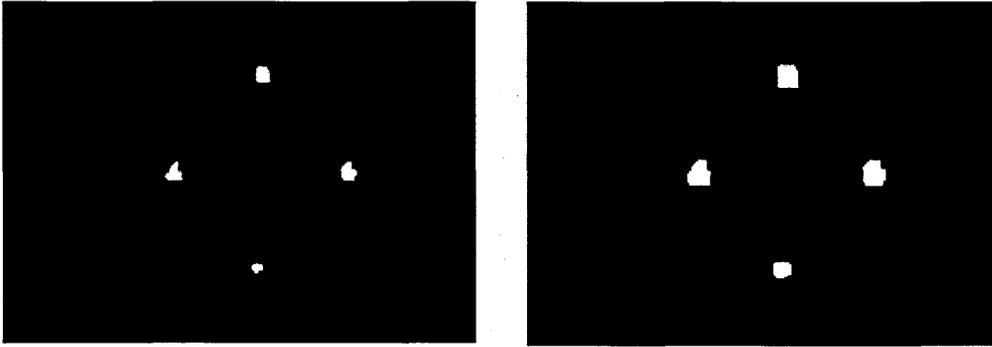


Figure 4.24: Input image (left) and processed image (right) obtained after dilation.

4.3 Vision Based Obstacle Avoidance

In this section the research undertaken for the development of the vision based obstacle avoidance strategies, is presented. Obstacle avoidance is one of the important aspects of mobile robots. Without it, robotic movements would be very restrictive. This functionality is also needed in every distributed vision processing scenario as the robots will be performing operation in the environment, where obstacles are also present. There are many other sensors which can be used for achieving obstacle avoidance tasks such as, infrared and laser range finders. But in the current scenarios, as the vision information is needed any way for the distributed vision processing tasks and knowing that it provides rich surrounding awareness which could be efficiently used for different purposes. So based on this fact, it is also utilised for performing the obstacle avoidance task. There are many ways to accomplish obstacle avoidance, but efforts were made to devise an obstacle avoidance algorithm with smallest computational complexity. For example, algorithms with more floating point operations have more computational complexity and execution time. And knowing that *Analog Devices* BF537E processor is a fixed point processor so floating point operations are not recommended as it slows down the performance. Therefore, algorithms with less floating point operations are preferred.

4.3.1 Approaches To Obstacle Avoidance

Three obstacle avoidance approaches were implemented and tested on the target robot. These were segmentation based, optical flow based and path finder based obstacle avoidance approach.

Segmentation based Obstacle Avoidance

A very simple segmentation based obstacle avoidance algorithm was implemented. To explain the basic concept, the image shown in Figure 4.25a is considered. This image shows couple of obstacles placed on a test arena. Some assumptions were made in this obstacle avoidance algorithm. For example, it was assumed that, the robot is placed on a flat ground and the camera was placed relatively straight or slightly tilted down. Now by looking at this image, it could be noticed that the ground surface of the test arena had more or less the same colour. After segmentation algorithm was applied, the resultant segmented image obtained is shown in Figure 4.25b.

The next step was to determine which region in the segmented image was the floor. One way to isolate the floor was to assume that the biggest region is the floor. But this assumption was not true when the robot was in front of a very big obstacle as in that case, the region representing obstacle was the biggest. In the current implementation, the speed of the robot was set to guarantee that, the robot was not very close to the obstacle. With this implementation, the region covering the middle bottom of the image was considered the ground region as this part of the image was the one closest to robot camera. Sometime while turning, there was a possibility that the obstacle was very close to robot in the robot field of view. In this scenario, the robot could collide with the obstacle, assuming the obstacle region as the floor. To overcome this problem, the robot kept track of the intensity of the ground region in the last frames. If there was a sudden change in the ground region intensity, then the robot determines that it was very close to the obstacle. The robot moved back a small distance to get ground floor in its field of view. With these additional checks, the assumption that the floor region was the one covering the middle bottom of the image works well. Following this assumption, the isolated ground region from the rest of the obstacles is shown in Figure 4.25c.

Now to determine the ground map visible to the robot, the bottom of the image was considered initially as shown in Figure 4.25c and filled (filling was performed with White pixels) vertically pixel by pixel until the obstacle boundary encounter

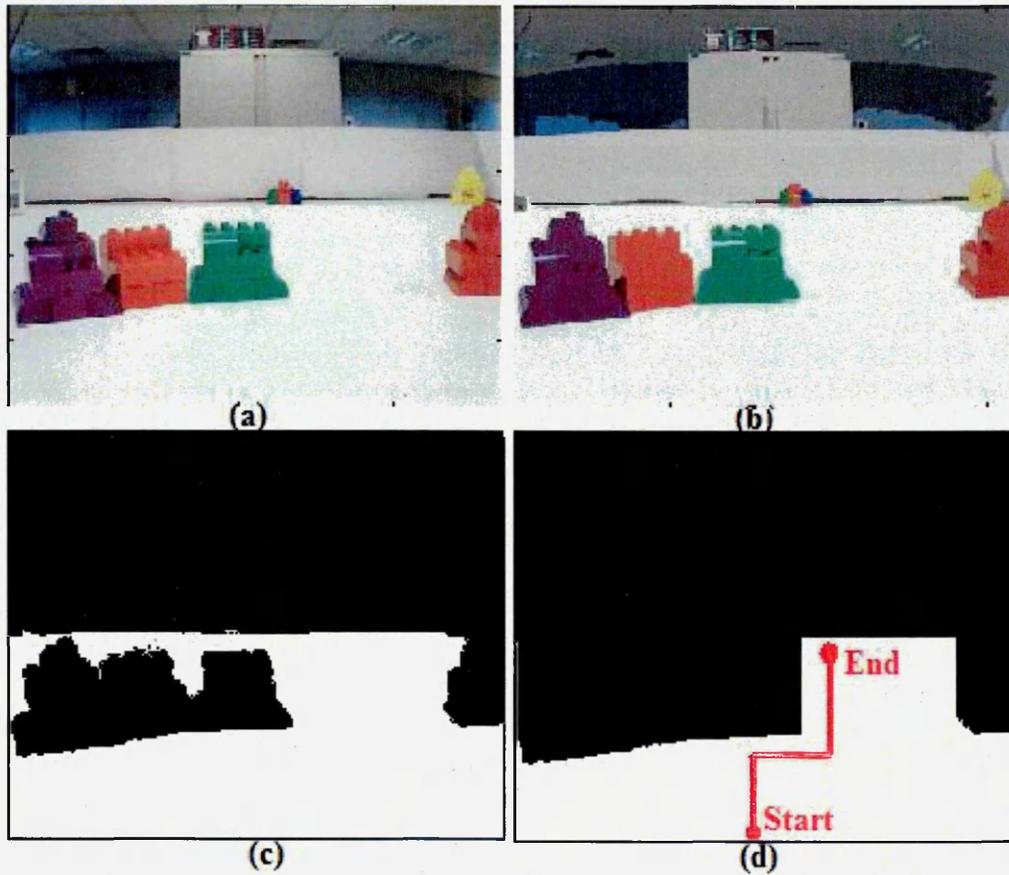


Figure 4.25: (a) Colour input image. (b) Processed image obtained after applying segmentation. (c) Initial ground map isolated from obstacle's. (d) Ground map visible to the robot vision system.

(i.e. the Black pixel detected). Upon detecting the obstacle boundary, the rest of the pixels to the top were filled with Black pixels. The same was performed with all columns. The resultant visible ground map is shown in Figure 4.25d. In this image, the final visible ground map is also refined by the image dilation algorithm. Now in this final ground map, the white region is the area where robot can move freely without colliding with the obstacle. In the current implementation, the resolution of the images captured by the robot is 160x120 pixels (initially, the experiments were performed with the resolution of 320x240 pixels, but to reduce the computational time, 160x120 resolution was selected). If from centre bottom of the final ground map, robot detected continuous white pixels greater than 30 in vertical direction, then it went straight ahead. It also checked whether any obstacle from left and right side of the robot were not very near and there was enough space to go forward. If the distance in vertical direction was less than 30 pixels, then the algorithm checked

from which part of the image (either left or right), the robot was closest to the obstacle. The robot took a turn in the direction opposite to the one from where it was closest to obstacle (i.e. If on the left side, white pixels in vertical direction was less than right side, then robot took a right turn). Using this segmentation based approach, the sequence of processes followed by the robot to avoid obstacles were the following.

- Capture image
- Perform image conversion to grey scale
- Perform grey scale image segmentation. In segmentation results, there is possibility that obstacles shadows on the ground are also identified as separate regions rather than a common ground region. The part of image representing shadow image has intensity values slightly different from the ground region. To overcome this problem, a large threshold value was used for segmentation process. This value also resulted into lower number of segmented regions in the resultant segmented image.
- Remove small segments from the segmented image. This was used to further reduce the number of segments to process. Moreover, if some dark spots were present on the ground plane, then they could be identified as a different segment and could act as an obstacle. This could result into a ground map which was different from the expected one. So to overcome this problem, a routine to remove small segments (segments comprising of pixels less than some predefined number) was implemented.
- Determine the ground map visible to the robot.
- Perform dilation on the visible ground map to refine the results.
- Determine the distance to obstacle in pixels. If distance was greater than predefined number then go forward, otherwise turn the robot in the direction opposite to the one where the obstacle is closest to the robot.

Optical Flow based Obstacle Avoidance

An optical flow based control algorithm for obstacle avoidance was also implemented. There are many methods to determine optical flow from a sequence of images. For

example, Kevin S.Pratt [42] used Lucas-Canade optical flow based obstacle avoidance algorithm for MAVs in urban environment. And Kahlouch [43] used Horn and Schunck optical flow based algorithm for avoiding obstacle in their autonomous ground robot. The selection of which optical flow algorithm to use is subjective to the application, whether the resultant motion estimation information obtained from optical flow is intended to be used for robot navigation, obstacle avoidance, object tracking or some other application. One of the most important requirements is the execution speed. If the algorithm to obtain optical flow information is highly accurate, but cannot provide the output in necessary response time, then it is useless. Similarly, the algorithm is also useless if it is very efficient but cannot provide the necessary accuracy. Hongche [95] provides a detailed analysis on “Accuracy vs. Efficiency Trade-offs in Optical Flow Algorithms”. For applications like navigation, high accuracy is utmost important as a little error in the rotation estimation can result into a big error in the robot estimated position and hence in navigation. For applications like obstacle avoidance, a compromise on accuracy can be made, but efficiency is important as if the algorithm fails to take decision within necessary response time, then the robot can collide with the obstacle. In the current implementation, Horn and Schunck optical flow [96] based obstacle avoidance algorithm was implemented. Horn and Schunck derived equations that relate the image brightness at a point to the motion of the brightness pattern. Let the image brightness at point (x, y) in the image plane at time t be denoted by $E(x, y, t)$. When the image pattern moves, the brightness of a particular point in the image pattern is constant. This led to the following equations

$$\frac{dE}{dt} = 0 \quad (4.13)$$

Using chain rule Equation 4.13 can be written as,

$$\frac{\partial E}{\partial x} \times \frac{dx}{dt} + \frac{\partial E}{\partial y} \times \frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \quad (4.14)$$

$$\frac{\partial E}{\partial x} \times u + \frac{\partial E}{\partial y} \times v + \frac{\partial E}{\partial t} = 0 \quad (4.15)$$

Where,

$$u = \frac{dx}{dt}, v = \frac{dy}{dt} \quad (4.16)$$

$$E_x u + E_y v + E_t = 0 \quad (4.17)$$

Equation 4.17 is a single linear equation with two unknowns (i.e. u and v). The equations used to obtain the three partial derivatives E_x , E_y and E_t are given below. The detailed derivation of these equations can be found in [96].

$$E_x \approx \frac{1}{4} [E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} + E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}] \quad (4.18)$$

$$E_y \approx \frac{1}{4} [E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k} + E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1}] \quad (4.19)$$

$$E_t \approx \frac{1}{4} [E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k} + E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}] \quad (4.20)$$

It can be noticed that, the estimates E_x , E_y and E_t are obtained by taking the average of the first four differences taken over the adjacent measurements. The relationship in space and time between these measurements is shown in Figure 4.26.

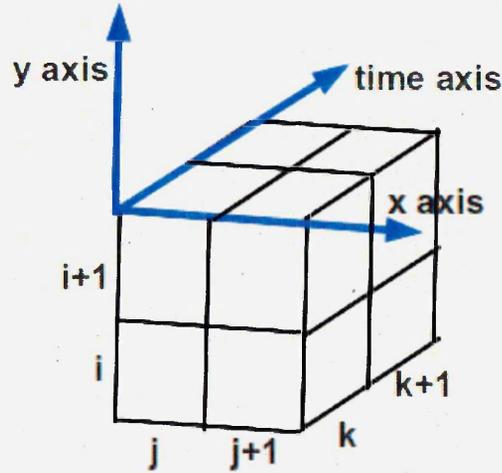


Figure 4.26: The estimates of three partial derivatives E_x , E_y and E_t of image brightness at the centre of the cube are each obtained from the average of first differences along four parallel edges of the cube. Here the index ' j ' corresponds to the x direction in the image, the index ' i ' to the y direction and index ' k ' lies in the time direction.

Similarly, to obtain optical flow vectors u and v the following equations can be used, respectively.

$$u^{n+1} = \bar{u}^n - \frac{E_x [E_x \bar{u}^n + E_y \bar{v}^n + E_t]}{\alpha^2 + E_x^2 + E_y^2} \quad (4.21)$$

$$v^{n+1} = \bar{v}^n - \frac{E_y[E_x \bar{u}^n + E_y \bar{v}^n + E_t]}{\alpha^2 + E_x^2 + E_y^2} \quad (4.22)$$

The above are the recursive equations for obtaining the optical flow vectors. Where α is a weighting factor which is also used to avoid division by zero. \bar{u}^n and \bar{v}^n are local averages and are defined as

$$\bar{u}_{i,j,k} = \frac{1}{6}[u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}] + \frac{1}{12}[u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}] \quad (4.23)$$

$$\bar{v}_{i,j,k} = \frac{1}{6}[v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}] + \frac{1}{12}[v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}] \quad (4.24)$$

The detail derivation of these equations can be found in [96]. It can be seen from the above equations that obtaining the optical flow information is a computationally expensive process as to process every pixel, many floating point operations are performed. As mentioned previously, Blackfin processor is not good in performing floating point operations, so for computing the three partial derivatives E_x , E_y and E_t , the decimal operation was performed in place of floating point operation. This reduces the accuracy of the computed optical flow vectors, but at the same time it also reduces the computational complexity of the algorithm and also the execution time. Another approach used to reduce the computational complexity is to scale the grey scale image values by some factor (for example, 10000) and then perform the decimal operation in place of floating point operation. This way, it is defined in the beginning that in floating point format, up to what decimal point the precision lost is affordable. With this approach, the resultant optical flow vector which is 1.3978654 when floating point operation is used, will now become 13978 if decimal operation is used with scale factor set to 10000.

The optical flow vectors, obtained using the above procedure, can be split into two components, the rotational (x component of flow vector) and the translational component (y component of the flow vector). Once the optical flow vectors are obtained, the next task is to determine when the obstacle is very close to the robot. In typical approaches [43] the focus of expansion is calculated. Focus of expansion is a singular point on the image plane from where the image motion (due to translational motion of robot) everywhere is directed away. The focus of expansion is shown in Figure 4.27.

Using the translational component of the optical flow vector and the calculated

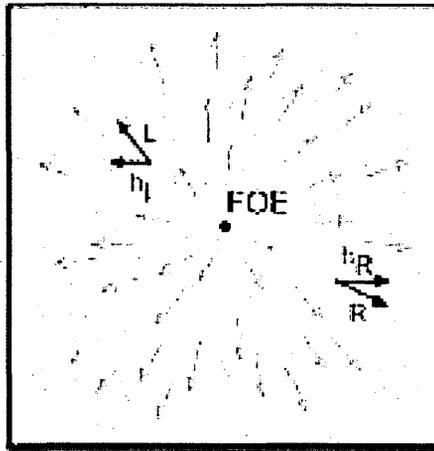


Figure 4.27: Focus of expansion (FOE).

FOE, the Time To Contact (TTC) to the obstacle is computed. The equation used to determine the time to contact is given as.

$$TTC = \frac{\Delta_i}{|\vec{V}_t|} \quad (4.25)$$

Where \vec{V}_t is the translational component of the flow vector and Δ_i is the distance of the point (x_i, y_i) on the image plane from the focus of expansion. A lower threshold value can be applied to the time to contact information such that, whenever the time to contact is less than that threshold value, the robot determines that the obstacle is very close to the robot and necessary action is required.

The optical flow based approach used in this research, is based on a very simple strategy. Based on the optical flow field, every time the magnitudes of the optical flow vectors of the left and right half of the image were calculated. If the sum of these magnitudes exceeded some predefined threshold, it was assumed that the obstacle was in front of the robot. Then the flow magnitudes computed for the left and the right half were used to determine the direction in which the robot was required to turn. If the flow magnitude from left half was greater than the right half, then it was assumed that the robot was closer to the obstacle from left side so it took a right turn.

Path Finder Approach

The last vision based approach used for obstacle avoidance is given the name, “Path Finder”. In this approach, rather than segmenting the whole image, the segmentation is performed from the bottom part of the image only. This way, the biggest segment resulting from the ground, providing the ground clearance, can be determined and finally can be used to determine the distance from obstacle lying in front of the robot. The basic idea is illustrated in the Figure 4.28.

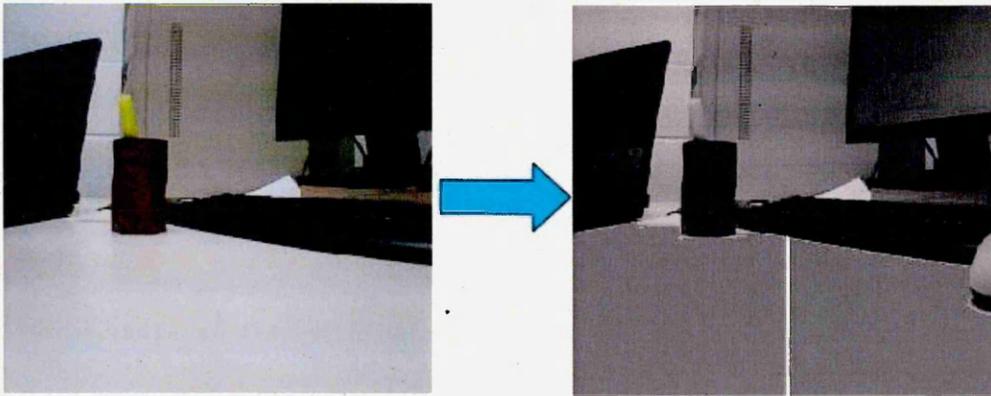


Figure 4.28: Path finder approach to obstacle avoidance

4.3.2 Experimental Results

In this section, the results obtained from the implemented obstacle avoidance approaches are presented. The objective of explaining the results here is that, the developed obstacle avoidance approaches can be efficiently executed in real time. An effort was made to minimise the execution time of these approaches so that they can be executed in parallel to the distributed vision processing tasks.

Segmentation Based Obstacle Avoidance - Experiments

To test the image segmentation based obstacle avoidance algorithm, a test platform for SRV1 robot was developed. To make ground plane, white sheets of paper were joined together and some objects were placed on the ground plane which acted as an obstacle for the robot. In Figure 4.29, a view of the testing platform with SRV1 robot and many obstacles placed on it is shown.

The SRV1 robot has a wireless connection to the development platform and through this wireless connection the processed images can be uploaded to the de-

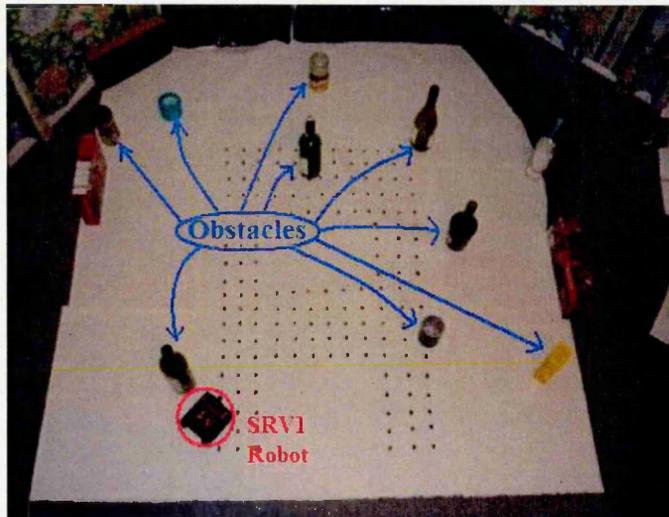


Figure 4.29: Test platform for SRV-1 robot with obstacles placed on it.

velopment platform for debugging. This image upload process is a slow process. Moreover, if the robot is processing stream of images for performing obstacle avoidance task, then uploading the stream of processed images can slow down the performance of robot. Due to this, the required response time for processing images and necessary action taken by the robot to avoid obstacle cannot be achieved by the Blackfin processor. To view the processed images by the robot, when it is moving towards the obstacle, the robot was programmed to move in non continuous fashion (i.e. robot moved a small distance, stopped and executed the segmentation based obstacle avoidance algorithm, transmitted the processed images to the development platform and then moved again). This way, some processed images were uploaded by the robot when it was moved in the testing platform with the position of obstacle arranged as shown in Figure 4.30.

On the right side of Figure 4.30, the trajectory followed by the robot is shown. The robot has gone straight, encountered the first obstacle and took a right turn. It has gone straight again, encountered the second obstacle, and took a right turn and then simulation over. The processed images uploaded from the robot are shown in Figures 4.31 to 4.37. In Figures 4.31 to 4.37, on the left most side, raw image stream captured by the robot vision system is shown. Image resolution is set to 120x160 pixels to reduce the computational load. In the middle, the processed images obtained after performing segmentation and removing small segments is shown. On the right most side, the final visible ground map obtained is shown. As already

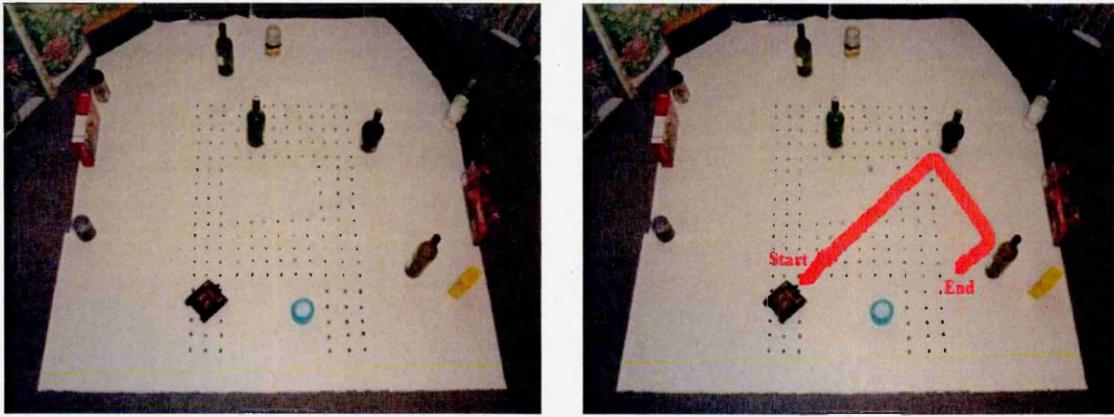


Figure 4.30: In left image robot environment and placement of obstacles are shown. In right image, the path followed by the robot is shown.

mentioned, if the number of white pixels from the middle bottom of the image to the obstacle (shown by blue line in the right column images) are greater than 30, then robot moves forward assuming that the obstacle is far from the robot. It also checks for enough free space for robot to go forward. For this purpose 80 pixels wide region (shown by red line in the right column images) covering the middle bottom of the image is also scanned. For example, in the images shown in Figures 4.31 to 4.37, from steps 1 to 9, the distance to obstacle was greater than 30 pixels so robot kept on moving forward. But in the image shown in step 10, the distance to obstacle is less than 30 pixels. Moreover, in step 10, the information obtained from 80 pixels wide scanned area also tells the robot that it is closer to the obstacle from the left side. Considering these facts, the robot control algorithm takes the decision to turn right. Now again from steps 11 to 12, the robot finds enough space to move forward, but in step 13, the distance to obstacle is less than 30 pixels. Information obtained from 80 pixels wide scanned area again tells that the robot is closer to the obstacle from left side. So the robot control algorithm takes the decision to turn right. After turning, the images processed by the robot give information about the new scene is shown in step 14. Here, the test simulation ends at step 14.

Using this segmentation based obstacle avoidance algorithm, many tests were performed. One of the paths followed by the robot, when obstacle avoidance algorithm was tested in the testing platform, is shown in Figure 4.38. The path followed by the robot is drawn in three different colours. If the same colour is used for representing the path, then the fact that the robot trajectory is crossing at many places makes it difficult to understand the exact path followed by the robot. Con-

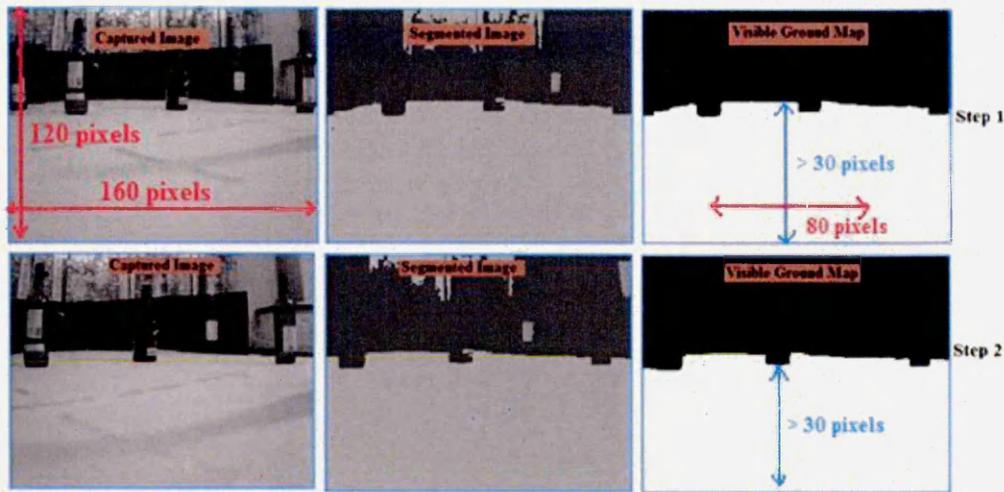


Figure 4.31: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 1 and 2



Figure 4.32: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 3 and 4

Considering this problem, the places where the robot trajectory is going to overlap the previously followed trajectory, a different colour is used to represent the path. In Figure 4.38, the robot has started from the red path and the test ends when the green path ends. The starting point and the ending point of the path followed by the robot are also identified in the figure. The maximum frame rate achieved with segmentation based obstacle avoidance approach is 2.46 frames per second, where the image resolution is kept 160x120 pixels. This means, the algorithm requires

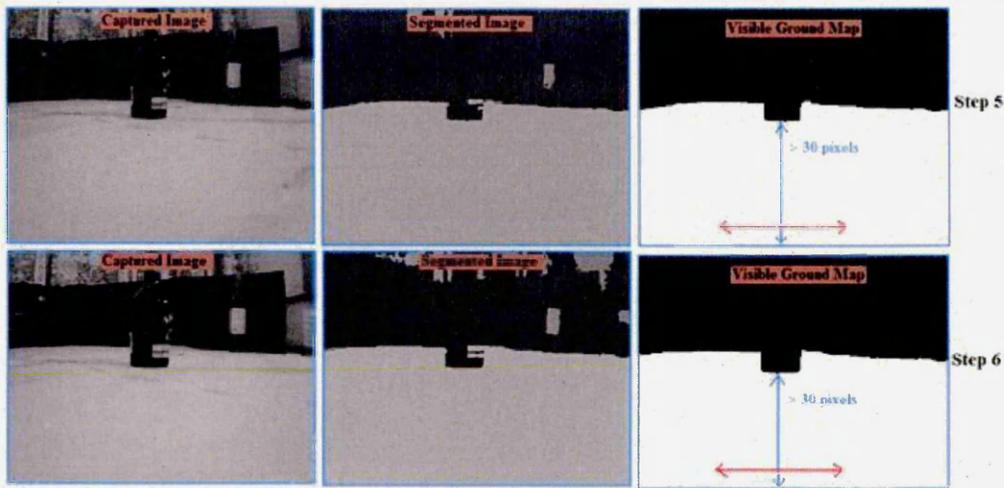


Figure 4.33: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 5 and 6

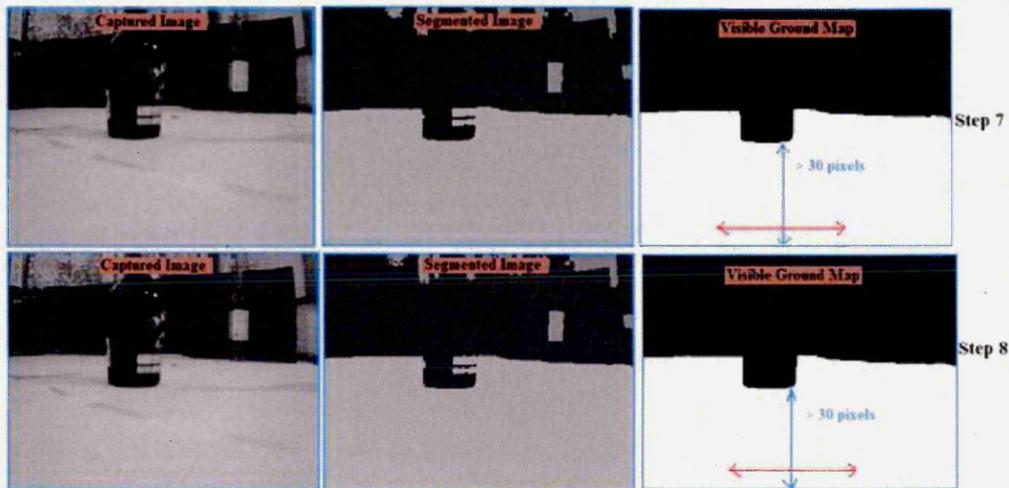


Figure 4.34: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 7 and 8

406 msec to process one image. This is sufficiently fast considering the processing power of the Blackfin processor. With this approach, it is possible to move the robot without colliding with the obstacle in real time. But it is found that this approach is not fast enough to execute in parallel to the distributed vision processing scenario as it does not leave enough processing resources for performing the actual task of distributed vision processing.

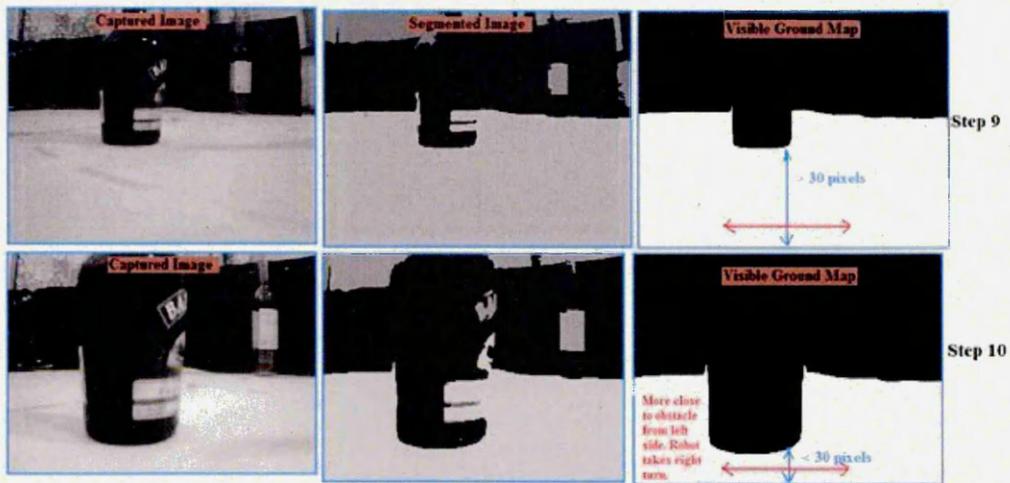


Figure 4.35: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 9 and 10

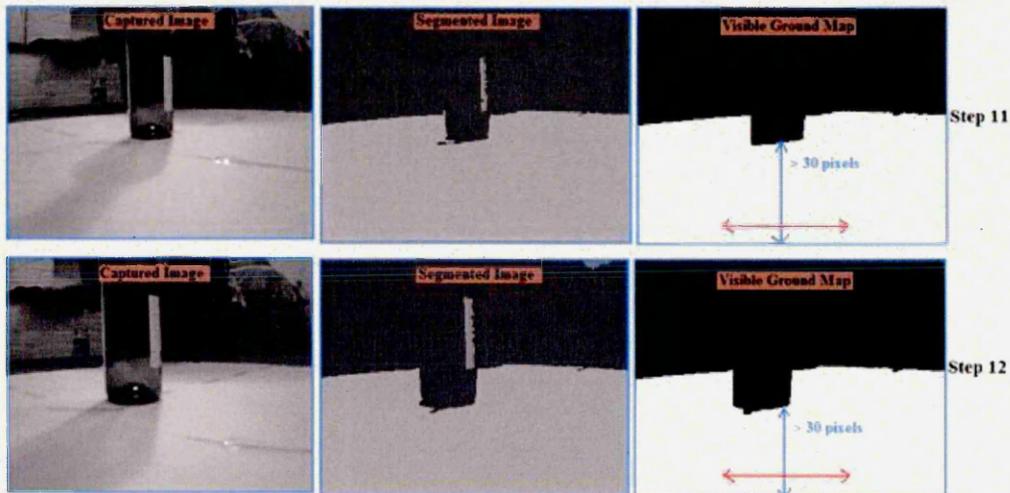


Figure 4.36: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 11 and 12

Optical Flow based Obstacle Avoidance - Experiments

In this section, the results obtained from optical flow based obstacle avoidance are presented. As already explained in the previous section, when the robot is moving and at the same time processing stream of images to take some decision, transmitting the processed images to the development platform is not possible. As image transmission is a slow process and to view the results obtained after every step, the raw image and optical velocity vectors in x and y directions for the whole image

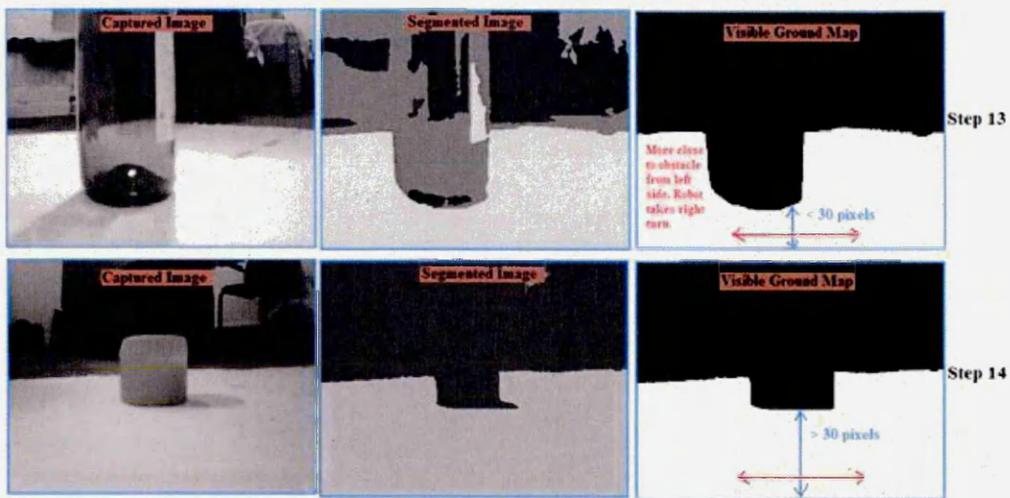


Figure 4.37: Processed images obtained from segmentation based obstacle avoidance algorithm. Steps 13 and 14

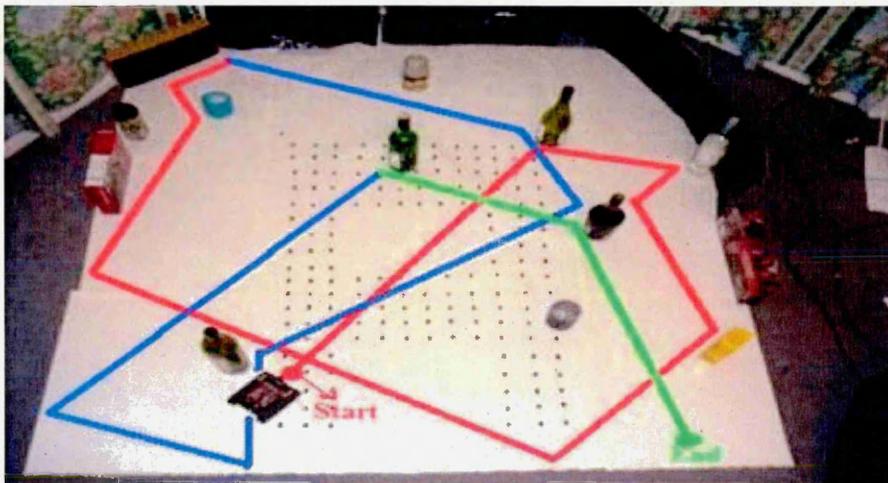


Figure 4.38: Path followed by the robot when segmentation based obstacle avoidance is used.

were transmitted. This sums up to three times the number of bytes the actual image comprises of. So acquiring the optical flow field computed by the robot for the complete experiment is not performed.

An example optical flow field obtained when the robot is moving in the testing platform towards an obstacle is shown in Figure 4.39. Frame 1 is the first image taken by the robot and frame 2 is the second image. The optical flow field obtained between these two consecutive frames is plotted on frame 1 in yellow and shown in the bottom of figure. In Figure 4.40, a zoomed in version of optical flow field is also

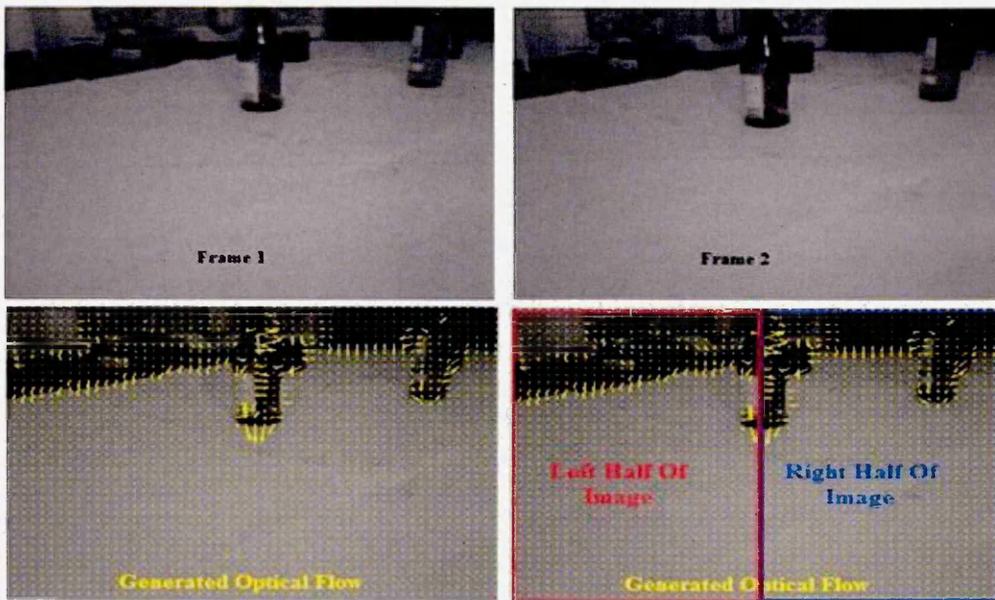


Figure 4.39: Optical flow field obtained between two consecutive images.

shown. Note that, the optical flow vectors obtained are in the correct direction (i.e. downward, opposite to the direction of motion), but there are some vectors which are in random directions. These vectors act as noise. These random vectors can be eliminated by considering only those vectors which share the same direction and greater in number. This way of eliminating random vectors works well if the robot is minimum at 6cm distance from the obstacles such that, the obstacle is not covering the field of view of robots' camera. But when robot is very close to the obstacle then obstacle boundaries result into optical flow vectors in all directions and these vectors cannot be considered as noisy vectors in random directions. However, these noisy vectors are not greater in number and in the current implementation of obstacle avoidance, it does not affect the performance.

As mentioned before, every time the magnitudes of the optical flow vectors of the left and right half of the image are calculated and if the sum of these magnitudes exceeds some predefined threshold then the obstacle is assumed to be very close. The way images are divided into left and right half is shown in the bottom right of Figure 4.39. The flow magnitudes computed from these left and the right half also provide information about the direction in which the robot is required to take turn. The maximum frame rate achieved with optical flow based obstacle avoidance approach is 3.5 frames per second, where the image resolution is kept 160x120 pixels.

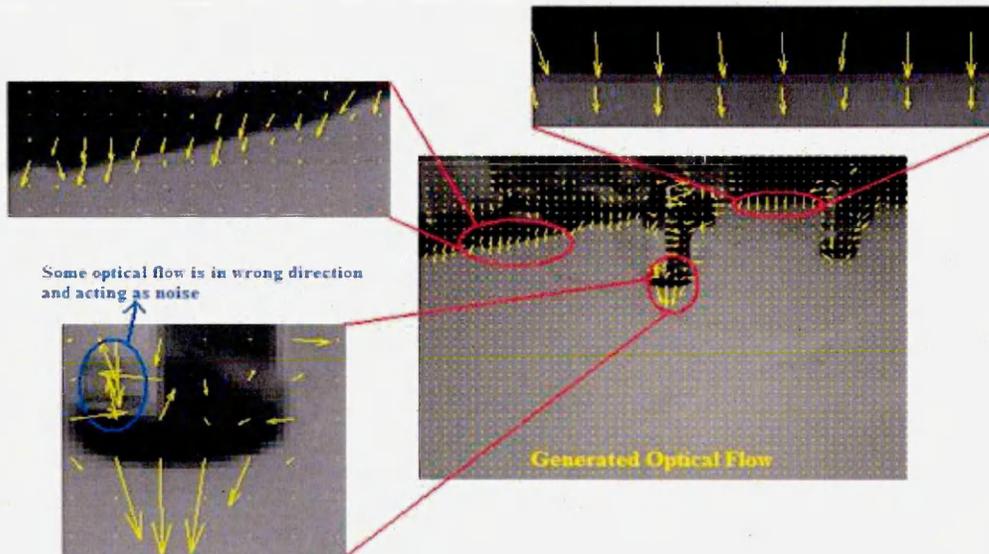


Figure 4.40: Zoom-in version of the optical flow field shown in Figure 4.39.

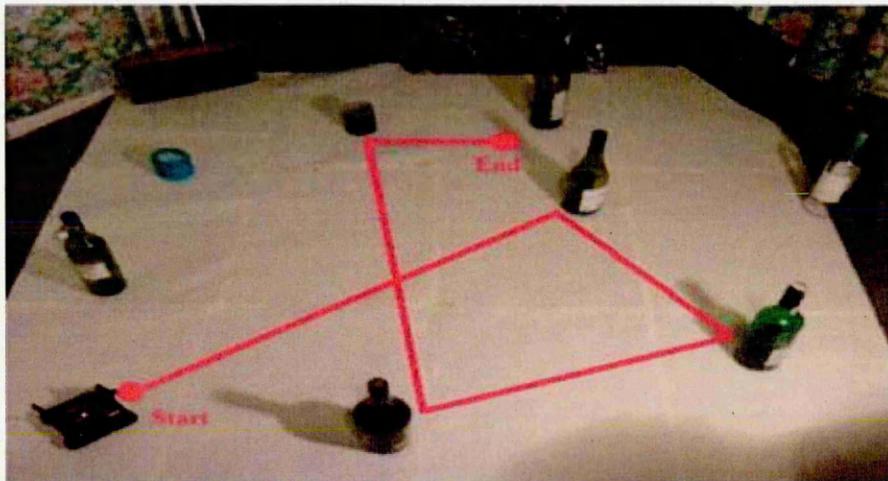


Figure 4.41: Path followed by the robot when optical flow based obstacle avoidance is used.

Following the above strategy, an example path followed by the robot is shown in Figure 4.41. In the experiments, it is observed that, when the robot encounters an obstacle and takes a turn and if after turning, it is very close to another obstacle, then there are chances for the robot to collide with the obstacle. This is because the algorithm is relying on the sum of the magnitudes of the optical flow vectors. And when the robot is very close to the obstacle such that the obstacle surface is covering the whole field of view of the robot, then there will be no optical flow generated

by the movement of the robot. So this behaviour of robot colliding the obstacle is expected and is the drawback of this approach.

From the timing analysis of this algorithm, the execution time required by it to process one frame is 285 msec. The execution speed of this algorithm is found better than the segmentation based approach, but it was again not good enough to execute in parallel with the distributed vision processing scenarios.

Path Finder - Experiments

As mentioned, this approach does not segment the whole image but tries to find the biggest segment from the bottom section of the image, which ideally would be resulting from the ground region. Many experiments were performed using this approach. The image size used with this approach is 320x240 pixels, which is higher in resolution as compared with what is tested with the other approaches. This algorithm is also found to be very reactive as it took only 22 msec to execute and comparatively much faster than segmentation and optical flow based approach which took 406 msec and 285 msec respectively, to process one image. The selection of this algorithm is made for use in the distributed vision processing scenario because of two factors, namely: it is faster to compute and this algorithm also leaves enough time in which communication among robots and distributed vision processing task (e.g. distributed appearance based recognition) can be performed.

4.4 Energy Foraging

As explained before, when the robots get hungry and running out of the battery, they look for energy sources in the surrounding. Here vision support for finding the energy sources is presented. To facilitate the identification of energy sources, it was decided to use different colour LEDs near the energy sources. Then using a colour blob detection algorithm, the blob resulting from the LED could easily be identified and location of energy source could be determined by the robot. In Figure 4.42, the output of the colour blob detection algorithm is shown when Red LED was used near the energy source. In the left most image, the colour image is shown. In the middle, two images are shown. The top image shows the robot view and in bottom the output of colour blob detection for the current view is shown. Similarly, the robot view and detected colour blob output, when the robot moved closer to the

last detected blob, is shown in the right most side of the figure.

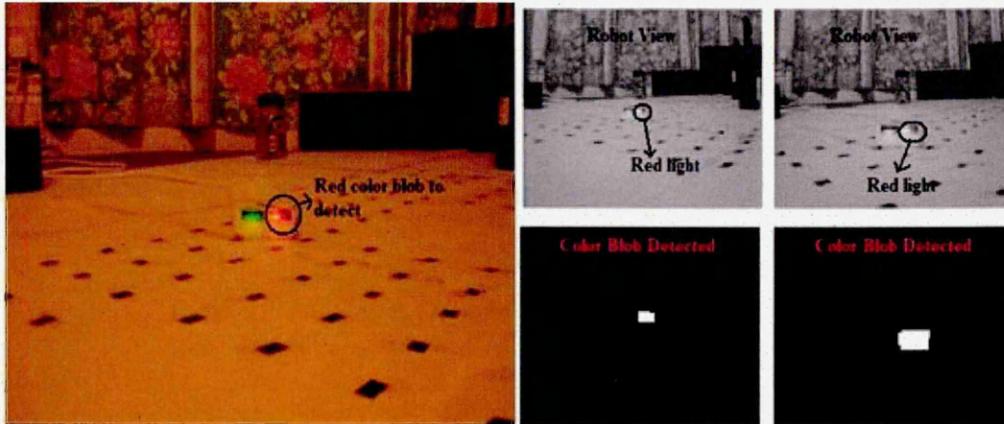


Figure 4.42: Image showing the detection of red colour blobs when colour blob detection algorithm is used.

4.4.1 Experimental Results

The results obtained for the colour blob detection algorithm, to facilitate robot to locate energy resources for recharging battery, is presented in this part. To test the effectiveness of the algorithm, the LED mounted on the extension cable was used as an alternative to represent the energy source. So, whenever the robots needed to recharge their batteries, they could, in parallel, process the stream of images using the colour blob detection algorithm and could locate the charging points. To test the performance of the algorithm, the robot was programmed to work in parallel with obstacle avoidance (Path Finder based approach) and colour blob detection mode to detect the LED. As long as the LED was not detected, the robot continued to move and kept on avoiding the obstacles. As soon as the robot found the LED, it will come out of the obstacle avoidance mode and tried to reach the location where the LED was present. In Figure 4.43, the path followed by the robot is shown. The objective of this experiment was also to show that the colour blob detection algorithm discussed in the “Embedded Vision Library” section was capable of running in real time. From the timing analysis performed during this experiment, the execution time of this algorithm was found only 6 msec. This is very promising and supports that this vision algorithm can run in real time while leaving enough processing resources for the distributed vision processing scenarios.

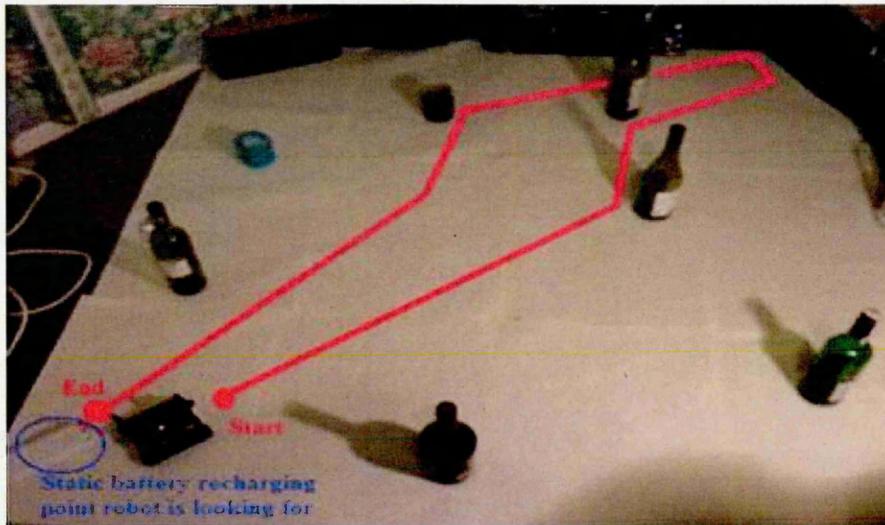


Figure 4.43: Path followed by the robot to locate the energy resources.

4.5 Vision Based Docking Support

As discussed in Chapter 1, in the swarm robotic system considered in this research, the robots were capable of forming a three dimensional organism by physically docking together. To achieve this, a computer vision based solution is discussed in this section. On the docking side of all the robots, a special four LEDs pattern was used. When the robot requires the other robots to dock to it, for forming an organism, it turned ON the four LEDs mounted on its backside. The other robots went around and tried to look for the robot which had the four LEDs turned ON. Using a computer vision algorithm, the robots tried to get close enough to the robot which required docking operation. Once the robots were close enough then here the vision based help for docking operation finishes. To achieve further precision in the alignment, the control was transferred to the Infrared sensor based docking operation, which is not part of this study as it is not related to computer vision field.

Considering the distributed vision processing scenario presented in Figure 1.4, the mobile robot went around in an unknown environment, doing obstacle avoidance and computationally expensive appearance based search operation to look for the places of interest. This was all based on input from vision sensor, so a very light weight vision based docking algorithm was required to achieve the real time performance. The fact that this algorithm ran all the time (i.e. when docking was required) in parallel with the other computation expensive vision algorithms and wireless based

multi-robot communication algorithm. So this made the task more challenging. Here a vision algorithm based on colour blob detection approach is presented. This algorithm looked for the LEDs which were turned ON and were also in the required known pattern. In the beginning, it seemed as the work could be done by identifying the red blobs in the current image, filtering the blobs which were in the required order to remove noise, measuring the distance between the detected LEDs blob to figure out the distance from the robot and finally write a piece of control algorithm to drive the motors. But the following observations made the task more challenging.

- (i) When LEDs were turned OFF, then LEDs in red colour were detected as red blobs in the image. But how it appeared when LEDs were turned ON? When red LEDs were turned ON, then the centre of all the LEDs appeared red to human eyes but to vision sensor it appeared as shown in Figure 4.44. For comparison, a red LED in OFF mode was also present in the image. In case of ON LEDs, it could be seen that the centre of all the LEDs appeared bright White to the vision sensor.

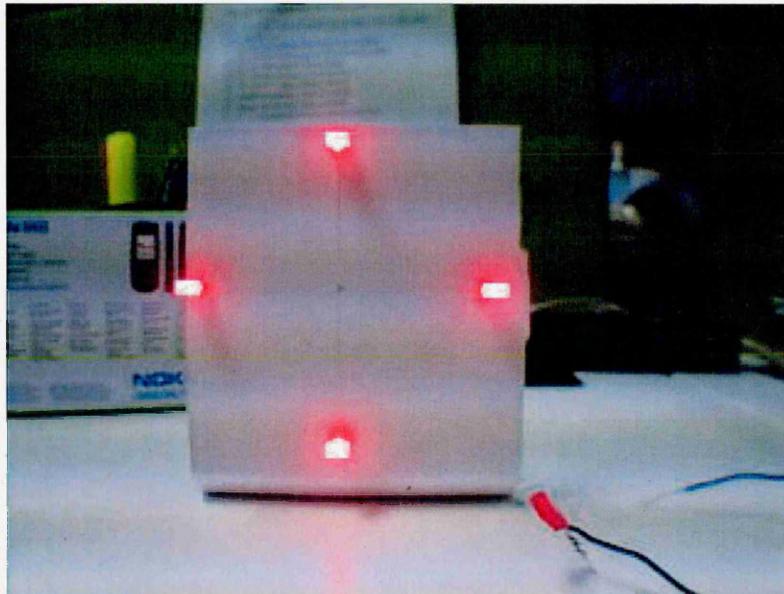


Figure 4.44: Image showing LEDs used for docking operation

- (ii) As the Surveyor robots, used for experimentation, had only one camera in the centre, so when the robot tried to get very close to object, the LED blobs go out of its vision. This supports the idea that the alignment could be performed up to some distance from the other robots which required docking. When robots

were close enough then vision based help could not be provided for precise alignment and help from some other sensor was required.

- (iii) The differential drive support on the Surveyor robots to perform robot motion did not allow very slow motion. The robot did not move if the speed reduced from certain value. The motors tried to move the robot but the robot appeared to be too heavy to the motors to perform very precise movement. This observation was more true when motors tried to rotate the robot. This is mainly because of the tank style treads tyres which increased the area by which the robot was in contact with the ground surface. This resulted increasing the robot grip to surface and caused lots of friction and hence restricted the precise movement.
- (iv) When the robot was approaching the four LEDs blobs, the blob did not appear to be forming a reasonable required pattern. The pattern could be very tilted depending on the direction and angle from which robot was approaching the blobs for docking.

Based on the above observations, it was required that a colour blob detection based algorithm be adopted which also gave a certain level of confidence to drive the robot in the right direction to perform the docking operation. One way to solve the problem addressed in first observation was to explore for very bright spot in the image only. But in this case, the colour of LEDs provided no information (i.e. robot was treating LEDs of different colours equally). Also the bright white image appearing in the background and reflections from the surfaces of objects around (which in turn appeared to be bright White) caused extreme noise which resulted in lowering the performance of the algorithm. It was therefore decided to breakdown the problem into a number of processing steps which in turn reduced the dimensions of complexity in every processing step while not sacrificing the performance. The operations done in the processing steps are the following:

4.5.1 Blob Detection of Red LEDs in ON State

First of all, the image was processed with the developed blob detection algorithm which was configured to detect Red colour blobs. The output of blob detection algorithm was further processed with the image dilation algorithm to fill the small holes in the processed image. Normally an image dilation algorithm processes and di-

lates the whole image. Considering the importance of processing time, a customised dilation algorithm was implemented which, rather than dilating the whole image, looked for only the part of image where the red blobs were detected and processed it. Doing this was rather easy as the output image from colour blob detection algorithm was a binary image. Now to show the performance of algorithm to detect the blobs caused by red LEDs, the following figures are presented. Figure 4.44 shows the colour image captured by the robot. This image shows the four red LEDs in ON state and one LED in OFF state.

As mentioned before, the colour blob detection algorithm utilised the U and V images from YUV image format for blob detection so U and V images of the above input image are also shown in Figures 4.45 and 4.46 respectively.

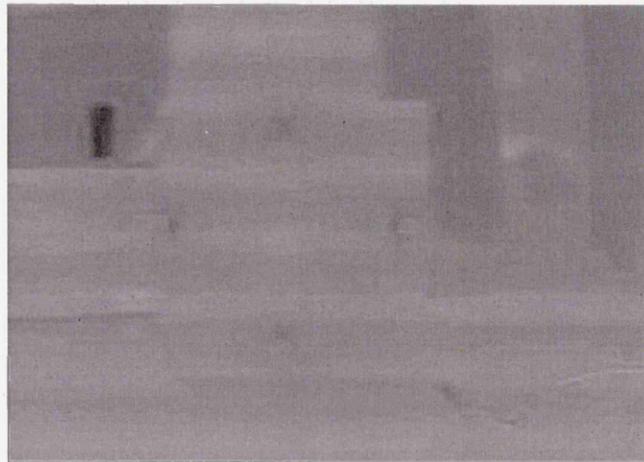


Figure 4.45: U image of Figure 4.44

In colour blob detection algorithm, all U values greater than 90 were considered which also covered darker spots occurring around LEDs due to the presence of red colour (i.e. Figure 4.45). As the U image did not provide the complete chrominance information, so when U image was processed together with the V image, then a robust colour detection could be carried out. In Figure 4.46, it can be seen that the image appears very bright where red LEDs are located. The LED in OFF state is causing a perfect continuous bright spot in the image where the LED is located. But the LEDs which are in ON state are causing brightness around the LEDs location only but not in the centre which in-fact appears to be dark (i.e. not red) to the vision sensor. So in the output image from colour blob detection algorithm, the unfilled holes are expected in the centre of the red blobs caused by the red ON LEDs. This

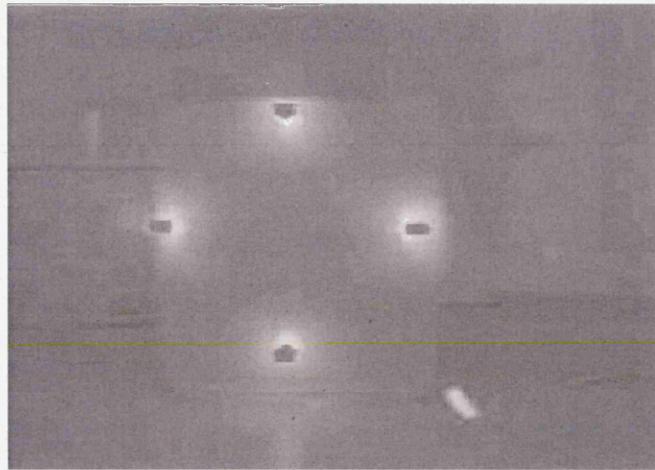


Figure 4.46: V image of Figure 4.44

was overcome by the use of a customised dilation algorithm. The processed output image from this processing step is shown in Figure 4.47.

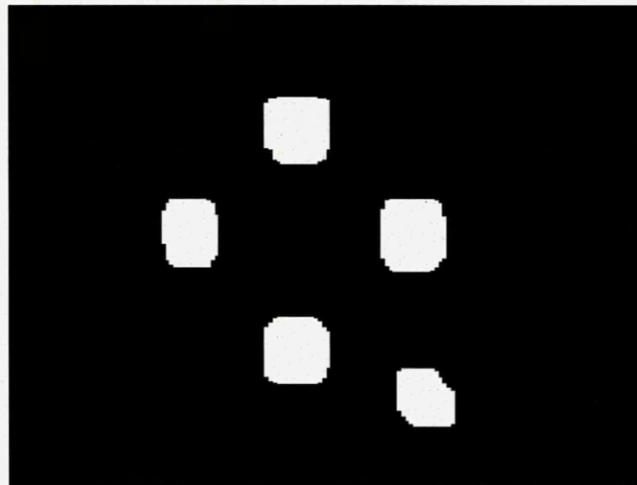


Figure 4.47: Output of colour blob detection algorithm

In order to determine the blobs caused by the LEDs in ON state, the brightness information from YUV image was utilised. It is important to note that the feature of ON LEDs, which in the beginning appeared to be a problem (i.e. the centre of the LED image appeared bright White rather than Red), could be cleverly used to eliminate the blobs caused by OFF LEDs. The output of colour blob detection algorithm was further processed and only those blobs were extracted whose centre appeared

to be bright White. The Y image in the YUV format provided the image brightness information. The Y image resulted from Figure 4.44 is shown in Figure 4.48.

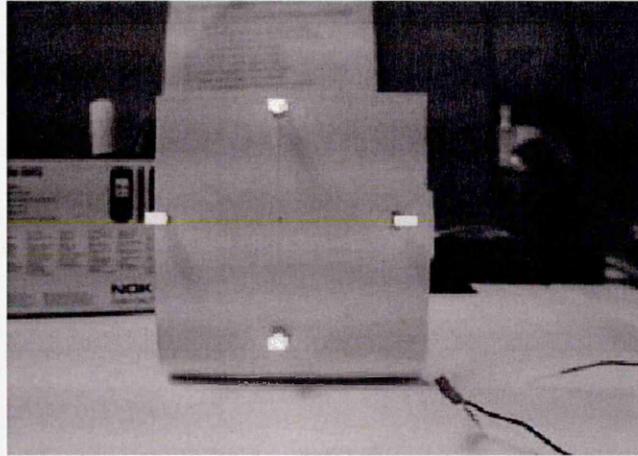


Figure 4.48: Y image of Figure 4.44

In Figure 4.48, it can be seen that it is very easy to identify the red LEDs which are in ON state by simply thresholding the Y image and using it together with the output of colour blob detection algorithm which is shown in Figure 4.47. The final image showing the colour blobs resulted from LEDs in ON state is shown in Figure 4.49. To increase the number of pixels representing the ON LEDs blobs, the resulting image was further dilated.

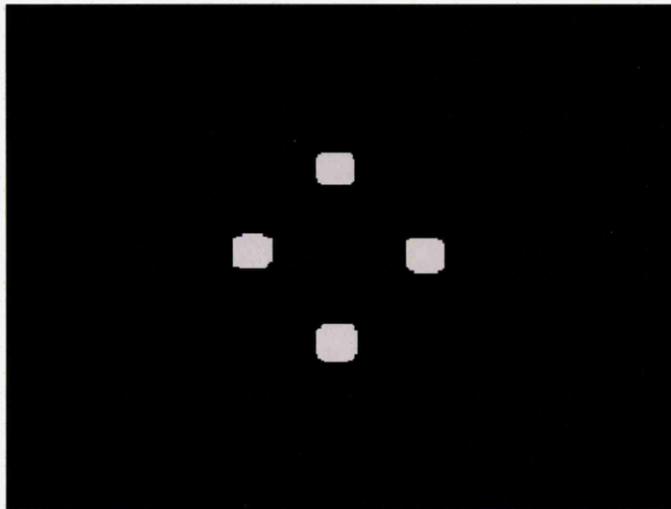


Figure 4.49: Blobs resulting from red LEDs in ON state.

4.5.2 Obtaining the Statistics of Red LED Blobs

To obtain the statistical information (i.e. the location of blobs in x and y coordinates in image) of the red LED blobs, the output image shown in Figure 4.49 was processed by a extract statistics algorithm. The image shown in Figure 4.49 is a binary image in which 0 value is representing the dark part and 1 is representing the blobs caused by Red LEDs in ON state. Note that all the blobs appear as separate segments. The developed statistics algorithm performed segmentation of the image. All the connected segments were given different IDs. This way, once the image was processed, a unique ID for all the pixels representing one blob (i.e. four unique ID's for four blobs) is produced. With the help of this ID, the centroid of all the pixels contributing to one blob could be easily calculated. The processed image providing statistical information of the LED blobs is shown in Figure 4.50. Using the statistical information, the centroid of LED blobs were marked with Red cross sign.

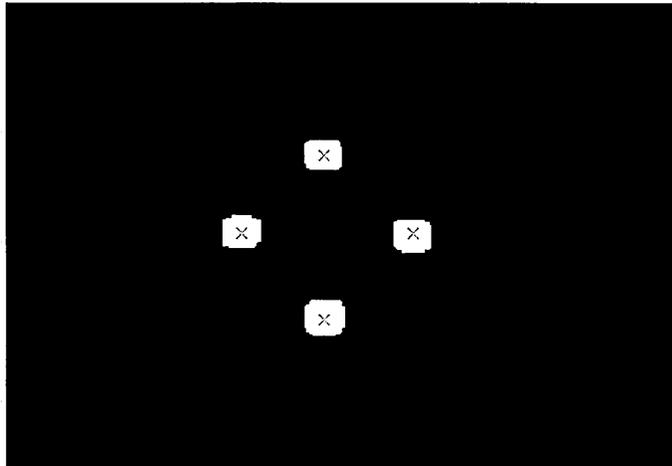


Figure 4.50: Image representing blobs statistical information.

4.5.3 Classification of Red LED blobs

After determining the statistics, blobs satisfying the required pattern were classified as Top, Bottom, Left and Right LED blobs. The classification algorithm made a reasonable assumption that while scanning the image from top to bottom, the first blob found was most likely to be from the Top LED following some conditions. Otherwise, the rest of the blobs were checked one by one. These conditions are as

follows.

- Around the currently assumed Top blob, a cone shaped search field was defined as shown in red colour in Figure 4.51. In this field, the algorithm tried to locate the Bottom blob. Some checks were made to avoid the blobs resulting from reflection of Top LED to be considered as Bottom LED blob. The bottom blob should not be detected very close to the top blob. In the current implementation, it was defined to be detected at-least 20 pixels down from the top blob and the blob size in pixels was almost the same as the top blob. Here, a 20 pixels limit was determined empirically for QVGA resolution in which processing is done.
- Once the top and bottom blobs were found, then their centre point was determined. Across this centre, 60 pixels wide search field was defined. The blob which was found on the left side of this search field was most likely the blob resulting from left LED. Then again, a cone shaped search field was defined, extending in right direction (shown in yellow colour in Figure 4.51). Algorithm searched this field to look for the blob resulting from right LED.

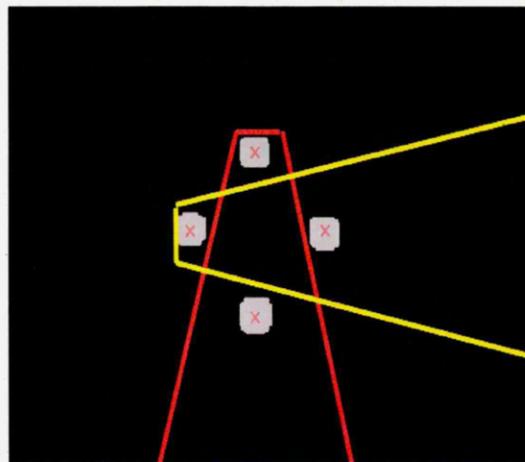


Figure 4.51: Search field for neighbouring blobs.

4.5.4 Control Algorithm to Approach the Blobs

Flow diagram of the control algorithm is shown in Figure 4.52. It performed the following sequence of operations.

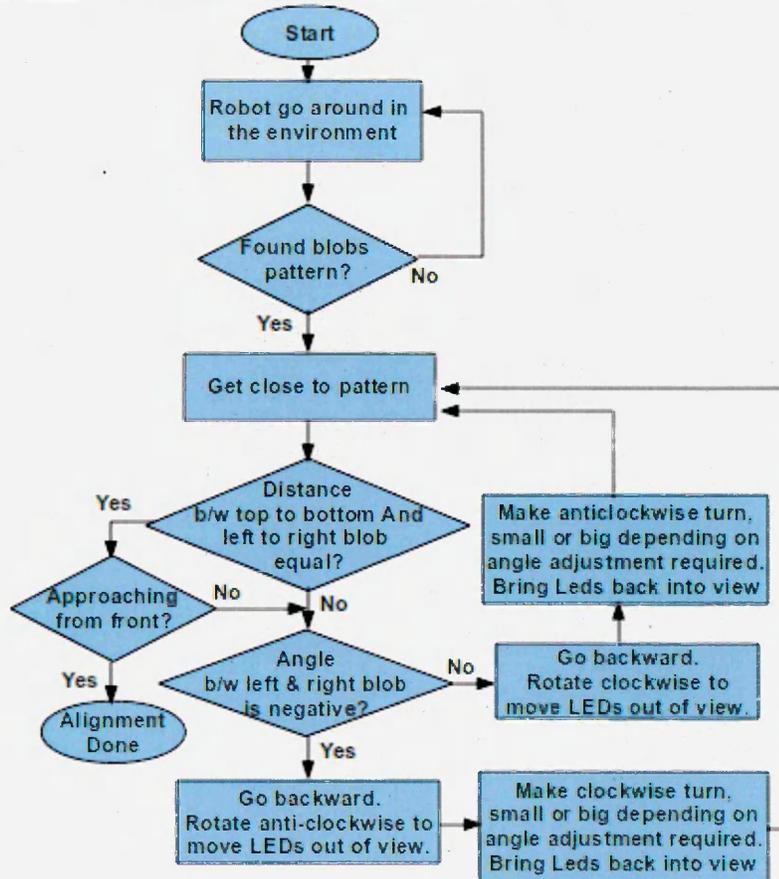


Figure 4.52: Flow diagram of control algorithm.

- Move robots in the environment and search for the blobs in the required pattern. On finding the pattern, the algorithm performed the blobs classification. As there was a strong possibility that the robot was not approaching the LEDs from the front, but at some angle and this angle could be small or large (can not be identified clearly from distance), so rather than performing alignment in the first go, the robot first tried to get close to the blobs.
- If the distance between Top to Bottom and Left to Right blobs were equal and robot was approaching from front, then robot assumed that the maximum precision was obtained using vision and control algorithm stopped there. Otherwise, the robot determined the direction of its approach.
- If the robot was approaching the LEDs from left side with reference to the LEDs locations, then the LEDs pattern appeared as shown in the Figure 4.53.

The right LED blob made a negative angle with the left blob. The control algorithm moved the robot backward, rotated it anti-clockwise so that the LED blobs went out of its vision. Then the algorithm moved the robot to make a clockwise turn. How big was the turn? It depended upon how big was the angle adjustment required. It kept on making this turn until the LEDs were back into its view field. Then it moved the robot again towards the blobs to see the further correction required.

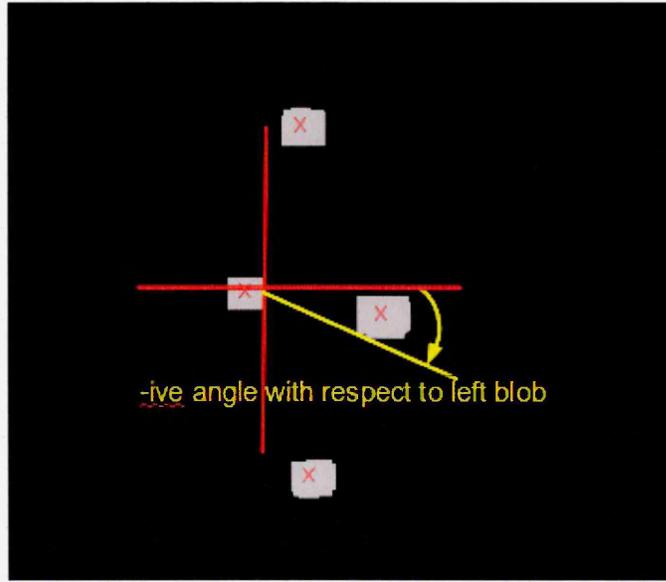


Figure 4.53: Robot approaching the LEDs for docking from left side with reference to the LEDs location.

- On the other hand, if the robot was approaching from right side then the LEDs pattern appeared as shown in the Figure 4.54. The right blob made a positive angle with the left blob. The control algorithm moved the robot backward, rotated it clockwise and then made an anti-clockwise turn to bring the LEDs back into view. This process continued until the robot align itself.

4.5.5 Experimental Results

The idea of using four LEDs on the back of robot and the vision based detection of these LEDs to facilitate the docking operation was just described. Using this approach together with a control algorithm, experiments were performed, for robot alignment with the four LEDs in ON state. The results obtained from four of these

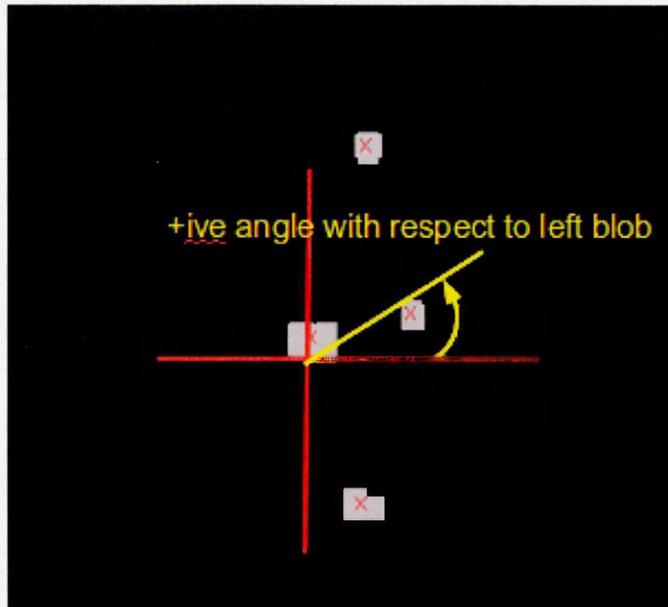


Figure 4.54: Robot approaching the LEDs for docking from right side with reference to the LEDs location.

experiments are presented. In Figure 4.55 the starting position of the robot, in four conducted experiments, before alignment, is shown. In Experiments 1 and 2, the robot was less misaligned but in experiments 3 and 4, more alignment was required as robot was approaching the LEDs with a very sharp angle.

The trajectory followed by the robot in experiment 2, is shown in Figure 4.56. As the robot was less misaligned with the LEDs shown on docking station, so control algorithm was able to achieve the vision based alignment in reduced time. In comparison, the result obtained from experiment 4 is shown in Figure 4.57, where more alignment was required to perform docking. In Figure 4.57, the different stages followed by the control algorithm to perform alignment, are shown in terms of trajectories followed by the robot in different colours. In the beginning, the robot approached the docking station straight (shown in Black colour) and determined the angle made by the left LED blob with the right blob. As the angle was large, so it moved back and made a big turn while approaching the docking station to reduce the error in angle (shown in blue). Finally, the robot was less misaligned so it moved back again and made a short turn (shown in green). This time the robot was reasonably aligned with the docking station and stopped further alignment.

The results obtained from the four experiments, after alignment was performed,



Figure 4.55: Initial pose of the robot before performing alignment.

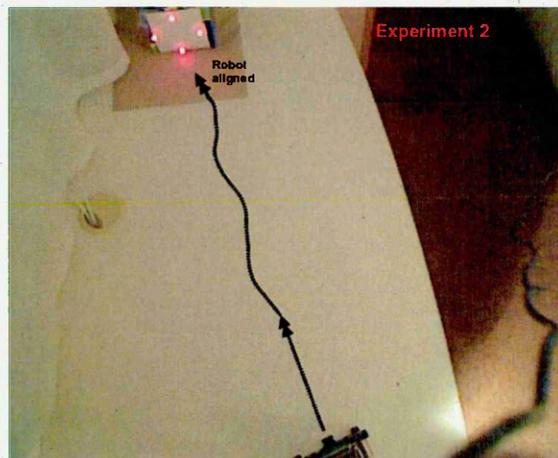


Figure 4.56: Trajectory followed by the robot in experiment 2.

are shown in Figure 4.58. In all these experiments, the robots were almost fully aligned with the four LEDs. This was the maximum support which vision could provide in docking. If the robot tried to get further close, then LEDs went out of its

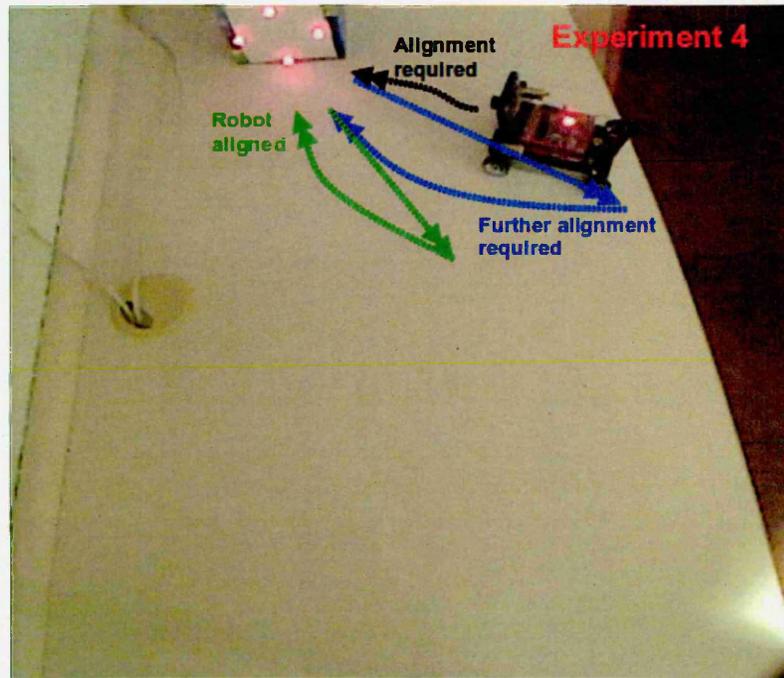


Figure 4.57: Trajectory followed by the robot in experiment 4.

vision (as the camera lies in the centre) and the robot could not take any decision about alignment. For final mechanical docking and further precise alignment, the control could rely on infra-red sensor information. Infra-red sensor provides accurate information at short distances (i.e. 5 to 10cm). Following this fact, the vision based docking algorithm will help the robots to get close (upto 5cm) to the docking port. Then the algorithm will switch to the robot control based on infra-red sensor information. Using infra-red sensor information, further alignment of the robot with the docking port and final mechanical docking operation will be performed.

To demonstrate the functioning of vision based docking support in a swarm robotic environment, an experiment was performed in which a number of robots were looking for docking port collectively. The docking port was installed on one of the robots. The robots were performing vision based obstacle avoidance, sharing information with each other and simultaneously looking for docking port. The robots were performing this task collectively that was, when a robot in a swarm would find the docking port, it would inform its team members to stop looking for the docking port and quit the mission. After informing the team members, the robot which found the docking port, aligned itself with the docking port using vision so that docking operation could be facilitated. Several tests were performed using this



Figure 4.58: Pose of the robot after performing alignment.

approach. In Figure 4.59, one of the experiments is shown. Three robots were used to perform the collective search operation. In the beginning, the LEDs used on the docking port of the robot were turned OFF as shown in Figure 4.59a. The robots started the mission with vision based obstacle avoidance, searching arena for docking station and in parallel, also informing each other whether any of them have found the docking port. Finally, one robot found the docking port, it informed the other team members that they were no longer required to search for the docking station, and align itself with docking station as shown in Figure 4.59b. All the other robots left the search operation. It can be noticed that, this vision based docking support may be used for the docking of two robots, so that they can become a single robotic organism. Or it can be used for docking with the energy source so that battery recharging operation can be performed.

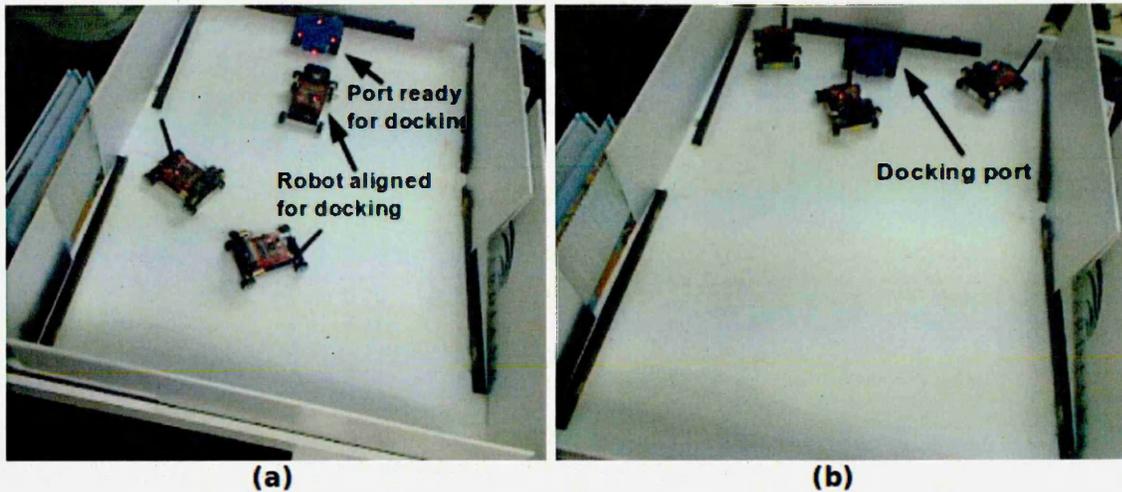


Figure 4.59: (a) Swarm of robots starting collective search for docking port. (b) One robot finds the docking port and the rest quit mission.

4.6 Conclusions

In this Chapter the development and implementation of basic vision processing algorithms has been presented. These vision algorithms provide the basis for the main distributed vision processing scenarios addressed in Chapters 6, 7 and 8. Using these basic vision algorithms, the necessary functionality required by the robots to move in the environment, that is obstacle avoidance, is also developed. The developed vision based obstacle avoidance algorithm is utilised in both swarm and the organism mode scenarios to facilitate a swarm of robots and a robotic organism avoid colliding with a number of obstacles. For the vision processing algorithms addressed in this Chapter, it is necessary that they achieve real time performance. For this purpose, it has been concluded that the customisation and optimisation of these algorithms is necessary for the target embedded system used on the robots.

Chapter 5

Multi-Robot Localisation and Tracking System

Vision based robot tracking and localisation using a fiducial marker based approach is adopted in many indoor multi-robotic research projects in which precise robot position and orientation are required for certain objectives. In this chapter, a *Multi-robot Visual Tracking System* is developed which performs the multi-robot localisation and tracking tasks using the information from two ceiling mounted cameras. This system utilises a new passive marker template which is designed to uniquely identify the robots and to precisely determine their positions and orientations in a multi-robotic environment. Using the designed template, a multi-camera solution to track robots in the robot arena, using off-the-shelf web-cams, is provided. As the passive markers do not consume energy, so their use together with off-the-shelf web-cams provides the energy and cost effective solution to the multi-robot localisation problem. The approach presented is found robust for localisation and tracking problem. Passive markers, which use colour information to code markers ID and orientation, are used on the top of each robot working in an environment. Two cameras, mounted on the ceiling, gave a collective view of the robot working arena. To show the effectiveness of this approach, its application to guide robots along predefined trajectories is also shown in this Chapter. Information from the two cameras is used to track and localise the robots' position and then the robots are given further instructions about their movements and orientation correction, so that they all could reach the target location in the presence of obstacles. This multi-robot localisation and tracking system is developed to provide localisation information to the robots in

the distributed vision processing scenarios in swarm mode. This chapter is divided into the following sections.

- Camera Calibration.
- Visual Localisation and Tracking System.
- Multi-robot Visual Guidance.

5.1 Camera Calibration

In a visual tracking and localisation system the robots were tracked in the images captured by the two ceiling mounted cameras. So to determine the robots' positions in real units or world coordinate system, first of all, camera calibration of the ceiling mounted cameras was required. The objective of camera calibration was to find the camera parameters which defined the relation between the 3D world coordinate points and the 2D points on the camera image. These parameters are represented in the form of a 3x4 matrix which is called camera matrix in computer vision terminology. The relation between 3D world points C_w and 2D image points C_i in terms of camera matrix K are explained next. The relevant equations are explained in detail in Chapter 4. Here, they are briefly described for convenience.

$$C_i = KC_w \quad (5.1)$$

This equation when expanded, can be expressed as

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = AB \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \quad (5.2)$$

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = A[RT] \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \quad (5.3)$$

where, u and v are the 2D camera coordinates and s is the scale factor. The 3D world coordinates are represented as x_w, y_w, z_w . The camera matrix K is represented as $A[RT]$, where A is the camera intrinsic matrix, R is the rotation and T is the translation matrix of camera with respect to the world coordinate points (World coordinate system is the reference coordinate system). R and T together represent

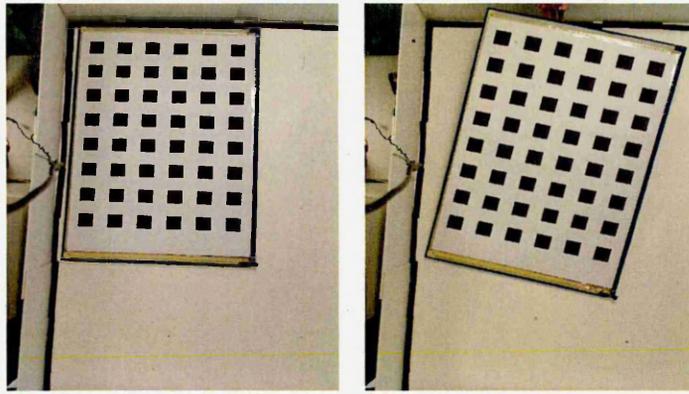


Figure 5.1: Image 1 (left) and image 2 (right)

camera extrinsic parameters B . The camera intrinsic A and extrinsic B matrices can be expanded as

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.4)$$

$$B = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.5)$$

where, α_x and α_y are the focal length of camera in terms of pixels. γ is the skew coefficient between x and y axis of image. And u_0 and v_0 are the principal points of the camera. Similarly, $r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}$ define the parameters of rotation matrix and t_x, t_y, t_z define the parameters of translation matrix. To cover the complete robot arena, both ceiling cameras were mounted 5 feet above the arena surface. Due to this, a big size calibration grid pattern was used for calibration purposes. The use of big calibration pattern was made so that the pattern could be clearly seen by the cameras. The images of calibration pattern captured by ceiling cameras were processed with MATLAB camera calibration toolbox [115]. Twelve images were used for calibrating the camera. These images are shown in Figures 5.1 to 5.6.

For camera calibration, the same procedure was adopted which was used for calibrating the robot camera as explained in detail in Chapter 4. The grid points were selected carefully on the pattern images using the calibration toolbox. The

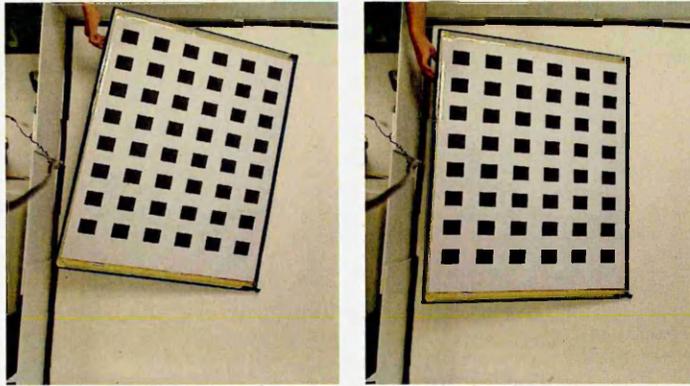


Figure 5.2: Image 3 (left) and image 4 (right)

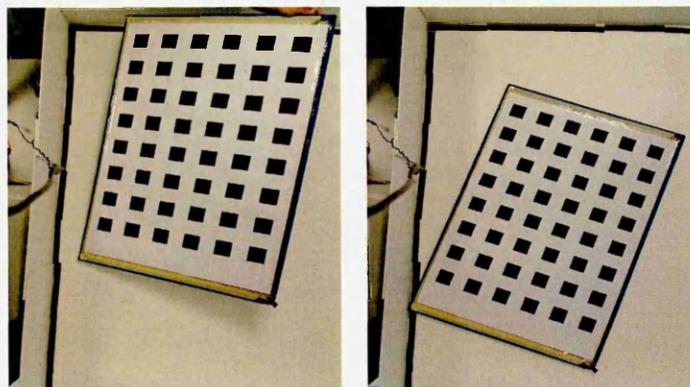


Figure 5.3: Image 5 (left) and image 6 (right)

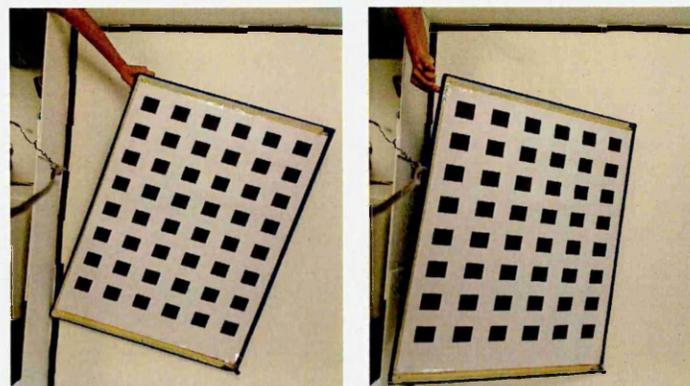


Figure 5.4: Image 7 (left) and image 8 (right)

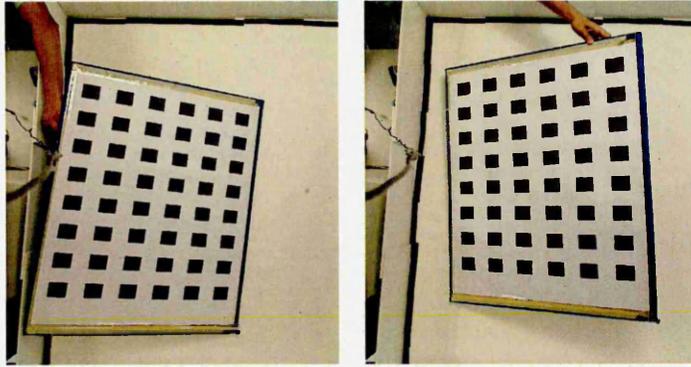


Figure 5.5: Image 9 (left) and image 10 (right)

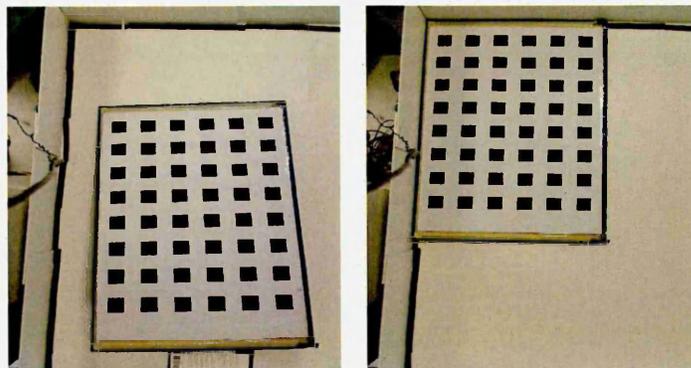


Figure 5.6: Image 11 (left) and image 12 (right)

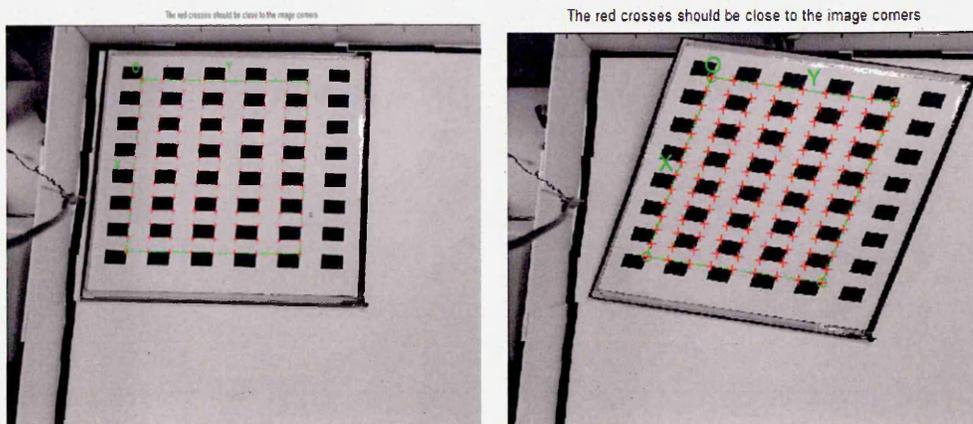


Figure 5.7: After grid points selection: image 1 (left) and image 2 (right)

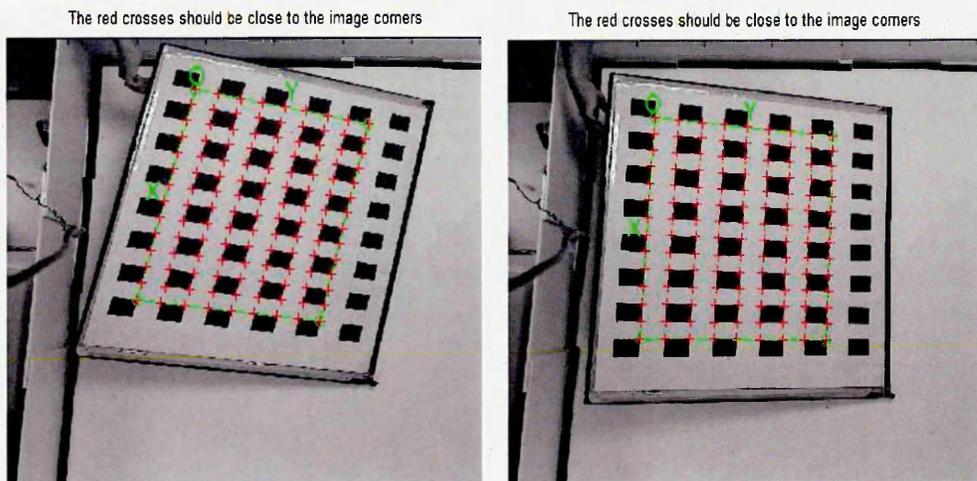


Figure 5.8: After grid points selection: image 3 (left) and image 4 (right)

images obtained after grid points selection, are shown in Figures 5.7 to 5.12. It can be seen that the grid points are selected accurately in all the images.

Finally, the camera intrinsic and extrinsic parameters obtained, after executing the calibration routine, are shown below.

$$A = \begin{bmatrix} 782.2 & 0 & 372.3 \\ 0 & 782.2 & 466.1 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.6)$$

$$B = \begin{bmatrix} -0.0216 & 0.9996 & -0.0171 & -310.5 \\ 0.9952 & 0.0198 & -0.0959 & -589.2 \\ -0.0955 & -0.0191 & -0.9952 & 1286.6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.7)$$

where all measurements are defined in millimetres. To determine the accuracy of these parameters, the 2D grid points were re-projected on the images. The result obtained when these points were re-projected on image 1 is shown in Figure 5.13. It can be noticed that, the re-projection of the grid points is done accurately. Now these parameters were utilised in the *Visual Tracking System*, if the localisation and tracking information were required in the real world coordinates.

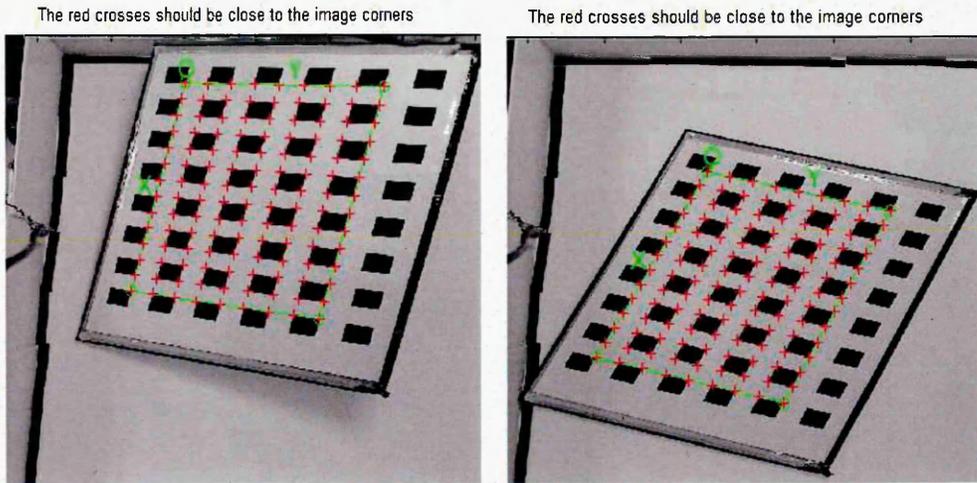


Figure 5.9: After grid points selection: image 5 (left) and image 6 (right)

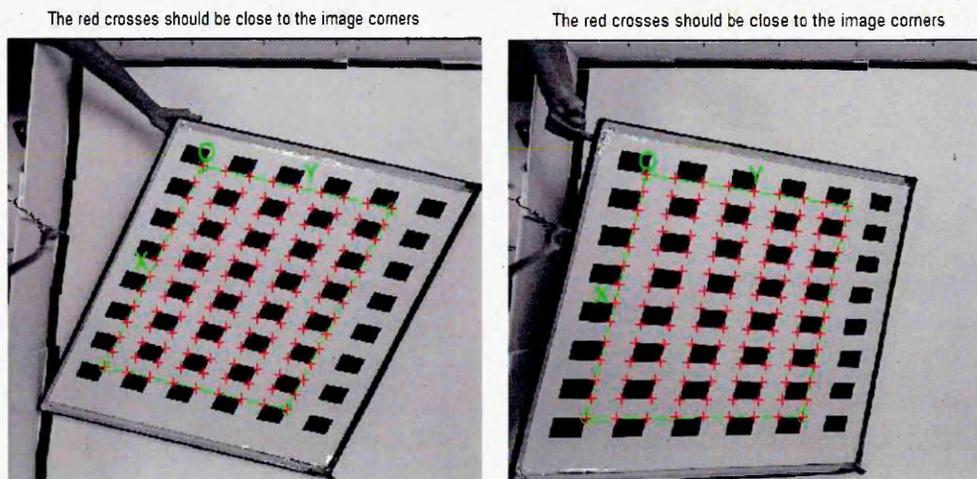


Figure 5.10: After grid points selection: image 7 (left) and image 8 (right)

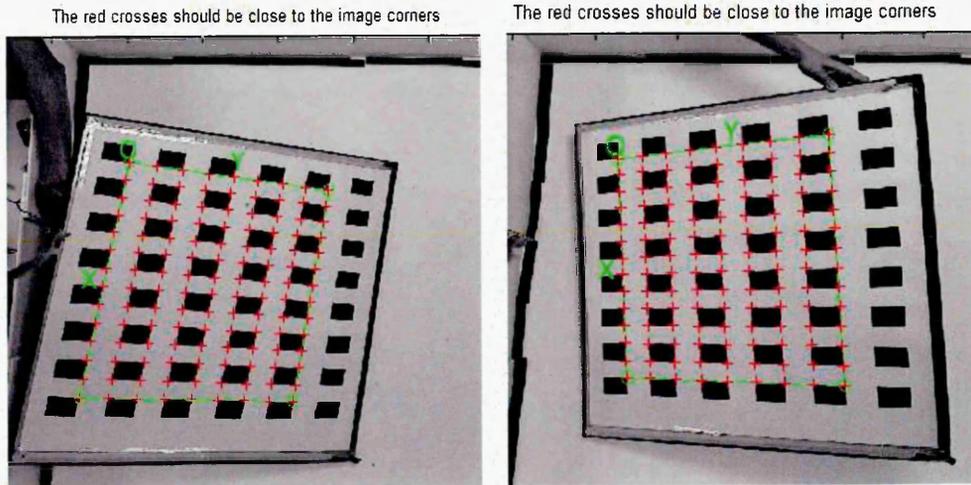


Figure 5.11: After grid points selection: image 9 (left) and image 10 (right)

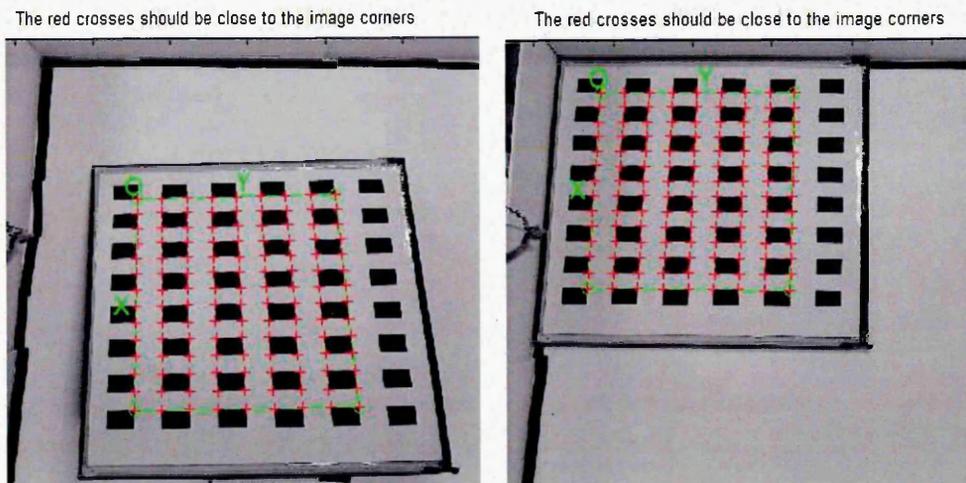


Figure 5.12: After grid points selection: image 11 (left) and image 12 (right)

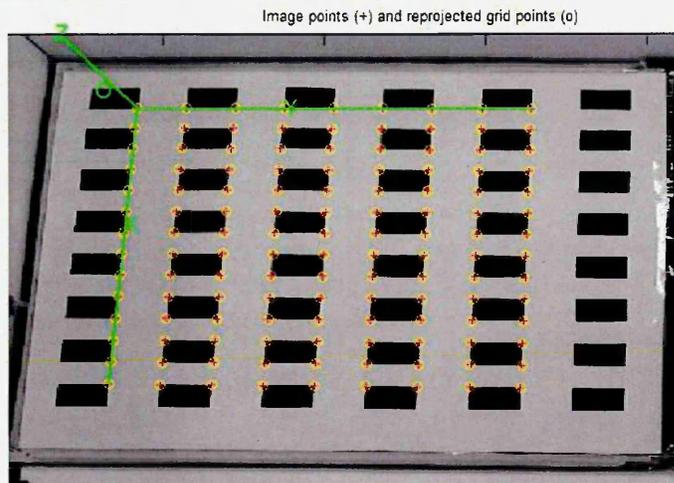


Figure 5.13: Reprojected 2D grid points.

5.2 Visual Localisation and Tracking System

After performing the camera calibration, the second step was to process the camera images and localise the robots. In robot localisation based on passive markers, the most challenging stage was the design of the markers such that they were prominent in the environment and easy to detect. They should code ID information so that multiple robots could be identified and their design should also convey the robot orientation information. The markers used in this research also utilised the colour information based on the fact that colours appearing in certain pattern could be very prominent in the environment. In Figure 5.14, four markers conveying the unique ID information following the same the design template are shown.

In Figure 5.14, it can be noticed that, three colours are used in the design of the markers. One marker, with blue colour cover, is labelled to define the template. Cover colour region defines the boundary of the marker. Cover region surrounds the head region from three directions, but is open from one side of the tail region. This also differentiates the head region from the tail region. Head region is the one which finds the tail region in one direction and cover region in three directions. And tail region is defined as the one which is open from one direction, surrounded by cover region from two directions, but finds the head and cover regions together in one direction. Each marker is the result of colour blobs appearing in a certain pattern. In Figure 5.14, markers using all three colours are shown, but with this approach, markers of same colours can also be designed. With this approach, by using three

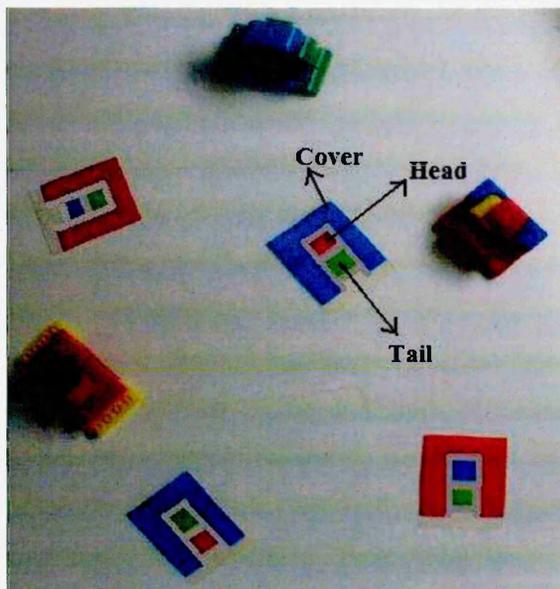


Figure 5.14: Fiducial markers template.

colours together with a simple design template, not only 27 different robots can be identified, but at the same time, it also conveys the orientation information. This approach shows the advantage over other approaches such as the one used in RoboCup [103] which uses the same number of colours, but can uniquely identify only 9 robots. The approach presented here requires the colour image processing techniques in order to detect the colour blobs appearing in a certain pattern. It identifies the markers from which these blobs are resulting and then determines their orientation information. To perform image processing, MATLAB was selected as it provides a good platform for the proof of concept. The procedure adopted to achieve the objective required extraction of colour blobs, blobs statistics and pattern recognition in a designed template. Once marker identification was achieved, then multi-camera based marker tracking system was used to provide visual guidance to the robots. This is explained in the following sections.

5.2.1 Colour Blob Extraction

In the beginning, different colour blobs (i.e. red, green and blue) appearing in the image were extracted. To extract the colour blobs, YUV image format was used. The reason for processing in YUV format is that, it makes the blob extraction algorithm less sensitive to the changes in lighting condition. In order to select the colour blobs,

the threshold was applied to the range of values in UV plane. The result obtained when colour blobs were extracted from Figure 5.14 is shown in Figure 5.15.

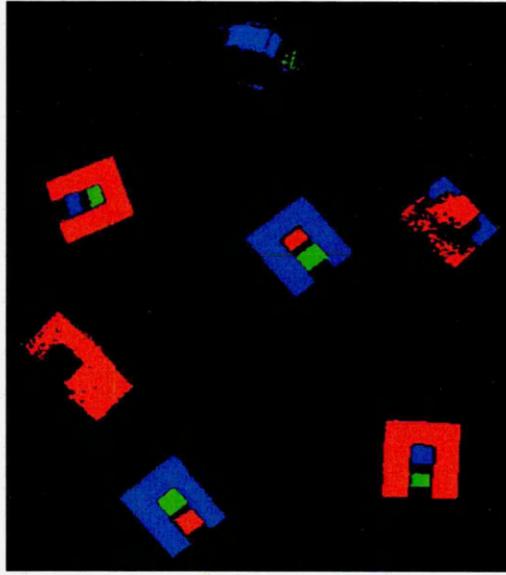


Figure 5.15: Color blobs extraction.

5.2.2 Extraction of Blobs Statistics

In the next stage, segmentation of the resulting colour blobs is performed. In other words, statistical information of each blob was extracted that is, the centroid of each blob and the number of pixels the blobs comprised of. This way, each blob was also assigned a fixed ID. This was to make sure that when the algorithm was searching for the cover blob around the head blob, then it checked whether the blob (which is a cover blob candidate) appeared around the head was all connected (i.e. it has same ID) and was not resulting from different objects.

5.2.3 Template Matching and Pattern Recognition

The localisation system was given a prior knowledge about the identity of the different colour patterns appearing in the form of designed template in the given image. For example, blue cover, red head and green tail blob, this pattern was attached to robot 1. The complete marker detection and identification process is as follows. After segmenting and assigning a unique ID to all blobs, each blob was processed one by one and checked whether it was the one resulting from the head blob. For ex-

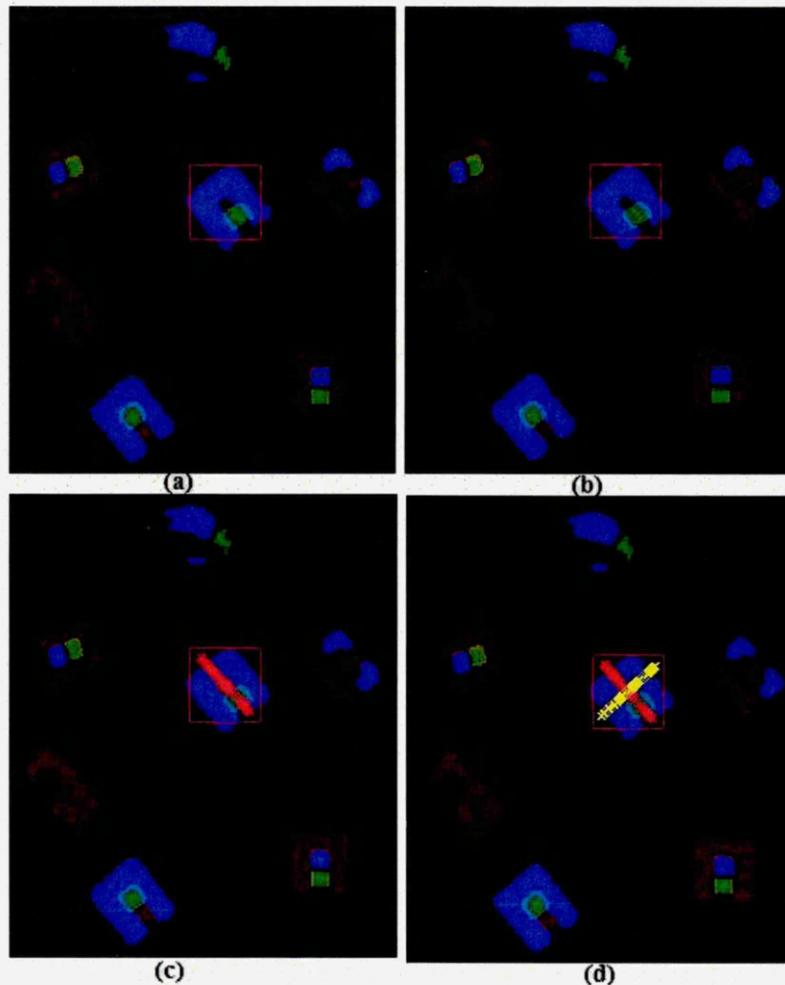


Figure 5.16: Pattern recognition process. (a) Window around the selected blob. (b) Selecting the expected tail blob. (c) Search line from tail to the head blob along which cover blob will be searched. (d) Search line along which cover blob is searched for final validation of the pattern.

ample, if any robot marker had red colour in the head blob, then localisation system considered all red blobs one by one and checked whether they follow the designed pattern. An example showing the step by step process to identify that a marker fitted the design template is shown in Figure 5.16. When red blob was considered as the one resulting from the head region then to make the process of template validation faster, a window (35 pixels square window) was defined around the selected blob (shown in Figure 5.16a). Only those blobs which lied in this window were considered for template validation. From the prior knowledge, algorithm knew that a green tail blob was in a possible set of patterns. In this case, it found the green blob in the search window as shown in Figure 5.16b. The algorithm then determined the slope

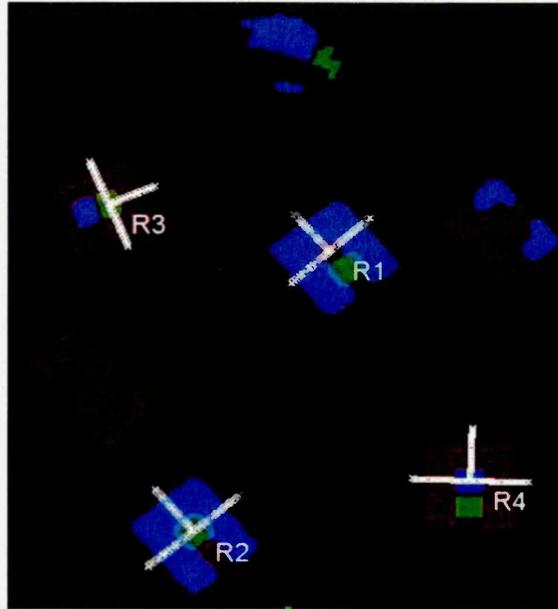


Figure 5.17: Determining the location and orientation of the four robots.

between the head and the tail blob. For this, it required the centroid of the head and tail blob. This is where it utilised the statistical information it had extracted during the blob segmentation process. After obtaining the slope, it drew a search line along which it processed the pixels and searched for the closed cover blob in one direction (head direction) and open cover blob in opposite direction (tail direction). The search line is shown in Figure 5.16c. If algorithm found the expected pattern along the search line in Figure 5.16c (i.e. head was covered by another expected colour blob and the cover was open from the tail blob direction), then it drew another search line perpendicular to the previous search line. This search line is shown in Figure 5.16d. Along this search line, the algorithm searched for the cover blob on both sides of head blob. Algorithm made a check that the cover blob found along the search line shown in Figure 5.16d had the same ID which was found when the search was made along the line shown in Figure 5.16c.

Following this approach, when the complete image was processed, then the four robots identified are shown in Figure 5.17. To identify the orientation of the robots, three white lines were plotted over each identified marker. Two lines spanned across markers width whereas, one white line identified the direction, where robot was facing. It can be seen that, with very simple designed markers, the robot orientation is very accurately identified.

5.2.4 Multi Camera Based Robot Tracking

After identifying the multiple robots in arena using the above defined marker based approach, an algorithm to track their position and orientation in the arena was developed using a multi-camera system. The setup of this multi-camera system and the development of the robot tracking algorithm are explained in detail in the following sub-sections.

- (i) Multi-camera System
- (ii) Robot Tracking Algorithm

Multi-camera System

This section is divided into In the beginning, a single camera was mounted on the ceiling right above the middle of arena, so that the markers placed on the robots could be identified. To grab images, a LogiTech webcam with 90 degree field of view was used. With 90 degrees field of view, it was not possible to cover the complete arena with single camera. If cameras with higher degree field of view are used, then their image covers more area but at the same time they cause fish eye problem which reduces the precision of markers' position detected near the boundary of the image. So to cover the complete arena, two LogiTech webcams were mounted parallel to each other above the arena. As the web-cams were mounted higher (i.e., 5 feet) above the arena, so it was not possible to work with low resolution. With low resolution, the markers appeared very small that is, not enough pixels were contributing to provide information about the markers. Due to this, it was possible that some of the markers left undetected due to the changes in the illumination. So, to overcome this, it was decided to grab images in a higher resolution that is, 960x720 pixels. The way the web-cams were mounted above the arena, is shown in Figure 5.18. Both cameras were connected with a Core 2 Duo processing system. And for image processing, to identify the markers and track the robots, MATLAB was used. As images from both cameras were processed on a system by the same algorithm, so the robots' position information was shared within the algorithm. This way, the two cameras collectively could track the robots.

The complete arena surface was divided into three zones. Camera 1 could view the robots in zones 1 and 2. So, camera 1 could effectively track the markers mounted on the robots in zones 1 and 2. Similarly, zones 2 and 3 were in the

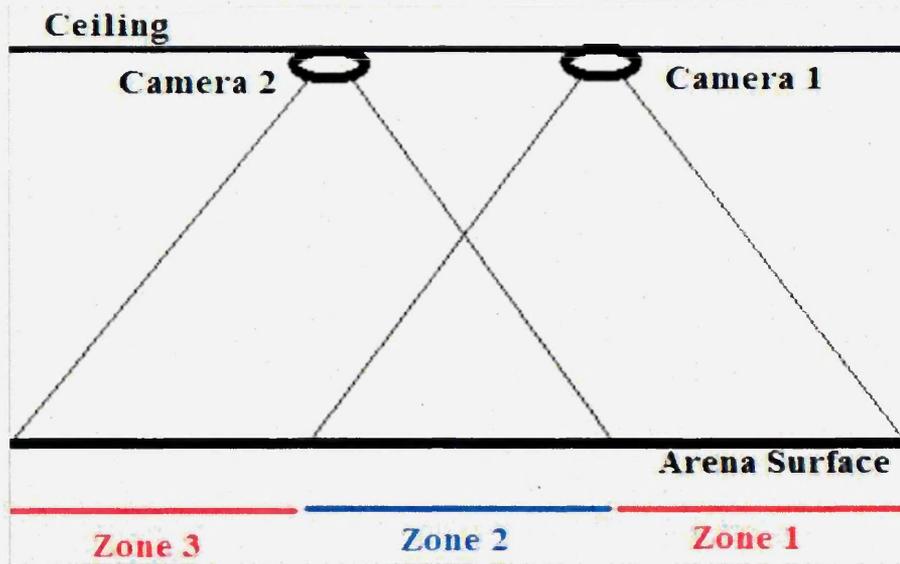


Figure 5.18: Ceiling mounted camera set-up for robot tracking.

visibility of camera 2. Zone 2 is the area which appears in both cameras 1 and 2 view. So, robots appearing in zone 2 were tracked by both cameras 1 and 2. This way, cameras 1 and 2 collectively provided a wide field of view and covered the complete arena. As mentioned before, the cameras were mounted higher above the arena, so to identify the markers placed on the robot, a high resolution image (i.e. 960x720 pixels) was used. Now if images captured by both cameras 1 and 2 were fully processed every time, then the algorithm to identify the markers could be computationally demanding. So to reduce the computational load, the images from cameras 1 and 2 were fully processed only once in the beginning or until the time, when all the expected markers placed in the arena were identified. In Figure 5.19, the images captured by the cameras 1 and 2, when four robot markers were placed in the arena, are shown. All four markers positions and orientations were properly determined. Camera 1 (shown in Figure 5.19a) will be tracking robots 1, 2 and 4 whereas, camera 2 (shown in Figure 5.19b) is responsible to track robots 1, 2, 3 and 4.

Robot Tracking Algorithm

Once all the markers were identified, then the tracking algorithm was developed to track the robots positions in the environment. For the two cameras used in the multi-camera system, the tracking algorithm made a tracking database to store the

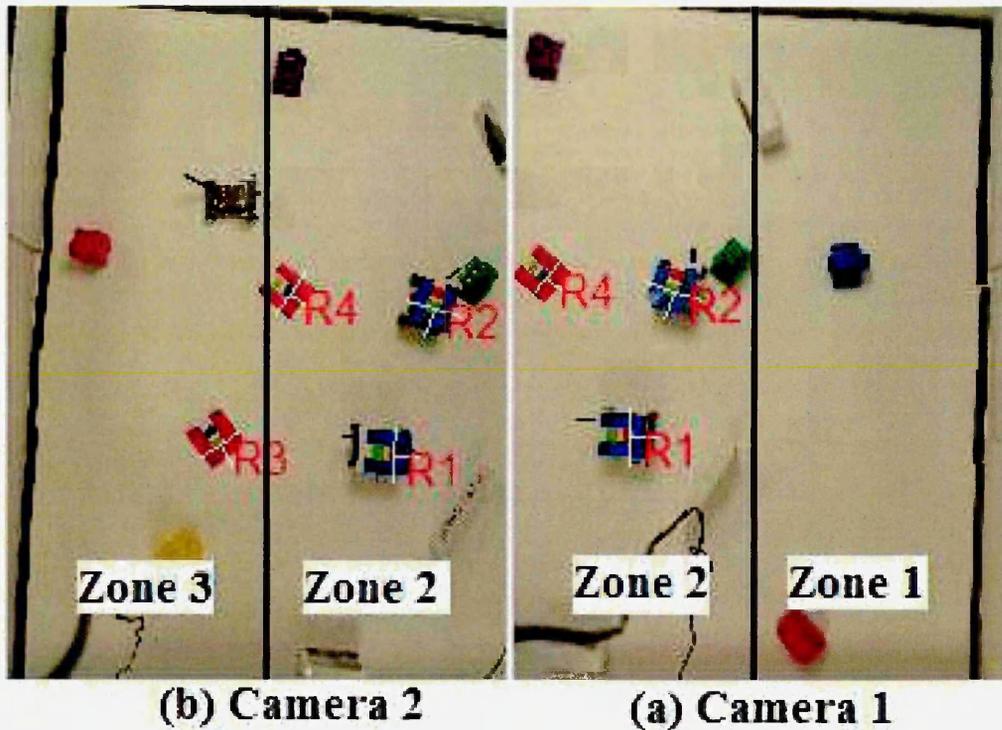


Figure 5.19: Collective tracking of robots from ceiling mounted cameras.

robots positions. All the markers identified in camera 1 image were kept in camera 1 tracking database, and similarly, the markers in camera 2 view, were kept in camera 2 tracking database. After all the robots positions were identified, in the next image frame, the algorithm made a search window (70 pixels wide) around each robot's last identified position. The algorithm expected the robots to appear in that search window. If the robot was not identified in that window, then in the next frame, the algorithm increased the search window size for that robot. In other words, the algorithm increased the uncertainty about the robot's position and increased the search window size. If the robot which was tracked by camera 1 was in zone 1, but now was entering zone 2, then this robot would be entering the shared zone and would become visible in camera 2 view too. So, the algorithm would keep on tracking the robot in camera 1 images, but at the same time, it would also add the robot ID in camera 2 tracking database. Now the algorithm had to determine, where the robot was expected to appear in the camera 2 image. For this purpose, the algorithm solved the Homography[11] between cameras 1 and 2 images. Using the homography, algorithm identified the robot expected position in camera 2 image

and determined that where to draw a search window to track the robot in the coming image frames from camera 2. On the other hand, if a robot moved from zone 3 to zone 2, then tracking database update process was initiated for camera 1. Similarly, the removal of robot ID from tracking database was also important when the robot was moving from zone 2 to either zone 1 or zone 3. If it was moving from zone 2 to zone 3, then it would go out from the camera 1 field of view and the algorithm would remove this robot ID from camera 1 tracking database. And if it was moving from zone 2 to zone 1, then removal of the robot ID would be performed from camera 2 database. The later case is also shown in Figure 5.20. Camera 1 was tracking robots 1, 2 and 4 (as shown in Figure 5.19) and camera 2 was tracking robots 1, 2, 3 and 4. Now robot 1 moves from zone 2 to zone 1. It is no more visible by camera 2 so its ID was removed from camera 2 tracking database.

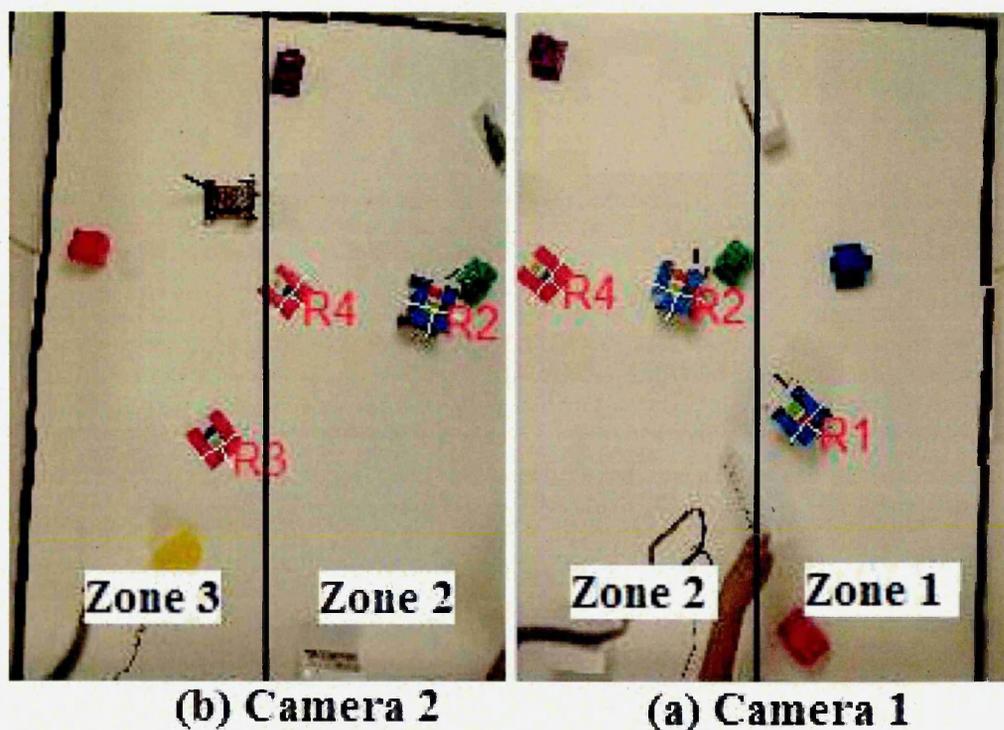


Figure 5.20: Collective tracking of robots from ceiling mounted cameras.

To determine the positions of all the robots on the combine image plane, the following steps were taken.

- Robots appearing in zone 1, their positions and orientations were provided by camera 1 as they were tracked by camera 1 only.

- Robots appearing in zone 3, their positions and orientations were provided by camera 2 as they were tracked by camera 2 only.
- Robots which appear in zone 2, as they were tracked by both cameras 1 and 2, so information from both cameras was fused together to determine the robot's position. The advantage of this shared zone is that, due to light reflections or illumination variations, if one of the camera was not able to identify the marker in zone 2, then the marker position could still be identified using the information from the other camera.

5.3 Multi-Robot Visual Guidance

To visually guide the multiple robots in the arena, using markers based localisation information, initially all the ground surface was determined on which robots could move to reach the target location. In Figure 5.21, the target location for the visually guided robots is shown. To determine the ground surface, Canny edge detection algorithm [120] was applied on the images captured from both cameras. As the arena surface had no texture, so it did not cause any edges in the output image. Only arena boundaries, obstacles and robots caused edges in the output image. As the robot positions were known precisely through the marker detection approach, so the edges appeared around robots positions were simply ignored as they were certainly caused by robot bodies. This way, ground surface was successfully extracted and all the obstacles were identified using the edge information caused by the obstacle boundaries. This provided a ground map with obstacles identified on it. Now the next task was to guide all the robots to reach the same target location without colliding the obstacles in the environment. For this purpose, the shortest distance to the target location was determined in the presence of obstacles. This is a path planning problem. For this purpose, A*[12] path planning algorithm was used. As the ground map was defined in high resolution, so it could be computationally very expensive to process every pixel while searching for the possible shortest path to the target location using A* algorithm. So to reduce the computational complexity, a grid map was defined. This grid map was obtained by dividing the high resolution ground map into 25x25 pixel windows and identifying each window as a single cell which could convey the ground clearance or the obstacle presence. This grid map is shown in Figure 5.22. It can be noticed that, nine obstacles (O1 to O9 in green

colour) are clearly identified on the grid map. As the positions of three robots was precisely known on the grid map, so using A* algorithm, the shortest path to the target location was determined. The shortest path obtained for the three robots is also shown in Figure 5.22.



Figure 5.21: Robot arena used for the visual guidance of the robots.

To test the developed fiducial marker based approach for tracking and guiding robots, tests were performed in which many obstacles were placed in the robot arena such that the robots had to go through complicated way to reach the target. Two of these tests are discussed here. In the first test, before the robots started receiving the guidance commands, the positions of the robots in the arena are shown in Figure 5.23. The paths to target for three robots are also shown in the Figure 5.24. It can be noticed that, robots 1 and 3 have to go through a long path to reach the target because they are surrounded by many obstacles. In case of robot 1, the obstacle around it had some free space in between but this was not enough for the robot to go through, so it is not shown as ground clearance to robot. The detected obstacles were dilated in the grid map image so that the robots passed by their side with safe distance. This is done to avoid collision with obstacles.

It is noticed that, during the operation, some time the robots collided the obstacle when they were passing near to them. This happened because, to avoid robot bodies

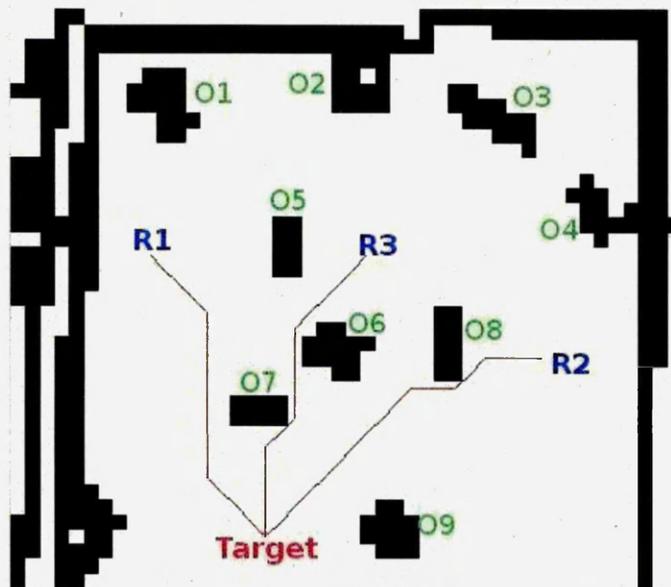


Figure 5.22: Image showing the shortest path to the target location for all the robots on the grid map.

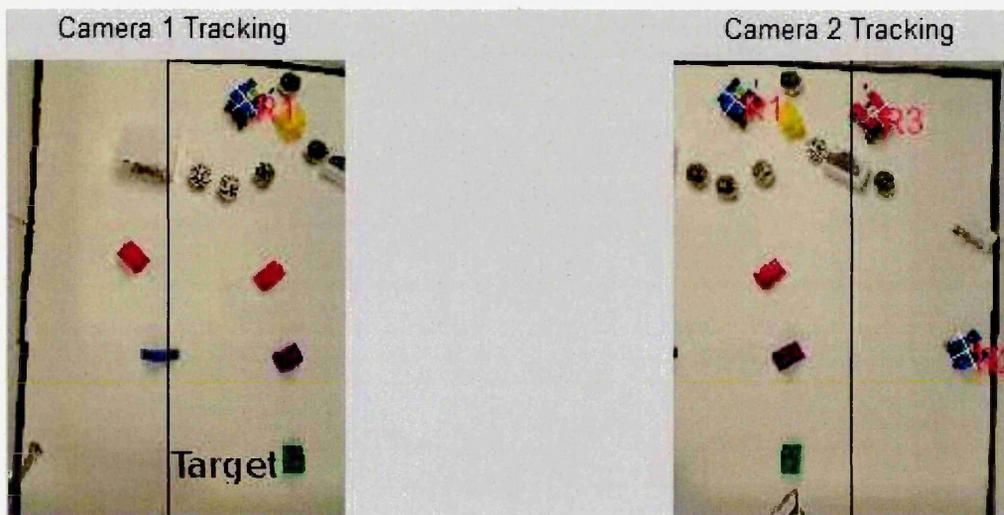


Figure 5.23: Test 1: Positions of the robots in the arena before the test started.

from appearing as an obstacle, a window region was defined around robot detected positions. And edges detected in those windowed regions were considered as the one appearing because of robot bodies, and so were ignored. When robot was very close to obstacle, some time this assumption failed as those edges actually appeared because of obstacles boundaries but were mistakenly ignored. One of this scenario in test 1 is shown in Figure 5.25 when robot 1 collided the obstacle

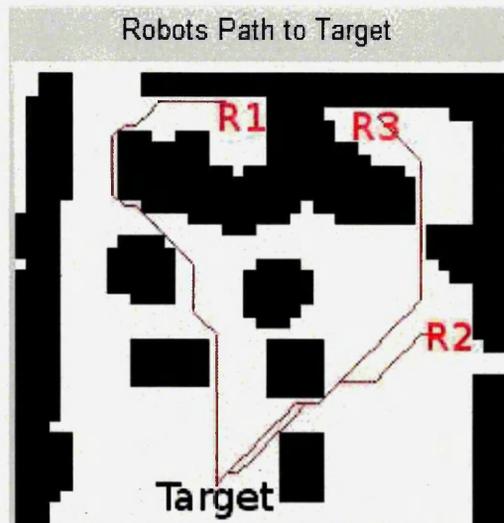


Figure 5.24: Test 1: Robots path to the target location on the grid map.

boundary on its way to target. Finally, the positions of the robots when they reached the target location is shown in Figure 5.26. The robots were gathered around the target location successfully.

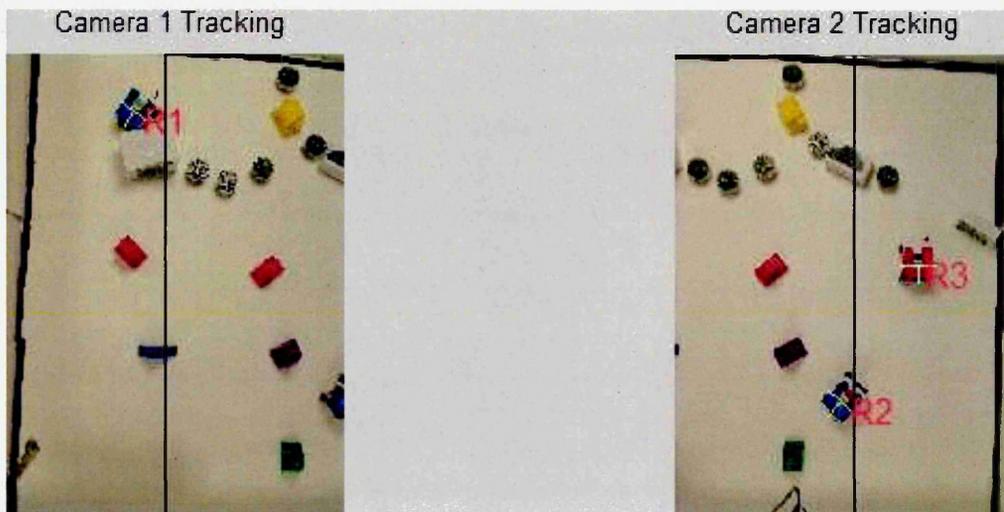


Figure 5.25: Robot 1 colliding the obstacle boundary.

Similarly, in the second test, the positions of the robot before they started receiving guidance and once they reached the target locations are shown in Figures 5.27 and 5.29, respectively. The shortest path to the target, before the robots started moving, is also shown in Figure 5.28. In this experiment, again the robots reached the target locations successfully as shown in Figure 5.29.

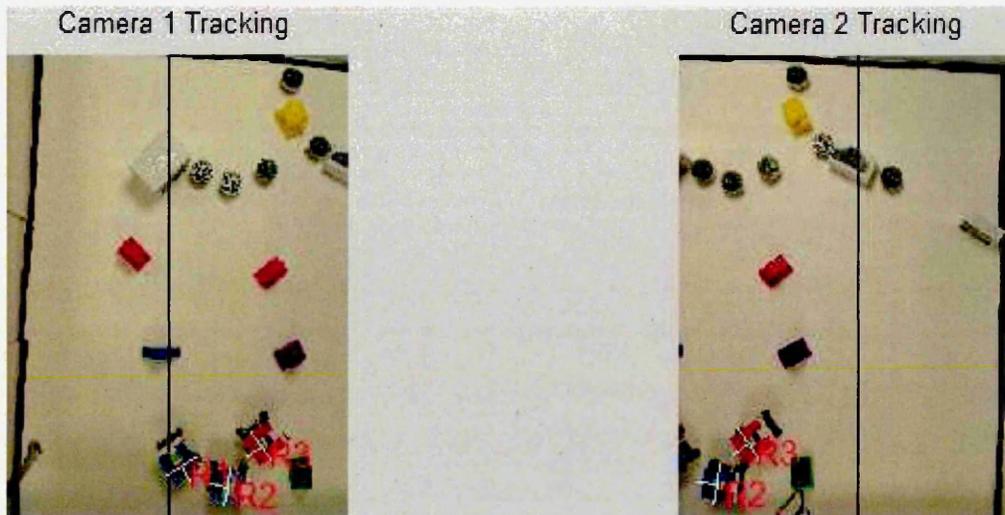


Figure 5.26: Test 1: Robots reached the target location successfully.

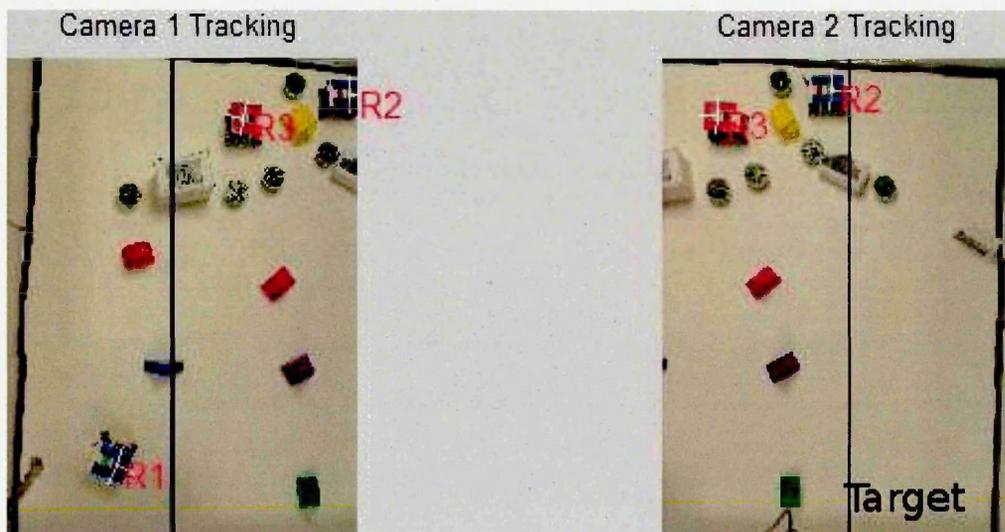


Figure 5.27: Test 2: Positions of the robots before the test begin.

In the presented approach, the path to target was determined every time. This way, if some of the objects moved in the environment, then robot automatically directed a new path to the target location. This on-line path determination some time caused problems for the last robot reaching the target. This happened because the robots which reached first, covered the target location and this caused problems when the path to target was determined for the last robot. Another problem observed was due to the difference in the positions of the robot in the captured image and their actual position. The new commands to the robots was determined on

received commands. This synchronised the camera frame capturing with the control command implementation on the robot. In this way, it could be ensured that before the next frame was captured by the camera, the robots had already changed their positions based on the last received commands.

The results obtained from the described tests proved that the position and orientation information obtained from the developed fiducial markers were precise enough to guide the robots. This solution to robot tracking and guidance is cost effective and can be easily extended to cover more area by simply adding more cameras or by using the cameras with more wide field of view. The handshaking between the two cameras while tracking the robots can also be easily extended to further increased number of cameras. This solution can be made more effective by using smart cameras, which have on-board processing capabilities, together with the wireless network technology to share robot tracking information between the different cameras.

5.4 Conclusions

In this Chapter the development and implementation of a “Visual Tracking System” has been presented. This tracking system is developed to provide the localisation information to a group of robots working in the swarm mode. In the two swarm mode scenarios addressed in Chapter 6, the group of robots obtain their localisation information from the “Visual Tracking System” to precisely localise the target objects in the environment and also to map their environment boundaries. To show the effectiveness of the localisation information provided by the “Visual Tracking System” to the robots, experimental results has been presented in this Chapter, which demonstrate the manner the group of robots reaches the target location without colliding with the obstacles, solely based on the guidance provided by the “Visual Tracking System”. From the successful experiments, it has been concluded that, the robot’s localisation information provided by the “Visual Tracking System” is precise enough to be used for the swarm mode scenarios.

Chapter 6

Distributed Vision Processing in Multi-Robotic Swarm

This chapter is dedicated to the two distributed vision processing scenarios which are performed in the swarm mode. In the first scenario, a group of robots are used to search for the object of interest in the environment and localise them by using their visual information together with the *Visual Tracking System* discussed in Chapter 5. For this scenario, an object recognition algorithm is developed and customised for the target embedded system and made it run efficiently on small size robots with limited memory and processing resources. The *Visual Tracking System* localised the robots and the group of robots recognised the objects. This way, the group of robots and the *Visual Tracking System* localised the objects, collectively. In the second scenario, a novel approach for mapping the environment by a group of robots is discussed. The group of robots extracted very simple visual features, which are used together with the localisation information from the *Visual Tracking System*, to collectively map the environment. To discuss these scenarios in detail, this Chapter is divided into the following three sections. First section describes the communication medium used to share the information among swarm of robots. The other two sections describes the distributed vision processing scenarios in swarm mode.

- Communication among Swarm of Robots.
- Vision Based Object Recognition and Localisation by Multi-robotic Systems.
- Environment Mapping by Distributed Multi-robotic System.

6.1 Communication among Swarm of Robots

Following the discussion in Section 2.1, wireless communication medium was found suitable to exchange information among swarm of robots. To establish communication, the robots need to use onboard communication middleware onboard. The discussion in Section 2.1 shows that the use of such communication middleware is not realisable in swarm robotic systems because these middlewares have very high memory requirements. For this purpose, low level wireless communication routines were developed in this research, and these routines are customised for the target system. It is important to note that the robots have very limited onboard energy resources, and due to this, the wireless communication boards used on the robots have very low signal to noise ratio. Due to this low signal to noise ratio, the use of adhoc communication mode caused severe communication delays which affected the performance of the overall system. To overcome the problem of low signal to noise ratio, infrastructure communication mode was preferred over the adhoc mode. The use of a wireless access point in infrastructure mode increased the range of the swarm by retransmitting signals at a higher power level; hence reducing the chances of data loss during transmission. For reliable data communication, the TCP communication protocol was used which works at transport layer. To avoid overloading the communication network, it was decided to share visual features which encode low density information. As the vision sensor generates huge amounts of data, so the selection of visual features with low density information was critical. The selection of appropriate visual features, which provide sufficient information to achieve the target objectives, is addressed in detail in Sections 6.2 and 6.3. To transmit these visual features, the data packets with maximum size of 400 bytes, were used. The use of small size data packets, low density visual features and TCP communication protocol over reliable infrastructure communication mode made it possible to achieve the efficient data exchange between swarm of robots, to successfully achieve the objectives of swarm mode scenarios.

6.2 Vision Based Object Recognition and Localisation by Multi-Robotic Systems

Object recognition is an essential element of robotics. In most robotic applications, the robot or group of robots are required to look for and recognise the objects of in-

terest in the environment to achieve given objectives. To perform object recognition, the integration of vision sensors in robotic applications has provided many solutions. In robotic applications, the need for object recognition arises when a robot has to perform a given manipulation task and before that it has to recognise the object of interest in the environment. It may also be necessary when the robot has to localise itself in the environment with respect to some specific landmarks which it has to recognise. Hence, it may be thought of as the basic functionality required for advanced robotic applications. The object recognition functionality is considered as one of the most challenging problems in computer vision as it presents several challenges such as, view point changes, intensity variations, occlusions and background clutter. Additionally, the provision of this functionality in mobile robotics applications introduces an important challenge given by the constraints for execution time (i.e. computational complexity) [113]. In real world scenarios, robots are normally equipped with high processing systems, so that they can fulfil the real time execution demand of the vision based object recognition algorithms. For such high performance robots, the choices of object recognition techniques may be large, and the number of vision based object recognition techniques developed for offline processing may be simply used with less changes. But when it comes to swarms of small robots, which come with very limited memory and processing resources, the task becomes more challenging.

Over the past few decades, many object recognition techniques have been developed. Some of them are computationally less expensive, such as content based approaches (i.e. using colour, texture and shape) and geometric approaches (i.e. using affine, projective or euclidean transformation to account for the appearance variation of the object of interest) but are sensitive to changes in lighting conditions. Other approaches are considered more computationally expensive, such as context based and appearance based approaches (e.g. using SIFT, SURF or PCA feature descriptors) but are found more flexible and show in-variation to changes in scale, rotation, skew and lighting conditions. Most recent efforts are centred on appearance-based approaches [114]. As these approaches are computationally expensive, they are suited for either offline image processing or can be considered applicable to robots with high processing systems, as described in detail in Section 2.5 of Chapter 2. In spite of the excellent results achieved with these recognition techniques, when it comes to small robots or groups of small robots working in a multi-robotic environment, the bottle-neck of the slow rate of information processing

forced the researchers to make huge compromises with the recognition performance and switch to computationally less expensive algorithms. For example, shape based recognition and blob detection based techniques have been used to recognise simple objects in a given environment. The research work presented in this Chapter, bridges the gap between these computationally expensive, but efficient object recognition algorithms and their applicability on small robotic systems and suggests some techniques following which, the advantages of computationally expensive algorithms can be enjoyed in small robotic systems for recognition purposes.

In the current implementation, the focus is on efficiently applying appearance based approaches to object recognition using SURF (Speeded Up Robust Features) features using a group of small robots and show how multiple robots collectively look for 3D and 2D (i.e., images of objects) objects of interest in an unknown controlled environment. In Figure 6.1, a swarm of robots is shown. The objective is to send these robots in the environment in random directions. These robots will go around, and using the developed object recognition techniques, they will look for the objects of interest (e.g. shown on the right side of Figure 6.1) in the environment. In this Figure, some obstacles are also shown in the environment. To avoid colliding with these obstacles, the robots will also be performing vision based obstacles avoidance (discussed in Chapter 4) in parallel to the recognition task. For recognition purposes, all the robots will be given a common visual vocabulary to use to identify the required objects. Here, the task of vision based recognition to search for objects of interest is distributed among all the robots, but on equal basis (i.e. all robots are given the same task). To help achieve this task efficiently, all robots will also be sharing their knowledge with other team members. They will be communicating with each other and will be updating each other about the progress information that is, what objects have already been found and what they are still looking for. This way, if a place is found by one robot, then the other robots will not be searching for this place unnecessarily and this will avoid effort duplication, showing a collective effort by the group of small robots to recognise the objects in the unknown environment.

Once a robot in a swarm, successfully finds an object of interest in the environment, then apart from telling the other team members about it, the robot also shares this information with the *Visual Tracking System* developed in Chapter 5. For establishing a connection between the robots and the tracking system, a Server is shown in the Figure 6.1. This Server is responsible for running the *Visual Tracking*

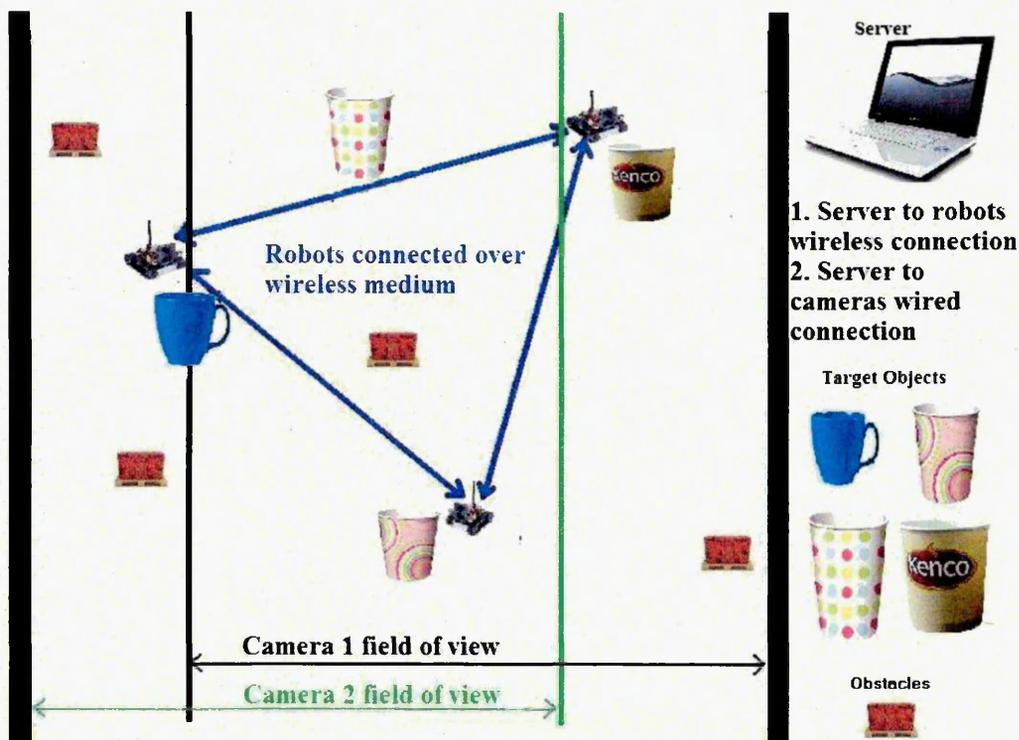


Figure 6.1: Scenario where group of robots performing search operation to find the object of interest.

System and sharing the localisation and tracking information with the robots. As in the tracking system, two ceiling mounted cameras collectively track the robots, so the area of the robot arena which is covered by the two cameras is also shown in Figure 6.1. Now during the operation, when the robot recognises the target and shares this information with the tracking system, then the tracking system locks the robot position in the image, localises it and sends this localisation information to the robot. This information contains robot's position in the arena and its orientation (i.e. heading). When the robot receives this localisation and heading information, then it determines its distance to the the recognised object. Finally the robot localises the object with reference to its own position. For this purpose, it uses the "distance to the object" information together with its position and heading information. After localising the target object, the robot shares the object ID and its location information with the Server which later displays the location of the target objects in the environment.

From the scenario description, it can be seen that the performance of the object recognition approach on the small robots plays an important role for successful op-

eration. So, an aim of this research is to use the advantages of powerful appearance based recognition approaches and make them run efficiently to perform object recognition using a group of small robots. Following the literature review in Chapter 2, the SURF feature based recognition approach was found to be the fastest to compute and appeared to be more favourable for implementation on an embedded system. Gradient based feature extraction [60] and Harris feature based approaches [65] were also fast for performing feature extraction but they did not provide features which could be as robustly detected as carried out by SURF based approaches. There are a number of open source implementations of SURF feature extraction and matching algorithms. OpenCV [97] is an open source computer vision library which provides one possible implementation of the SURF algorithm. Another open source implementation of the SURF algorithm is found in OpenSURF [98] library which is a faster and a better optimised implementation of SURF when compared to OpenCV. In this research, it is decided to use the OpenSURF library as a reference to the SURF implementation for performing appearance based recognition tasks. The target hardware used is also an important factor as it strongly influences the method adopted to solve the problem at hand. As mentioned in Chapter 3, for the swarm mode scenarios, a group of SRV1 robots by Surveyor Corporation were used with Blackfin BF537E processor on-board. uClinux (micro controller Linux), which is a popular operating system customised for embedded systems, was used as the on-board operating system. Code compilation was performed using GNU cross compilers on a Linux based development platform. At the beginning, the OpenSURF library was cross compiled and ran on the target blackfin processor. The image resolution was set to 320x240 pixels. The execution time calculated to process one single frame was 33sec. The reason for this was twofold, one is the computationally expensive nature of the algorithm and second is that the algorithm performs many floating point operations and the target blackfin processor lacks the floating point unit (FPU). To reduce the execution time and also to increase the performance of the algorithm to recognise the objects lying far from the robot, the following optimisation operations were carried out.

- Processor specific optimisation to reduce execution time.
- Image pre-processing to reduce the amount of data to process.
- Multi-resolution analysis.

6.2.1 Processor Specific Optimisation

To perform processor specific optimisation, the SURF algorithm was coded such that it exploited the architectural advantages of the target embedded system. For example, the Blackfin is a fixed point processor, so the floating point operation should be avoided as much as possible. In SURF algorithm, where it was necessary to make use of floating point operations, the use of fast floating point emulation library [119] was made. To perform fixed point operations on blackfin processor, there are further limitation posed by the uClinux operating system as it allows 1.31 fixed point operations only. In 1.31 fixed point operation, there is 1 integer bit and 31 bits are used as fractional bits. The data which can be represented using 1.31 fixed point format lied in the range of -0.9 to +0.9. This limitation of data representation made it necessary to normalize the fixed point data at every point in the program flow in order to guarantee that it lied in the allowed range, otherwise erroneous results can occurred.

In general, Blackfin code optimisation can be done in three different phases i.e. Compiler optimisation, System optimisation and Assembly optimisation [99]. Following these optimisation steps, the improvement in the algorithm's execution performance achieved is shown in Figure 6.2. When no optimisation was applied, the program took 33sec to process a single frame. After performing the compiler optimisation (i.e. using "fast-math" and "mfast-fp" floating point libraries for Blackfin processors) the execution time reduced to 10sec per frame. The fast-math library allows the compiler to use faster hardware floating point instructions, at the potential expense of IEEE floating point compliance. If the program does not need strict compliance, the use of "fast-math" library increases the performance of floating point operations. The "mfast-fp" library further relaxes some of the IEEE floating-point standard's rules for checking inputs against Not-a-Number (NAN), in the interest of performance. As Blackfin processor lacks FPU, so "fast-math" and "mfast-fp" libraries performs floating point emulation. To reduce the execution time further, the portion of the code which was costing more time was identified and customised by exploiting the fixed point architecture of the Blackfin. Therefore, 1.31 fixed point operations were adopted in place of floating point operations. This helped in reducing the time to 3 sec per frame. Optimising the data flow helped in reducing the time to 2.8sec per frame. A further reduction in time was achieved by scaling down the image resolution, by a scale of 2. The last step reduced the time, but

affected heavily the recognition performance. From Figure 6.2, it can be seen that the processor specific optimisation has performed a significant role in the reduction of the execution time, although it needed to be reduced further.

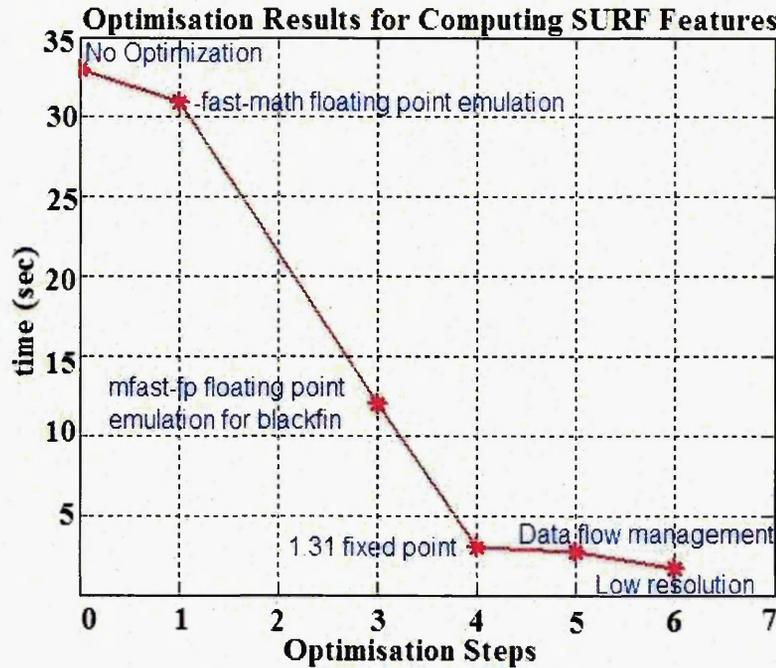


Figure 6.2: Processor specific optimisation to reduce execution time.

6.2.2 Image Pre-processing

In the second stage, image pre-processing was performed. The objective of image pre-processing stage was to identify the pixels in the image which defined the objects. The intensity values of only these identified pixels contributed to the calculation of SURF features. This avoided applying the SURF algorithm on the part of the images which represented plain surfaces. These plain surfaces could result from the smooth ground surface, or the parts of the objects which did not show any intensity variation. It is to be noted that detection of the SURF features strongly relied on the intensity variations in the image. So processing the smooth parts of the images with the SURF algorithm increased the computational time, but did not provide any reliable features. Avoiding plain surfaces in the image, to be processed by SURF algorithm, reduced the processing load by a significant amount. To identify the image pixels which defined the objects, a light weight feature extraction technique

(i.e., in this study the Harris feature extractions technique) was applied. At the beginning, the images were divided into top and bottom portions. The top portion, separated by a thick blue line (shown on the right column of Figure 6.3), always lied outside the arena and was discarded. The bottom portion of the image was processed with the Harris algorithm. If the edges resulting from an object's boundaries were found very near to the bottom of the image, this indicated that the robot was very close to the object. In this case, the pixels identified by the Harris features in the complete bottom portion of the image were used by the SURF algorithm for the object recognition. If the edges resulting from the object boundaries were detected far from the bottom of the image, then it was expected that the objects were not present very near to the robot. In this case, the complete bottom portion of the image was further divided into three portions that was middle, left and right (these portions were also separated by a thin blue line as shown on the right column of Figure 6.3). When these three portions were processed by the Harris algorithm, the extracted image features are also shown on the right column of Figure 6.3. The centroids of the feature points were computed from the middle, left and right portions and windowed images were extracted (identified by red boundary). These windowed images were likely to contain objects in the image and were further used to extract SURF features. There were two main reasons for splitting the bottom portion of the image into further three portions. The algorithm first processed the objects detected in the middle portion. If the object of interest was recognised, then it avoided processing the objects detected in the left and right portions. This conditional processing of the objects detected in left and right portions made the execution of recognition algorithm faster as the algorithm was not required to process the middle, left and right portions every time. The second reason for dividing the bottom portion into middle, left and right portion was that, if the objects detected in any of these portions were recognised as the objects of interest, then it provided simple directional cues to the robots. Then, the robots could turn in the direction where the objects were recognised so that the recognised objects appeared in the centre of the image and the robots could get close to them.

6.2.3 Multi-resolution Analysis

The SURF feature extraction and matching technique works in two stages: training and recognition. In the training stage, the features of the target object were

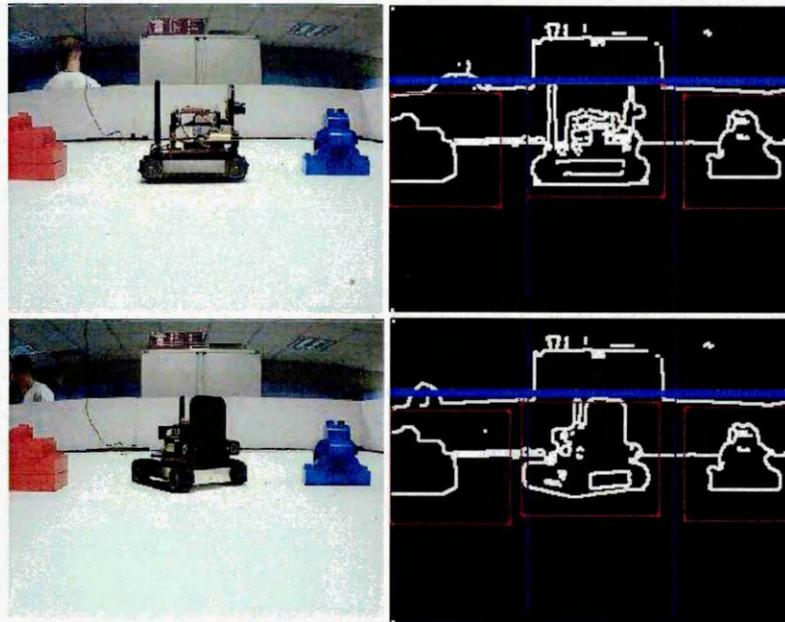


Figure 6.3: Image pre-processing to reduce the amount of data to process.

extracted and kept in memory. In the case of more target objects to recognise, a database containing features, resulting from all the target objects, was generated and kept in memory for recognition purposes. In the case of 2D object (i.e., image of object) recognition, only a single object image was required during training stage for extracting SURF features. If the target object was a 3D object, then pose based feature extraction was performed. That is, the images from different poses of the target object were taken and SURF features were computed for all of these images. During the recognition stage, features from all these images were compared with the features extracted from the current view and the best match provided the information about the objects and also the direction from which the robot was heading towards the object. This pose based recognition for 3D objects increased the database size by a big factor, but keeping the resolution low during the training stage helped in reducing the database size. To make the recognition technique scale invariant, the SURF algorithm generated the scale-space image pyramid, where the input image was iteratively convolved with the Gaussian kernel, and at the same time, the image was sub-sampled iteratively (this reduced the size of the image) [98]. The scale-space image pyramid is also shown in Figure 6.4.

During the training stage, if the image resolution was 320×240 pixels and the number of times the image was sub-sampled is set to 4, then in the image pyramid,

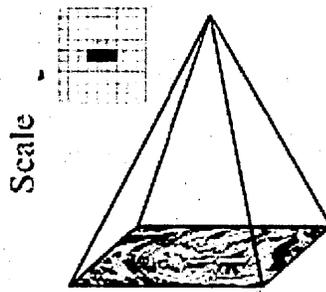


Figure 6.4: Scale-space image pyramid.

the sequence in which the resolution was down sampled was 320×240 , 160×120 , 80×60 and, finally, 40×30 pixels. As the target embedded system had a limited memory and processing resources, the training was given in 320×240 pixels resolution so that the resultant features database was smaller in size and could be kept in the memory for recognition purposes. With this resolution, if the objects lied close to the robot then it could be recognised, but increasing the distance made the recognition difficult, because the object appeared really small in the 320×240 pixels image. To overcome the problem of recognising the objects lying far from the robot, a multi-resolution analysis was performed. The distance to the objects was measured in the image with 320×240 pixels resolution. The objects which were lying near were processed in low resolution and for the objects, which were detected far from the robot, their position was determined in the high resolution image and windowed image was extracted from the higher resolution image. This way, the number of pixels, defining the far lying object, increased and made the recognition possible. In other words, to increase the recognition performance, the objects detected in the image were processed in different resolutions, depending upon their distance from the robot. This idea is explained in Figure 6.5. The two objects on the left and right side were placed close to the robot vision system. The windowed image was extracted from the lower resolution image (i.e. 320×240 pixels) for the SURF features extraction and matching purposes. The object in the centre of the image (i.e. another robot) lied far from the robot, so higher resolution analysis was performed and windowed image was extracted from the high resolution image. The high resolution windowed image extracted for the central object is shown on the right side of Figure 6.5.

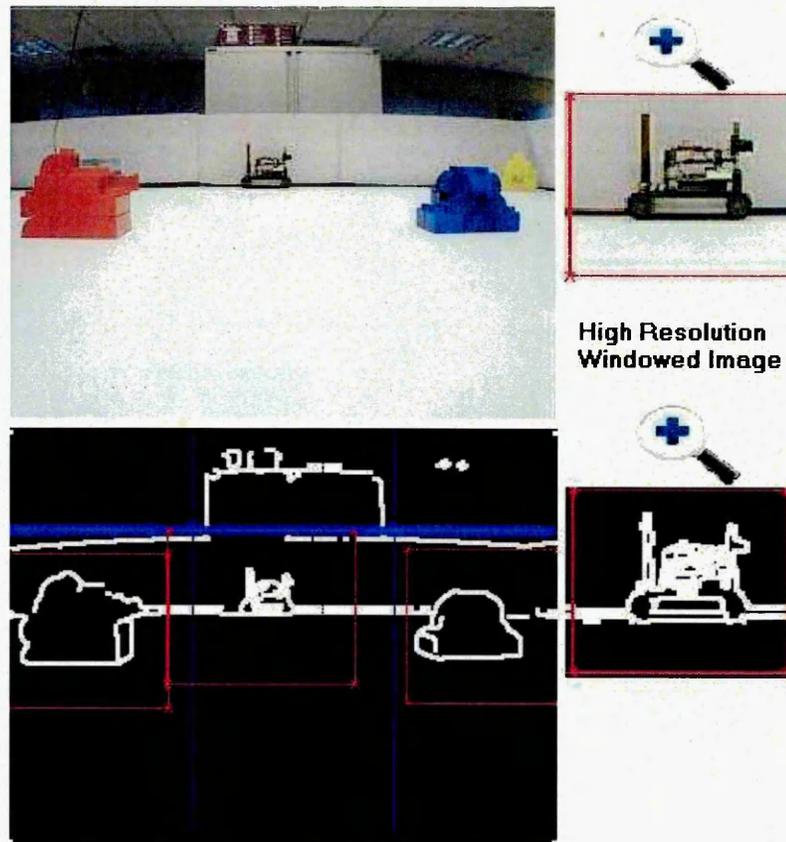


Figure 6.5: Resolution switching based on distance to object.

6.2.4 Experimental Results

The experimental results section is divided into three parts. As mentioned, the performance of the distributed vision processing scenario relies strongly on the reliability and the execution performance of the target object recognition algorithm. So in the first part, the recognition results obtained will be described. The second part demonstrates the experiment in which swarm of robots perform a collective search operation for the objects of interest in the environment. The third part demonstrates the experiment in which the swarm of robots also get localisation support from the *Visual Tracking System*. In this experiment, the robots not only recognise the target objects in the environment, they also try to localise them.

Recognition Results

To show the performance of the developed object recognition algorithm, first of all a comparative analysis with the reference SURF recognition technique was performed. The performance in terms of execution time and recognition was recorded. To test the execution time, ten experiments were performed where in each experiment a robot was trained to recognise a different object. The execution time obtained from reference SURF based approach and the optimised SURF based approach are shown in Figure 6.6. The SURF based approach, on average, took 35 seconds to recognise an object, whereas the optimised SURF approach took 780 milliseconds. It can be noticed that in the fourth experiment, the SURF approach took 39 seconds. This is because, the object in the image had more features as compared with the objects used in other experiments. A rise in execution time (i.e. 890 milliseconds) was also noticed in case of optimised SURF approach.

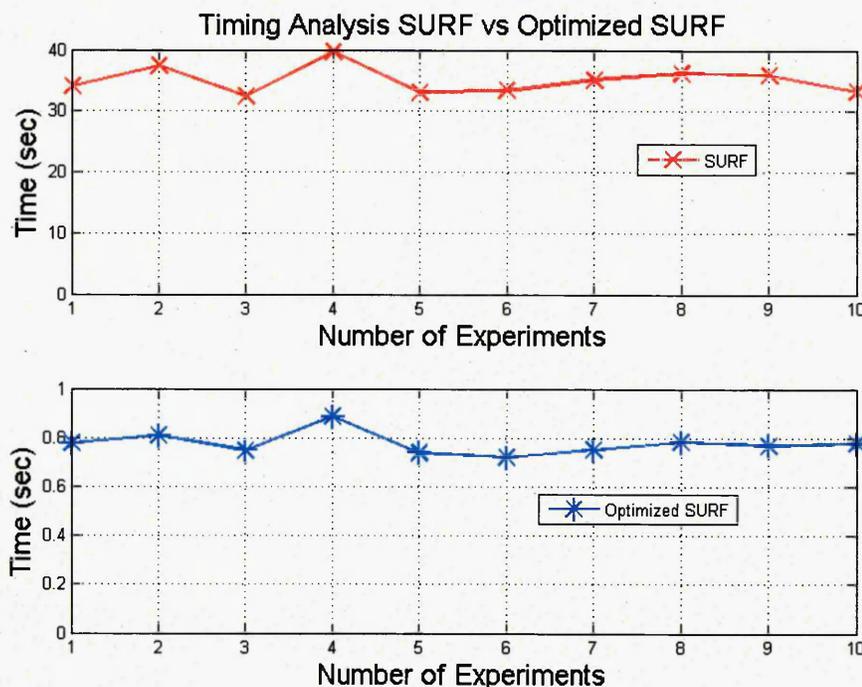


Figure 6.6: Execution timing - SURF vs Optimised SURF.

To check the recognition performance, rigorous testing was performed in comparison to the SURF based approach. The use of “distance based resolution switching technique” was expected to outperform simple SURF based approach in terms of

recognition performance with respect to the distance from the object. For training purposes, QVGA (Quarter Video Graphics Array) resolution was selected. The robot was given training, while keeping the object at distance of 20cm from it. Recognition experiments were performed while moving the robot 5cm away from the object each time. For each robot position, one hundred tests were performed so that the recognition percentage could be obtained. The recognition performance obtained with reference to the SURF based approach is shown in Figure 6.7. At distance of 20cm from the object, both approaches gave 100 percent accuracy. In case of the SURF based approach, the recognition performance degraded gradually as the robot moved away from the object and at distance of 55cm, recognition dropped below 70%. This happens because after 55cm distance, the object appears very small in the QVGA image i.e. very small number of pixels are defining the object and this effects the recognition. From 70cm distance onward, recognition was not possible. In case of Optimised SURF based approach, the use of "Resolution Switching Technique" determined that the robot was far from the object and it enabled the robot to process the object in the high resolution image. As a result, a sudden increase in the recognition performance was noticed when the robot was placed more than 50cm away from the object. This happened because the SURF based recognition relies strongly on the number of features extracted from the image. When the windowed image was extracted from the high resolution image, then it gave a very detailed description of the object, and hence, it produced more features and increased the recognition performance. With the use of "Resolution Switching Technique", the optimised SURF approach was able to provide reliable performance even at the distance of 2.5 times the distance at which SURF was able to perform the recognition.

Distributed Search Operation by Swarm of Robots

In the first experiment, a test was planned in which the task of recognising a set of objects was distributed among two robots. The training was given to recognise three 2D objects and one 3D object in the environment. Out of three 2D objects, one was a plain text on a sheet of paper (i.e. "Replicator") and the other two objects were images of buildings. They are shown in Figure 6.8.

As appearance based recognition technique relied on the SURF features, it was possible to recognise the text and building images providing that these images pro-

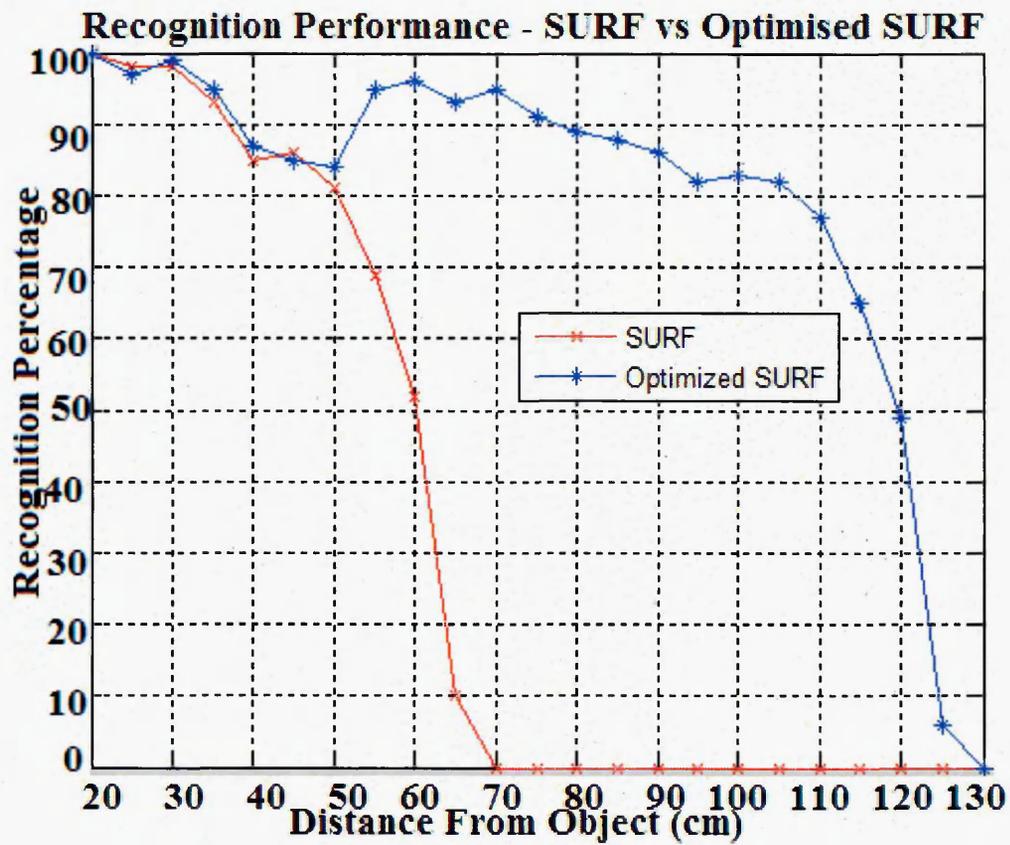


Figure 6.7: Recognition rate - SURF vs Optimised SURF.

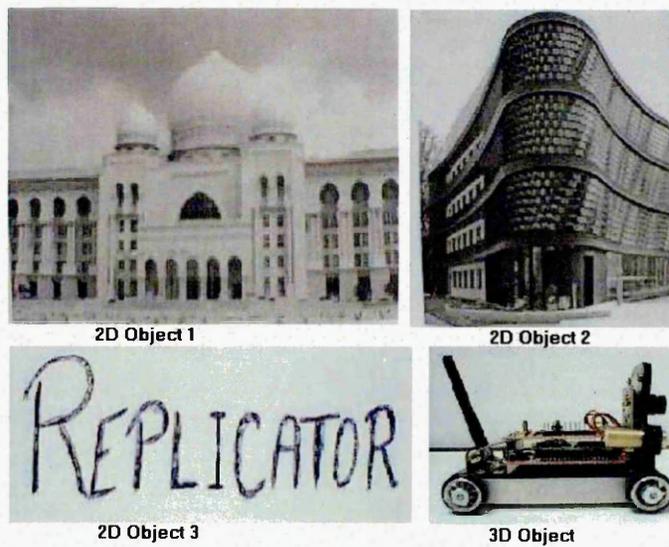


Figure 6.8: Objects used in training to extract SURF features.

duced enough features during the training stage. For the 3D object, another robot was selected and is also shown in Figure 6.8. For 2D objects, a single image was used to extract the SURF features during the training stage. But for the 3D object, pose based recognition was performed. Images from 16 different poses of the robot were taken and their SURF features were extracted during the training stage. The 16 different poses of the robots, which were used for training are also shown in Figure 6.9.

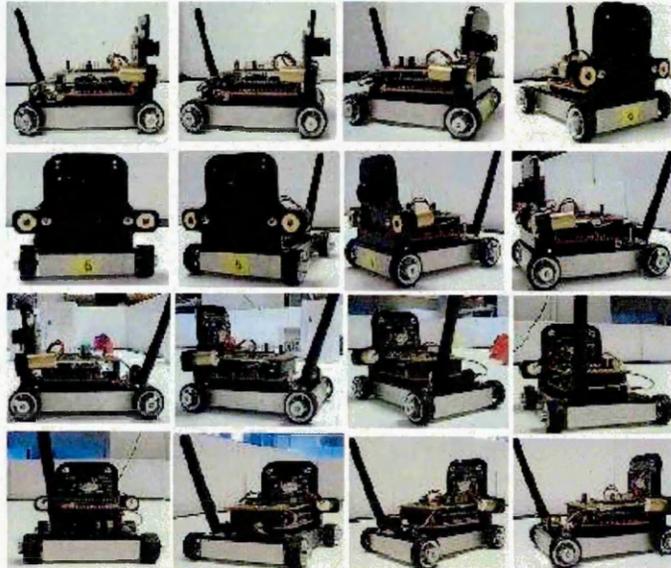


Figure 6.9: Images from 16 poses of the 3D object.

The features extracted from the 2D objects and the 3D objects were combined to form a library which provided information about the object's identity and also the pose information, in case of the 3D object. This library was uploaded to the two robots memory and provided visual clues to both robots about the objects to recognise in the environment. The two robots were programmed to move randomly in the unknown structured environment and search for the objects of interest collectively. To perform the collective operation, a communication medium was also established between the two robots. The robots could share information about the number of objects found and which they were still looking for, over a wireless network. When one robot found an object of interest, it informed the other robot to remove this object from the search list so that redundancy of searching the same object by two robots could be avoided. The placement of the objects of interest and the two robots in the test arena is shown in Figure 6.10.

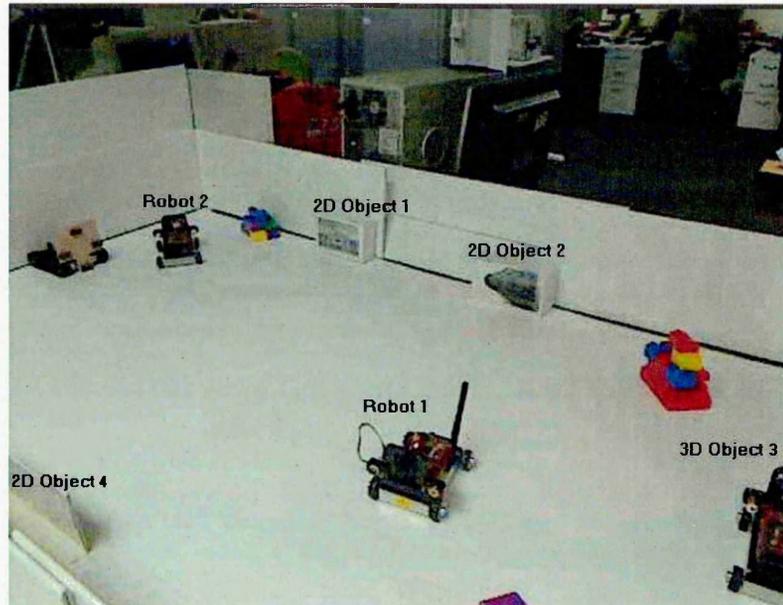


Figure 6.10: Position of robots and objects of interest before experiment is performed.

In this experiment, while collectively searching for the objects, robot 1 found the 2D objects 1 and 2, and 3D object 3. Robot 2 found 2D object 4. The positions of the robots, when they communicated the presence of the objects to their team members, are shown in Figure 6.11.

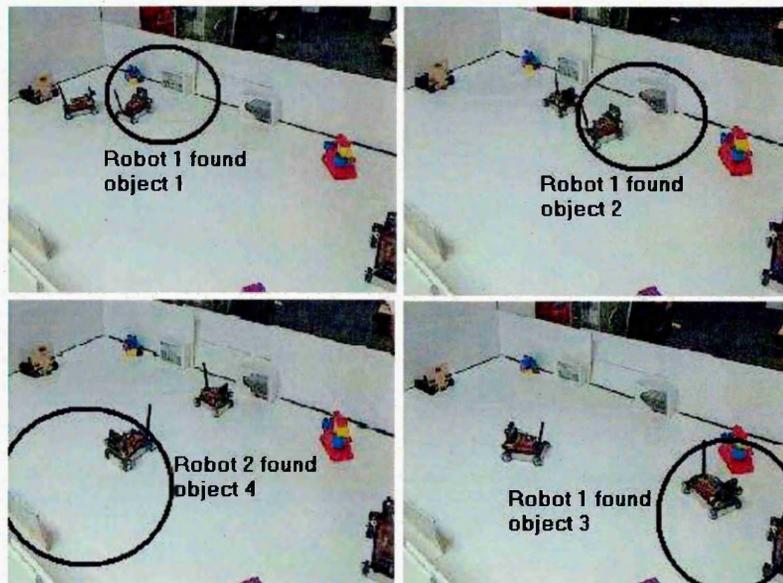


Figure 6.11: Position of the robots when they recognise the object in the environment.

During this experiment, robot 1 missed the 3D object in the first attempt and successfully found it in the second attempt. The sequence of robot 1 positions when it missed object 3 (i.e. 3D object) are shown in Figure 6.12. In part a, robot 1 detected object 3. Then, it was required to get close to object 3 to confirm its presence. In part b, robot 1 got close to object 3. Object 3 was detected on its right side, so in part c, robot 1 turned right. After detecting the object in part c, the robot went straight towards the object. But it moved more in the forward direction such that it left object 3 on its left side undetected. Now, in part d, only a small portion of the 3D object was in its field of view and this was not enough for recognition. Similarly, robot 2 also missed object 1 after detection. This happens because of the

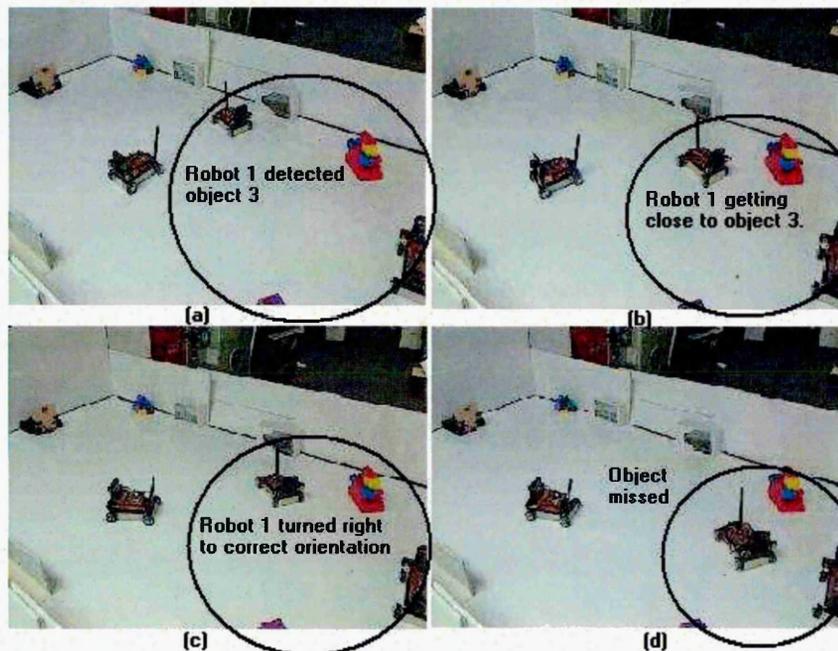


Figure 6.12: Sequence of Robot 1 positions when it detected and then missed object 3.

interruption from the robot 1. The sequence of robots 1 and 2 positions, when robot 2 missed object 1, are shown in Figure 6.13. In part a, robot 2 detected object 1, while robot 1 was also nearby. In part b, robot 1 also detected object 1 on its right side. In part c, robot 1 corrected its orientation towards object 1 and robot 2 moved towards object 1. In part d, robot 1 moved towards object 1, but it also moved between robot 2 and object 1. Now in part d, it can be seen that robot 1 partially blocked robot 2 field of view such that robot 2 could not see object 1. Due to this robot 2 missed object 1 and then it was found by robot 1. It is observed that once

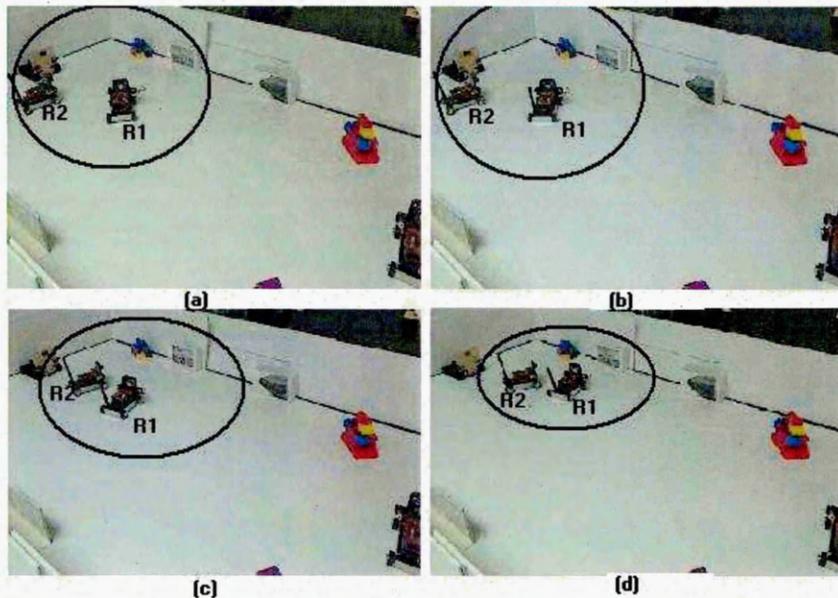


Figure 6.13: Sequence of robot 2 positions when it detected and then missed object 1.

the robot found the 2D objects, then they hardly missed them while got close to the objects. But, in the case of the 3D object, sometimes after detecting the object, when the robot got closer to the object, it missed the object. This is possible as the robot is also switching between higher and lower resolutions, while extracting the object of interest. This switching relies on the distance to the object information and if, at some point this distance information is wrong, then the robot can lose the object of interest.

Distributed Search And Localisation Operation by Swarm of Robots

In the second experiment, a test was planned to show the performance of recognition and localisation algorithm together in the distributed robotic environment. In this experiment, the task was to recognise three 2D and one 3D object by a group of three robots working collectively in the environment. For the three 2D objects, building images were used and for a 3D object, again another robot was selected as in the first experiment. These objects are shown in Figure 6.14. Again, for 2D objects, single image was used for training, but to recognise a 3D robot, training from 16 different poses was given. During the tests, robots recognised the objects of interest collectively, while they were also tracked by the ceiling mounted cameras. In this experiment, all the robots shared information with each other over the wireless

channel in the infrastructure mode once per second. As mentioned before, on finding an object of interest, apart from telling the team members about the object's ID, the robots also informed the server about the object's ID and its distance to object. The robots obtained localisation information from the server. They used the "distance to object" information together with the localisation information to localise the objects. This object localisation information was also conveyed to the server by the robots. The server displayed these determined positions of the objects in the camera images. Apart from this, the server also displayed these object positions in the combined map made by using both camera images.

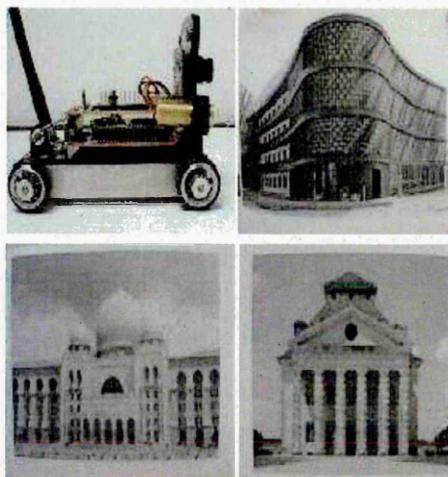


Figure 6.14: 2D and 3D objects used for experiment.

In this experiment, the positions of the robots and the objects of interest, before the test start, are shown in Figure 6.15. In Figure 6.15 it can be noticed that, apart from the objects of interest, some obstacles are also placed in the arena. To avoid colliding with these obstacles, the robot performed the vision based obstacle avoidance technique as described in detail in Chapter 4. At the end of the test, when all objects of interest were found, the localisation information obtained for all the objects on the camera images and also on the combined map is shown in Figure 6.16. The combined map actually shows the combined field of view of both ceiling mounted cameras and its image resolution is 960x1040 pixels. In Figure 6.16, objects 1-4 positions are identified as O1, O2, O3 and O4 respectively, both in the camera images and also on the combined map. They are also marked by a yellow cross sign. It can be seen that the objects of interest are successfully recognised and localised by the team of robots and the cameras collectively.

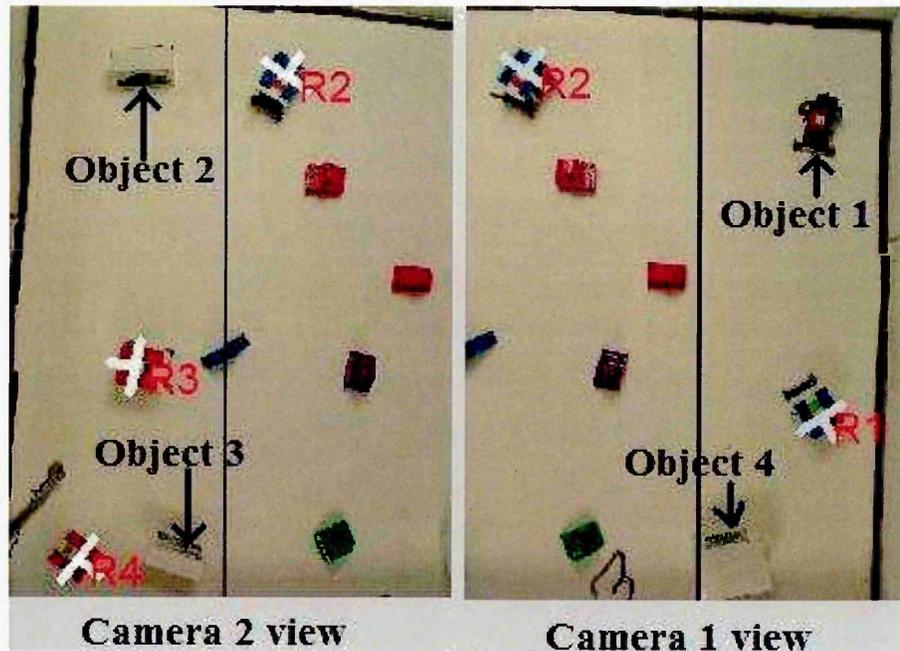


Figure 6.15: Position of robots before experiment.

To determine the accuracy of the identified positions of the target objects, the objects' locations information are shown in Table 6.1. The objects' locations are shown in terms of the x and y coordinates of the ceiling camera image space. As shown in Table 6.1, the x and y coordinates detected for object 1 (i.e., a 3D object) are (163,914) on the combined map. Whereas, the actual coordinates (i.e., the true coordinates of the object on the combined map) determined for object 1 are (159,931). This shows a deviation of 17.5 pixels on the combined map. When translated to the real world coordinates, the error is 2.8 cm. This shows a very small error in the determined location of the first target object. Similarly, for objects 2, 3 and 4 (i.e., object images), the detected coordinates and the true position coordinates are shown in Table 6.1. For objects 2, 3 and 4, the observed error was 14.2, 5.2 and 4.7 cm, respectively.

Object ID	Actual Position (x,y)	Detected Position (x,y)	Error (pixels)	Localisation error (cm) in the arena of area (150x150cm)
Object 1 (3D)	(159,931)	(163,914)	17.5	2.8
Object 2 (2D)	(135,232)	(60,262)	90.1	14.2
Object 3 (2D)	(927,307)	(946,280)	33	5.2
Object 4 (2D)	(885,785)	(896,757)	30	4.7

Table 6.1: Object Localisation Information

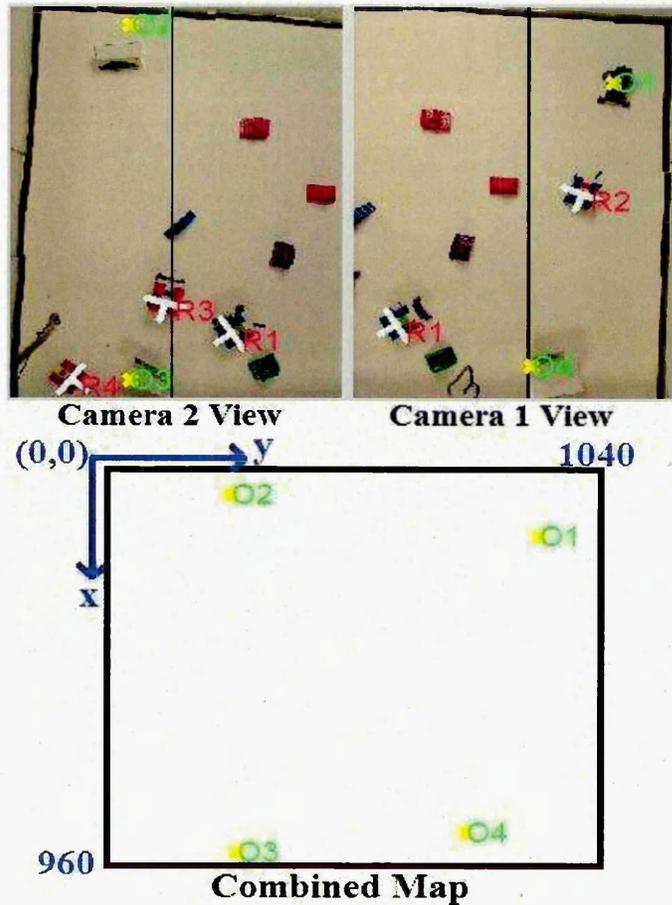


Figure 6.16: All objects are localised by using information from the team of robots and visual tracking system collectively.

During the test, the only problem observed was related to the synchronisation of the robots with the localisation system. For example, in the Figure 6.17, when robot 3 identified object 2 image (left image in Figure 6.17) and transferred the “distance to object” information to the localisation system (i.e., server), then the robot was slightly far from the object. But by the time it sent the object’s ID information to the server and enquired the localisation information from the *Visual Tracking System*, it moved toward the object (right image in Figure 6.17)). The tracking system locked the new position of the robot and passed it on to the robot. Now the robot used the “distance to the object” information with the its localisation information (valid for its new position) to localise the object with reference to its own position. As the distance to the object information was valid for the last position of the robot, but its use with the new robot position caused a shift in the object localisation.

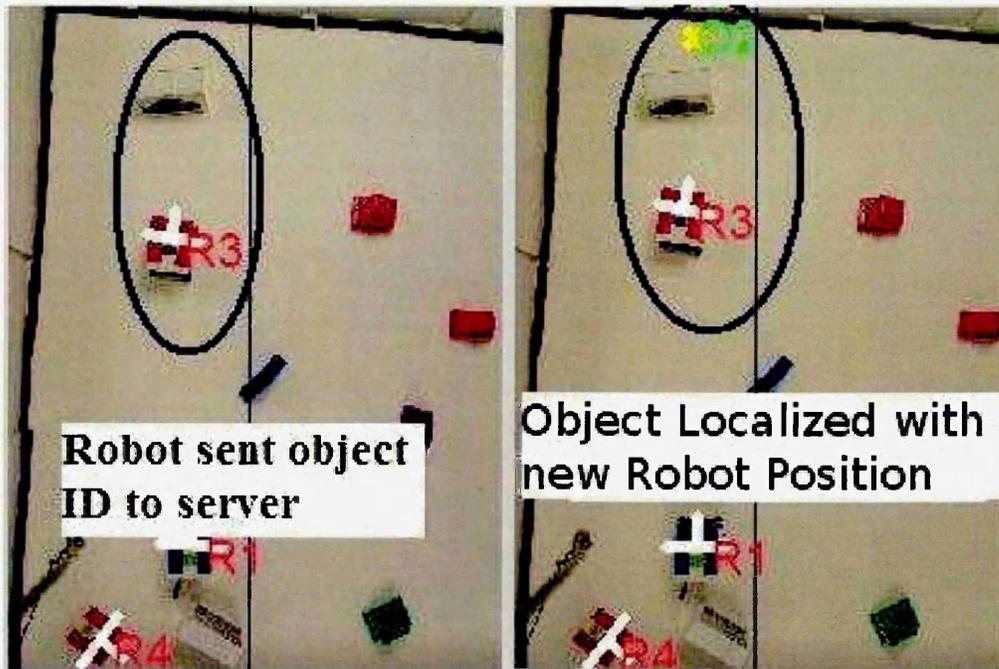


Figure 6.17: Error observed in localising object.

Due to this, object 2 (i.e., “object image”) mistakenly localised slightly far from its actual position. The determined object’s position is shown on the right side of Figure 6.17. This can also be observed in Table 6.1, where the error in the detected position of object 2 is 14.2 cm which is comparatively much larger than the error found in the positions of objects 1, 3 and 4. This problem can be easily fixed by adding an acknowledgement between the robot to the server communication such that the robot did not change its position unless it was informed by the server that the “distance to object” information has been used.

6.3 Environment Mapping by Distributed Multi-robotic System

Another distributed vision processing scenario in swarm mode is related to multi-robot environment mapping. Environment mapping is the concept in which robot senses or experiences its surrounding and tries to obtain a global map. In other words, robots map the environment using the perception sensors available on-board. The generated global map essentially helps the robots to navigate autonomously in

the environment. Environment mapping is a difficult problem to address as in most cases, the robots require human assistance, if they are exploring the place first time. The robots can not know their locations without having an environment map and at the same time, the robots can not built the environment map without knowing their location. In some studies in the robotics field, researchers have worked around this problem in which the robots keep on localising themselves and at the same time, they also build the environment map. This is called Simultaneous Localisation And Mapping (SLAM) in computer vision and robotics field. In the environment mapping problem, in the beginning, the robots go around in the environment, while localising themselves, keep on building a map of what they sense in the environment and this way, when they revisit a place, then from the generated map they already have some awareness of their surrounding. If the robots are working in the static environment then the full map of the environment can be generated, which can help the robots to perform the path planning efficiently, for example to determine a path to reach some specific location.

This environment mapping problem is addressed in robotics by many researchers. Researchers have used many different sensors such as laser range finders, infrared sensors, sonar and vision sensors. But most of the research is focused on using laser range finders. The laser range finders provide a very good depth or distance information about the robot surrounding, but at the same time, they are very expensive. In case of swarm of robots, the laser range finder solution can not be used as a single laser range finder can cost from £800 to £3000. This is far too expensive to integrate on a single small robot as the basic idea in swarm robotics is to use many, but simple designed robots to keep the cost of each robot low. At the same time, the laser range finders are also very big in size and their power consumption is also high and do not suit the robots which have limited on-board memory and energy resources.

In this section, a distributed vision based multi-robot environment mapping problem is addressed in which swarm of robots collectively try to obtain a common global map of the environment using the visual clues they obtain from their surrounding. The generated map is intended to facilitate the multi-robot mission planning as the environmental map together with the robots position on the map will be available. The problem addressed here is different from SLAM as in this case, the robots are provided the localisation information and they do not have to keep on localising themselves. This is done to keep the focus of the research on the distributed vision

processing part rather than addressing the SLAM issue.

The rest of this section is divided into two parts. In the first part, the methodology followed to perform the environment mapping is addressed. Whereas, in the second part, the results from the multi-robot environment mapping technique are discussed.

6.3.1 Methodology - Environment Mapping

To address the distributed multi-robot environment mapping problem, two Surveyor SRV1 robots (shown in Figure 3.1b) equipped with a vision sensor were used. For obtaining the localisation information the *Visual Tracking System*, discussed in Chapter 5, was used. This system is responsible to determine the robot position, track it and pass the robot's localisation information to other robots. The concept of the overall scenario is the following. Each robot in the environment creates a map in its memory. As the robots will be generating 2D map of the environment (i.e., only the boundaries of the objects detected in the environment), so this will not require large amount of memory. The robots will allocate memory for map generation only once in the beginning. This will avoid allocating or expanding allocated memory for map at run time. The robots could have used shared memory (e.g., the map is generated on one robots' memory in a swarm and this memory is shared amongst all) for generating the common map. But in this case, if the robot which holds the map malfunctions or fails, then all the learned map will be lost. As the generated map is two-dimensional and requires very low amount of memory (i.e., 1040x960 bytes only), so it was found more efficient to have a local copy of the map in all the robots. When the robots are generating the map, this map is also updated by all other robots working in the environment. For this purpose, each robot gets its location and orientation information from the localisation system. Once the robot knows its location and orientation, then in the direction of its heading, it utilises the visual cues it obtained from its vision sensor to determine the objects boundaries detected in its neighbourhood. A robot uses these detected boundary information to update its own map. Apart from updating its own map, the robot also broadcasts this map update information to the other robot members in the environment. On receiving this map update information, each robot in the environment also updates its map. This way, each robot in the environment not only knows the other robots' position, but at the same time, it also keeps on maintaining a common map which

has been built by the contribution of all the robots in the environment. Each robot also passes the map update information to the server on which tracking system is running. This way, the map building process can be seen on the server side. As the robots used had limited on-board memory and processing resources, so it was decided to use a very light weight vision algorithm to solve this problem. In this scenario, the vision algorithms which require the on-board processing are:

- Objects' Boundary Detection.
- Mapping.

Objects' Boundary Detection

To determine the objects boundaries in their neighbourhood, a segmentation based algorithm was used. This is the same approach which was utilised to develop the efficient vision based obstacle avoidance algorithm described in Chapter 4. The vision based obstacle avoidance algorithm also works in parallel to help the robot control algorithm to take necessary decision. If the vision based obstacle avoidance algorithm gave the ground clearance signal to the robot control algorithm, then mapping algorithm was called which determined the object boundaries in the surrounding. To explain the concept of segmentation based object boundary detection algorithm, an example image in Figure 6.18 is considered.

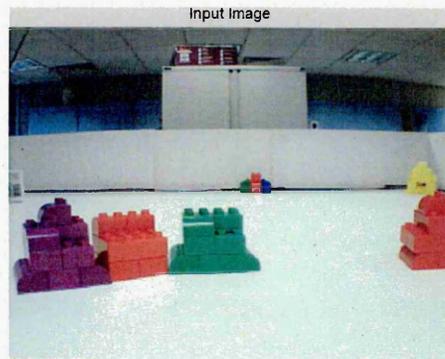


Figure 6.18: Input image used for boundary detection.

The segmentation of the input image, shown in Figure 6.18, is performed and the resultant image is shown in Figure 6.19.

To determine the distance to the near obstacles or to determine the boundaries of the obstacles in field of view, the region covering the middle bottom of the segmented

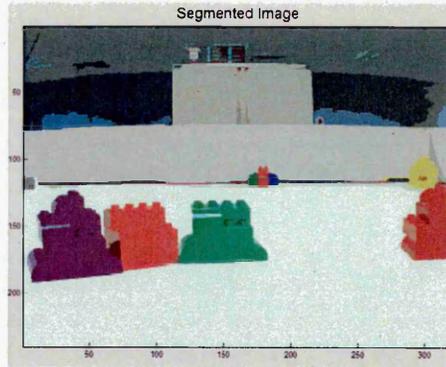


Figure 6.19: Segmented image.

image was considered for further processing. The boundary of this selected region in terms of pixels, in the forward looking direction, was determined. The boundary information was obtained in the form of a vector. This boundary vector information, when plotted on the segmented image, is shown in Figure 6.20. It is to be noted that, this boundary vector provides information about the distance to the objects in the robot field of view, with reference to the robot's current position.

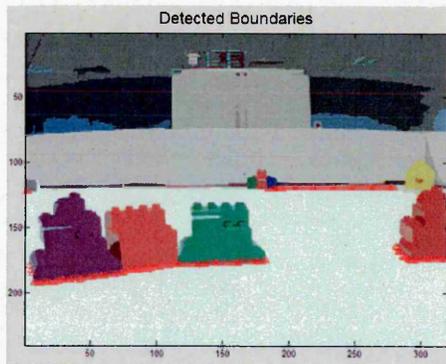


Figure 6.20: Boundary vector plotted on the segmented image.

Mapping

Once the distance vector information was obtained, then the second step was to map this information on a global map. Here, the robot utilised its location and orientation information provided by the tracking system. To demonstrate the manner the robot mapped the distance vector information with reference to its location and heading, the camera images from the tracking system was used. The use of camera images, from tracking system, was made for demonstration because robot location

was defined in image space of tracking system. To accomplish the complete mapping process, following procedure was used.

- From the heading information obtained from the Visual Tracking System, the distance vector was mapped across the field of view provided by the robot vision system. From the specification of the Omni-vision camera sensor used on the robot vision system, the field of view information determined was 90 degrees. To illustrate the manner this 90 degrees field of view was spanned from the current robot heading, consider the image shown in Figure 6.21. In Figure 6.21a, the tracking information from left ceiling mounted camera is shown and similarly in Figure 6.21b, the tracking information from the right camera is shown. It can be seen that there are four robots in the environment and the tracking system has detected their positions properly. Lets consider robot1's position in camera 1 (represented as R1). The zoomed in version of this position is also shown in Figure 6.22. The 90 degrees field of view across the current heading of the robot1's position is determined and shown by the red (on the left side of robot heading) and yellow lines (on the right side of robot heading) in Figure 6.22. The distance vector information (which is obtained by processing the images from robot vision system) are mapped within this 90 degrees field of view.

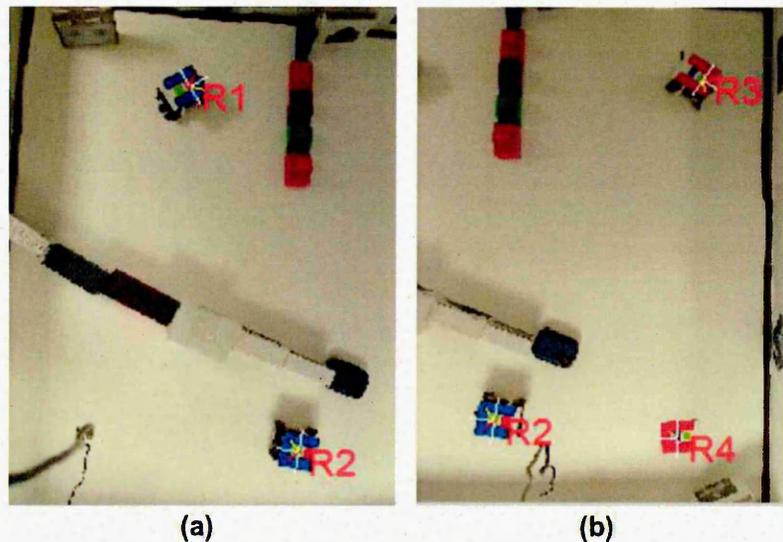


Figure 6.21: Visual tracking and localisation information (a) Left camera. (b) Right camera.

- The second requirement, to perform perfect mapping of distance vector on the

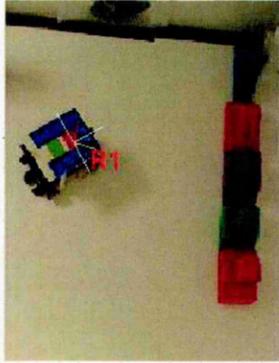


Figure 6.22: Zoom-in information of robot 1 position.

global map, was to determine the equation which translated the distance in pixels, detected by the robot vision system, to the distance in pixels of the images processed by the visual tracking system. This was performed because the robot localisation information was provided in terms of heading and (x, y) coordinates of the images obtained by the visual tracking system. Some experiments were performed to determine the distance in pixels measurement from robot vision system and their corresponding coordinates values in the images of visual tracking system. To obtain these corresponding values, an object was placed at some distance from the robot vision system and the distance detected by the robot vision sensor and the corresponding coordinate values in the visual tracking system were recorded. This experiment was repeated while increasing the distance of the object from the robot vision sensor. Ten values recorded are shown in Table 6.2.

In the first column of the Table 6.2, the distance in pixels " d_r " recorded from robot vision system is shown. In the second column the corresponding value " d_c " (i.e. distance in pixels detected by the ceiling cameras) determined by the visual tracking system is shown. And in the third column, the scale factor " α " determined between the first and second column values is shown. This scale factor translates the distance in pixels from robot vision system to distance in pixels valid for the visual tracking system. It can be noticed that this scale factor is not constant. When this scale factor was plotted against the distance in pixels from robot vision system, the profile shown by the Red colour in Figure

Table 6.2: Scale factor: From robot to ceiling camera.

Distance in pixels - d_r (Image from Robot)	Distance in pixels - d_c (Image from ceiling camera)	Scale Factor (α)
62	73.8	1.19
69	82.8	1.20
75	90.8	1.21
81	99.6	1.23
85	106.3	1.25
93	119.0	1.28
101	138.4	1.37
108	169.6	1.57
114	199.5	1.75
121	254.1	2.10

6.23 was obtained. It can be noticed that this is an exponentially rising profile.

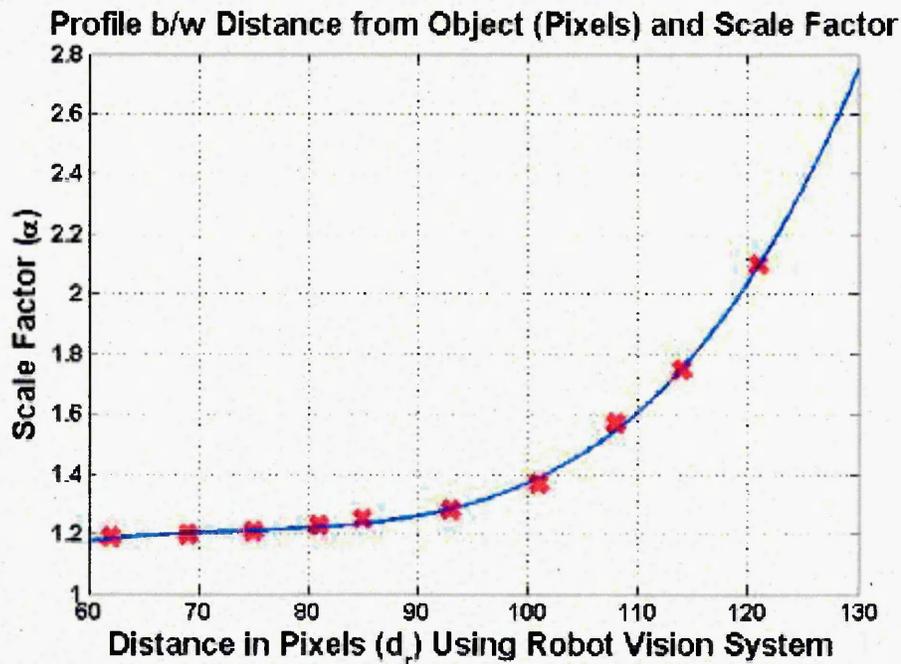


Figure 6.23: Profile between distance from object (pixels) and scale factor.

To determine the equation which satisfies the relation between the distance in pixels " d_r " (from robot vision system) and the scale factor " α ", MATLAB Equation Fitting Toolbox was used. After the equation fitting, the result obtained is shown by the Blue colour profile in Figure 6.23. For this profile, a fourth order fitting equation was obtained from the MATLAB Equation Fitting

Toolbox as:

$$\alpha_i = 4.4e^{-8}d_{ri}^4 - 7e^{-6}d_{ri}^3 + 0.00011d_{ri}^2 + 0.028d_{ri} + 0.0051 \quad (6.1)$$

Where, $i = 1 \rightarrow 320$ (Image width). Note that, as the vector of distance information (Figure 6.20) is generated, so a vector of corresponding values of α_i will be generated. Equation 6.1 was used to determine the scale factor α_i for every value of the distance vector " d_{ri} " shown in Figure 6.20. When these scale factors α_i were multiplied with their corresponding values of d_{ri} then a vector d_{ci} was obtained which determined the distance in pixels detected by the visual tracking system. This is shown as:

$$d_{ci} = d_{ri}\alpha_i \quad (6.2)$$

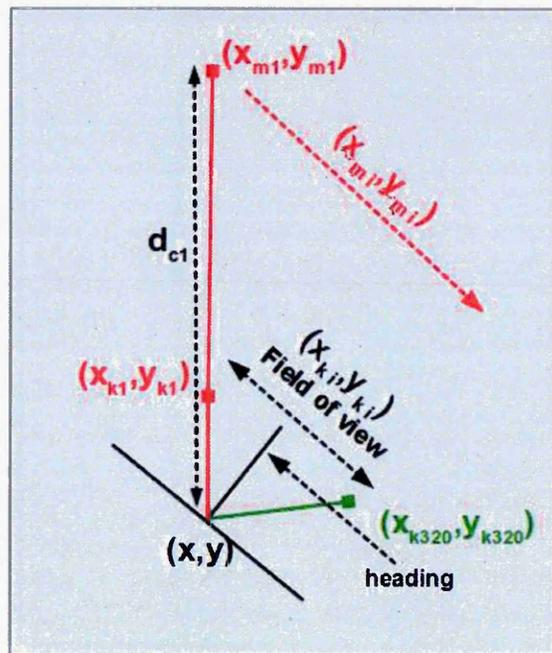


Figure 6.24: Mapping process.

From Figure 6.24, d_{ci} can also be described in terms of the robot localisation information (i.e. the coordinates of robot position (x, y)) and the final coordinates (x_{mi}, y_{mi}) where the object boundaries were mapped;

$$d_{ci} = \sqrt{(x_{mi} - x)^2 + (y_{mi} - y)^2} \quad (6.3)$$

Similarly, the corresponding slope values s_i for each value of d_{ci} were computed using the coordinates (x_{ki}, y_{ki}) which were uniformly spaced coordinates spanning the robot field of view (see Figure 6.24);

$$s_i = \frac{y_{ki} - y}{x_{ki} - x} \Rightarrow \frac{y_{mi} - y}{x_{mi} - x} \quad (6.4)$$

Using the values of d_{ci} (where $i = 1 \rightarrow 320$ "Image width") together with the corresponding slope values s_i and the robot position coordinates (i.e. (x, y)), the coordinates (x_{mi}, y_{mi}) were computed for mapping the object boundaries. The equations used to compute x_{mi} and y_{mi} are given as:

$$x_{mi} = \frac{d_{ci}}{\sqrt{1 + s_i^2}} + x \quad (6.5)$$

$$y_{mi} = \frac{s_i d_{ci}}{\sqrt{1 + s_i^2}} + y \quad (6.6)$$

Note that, (x_{mi}, y_{mi}) are defined in ceiling camera image space. This way, once (x_{mi}, y_{mi}) were computed then the complete distance vector information was mapped to the ceiling camera image space. Following the above defined procedure an example distance vector from the robot 1 vision system, mapped to the global map, is shown in Figure 6.25a. In Figure 6.25b, the tracking information from the second ceiling mounted camera is shown. In Figure 6.25c, the global map generated when the two camera information was fused together, is shown. The zoomed in version of Figure 6.25a is shown in Figure 6.26. It can be seen that, the distance vector is mapped properly along the boundary of the objects.

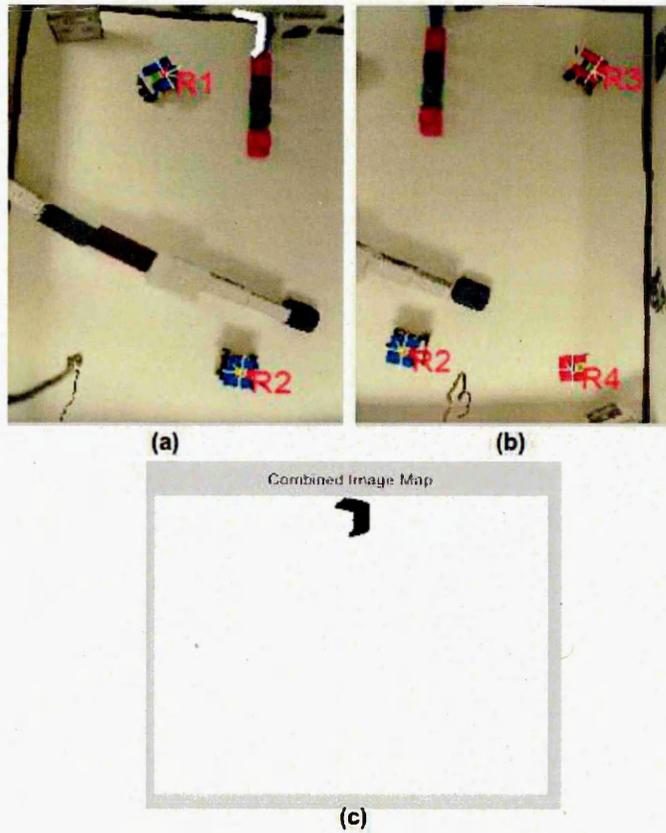


Figure 6.25: (a) Distance vector from robot 1 mapped To coordinates of visual tracking system. (b) Robot tracking image from second ceiling mounted camera. (c) Global map generated using the two ceiling mounted cameras.

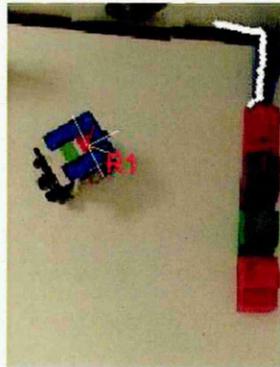


Figure 6.26: Zoom-in version of Figure 6.25a.

6.3.2 Experimental Results

To demonstrate the distributed vision based multi-robot environment mapping scenario, several experiments were performed. A test platform was designed

with obstacles placed in certain forms to create an environment to be mapped by the multi-robotic system. In this section, the results obtained from three experiments, are presented where in each experiment, a different environment is provided to be mapped by the robots. All the robots were programmed to follow the boundaries in the environment and at the same time, if the robot found itself very near to an obstacle or stuck between a narrow passage, then the robot could decide a new direction depending upon the ground clearance. These three experiments are detailed below.

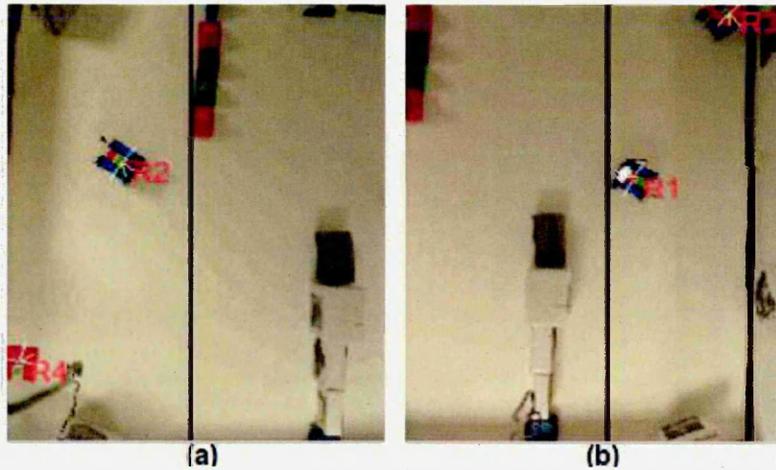


Figure 6.27: Experiment 1: Visual tracking system (a) Left camera. (b) Right camera.

Experiment 1

In experiment 1, the environment to map is shown in Figure 6.27. The images provided by the left and right cameras of Visual Tracking System are shown in Figures 6.27a and 6.27b, respectively. It can be seen that four robots (robots are labeled as R1, R2, R3 and R4) have been detected in the environment. Robots R3 and R4 are dummy and not contributing to map building process. Only Robots R1 and R2 are working collectively to map the environment. In these images, it can be noticed that some blocks and boxes are placed in the vertical direction to generate an appropriate environment for map building process. The two robots were expected to go around in the environment and use the visual clues from their vision sensor together with the visual tracking system information. This way they mapped the test arena boundary and walls made by the blocks. The robots worked collectively to generate a common map

which was blank in the beginning.

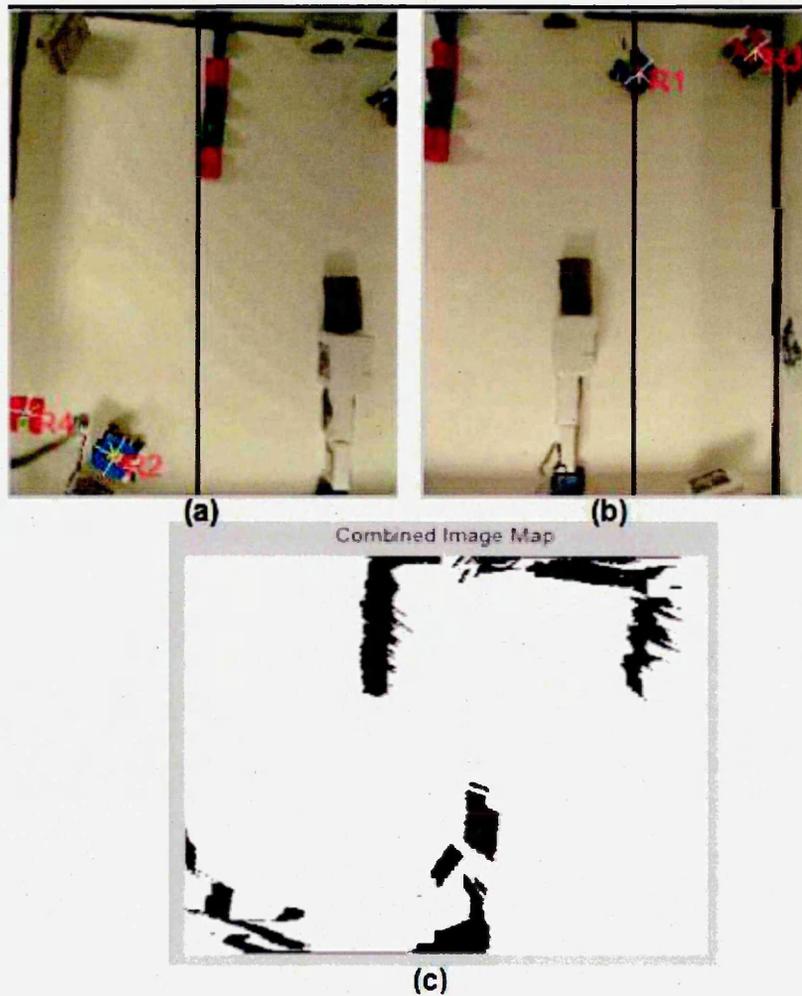


Figure 6.28: Experiment half finished: (a) Left camera. (b) Right camera. (c) Map.

In the experiment, the robots were left in the arena to map the environment. This environment mapping process was a slow process because after every move, the robots needed to wait so that the tracking system locked their current positions and provided them their location information. This was done so that the robots were provided with their locations and orientations information which precisely represented their current positions. This was important as robots had to use this vision based location and orientation information as the basis for mapping the objects boundaries detected in the field of view of their vision system. If this information represented their last position, then inaccuracies in map were expected. For example, a small error in the orientation

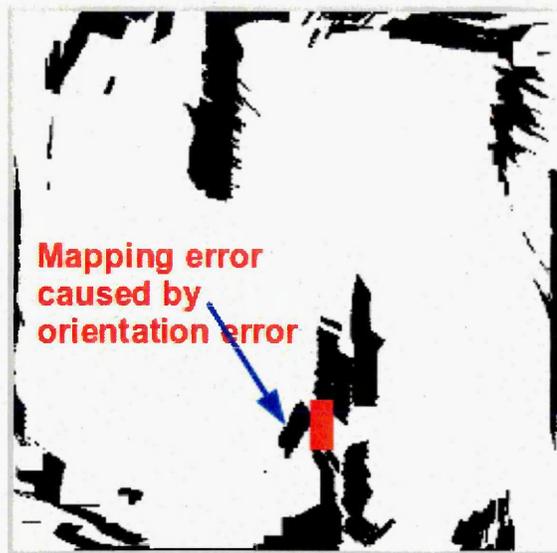


Figure 6.29: Mapping error caused by the orientation error.

information could generate considerable error in the mapped boundaries. The two robots were kept operative for 4 minutes and 21 seconds. After half of the experiment was complete, the progress on map generation process, done by the robots is shown in Figure 6.28. In Figures 6.28a and 6.28b, the positions of the robots, when half of the experiment was complete, are shown. In Figure 6.28c, the progress on the map generation process is shown. Finally when the experiment completed, the final map generated by the robots is shown in Figure 6.29. As mentioned before, a mapping error caused by a small error in the detected robot's orientation is also shown in Figure 6.29. The erroneously mapped boundary is pointed out by the Blue arrow. The actual location where this boundary should be mapped is also drawn in Red colour.

Experiment 2

In the second experiment, the environment was altered by placing the objects in different order such that they contributed to a different map. The starting position of the robots detected by the *Visual Tracking System* and the new environment to be mapped by the robots is shown in Figure 6.30. As the objective of this scenario was to demonstrate the collective vision based map building operation by a team of robots rather than the efficient robotic control, so the robot control algorithm was kept very simple. The robots were programmed to

perform the vision based wall following. To make sure that the robots explored the most part of the arena, the initial positions of the robots were selected very carefully. In Figure 6.30, it can be noticed that the position of the robot 1 (R1) is set in a way that it started the mission by mapping the top right part of the image (image shown in Figure 6.30b). Similarly, the robot 2 (R2) started the mission by mapping the top left part of the image shown in Figure 6.30a.

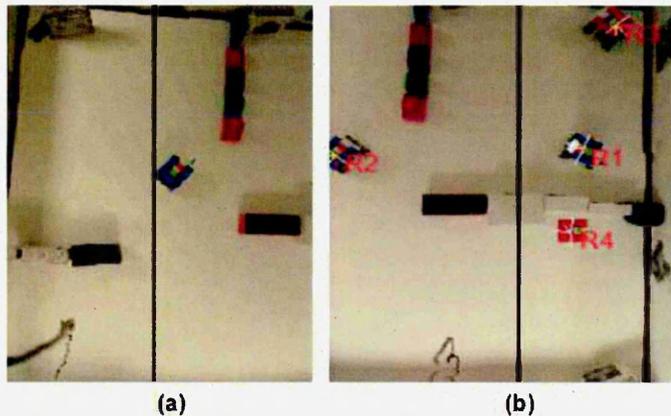


Figure 6.30: Experiment 2: Visual tracking system (a) Left camera. (b) Right camera.

After half of the experiment was completed, the progress in map generation process is shown in Figure 6.31c. It can be noticed that the robot 2 has mapped the part of the arena shown in top left side of the Figure 6.31a. After successfully mapping the first part, robot 2 headed toward the part of the arena which is shown in the bottom of the Figure 6.31a. Similarly, the robot 1 successfully mapped the part of the arena shown in the top right side of the Figure 6.31b. Finally, robot 1 moved towards the part of the arena which was already explored by the robot 2. Revisiting the part of the arena which was already mapped by the other robot added the redundancy but at the same time, it also filled the gaps in the map which were left by the other robot. When the test finished, the final map generated by the two robots is shown in Figure 6.31d.

Experiment 3

Similar to the experiment 2, for the new map generation, again the environment was altered significantly. At the start of experiment 3, the positions of the robots in the environment and the new map to be generated by the robots is shown in Figure 6.32. At the end of the experiment, the final map generated

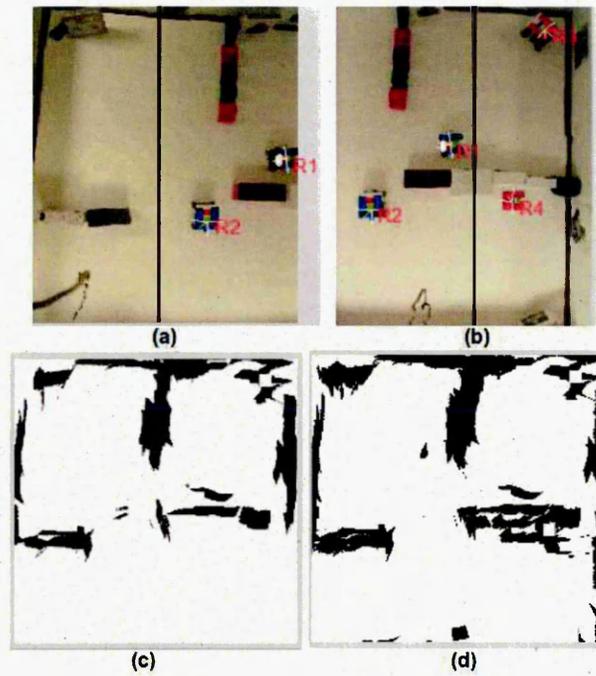


Figure 6.31: Experiment 2 half finished: (a) Position of robots in left camera after half of the experiment is finished. (b) Position of robots in right camera after half of the experiment is finished. (c) Progress on map generation after half of the experiment is finished. (d) Experiment 2 final map generated.

is also shown in Figure 6.33. It can be seen that, at the end of the experiment, the robots have successfully mapped the arena.

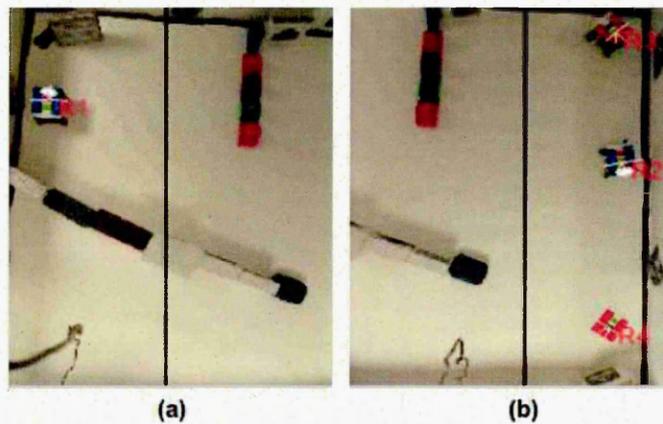


Figure 6.32: Experiment 3: (a) Left camera image from visual tracking system. (b) Right camera image from visual tracking system.



Figure 6.33: Experiment 3 final map generated.

6.4 Conclusions

In this Chapter distributed vision processing in swarm mode scenarios have been presented. It has been concluded that, when multiple robots are working in the swarm mode then due to the communication bottleneck, the robots can not share the raw vision data between each other for distributing the processing load. For information exchange, as the robots use wireless communication medium which suffers from severe noise, so this also limits the sharing of compressed images between multiple robots. It has been concluded that, the sharing of processed visual data in terms of features not only makes efficient use of the communication medium, but it also shows the distribution of vision processing load. It is shown that, the use of simple visual features (e.g., distance to the neighbouring objects) can be made to map the environment by a group of robots. However, if the features encode high density information (e.g., SURF features), it has been concluded that, sharing of such features could overload the network. In this case, the vision processing distribution can be achieved by processing the features on-board, and then sharing the outcome of these processing in terms of decision (e.g., identity of the target object or distance to the target object).

Chapter 7

Distributed Object Recognition and Information Gathering in a Multi-Robotic Organism

As described in Chapter 1, in this research a swarm robotic system is considered in which robots have the ability to physically join together to become a single three dimensional robotic organism whenever the need arises. So when the robots are in the form of an organism, each robot unit contributing to the organism, has its own resources. These resources can be used to fulfil the requirements of that individual robot unit and at the same time, these resources can also be used to contribute to a common objective set by the organism. In other words, the resources from all the robots, accumulated in the organism, can efficiently contribute to a major task which is set by the robotic organism. In Figure 7.1, a robotic organism is shown. This organism is formed after utilising a vision based physical docking support developed in Chapter 4. The physical formation of the complete organism is out of the scope of this research. To address the issue of distributed vision processing in the robotic organism, a scenario is described in this Chapter, in which the robotic organism is given a task to perform vision based obstacle avoidance, efficient object recognition, and visual information gathering in the distributed fashion, while utilising the energy and processing resources available within the robotic organism only.

In Figure 7.1, an example of an experimental platform is shown. Several obsta-

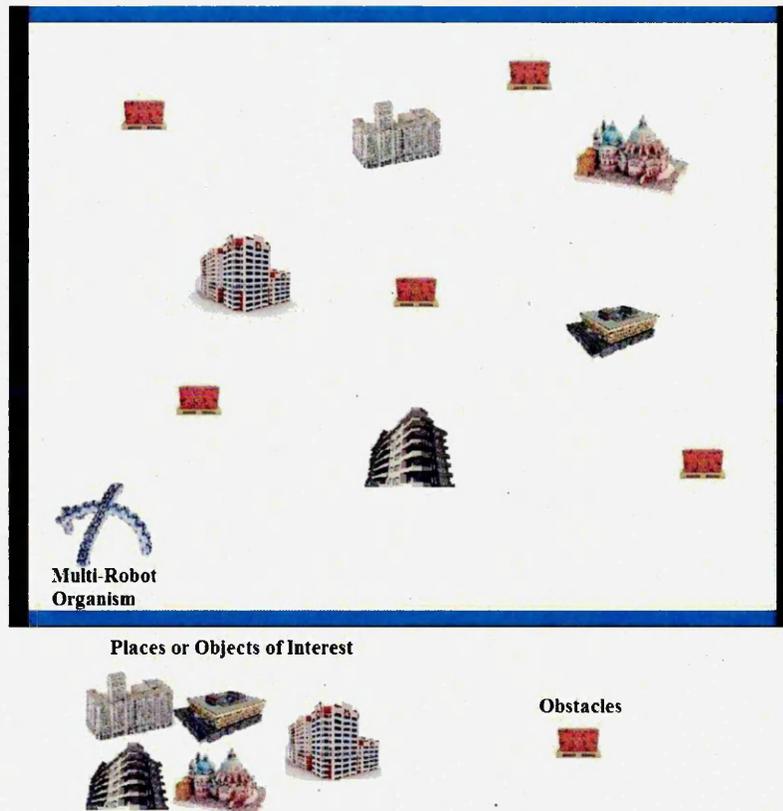


Figure 7.1: Robotic organism scenario.

cles are placed in the environment, which the organism has to avoid colliding while performing its task. At the same time, some places or objects of interest are also shown which the organism has to identify. This scenario is different from the one discussed in Chapter 6 where each robot in the swarm is doing obstacle avoidance and recognition task on its own. In the swarm mode scenarios, the robots had a wireless communication with each other. This communication medium is a bottle neck in the distribution of vision based task as communication bandwidth is much smaller to satisfy the exchange of vision based information between the robot units. Whereas, in the case of robotic organism, after a docking operation, the robots establish an Ethernet communication medium (10/100M bits/sec) with the other robot modules. This high speed communication backbone is promising to exchange the visual information and share the processing load between the robot modules forming an organism. So, through the Ethernet medium, the organism split the vision based control, obstacle avoidance, recognition and information gathering tasks

between different robotic modules. In the scenario addressed in this Chapter, the organism searches for the objects of interest in the environment as shown in Figure 7.2, where the organism is detecting and recognising one of the objects in the environment. Once the object is identified, then the organism performs the vision based scanning of the environment as shown in Figure 7.3.

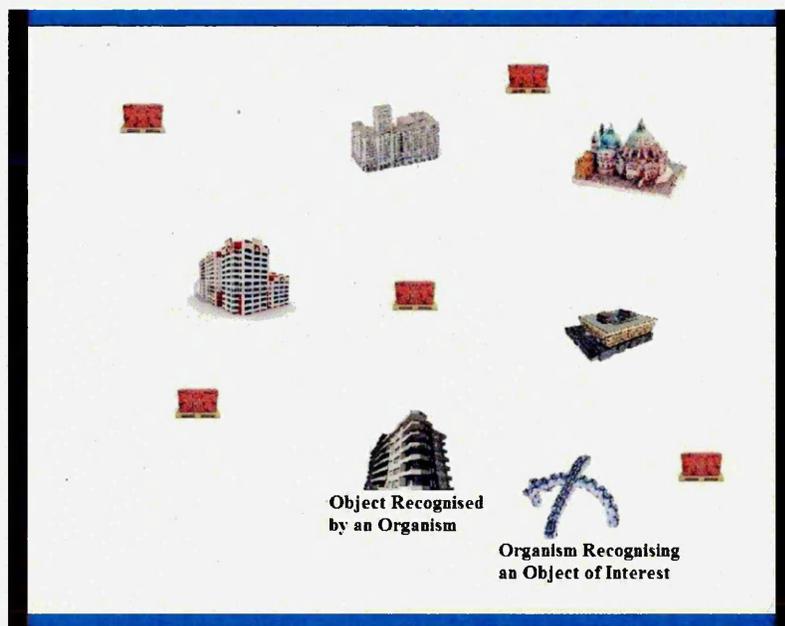


Figure 7.2: Robotic organism recognising the object in the environment.

In the vision based scanning of the environment (shown in Figure 7.3), the organism gathers the visual information in terms of the sequence of images. The organism stitches the sequence of images and makes a mosaic of it, which provides a bigger picture of the environment. This mosaic provides the objects surrounding awareness to the organism, that is, what lies on the left and right sides of the object. This information can be very useful in navigating the organism. For example, if the organism has to reach the same object again, it can try to relate what it saw with what it had in the mosaics stored in its memory. If it finds a match, then it can get clues that in which direction it is likely to find the object. This Chapter is dedicated to the described distributed vision processing scenario in the robotic organism. To address in detail the different issues, which makes the vision distribution and its processing possible in the organism, this Chapter is divided into the following sections.

- Communication in the Robotic Organism.
- Tasks distribution for distributed object recognition and information gathering.
- Experiments with the Robotic Organism.



Figure 7.3: Robotic organism gathering the visual information and making mosaic.

7.1 Communication in the Robotic Organism

As communication medium is a backbone of any distributed computing system, so careful consideration was given while selecting the medium for information distribution in the robotic organism. Ethernet is a very high speed and reliable communication medium and as the use of embedded systems is very common in network application, so Ethernet modules can be found integrated in most of the embedded systems. For this reason, Ethernet was selected for information distribution in the organism. As Blackfin processor was used for information processing, so like other embedded systems, Ethernet module is also provided on Blackfin development board Eval-BF5xx as shown in Figure 3.5. As discussed in Section 3.2, a multi-processor robotic organism was developed to perform distributed vision processing tasks, so in this organism the basic communication

network between the two processing modules (or robot units) can be viewed as shown in Figure 7.4.

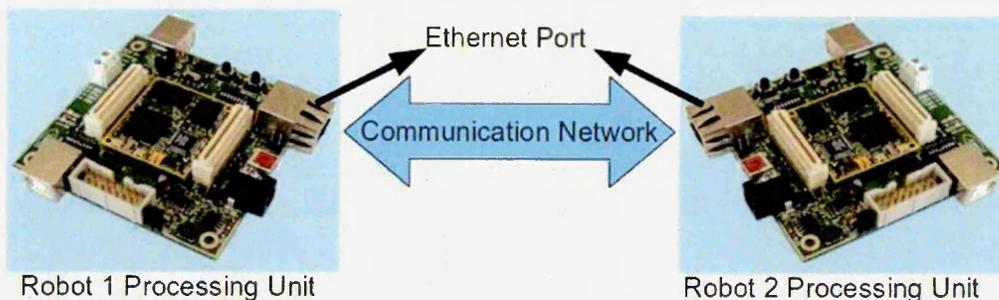


Figure 7.4: Communication set-up between robotic modules in the organism.

The Ethernet port on the two EvalBF5xx boards (through which the robot processing units are integrated) is also shown in Figure 7.4. Through this Ethernet port, a communication network channel between the two robot units is established in the organism. The information distribution over this network is processed through four network layers. These are TCP/UDP, IP, Ethernet, and Physical layers. These network layers are shown in Figure 7.5.

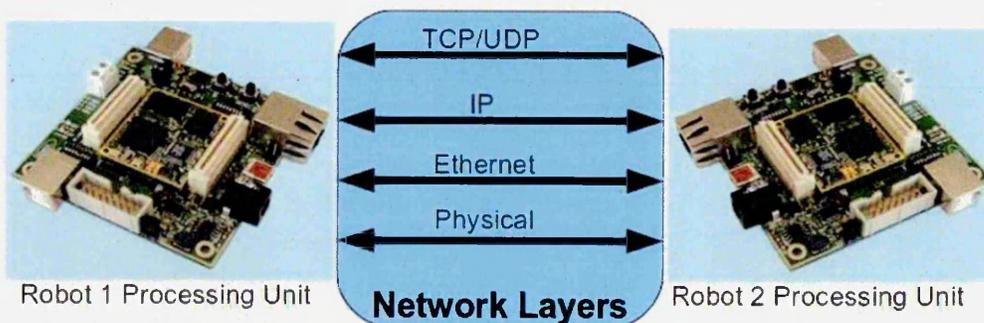


Figure 7.5: Communication network layers.

Generally, in network applications requiring information distribution over a network, the application to network communication interface is made above TCP/UDP layer as shown in Figure 7.6. In this case, all the communications between the applications go through all the network layers. The data from the sender side go from TCP/UDP to Physical and on the receiver side, it goes through the Physical to the TCP/UDP layer. Whereas, from the basic network protocol, the final data transmission happens on the Physical layer.

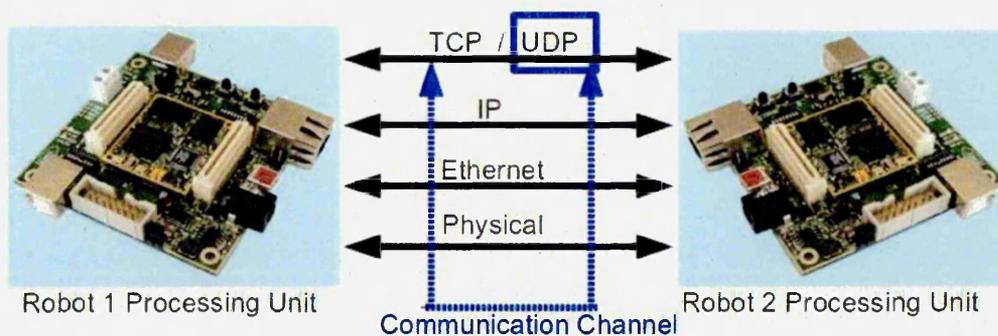


Figure 7.6: Application interfaced to TCP/UDP layer.

In this configuration, it can be noticed that, the data are processed on the TCP/UDP and IP layers twice, once on the sender side and then on the receiver side. This causes a processing overhead. If applications are running on high speed processing systems, then this processing overhead can be ignored but as mentioned in the swarm robotic systems, the robot units have limited on-board processing resources, so this information processing overhead should be avoided. To avoid this processing overhead, all the applications can be interfaced with the transmission channel at Ethernet layer as shown in Figure 7.7. In this case, as the information bypasses the TCP/UDP and IP layers, so an increase in the information throughput is expected.

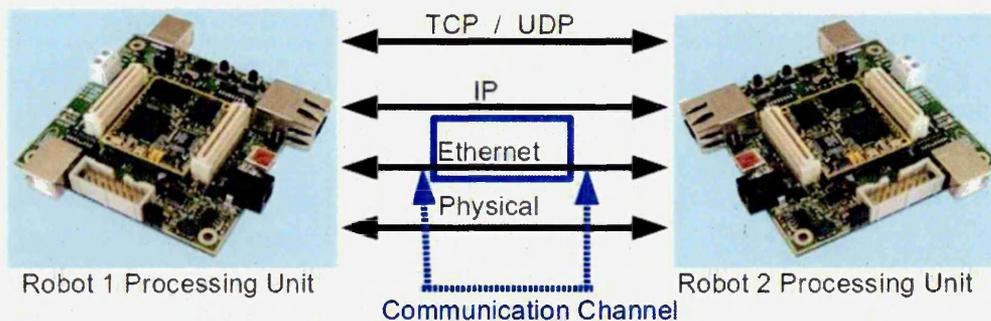


Figure 7.7: Application interfaced to ethernet layer.

In case the application is interfaced with the TCP/UDP layer (as shown in Figure 7.6), then it creates socket and the target robot IP is the only item required to send data to the other robot. The TCP/UDP and IP layers automatically create the frame from the data provided and these layers also perform the checksum to validate the transmitted data. On the receiving side, the tar-

get robot receives the data sent by the sender and does not require any data validation. On the other hand, in case the application is interfaced through the Ethernet layer (as shown in Figure 7.7), then this configuration provides a high transmission data rate, but at the same time, the application is required to know the target robot MAC (Media Access Control) address. The application is also responsible for manually creating the frames for the Ethernet layer. This frame format is shown in Figure 7.8. This is the basic frame format in which an application is required to pack the data and send to the target robot. This frame requires target robot MAC address (Destination MAC), sender robot MAC address (Source MAC), Frame ID describing what type of information is included in the frame, Data to send (User Data) and finally the check sum of the complete frame (Check Sum). The application is responsible for adding this check sum information in the frame to ensure data validation on the receiving side. All the robot units in the complete organism act like a network connected modules which exchange information at very high rate at Ethernet layer. It is to be noted that, in this form of communication, once the data are sent by the sender robot, then in the organism, the robot module with the MAC address as mentioned in the target MAC address of the Ethernet frame, receives the frame. As the sender MAC address is also a part of Ethernet frame, so the target robot also knows which robot module has sent this information.

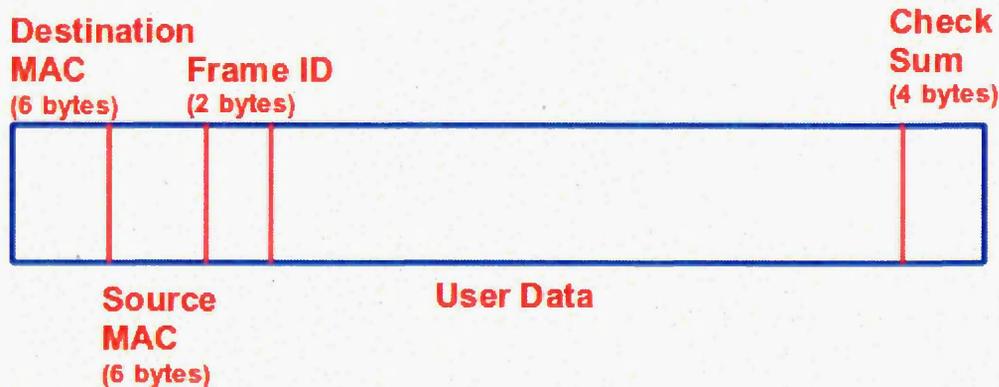


Figure 7.8: Ethernet frame (IEEE 802.3 standard).

7.2 Tasks Distribution for Distributed Object Recognition and Information Gathering

Based on the promising data throughput achieved using the low level Ethernet communication layer, the distribution of vision based task is possible. The distributed vision processing scenario addressed in this Chapter requires three robot modules docked together to form a robotic organism. The robot organism developed for this scenario is shown in Figure 3.12. As discussed in the beginning of this Chapter, the robot organism is required to perform vision based control (or decision making), obstacle avoidance, object detection, recognition and surrounding information gathering in parallel. From the work presented in Chapter 4, very light weight vision based obstacle avoidance techniques are already addressed. This task does not require to be distributed as it can be performed by a single robot very efficiently and so is the case of vision based decision making. The other two tasks, i.e., object recognition, visual information gathering and mosaicing are computationally very heavy and require collective processing from the robotic modules in the organism. The main objective of the complete distributed vision processing scenario can be divided into following three phases.

- Communication Awareness within the Robotic Organism.
- Distributed Object Recognition.
- Distributed Information Gathering.

7.2.1 Communication Awareness within the Robotic Organism

In this scenario, a robot in the organism is a master robot and the other two are slave robots. It is assumed that the robot which generates the need for organism formation is the master robot. And all the robots which are called to participate in forming an organism by going through the physical docking procedure, are automatically the slave robots. An example image showing the REPLICATOR robot units, which will participate to form an organism, is shown in Figure 7.9.

An Active Wheel robot (i.e. one design of Replicator robot unit), is shown as the master robot in Figure 7.9. This robot has the capability to perform

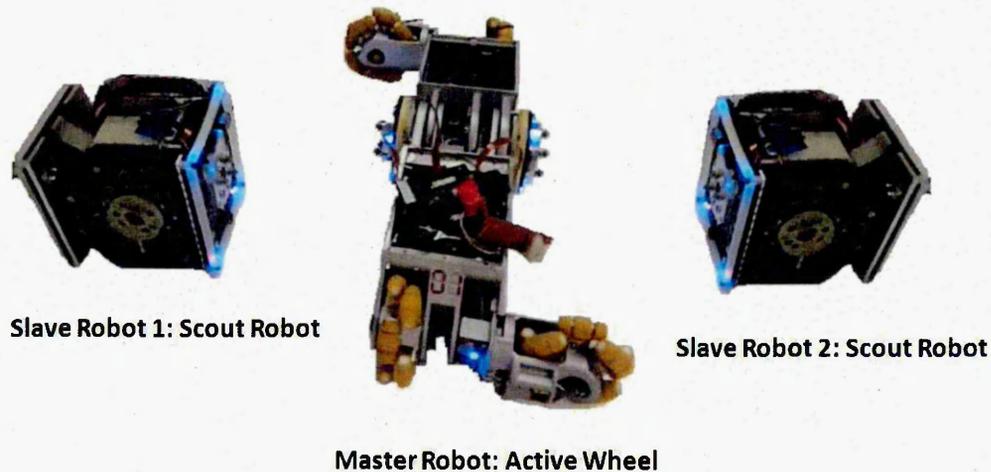


Figure 7.9: Example Replicator robots participating to form an organism [8].

docking from both sides and mechanism to lift the organism. The other two robots, acting as the slave robots are Scout Robots (another design of Replicator robots). Once the two slave robots are physically docked to the master robot, then the master robot also provides the Ethernet communication backbone to organism. Through this communication backbone all the robots module can communicate with each other directly. The manner master robot docks to the slave robots using the docking port, which also surrounds the communication backbone, is shown in Figure 7.10.

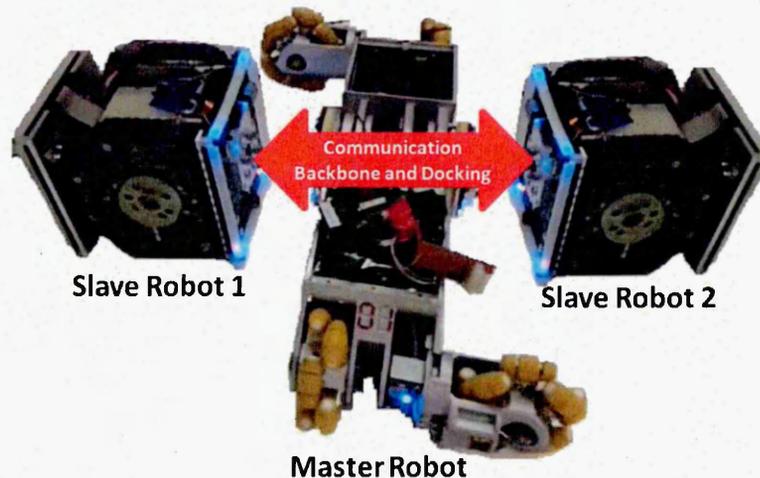


Figure 7.10: Replicator robotic organism from three robots [8].

In this implementation, once the organism is formed, in the beginning all the robot modules in the organism are unaware of the identity of their neighbouring robots. So after each robot is docked, it will start broadcasting a “Breathing Frame” to all the other robots in the organism through the communication backbone. This “Breathing Frame” is 100 bytes and it contains the frame ID stating that this is breathing frame and the MAC address of the sender robot. The manner breathing frame is broadcasted by the two slave robots, is shown in Figure 7.11. This Figure shows that how the communication between the two slave robots and the master robot takes place.

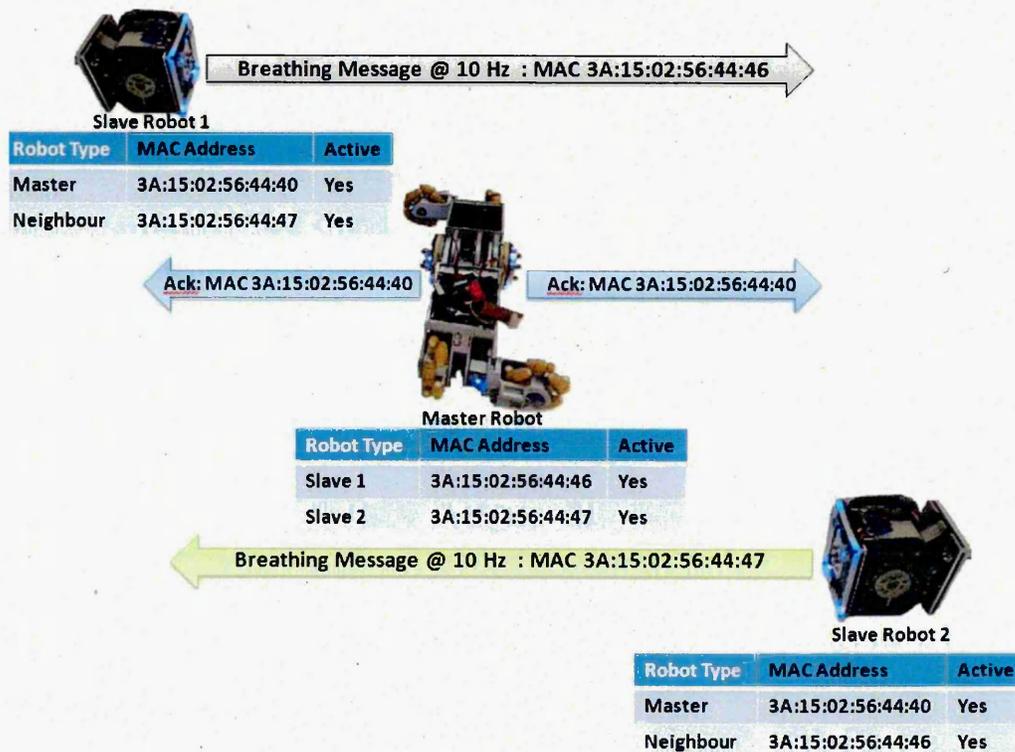


Figure 7.11: Communication within the organism.

This way, on receiving the breathing frame from the neighbouring robots, the master robot knows the identification of its slave robots and starts populating a Table. In this Table, master robot keeps identification information about its slave robots and also their status whether they are still active or not. Master robot senses the breathing signal from all slave robots continuously. This is implemented in a way that each slave robot keeps sending the breathing frame at 10Hz frequency. In Figure 7.11 it is shown that, after forming an organism, the

slave robots 1 and 2 start broadcasting the breathing frame with MAC address “3A:15:02:56:44:46” and “3A:15:02:56:44:47”, respectively. Once the master robot receives these breathing frames, then it updates the communication Table and populates it with the slaves MAC addresses and declares these slaves as active. As each slave robot has broadcasted its breathing frame and apart from master robot, the other slave robot has also sensed this frame. So the slave robots also update their table and populate it with their neighbouring slave robot information. From the acknowledgement received from the master robot, slave robots also know the master robot’s MAC address and store them in the table. The final populated tables for the current master-slave configuration in the robot organism, are shown in Figure 7.11. In the current implementation, if the master robot does not sense breathing message for 2 seconds, then it assumes that the slave robot is no more functioning. In this case, master robot updates the table, declaring this slave robot as inactive. To assign the responsibility of this inactive slave robot to another active robot, the master robot checks if there are more active robots in the organism. If there are free active robots in the organism, then it routes or distributes the vision tasks to these active robots.

7.2.2 Distributed Object Recognition

In the current scenario, it was decided that, once the organism is formed then the master robot is made responsible to perform vision based obstacle avoidance and performs the locomotion of the complete organism. It makes the organism move in the environment without colliding with the obstacles. While keeping the organism in continuous motion, the master robot also performs distributed object recognition with the support from one of the active slave robots. In Figure 7.12, the operations performed by the master robot while performing the distributed object recognition task with slave 1 robot is shown. For achieving the distributed object recognition task, the master robot is responsible to perform a light weight vision based detection of the objects in the images (shown in Figure 7.12). For this purpose, it uses the Harris Feature Detection algorithm for the detection of the features in the image. Rather than considering the complete image for performing the distributed vision processing, it looks for the part of images where more features are located. There is

a high probability that these parts of the image are resulting or describing the objects in the environment. So, the master robot extracts the chunk of these images and passes this visual information to one of the slave robots for performing the computationally heavy object recognition. In case of Figure 7.12, the master robot has extracted the parts of image representing two blocks and one robot image and have transferred this information to the slave 1 robot for further processing.

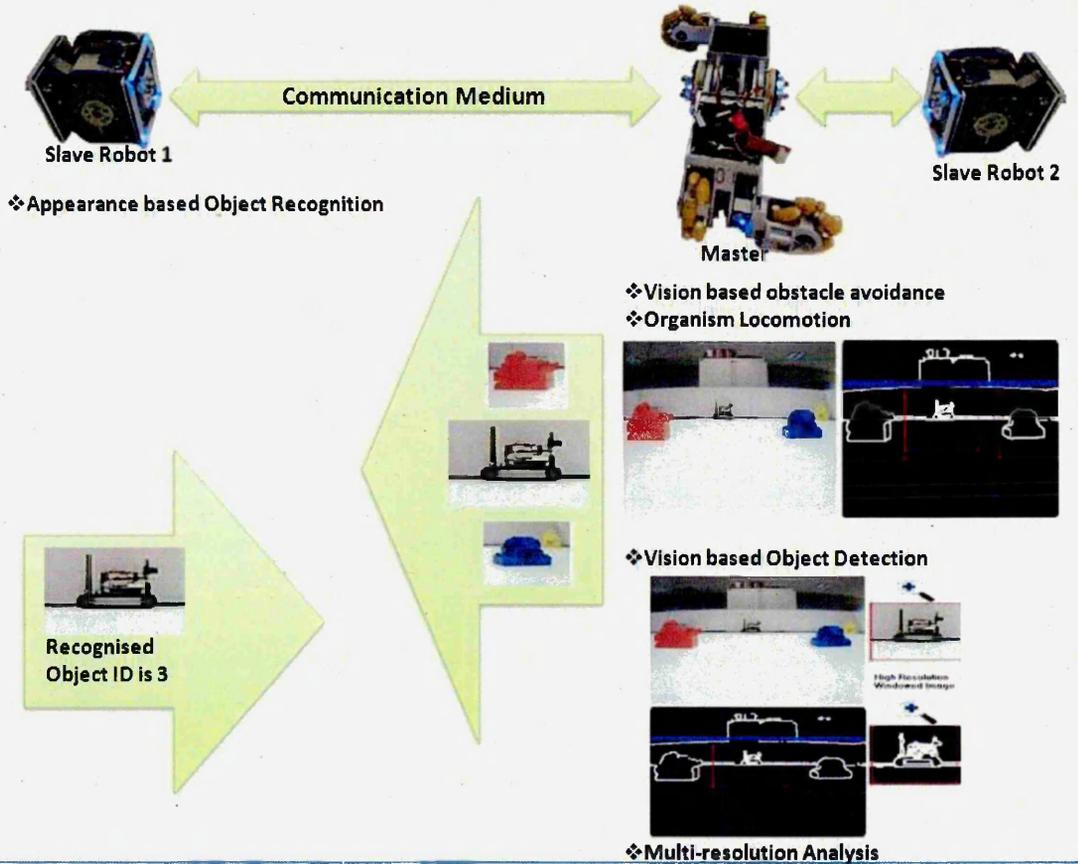


Figure 7.12: Distributed object recognition within the organism.

Here, the master robot also determines that these parts of images are resulting from the objects placed near or far from the organism. Based on the distance to the detected object, the master robot utilises the multi-resolution switching technique which is described in detail in Section 6.2.3. For the objects detected near the organism, the parts of images are extracted from the QVGA (320x240 pixels) resolution image and if the object is found far, then extraction is performed from VGA (640x480) resolution. In case of Figure 7.12, two blocks in

the image are found near and their information is extracted from QVGA resolution. But the robot in the image is detected far from the vision system, so high resolution switching is performed and information from VGA resolution is extracted. After performing feature detection and multi-resolution analysis, if the master robot does not find enough features in the image, then it discards these images and keeps on moving in the environment. In case, enough features are found then it distributes the extracted image parts to the slave robot. On the other hand, the slave robot keeps the features in the library for the target objects. This library is provided in the form of SURF extracted features and is obtained after the initial training is given to the robot to recognise the target objects in the environment. This library is usually large in size and can grow depending upon the number of target objects in the library. On receiving a request from the master robot to process the image parts with object recognition algorithm, the slave robot executes the optimised appearance based object recognition technique as shown in Figure 7.12 (optimised SURF features based technique is discussed in detail in Chapter 6). This appearance based recognition technique further utilises the library of SURF features to identify whether the target object is located in the image part. On process completion, the slave robot informs the master robot about the outcome of the recognition algorithm. If one of the target objects was found in the image parts (robot image in case of Figure 7.12), then the slave robot acknowledges the master robot and informs it about the object's identity found after processing the image data. In this case, the master robot saves the complete image (from which the parts were extracted to distribute to the slave robot) on the on-board memory. Master robot retains a copy of this image in the memory after sending recognition request to the slave robot. In case of negative outcome (i.e. when no object is found), the master robot discards this image copy. Here, one part of the distributed vision processing, involving one master and slave robot, ends.

7.2.3 Distributed Information Gathering

Once the positive outcome of the distributed recognition is obtained from the slave robot, then the organism has to gather the visual information from the object surrounding. This is shown in Figure 7.13 where slave 1 robot is providing *Object ID* information to the master robot and master robot is preparing to

request slave 2 robot for performing *Distributed Information Gathering* task.

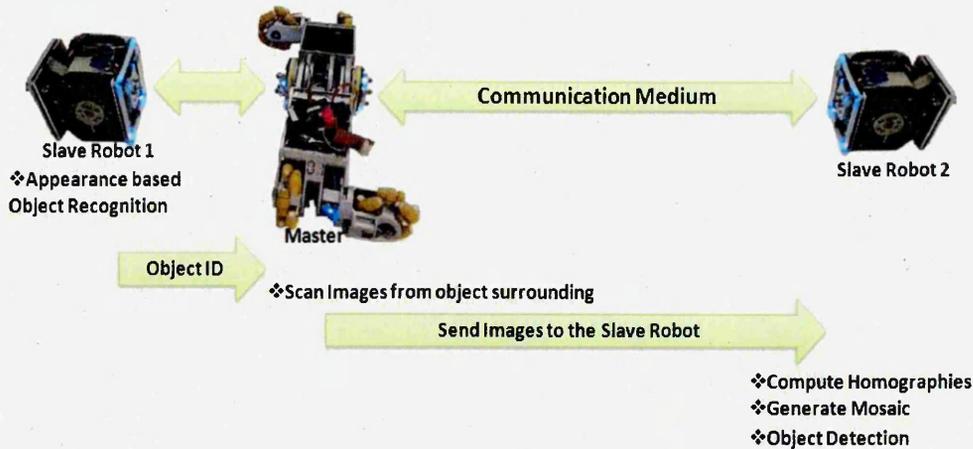


Figure 7.13: Distributed information gathering - task communication.

To gather object surrounding information, the control algorithm in the master robot moves the organism in backward direction. It is performed for two reasons. One is to reverse the motion done by the robot during the time when the slave was performing the recognition task. And second is to bring the recognised object back in the field of view of the organism so that the organism can see the surrounding objects with reference to the identified object. After moving backward, the master robot gives a big rotation to the organism in left direction. Finally, it rotates the organism in right direction and scans the environment around the object in terms of images. Here, it is to be noted that, as the organism generates the mosaic from the images obtained after the scanning, so the axis of rotation of the organism is kept constant. That is, the organism's location is not moved forward, backward and shifted right or left during the scanning. If the axis of rotation is not kept constant, then mosaic of images can not be generated as the objects in the images will appear with different scales.

While performing the scan, the master robot utilises the vision based rotation control strategy to rotate the organism before capturing each image. This is to ensure that the organism was rotated at least 10 degrees each time. The use of vision based rotation control was made because, the organism comprised of many robots (i.e., three or more robot modules) and was too heavy. If the time check is made, that is rotation command is given to the motor control board

for fixed time period for each image scan, then sometimes the motors are not able to rotate the heavy organism. This way, the organism scans the images from the same location, essentially providing no surrounding information. The use of visual feedback makes sure that each image in the scan provides some new information and at the same time, it has enough overlap with the last scan image so that image stitching (mosaicing) can be performed properly. While the master robot is obtaining images in the scan, at the same time, it keeps on transferring the QVGA resolution images to another slave robot as shown in Figure 7.13, where master robot stream the scan images to slave 2 robot for further processing. Slave robot receives these images and executes the optimised SURF based algorithm to extract scale invariant features from the images. The slave robot performs matching of SURF features extracted from each two consecutive images. These matching features are processed with RANSAC (“RANdom SAMple Consensus”) algorithm to remove any outlier features. The final matching features are then used to extract Homography matrix between the two images. Homography matrix basically relates the two images with same planar surface in space. The relation of Homography matrix in terms of the corresponding features (or coordinates) of the two consecutive images, is shown as:

$$\begin{bmatrix} {}^1x_i \\ {}^1y_i \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^2x_i \\ {}^2y_i \\ 1 \end{bmatrix} \quad (7.1)$$

$$\begin{bmatrix} {}^1x_i \\ {}^1y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} {}^2x_i \\ {}^2y_i \\ 1 \end{bmatrix} \quad (7.2)$$

$${}^1m_i = {}^1H_2 \times {}^2m_i \quad (7.3)$$

where 1x_i and 1y_i are coordinates of the features points from the first image and 2x_i and 2y_i are the corresponding coordinates of the features points in the second image. And h_{11} , h_{12} , h_{13} , h_{21} , h_{22} , h_{23} , h_{31} and h_{32} are the coefficients of the Homography matrix (1H_2) from image 1 to image 2. The slave robot computes this Homography matrix for every consecutive image. On expanding the above

equations, the relation between the coordinates of the second image and the coordinates of the first image in terms of the coefficients of the Homography matrix is obtained as:

$${}^1x_i = \frac{h_{11} \times^2 x_i + h_{12} \times^2 y_i + h_{13}}{h_{31} \times^2 x_i + h_{32} \times^2 y_i + 1} \quad (7.4)$$

$${}^1y_i = \frac{h_{21} \times^2 x_i + h_{22} \times^2 y_i + h_{23}}{h_{31} \times^2 x_i + h_{32} \times^2 y_i + 1} \quad (7.5)$$

To form a mosaic, the re-projection of all the images on a common plane is required, as shown in Figure 7.14. This Figure shows the manner the re-projection of the three images on the common plane is performed to form a mosaic. To obtain the re-projections on the common plane, the product of all the Homographies is computed in an incremental fashion, and is given as:

$${}^1H_{k+1} = \prod_{i=1}^k {}^iH_{i+1} \quad (7.6)$$

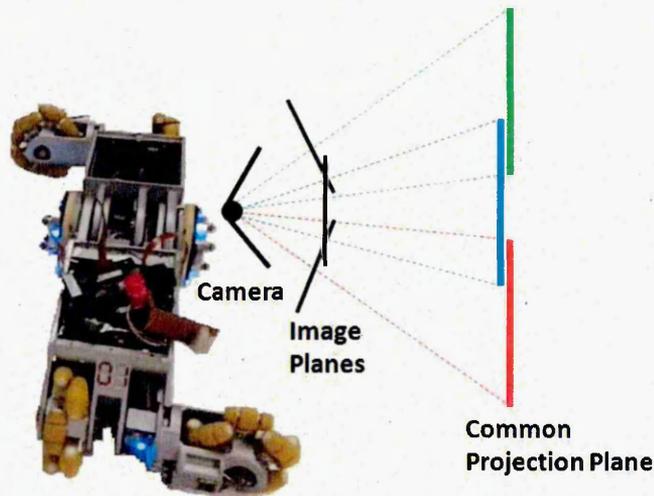


Figure 7.14: Image re-projection on common plane.

At each step when the product of Homographies is computed, the corresponding image (i.e. the image to which last Homography in the product belongs to) is also re-projected on the common plane. For example, the first image is projected straight away on the common plane. The second image uses the

Homography 1H_2 . The third image uses the product of Homographies 2H_3 with 1H_2 and so on. This relation of the product of Homographies with the images is shown in Figure 7.15. In this Figure, the reference frame (or common re-projection plane) is shown on which the images are re-projected using the product of Homographies.

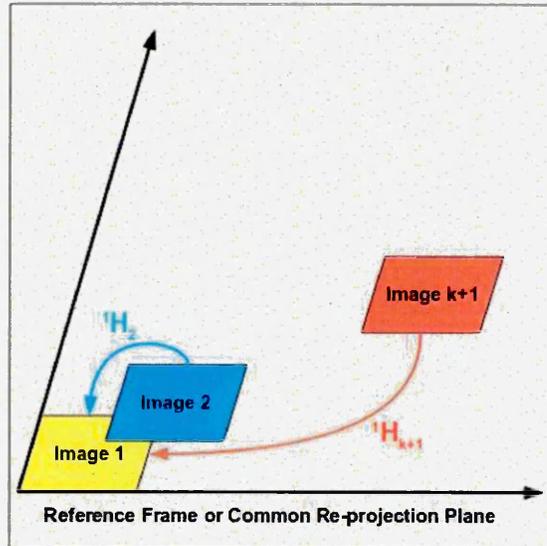


Figure 7.15: Images re-projection using product of homographies.

So in distributed information gathering task, the slave robot uses this image re-projection approach to produce the image mosaic, once it receives the streams of images from the master robot. This is shown in Figure 7.16 where slave 2 robot receives image stream from the master robot and has generated the image mosaic. In Figure 7.16, the four step processing phases are shown which are followed by the slave robot to process all the images. For providing the detail of the image, the zoom in version of these four processing phases are also shown in Figures 7.17 and 7.18.

The first step in Figure 7.17 generates the image mosaic. The image mosaic is a large image providing the bigger view of the object surrounding. It is difficult to process this large image so that the different objects in the environment can be related with each other. So here, rather than processing the complete image with computationally heavy feature extraction approach, it was decided to identify the parts of image containing the objects. And then these parts of the mosaic image will be processed by the appearance based approach. This makes

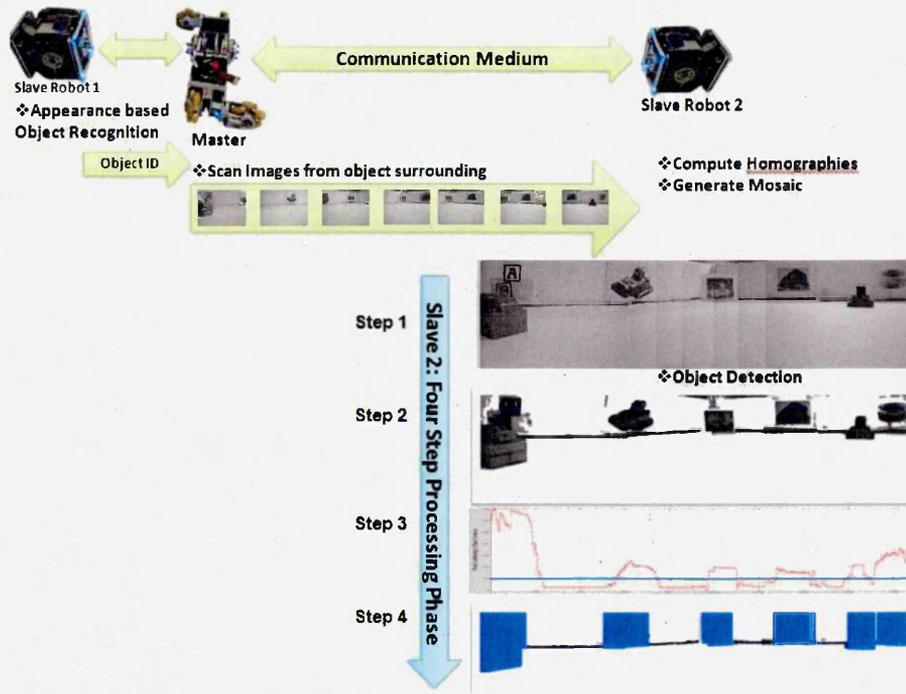


Figure 7.16: Distributed information gathering - vision processing phases.

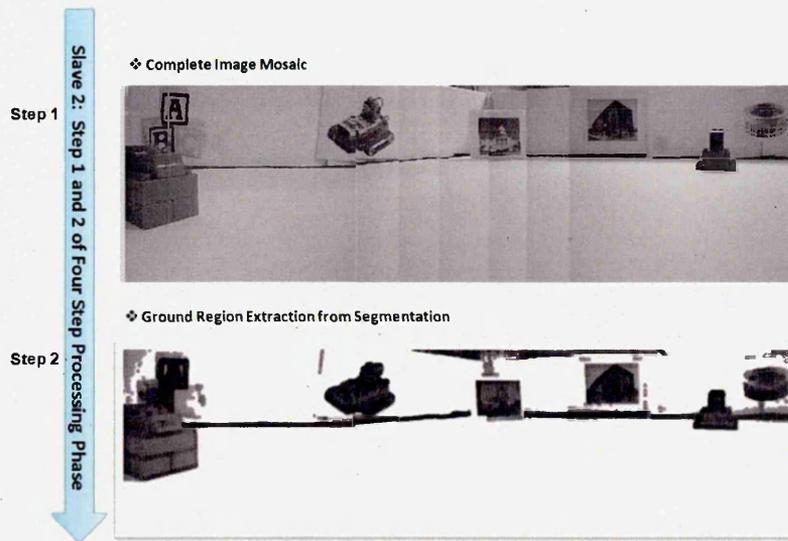


Figure 7.17: Slave 2 :Four step processing phases - Step 1 and 2.

the approach suitable to be implemented in an embedded system environment. For the detection of the objects in the image, first of all the segmentation of the complete image is performed and the regions resulting from the ground and the boundary wall are isolated. In this case, the ground region surface and

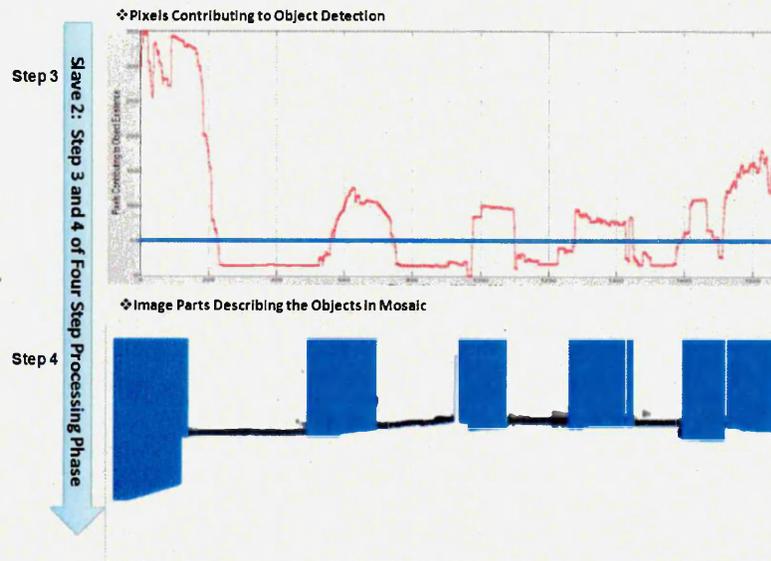


Figure 7.18: Slave 2 :Four step processing phases - Step 3 and 4.

the boundary wall appear to be the same so they appear in the same region after segmentation. This is shown in Figure 7.17 as the second step of the processing phases. The output from the second step is further processed in step 3 to determine the number of pixels contributing to the object detection. The number of image pixels in a mosaic, contributing to the object detection in each column are determined and the generated profile is shown in the step 3. This profile is thresholded and the columns where the profile exceeds the threshold, indicate the presence of an object. Finally in step 4, the pixels contributing to the object are filled with blue colour. It can be seen that, six objects in the mosaic are correctly detected and isolated.

These mosaics are generated to provide surrounding information for each target object. After detecting the objects in the image mosaic, these objects can be assigned labels and their appearance based information (in terms of SURF features) can be extracted by the slave robot which eventually contributes to building the features library. These mosaics also provide the location of these newly detected objects with reference to the target objects. After adding these new objects in the library, when they are re-visited and recognised by the organism, then it provides surrounding awareness to the organism about what it may expect around the object. The information gathered from mosaics, may also help the organism to reach other objects. In other words, this information

can help organism to navigate in the environment. For example, if an organism has to reach certain objects then it matches the objects in its field of view with the features of the object it is looking for. If there is a positive match, then the organism is in-front of the target object. Otherwise, the organism compares the objects in its field of view with the objects in the library. If it finds it in the library, and the library also provides a directional hint to the target object with reference to the current object in the field of view, then it facilitates the organism to reach the target object in less time. Hence, it is expected that these gathered information contribute to the learning mechanism in the organism and enable to provide support to the higher level of autonomy within the organism. It is to be noted that, in the described distributed vision processing architecture, if any of the robot modules malfunctions or fails, then the task can still be performed if another robot module with similar functionality as of the failed module, exist in the robotic organism.

7.3 Experiments with Robotic Organism

In this section, the experimental results obtained from the described distributed vision processing scenario in the organism mode, are presented. Many experiments were performed to practically demonstrate the two phase distribution of the vision based tasks within the organism. As already mentioned, the "Replicator" robots are not fully functional, so a multi-processor robotic organism designed in this research was used for demonstration purposes. For convenience, this robotic organism is again shown in Figure 7.19. Due to the hardware limitations, the organism has one master robot with locomotion and processing capabilities and one slave robot with only processing resources. This slave robot performs the distributed object recognition together with the master robot. Once the object is found, the master robot scans the environment and performs the information gathering tasks using the same slave.

The robot organism was given training to recognise the three target objects shown in Figure 7.20. Building images were used as objects. These images are 2D objects and single image will be sufficient for training. Unlike the work presented in Chapter 6, 3D objects were not used, so that the research is mainly focused to the distributed vision processing rather than the complexities of 3D

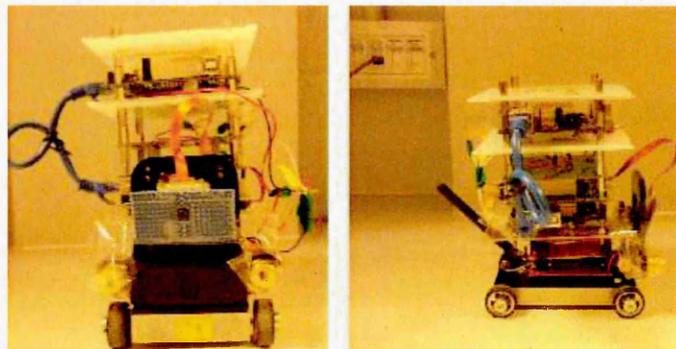


Figure 7.19: Multi-processor robotic organism.

recognition. The extracted SURF features of the target objects were provided in the form of a library to the organism and they were stored in the memory of the slave robot. The slave robot used this library to help the master robot in performing the distributed recognition task.

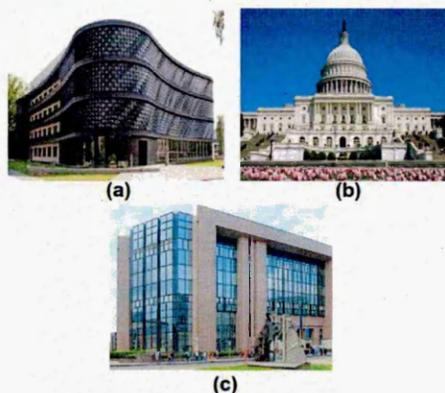


Figure 7.20: Target 2D objects.

The basic idea of vision task distribution is to share the processing load between multiple robot modules. So, before proceeding to the results obtained from the targeted distributed vision scenario, first of all the execution time performance comparison is made between a single robot implementation and the organism. The performance evaluation is done in two parts. In the first part, the performance of the refresh time of the vision based obstacle avoidance approach is evaluated between a single robot and the organism. In the second part, the distributed object recognition approach is evaluated with reference to the object recognition performed by a single robot. The refresh time of the vision

based obstacle avoidance means how frequent this algorithm is called by the robot control algorithm. The more frequent this algorithm is called, the more the robot can show continuous locomotion and receives frequent surrounding awareness. This helps the robot in decision making.

To evaluate the refresh time of obstacle avoidance algorithm, a test was performed. In this test, the control algorithm ran the vision based obstacle avoidance, object detection, distance based resolution switching and finally object recognition in a sequence. A check was made before obstacle avoidance was called, to determine the frequency of its execution. In a complete sequence, the obstacle avoidance algorithm was called only once to tell the robot whether it was close to an obstacle or not. In the test, a single robot from a swarm was kept static at a fixed distance from an object. The test was run for some time and results were recorded. In the same configuration, another test was performed, but with the robot organism, and the frequency of the algorithm call was recorded. The results obtained from a single robot test and also the organism, are shown in Figure 7.21.

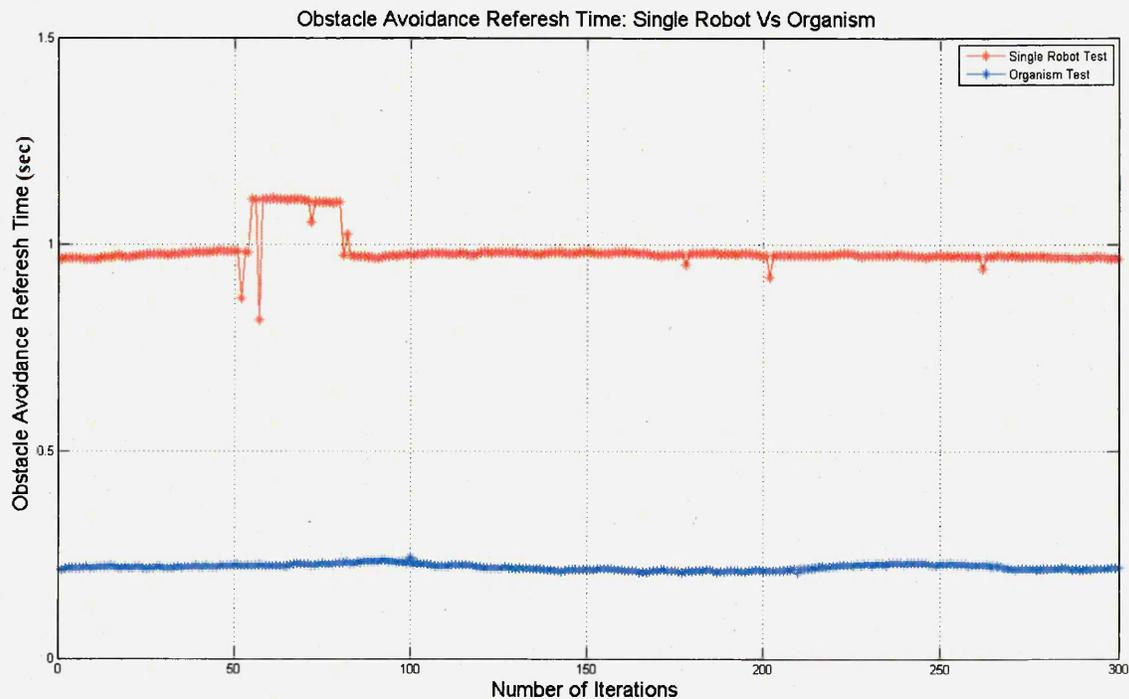


Figure 7.21: Obstacle avoidance refresh time: Single robot vs robot organism.

The refresh time profile obtained from a single robot test is shown in red colour

and the profile obtained from the organism test is shown in blue colour. In the single robot experiment, as the single robot unit was responsible for executing all algorithms, so the average refresh time calculated for the obstacle avoidance task was 950 milli-seconds. In case of the robot organism, the master robot was responsible for the obstacle avoidance, object detection and distance based resolution switching task. As already mentioned for the computationally heavy object recognition tasks, the master robot transferred the processed vision data to the slave robot. So a high refresh time for the obstacle avoidance was expected. It can also be seen in Figure 7.21 where the refresh time, in case of organism, is 230 milli-seconds on average. This means that the obstacle avoidance is called after every 230 milli-seconds, so the frequency to call the algorithm is higher. This refresh time helps the organism to show a continuous motion without waiting for the outcome of the object recognition algorithm. During the test the object, that was placed at some distance from the robot, was later moved closer to the robot. The vision system now produced a more detailed visual information about the object. This caused more features in the object recognition algorithm and hence, the algorithm took longer to process the vision data. This increase in time, from iteration number 50 to 80, can be seen in Figure 7.21 for the case of a single robot test. It is noticed that the time profile in case of the organism remained the same. This is because the organism transfers this vision data to the slave for processing and the data transferring time is much less due to fast Ethernet communication medium. So the execution time on the slave side may rise, but the refresh time for the obstacle avoidance does not change. Here, the first part of performance evaluation finishes. In the second part, the evaluation of the distributed object recognition approach with reference to the object recognition performed by a single robot, is performed. The performance of the optimised object recognition approach, on a single robot in a swarm, is discussed in detail in Chapter 6. The evaluation of this approach, in terms of execution time and recognition, with reference to the conventional SURF based approach is also explained in Figures 6.6 and 6.7 of Chapter 6. Here, first of all the execution time of the recognition algorithm on a single robot, with varying object features, is determined. The profile obtained is shown in Figure 7.22.

In Figure 7.22, the number of iterations means the number of times the exper-

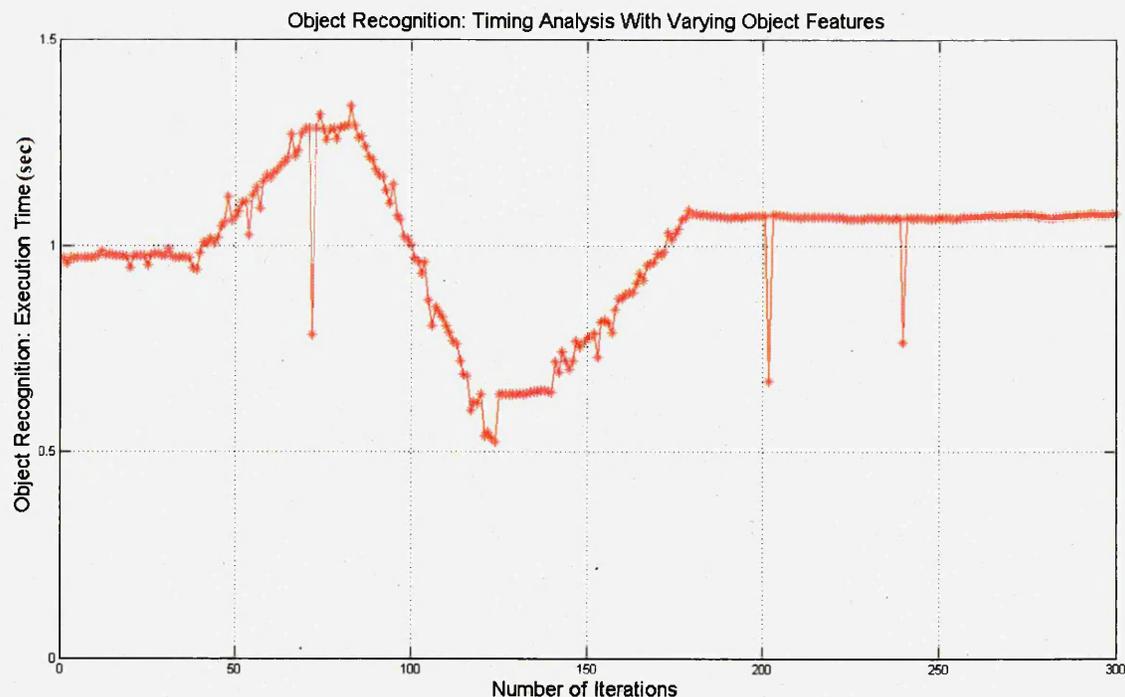


Figure 7.22: Timing analysis of object recognition algorithm with varying object features.

iment was performed. From iteration number 1 to 47, the object was kept at fixed distance from the vision system of a robot. The average time taken by the algorithm was 940 milli-seconds. From iteration number 48 to 66, the object was gradually brought closer to the vision system. The image from the vision system started providing a more detailed information. So, the number of object features rose as the object was brought closer and hence, the execution time of the algorithm also rose to 1.3 sec. Then the object was kept in that position for some time and then brought away from the vision system gradually. This effect can be seen in the Figure where the execution time reduced from 1.3 sec to 0.5 sec in iteration 80 to 120. When it reduces below 750 milli-seconds, then the successful recognition is not performed due to reduced number of features in the image. Again when the object was brought closer, then the rise in the execution time can be seen. In the test, some spikes in the profile of execution time are observed. These are caused because the image provided by the vision system was blur, which produced reduced features and hence a reduction in execution time.

For the evaluation of the distributed object recognition with reference to the

implementation on single robot, another test was performed. The execution time profiles obtained are shown in Figure 7.23. The object was brought closer and then moved away gradually from the vision system. To keep the vision data the same for the organism and single robot implementation, the image data were initially saved on the on-board memory. This data were later used for object recognition on a single robot and also for distributed recognition purposes. In case of a single robot, the profile obtained is shown in black colour. In this time, the robot executes the vision based obstacle avoidance, object detection, distance based resolution switching and object recognition (as the image data were read from flash rather camera, so a delay of 100 milli-seconds was added as this is image capturing time from the camera.). On the other hand, the execution time profile obtained from the distributed recognition is shown in red and blue colours. The profile in blue colour is the execution time on the master robot. It is 280 milli-seconds on average. In this 280 milli-seconds, the master robot executed the vision based obstacle avoidance, object detection, distance based resolution switching and also vision information communication to the slave robot. The object recognition part was performed in distributed fashion by the slave robot. The slave robot performed the feature extraction and the matching part using the features library. The profile obtained from the slave robot is shown in red colour. When the red and blue profiles were added together the black profile is expected. In this case, a difference from 1 to 10 milli-seconds was observed. This was due to information exchange between the master and slave robots. From the profiles shown in Figure 7.23, it can be seen that the distributed implementation of object recognition in the organism has effectively distributed the processing load between the master and slave robots.

Now to test this distributed object recognition approach in the main distributed vision processing scenario, apart from the target 2D objects shown in Figure 7.20, some other 2D objects (in the form of images but unknown to the organism) were also used in the test environment. Some of these unknown 2D objects are shown in Figure 7.24. The reason for using these unknown 2D objects in the environment is two fold. One is to demonstrate that the organism not only recognised the target objects, but also recognised their identity in the presence of the other objects in the environment. This links to the recognition performance in the distributed object recognition phase. And second is linked

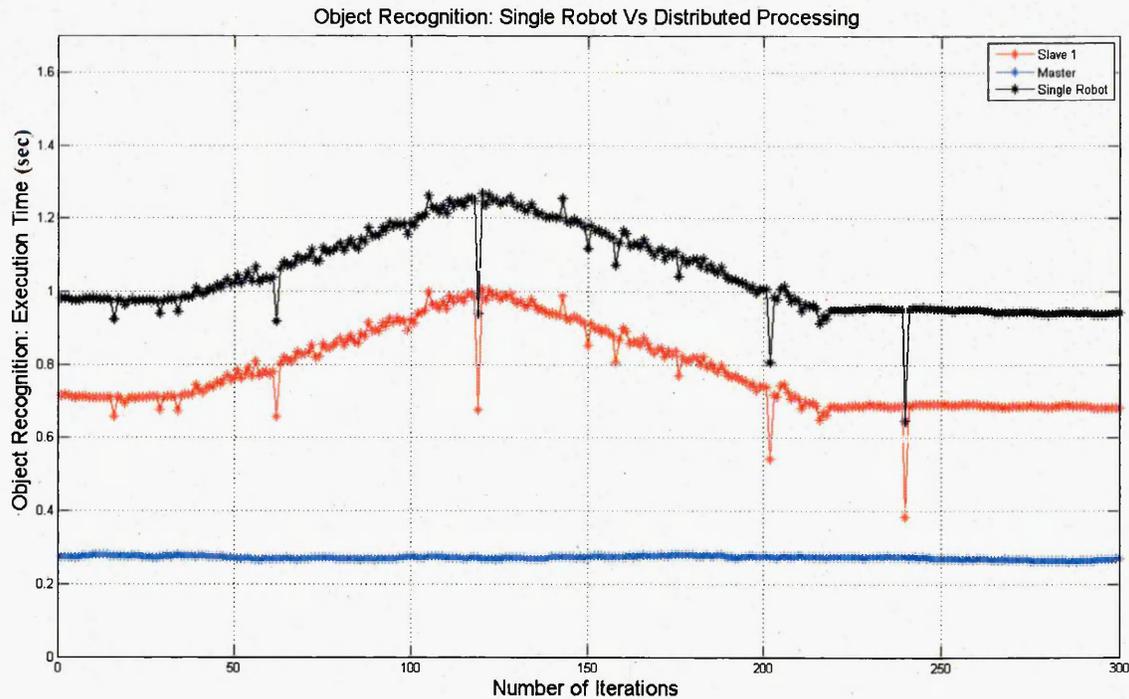


Figure 7.23: Object recognition: single robot vs distributed processing.

to the distributed information gathering phase. During the second phase, the robot gathered surrounding information of the object in one large image by forming a mosaic. While forming a mosaic, when robot stitched two images together, the vision algorithm required salient features in the environment, which were found common in the two consecutive images. So for this purpose, it was necessary to add more objects in the environment so that common objects could be found in the consecutive images.

Apart from these unknown 2D objects, some obstacles (blocks) were also placed in the environment. These obstacles, unknown objects and target objects were kept in the test arena, where distributed vision processing experiments were performed. The test arena is shown in Figure 7.25. In the test arena, the placement of the target objects is also identified as Objects 1, 2 and 3.

Here, the results obtained from one of the distributed vision processing experiment are discussed in detail. In the experiment, the starting position of the robot when the test began is shown in Figure 7.26a. The robot started its task by doing the vision based obstacle avoidance and at the same time, distributing the objects detected in its current view to the slave robot for further processing.

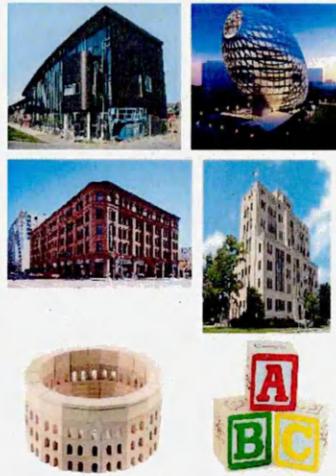


Figure 7.24: Unknown 2D objects.

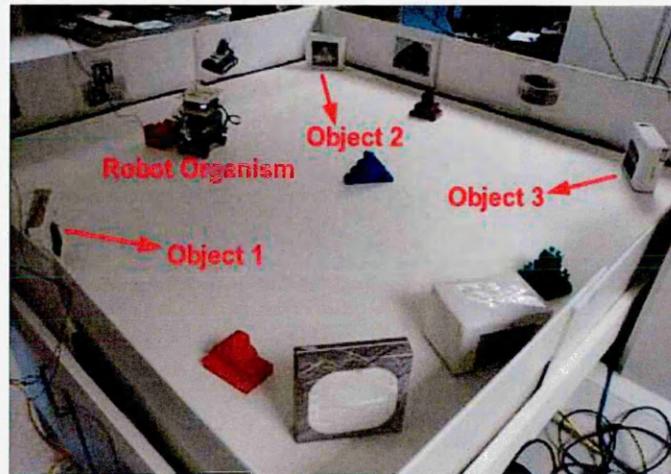


Figure 7.25: Test arena for distributed vision processing in organism mode.

The position of the robot organism when it found the first object is shown in Figure 7.26b. The object which the organism processed to identify and the trajectory made by it during this time is also shown in Figure 7.26b. The object ID identified by the slave robot is “2”. As mentioned before, when the slave notified the master robot about the successful detection of the target object, then the master robot saved the image in which the object was identified, to its memory. The coordinates where the object was identified, were also determined by the slave and they were projected by the master robot on the image. The image of object 2, stored by the master robot is shown in Figure 7.26c. The location in the image, where the slave robot has found object 2, is also shown

in the Figure.

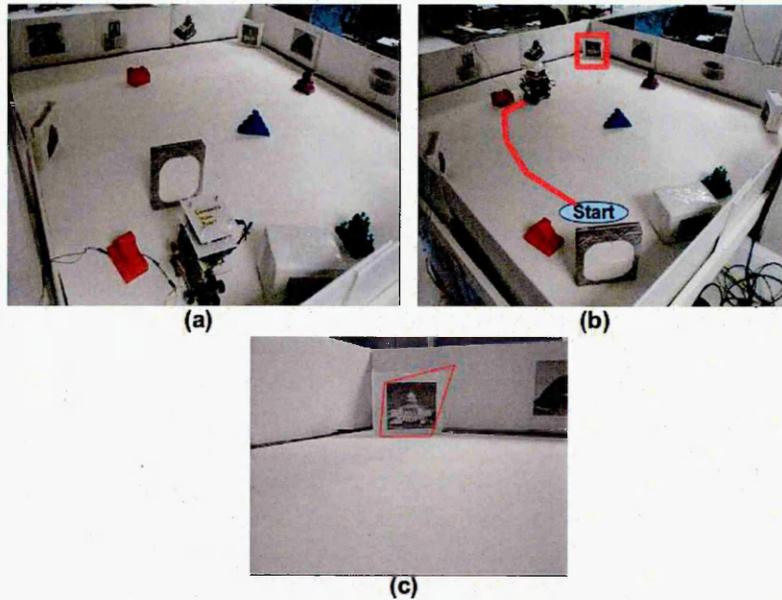


Figure 7.26: (a) Position of the organism when the test begin. (b) Position of the robot when it has identified the target object 2. (c) Identified location of the target object 2 in the image.



Figure 7.27: Object 2 surrounding scanned by the organism.

Once the object is identified, the organism scanned the environment to perform distributed information gathering task and formed a mosaic. In this experiment, the area the robot organism had scanned is shown in Figure 7.27.

During this scan, the organism grabbed 10 images for making a mosaic. These 10 images are shown in Figure 7.28. The distributed information gathering phase started and the formation of mosaic image began for the current scan.

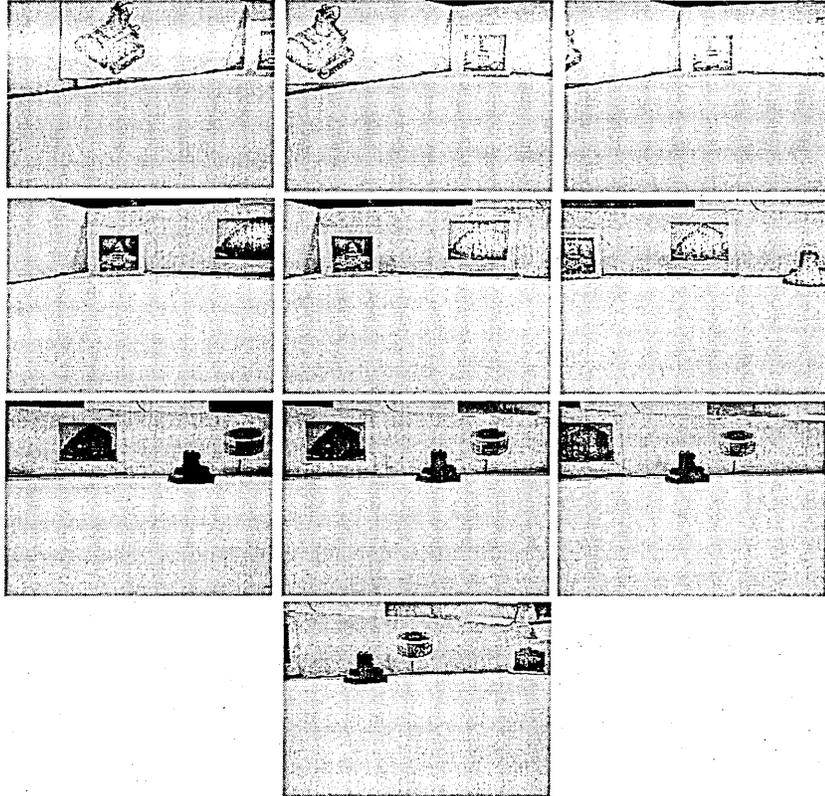


Figure 7.28: Images grabbed by the organism to form a mosaic.

In Figures 7.29 and 7.30, the complete process of the mosaic formation is shown. This Figure shows the progress in the mosaic formation when the images are stitched one by one. Finally, in Figure 7.31, the process to detect the objects in the mosaic is shown. Figure 7.31a shows the results obtained after ground region and arena boundary walls are eliminated using the segmentation approach. Apart from segmentation, image dilation is also performed to connect the pixels which are describing the objects in the image. In Figure 7.31b, the profile representing the density of pixels in describing the objects in each column of the mosaic is shown. Based on the pre-defined threshold (shown in blue colour in Figure 7.31b), the number of objects identified are shown in Figure 7.31c. The pixels describing the objects are identified by blue colour.

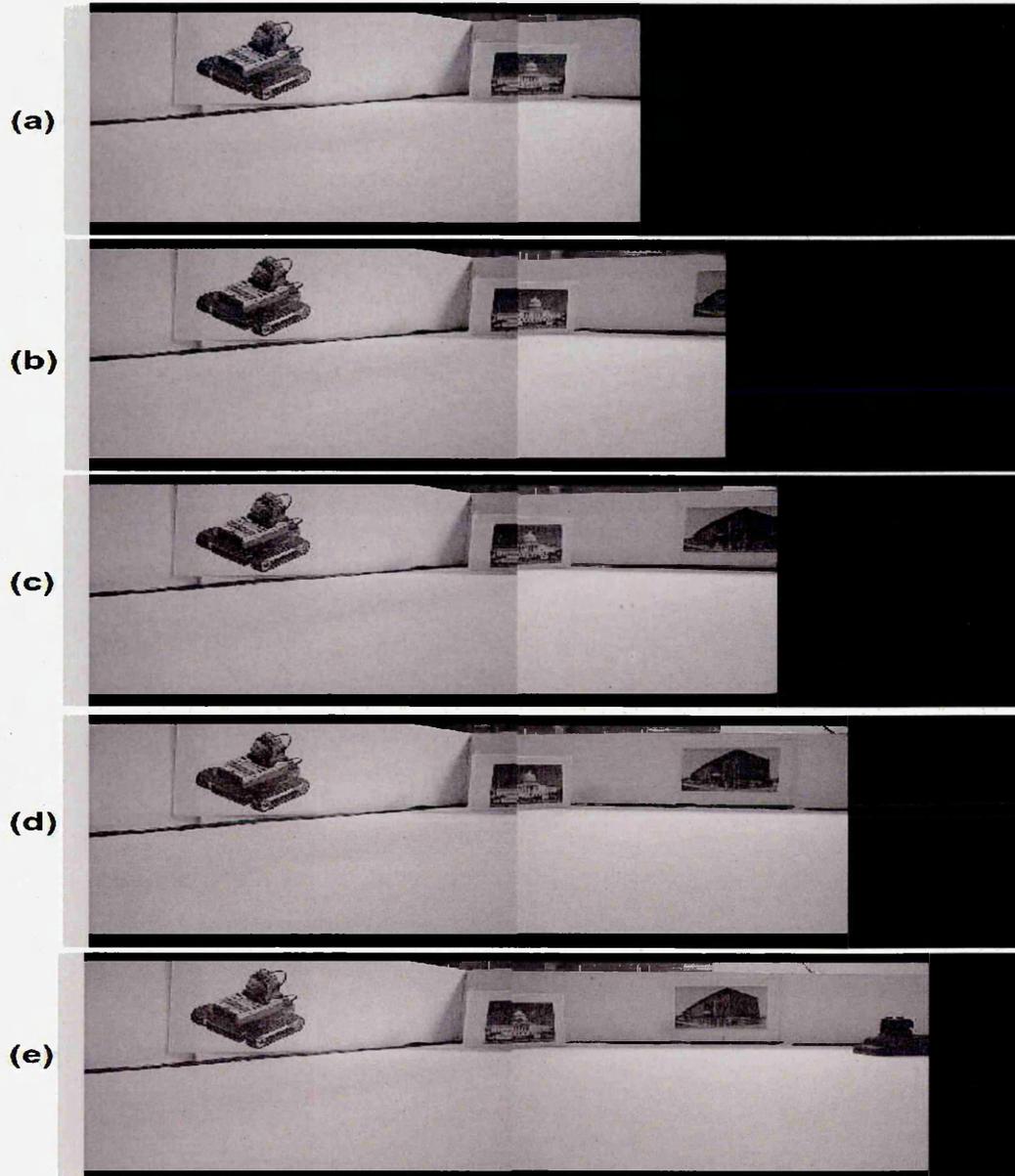


Figure 7.29: Mosaic of images scanned around object 2.

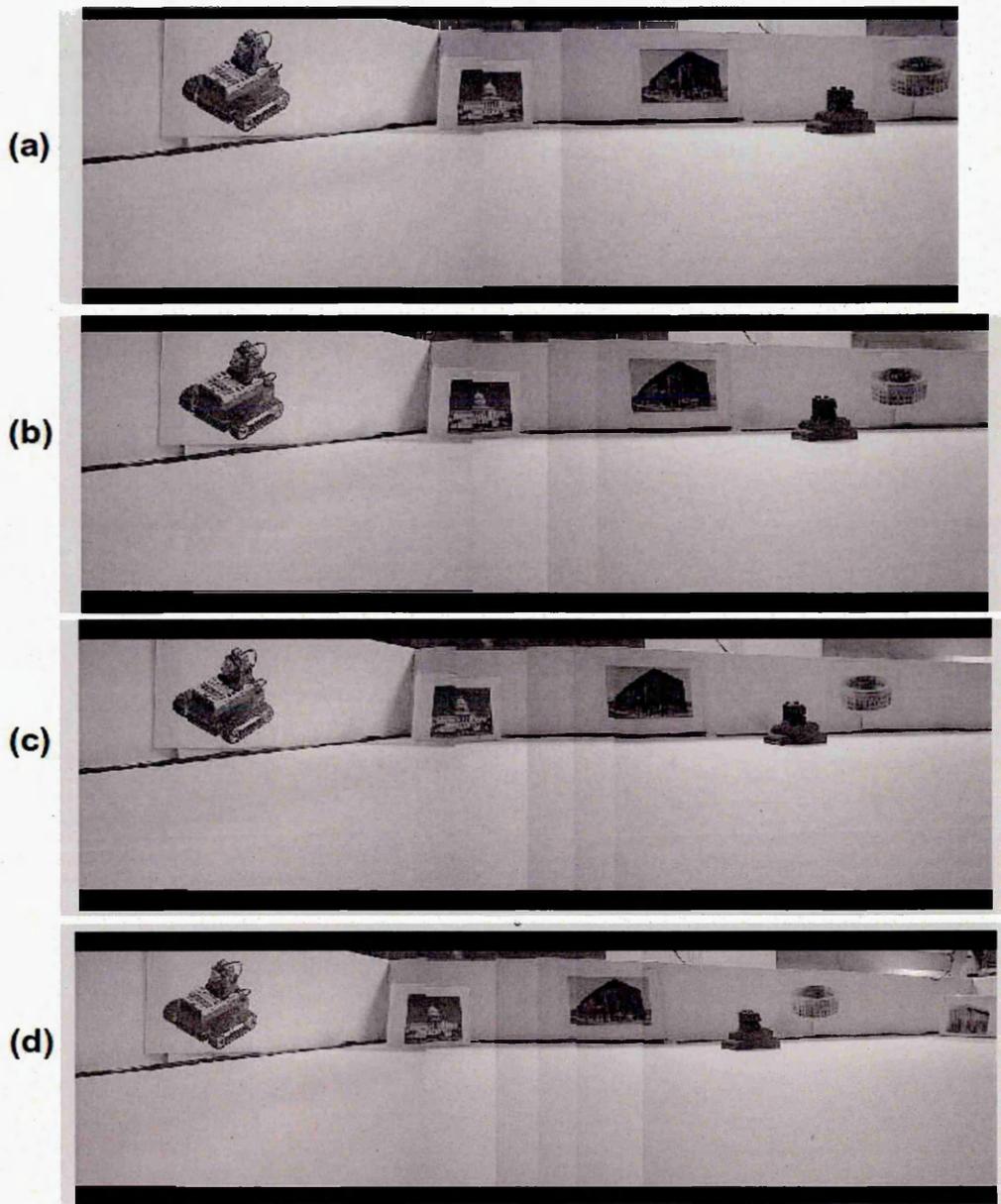


Figure 7.30: Mosaic of images scanned around object 2.

The complete trajectory made by the organism, in searching of the target objects and forming the mosaics, is marked by red colour in Figure 7.32. The locations in the arena, where mosaics generation were performed to gather the surrounding information of the target objects 1, 2 and 3 are also shown in green, yellow and blue colours, respectively.

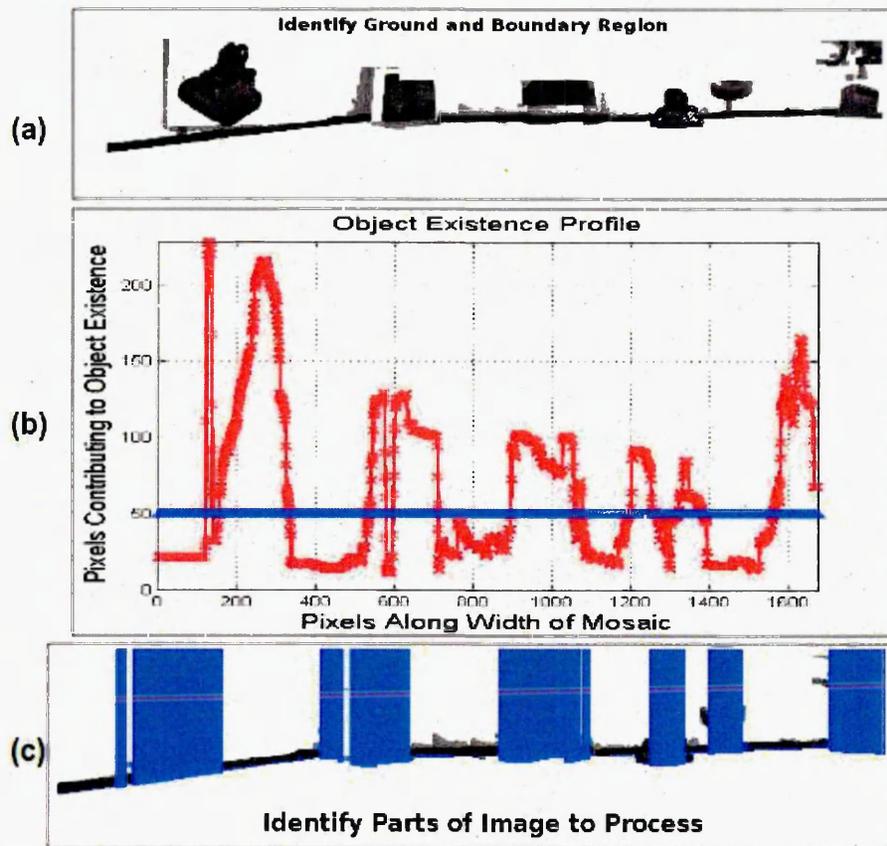


Figure 7.31: Object detection in complete mosaic.

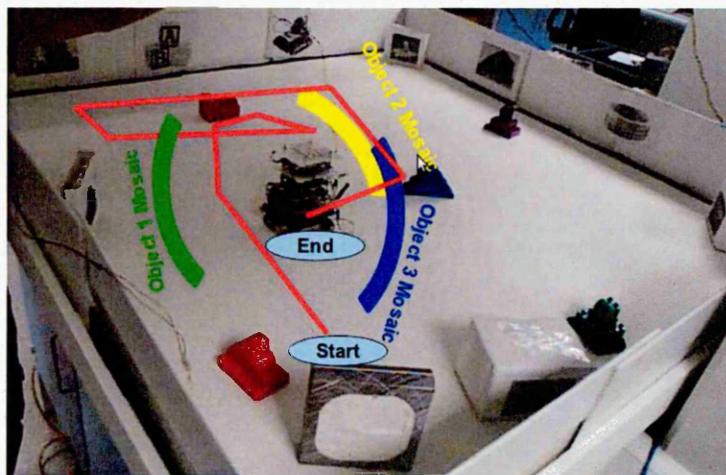


Figure 7.32: Trajectory made by the organism and location from which mosaics were generated.

The mosaics information gathered for target objects 1 and 3 is shown in Fig-

ures 7.33 and 7.34, respectively. All the objects in the mosaic view were properly detected and isolated from the ground surface.

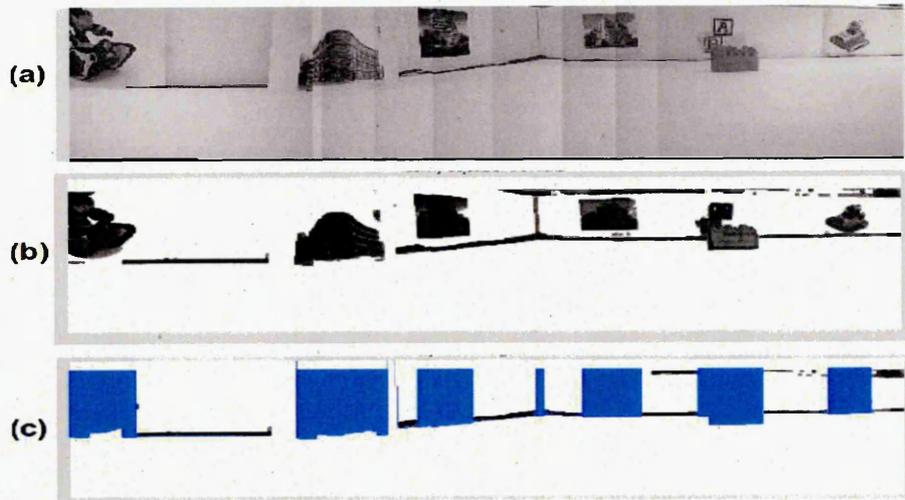


Figure 7.33: (a) Mosaic from the images scanned around object 1. (b) Segmentation based ground elimination. (c) Objects detection in mosaics.

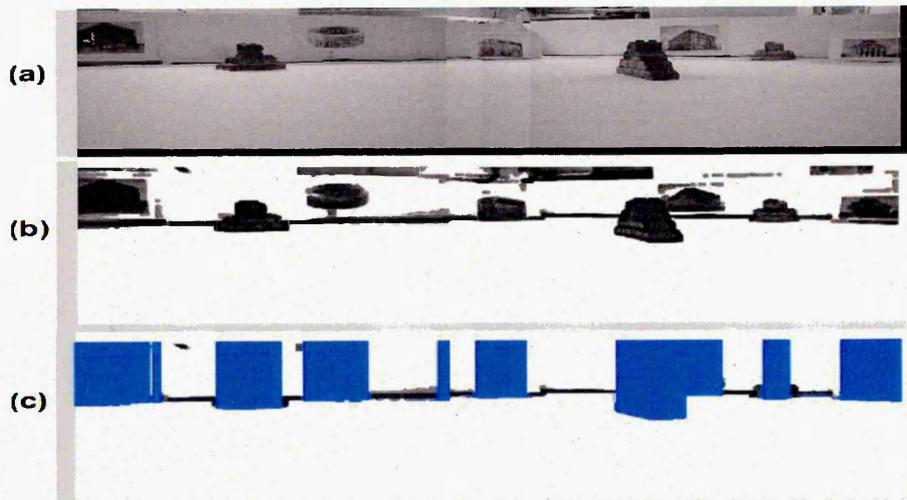


Figure 7.34: (a) Mosaic from the images scanned around object 3. (b) Segmentation based ground elimination. (c) Objects detection in mosaics.

It can be noticed that, the objects in the mosaics appeared very small and it seemed that they were difficult to recognise. In the beginning, QVGA resolution images were used for mosaic formation as this resolution was originally selected for generating the Homographies between the images. In this case,

the objects that appeared small can not be recognised. In the results shown in Figures 7.29, 7.33 and 7.34, the QVGA resolution was selected for solving the Homographies between the images, but for generating the mosaics, the VGA resolution was used. To make the Homographies information applicable to the VGA resolution, every element in the Homography matrix was scaled up by 2. This was done as QVGA resolution image could be obtained from VGA resolution by using a scaled down factor of 2. As the mosaics were generated using VGA resolution, so all the objects were present with their detail information. Here again for object recognition, the distance based resolution switching could be performed. The objects which lied near in the mosaic, could be scaled down by a factor of 2 so that they could be processed and recognised fast. On the other hand, the objects that lied far needed to be processed in a higher resolution, so that there were more pixels defining them and hence make the recognition possible.

In one of the distributed vision processing experiments, a problem was observed during the image mosaicing operation. In this experiment, the mosaic information generated is shown in Figure 7.35a. In the beginning of the mosaic, the images were stitched properly. The problem occurred when the last two images were stitched. These images are shown in Figures 7.35b and 7.35c. The information contributed by these images in the mosaic is identified by the blue arrow. After observing the images in Figures 7.35b and 7.35c with the complete mosaic, it can be noticed that these images are stitched at wrong points. The two correct corresponding points, showing the place where the image stitching should have been performed, are identified in the mosaic with red arrow. The reason for this error can be found very easily in the last two images. There is a sufficient overlap between these images but there are no objects in this overlapping region. The presence of objects in the common part of images result into the detection of salient features. These features help in determining the corresponding 2D points between the images which are later used to determine the Homography information for generating a mosaic. This is the reason that many other objects (in the form of images shown in Figure 7.24) were placed around the target objects. As the robot organism had to generate the mosaic of the images providing the surrounding information of the target object, so the presence of objects around the target objects helped in the detection of

features in the image, and hence facilitated the mosaic generation.

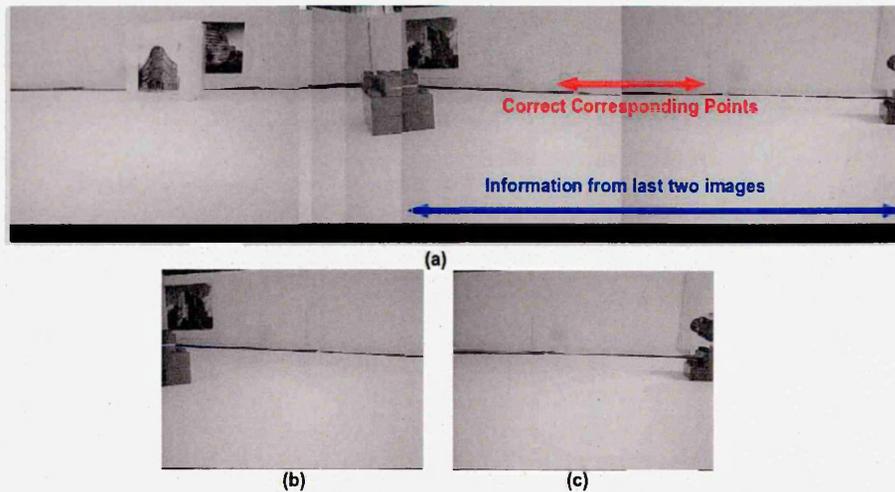


Figure 7.35: (a) Erroneous image stitching. (b) Second last image. (c) Last image.

7.4 Conclusions

In this Chapter a distributed vision processing scenario in the organism mode has been presented. It has been concluded that, the provision of a reliable communication medium (i.e., the Ethernet) in the robotic organism, can be effective for exchanging the visual information between the robotic modules. But the communication medium is the hardware feature of the underlying system. From the scenario presented in this Chapter, it is found that the most critical software aspect of the distributed vision processing in multi-robotic systems is the modular implementation of the vision processing tasks. From the presented scenario, it has been concluded that the robotic modules in the organism, not only efficiently utilise the communication bandwidth by sharing the heavy visual clues, but they also make an efficient use of the distributed processing resources by sharing the vision processing load between the modules. However, it is observed that, for obtaining effective surrounding awareness, the features of the detected objects in the mosaics needed to be added to the vocabulary and also learned by the organism. The addition of a new object features significantly increases the memory requirement. Therefore it has been concluded that the efficient utilisation of memory resources is critical when dealing with distributed vision processing applications.

Chapter 8

Distributed Object Classification and Recognition in a Robotic Organism

Following the research work presented in Chapter 7, this Chapter also focuses on distributed vision processing in modular robotic systems, which are reconfigurable and capable of producing different forms of three dimensional robotic organisms. As described in Chapter 7, once a robotic organism is formed, then based on the information from the vision sensor, the distributed implementation of the object recognition and mosaic formations can provide a rich surrounding awareness to the robotic organism. For recognition purposes, the robotic organism relied directly on the SURF features of the target objects. This approach works well if the number of target objects is small. But if the number of target objects is large, or if the robotic organism extracts SURF features of the objects detected in the mosaics so they can be recognised in the later stage, then this causes a significant increase in the total number of features (i.e., features space) representing the target objects. Hence, it becomes difficult to process these large number of features in real time and this makes the recognition process very slow. So ensuring a faster recognition in the robotic organism is needed to be focused as it performs an important role in the overall performance of the robotic organism. Following the Literature Review presented in Chapter 2, to achieve a faster recognition, the SURF features space clustering approach can be used [61]. The concept of clustering the

feature space is called feature space quantisation in some approaches [63] and it provides significant savings in memory and thus makes the approach more suited for implementation on embedded systems. It is expected that, this feature space clustering approach facilitates the robotic organism to classify and recognise the different objects in the environment in reduced time. It is to be noted that, considering the nature of the distributed processing environment of the robotic organism, the classification and recognition tasks are needed to be implemented in a modular fashion. The modularity in the implementation of vision based tasks is a challenge as it requires different robotic modules in the organism to form a feedback system, where each robotic module is assigned a specific responsibility. The complete system is required to be strictly synchronised, such that each robotic module participates in processing the visual data at some stage and passes on the results to the other robots for further processing.

In this Chapter, a distributed vision processing scenario is considered which aims to present a modular implementation of vision based object classification and recognition approach, using the distributed memory and processing resources of a robotic organism, comprising of four robotic modules. Each of the robotic modules in the organism is given a specific task, which facilitated the organism to classify and recognise the objects. In the beginning, the organism is able to recognise only four target objects, where one of the robot modules in the organism held the knowledge of these objects. During the operation, if a new object is shown to the organism, then the organism failed to classify it and used its distributed processing resources to learn this newly encountered object. For this purpose, a robotic module in the organism triggered a learning mechanism. Once the new object is learnt, then it could be correctly classified and recognised by the organism, if it is encountered again. This shows the manner the organism learning mechanism evolved with time. The remainder of this Chapter is divided into the following Sections.

- (i) Multi-processor Robotic Organism
- (ii) Communication within the Robotic Organism
- (iii) Vision Task Distribution for Distributed Object Classification and Recognition
- (iv) Experimental Results

(v) Performance Comparison of Distributed Modular Robotic System versus High Processing Systems

Section 1 briefly describes the hardware considered for implementing the proposed object classification and recognition approach. Section 2 explains the manner communication medium is established between the multiple processing modules within the organism. Section 3 discusses in detail the division of the vision processing task among four processing modules and the manner, the participating modules form a feedback system to accomplish the underlying tasks. In Section 4, the results are presented with detailed discussions. Finally, in Section 5, the performance obtained using modular implementation of object classification and recognition approach on a distributed robotic organism, is compared with its implementation on high processing systems.

8.1 Multi-processor Robotic Organism

To implement a vision based object classification and recognition approach in a modular fashion, first of all a robotic organism was required, which could provide distributed processing resources for the tasks. An example robotic organism is shown in Figure 8.1, where four robotic modules join together to provide the distributed processing environment. The physical docking ports which provide a communication gateway between the different robotic modules, are also identified in red colour in Figure 8.1. To keep the focus of this study on the distributed vision processing, the robotic organism shown in Figure 8.1 is simulated by a multi-processor robotic system as described in detail in Chapter 3. This multi-processor robotic system is also shown in Figure 7.19. For the scenario described in this Chapter, as four robotic modules needed to be simulated, so four Analog-Devices Blackfin processors were used on this multi-processor robotic system. To integrate these processors together, four Evaluation boards EVAL-BF5xx (shown in Figure 3.5) were also used.

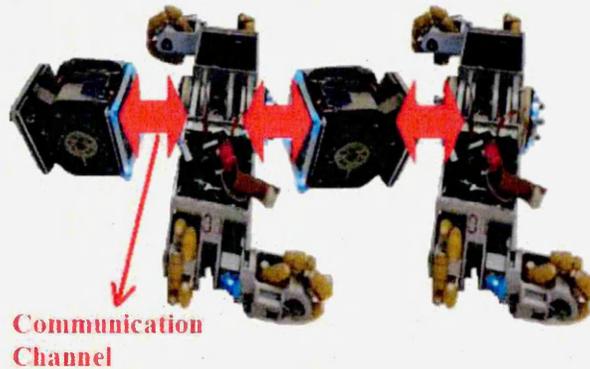


Figure 8.1: REPLICATOR robotic organism [2].

8.2 Communication within the Robotic Organism

As described in Section 7.2.1, to establish communication between the multiple processing modules of the multi-processor robotic system, a high speed Ethernet communication medium (10/100M bits/sec) was used. The multi-processor robotic system used in this scenario comprised of four Blackfin processors, where each processor simulated the computing resources provided by one robotic module in the organism. A robotic organism can comprise of many robotic modules and there can exist many communication strategies to synchronise the information exchange between these robotic modules.

The communication strategy used in this scenario was similar to the one developed in Section 7.2, and it is briefly described here. To establish communication, all the robotic modules were required to determine the identities of their neighbouring robots in the organism. To achieve this, all the robots in the organism were programmed to broadcast a “Breathing Frame” which was 100 bytes. This frame contained the MAC (Media Access Control) address of the sender robot. The MAC address was used as robot ID because the communication was performed at Ethernet layer. Figure 8.2 shows the manner Breathing Frame is exchanged between different processing modules. On receiving the Breathing Frames, each robot module started populating an address table. In this table, it kept the identification information of its neighbouring robots and also their status, i.e., active or inactive. For example, in Figure 8.2, robot 1 has identified three robot modules in its neighbourhood with MAC addresses “3A:15:02:56:44:42”, “3A:15:02:56:44:43” and “3A:15:02:56:44:44” and has de-

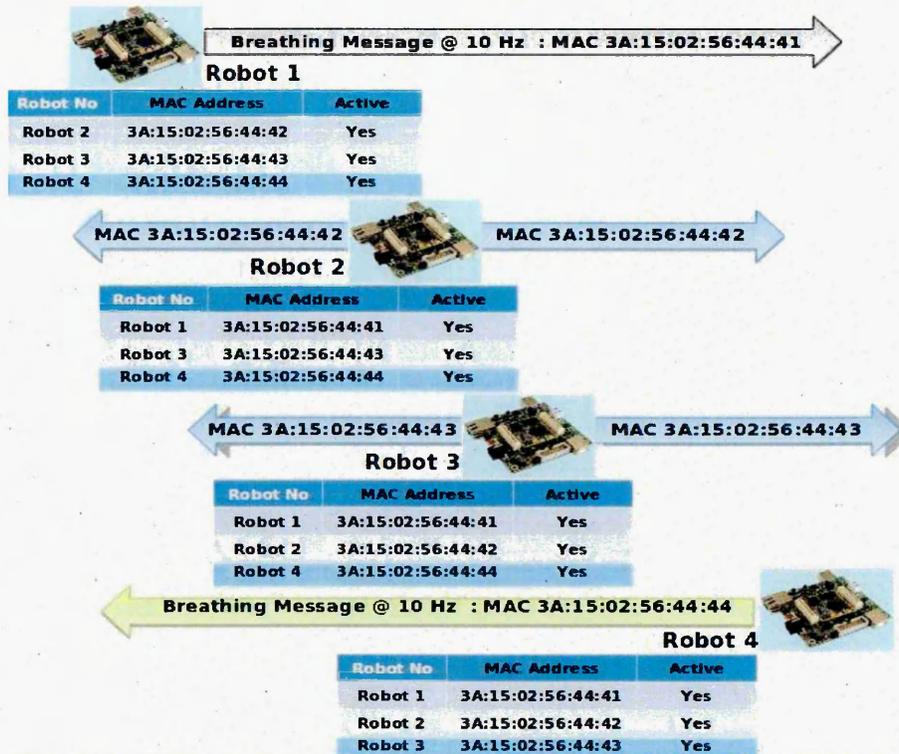


Figure 8.2: Communication within the multi-processor system.

clared them active. Each robot transmits the Breathing frame at rate of ten frames per second (i.e., 10Hz). If the robot modules do not sense this Breathing signal from their neighbouring robots for a certain period, then they declare those neighbours as inactive. Similar to robot 1, robot modules 2, 3 and 4 have also updated their communication tables. These are also shown in Figure 8.2.

8.3 Vision Task Distribution for Distributed Object Classification and Recognition

To distribute the vision based object classification and recognition tasks among the four processing modules, the tasks were required to be implemented in a modular fashion. The first processing module in the multi-processor robotic system acted as the main master processor. Once all the processing modules were aware of their neighbouring processing modules, the master processing module initiated the task. Figure 8.3 shows the manner object classification

and recognition tasks were divided into sub-tasks and assigned to the different processing modules. Figure 8.3 describes a complete feedback system and shows that how the visual information is processed in multiple stages to achieve the common goal. A detail description of the sub-tasks, assigned to different processing modules, is provided in the following sub-sections.

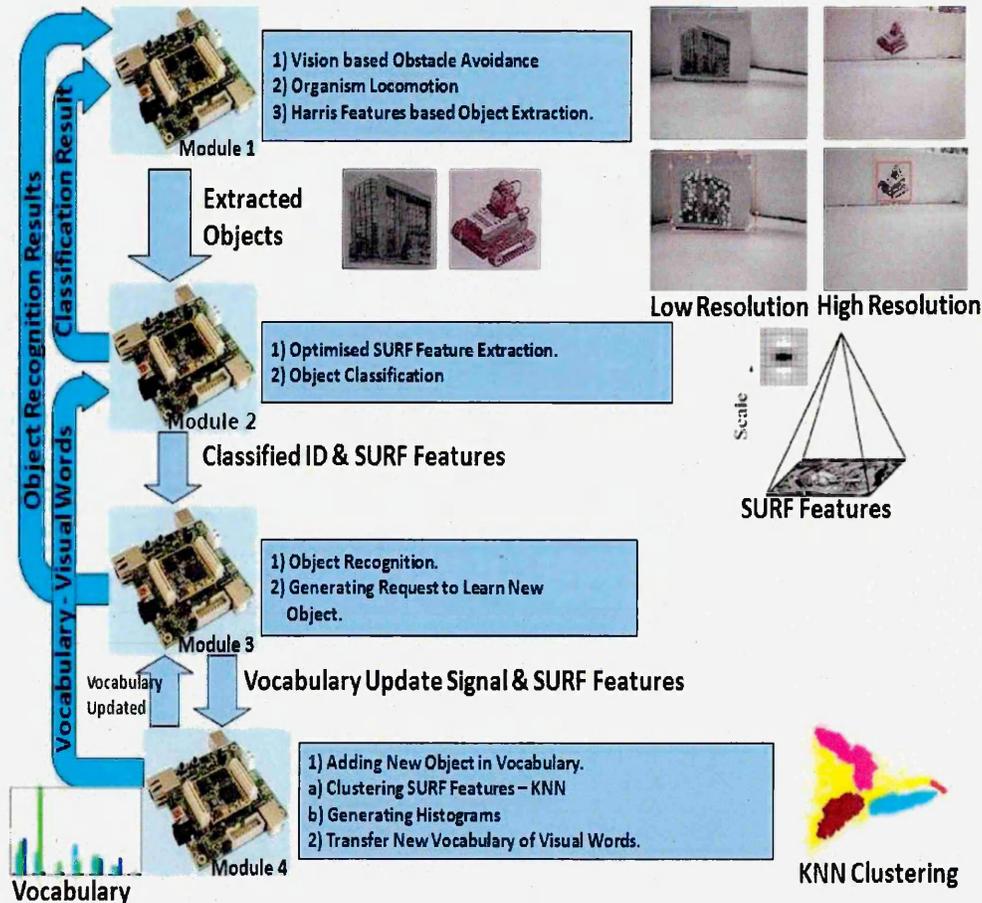


Figure 8.3: Vision task distribution.

8.3.1 Module 1: Information Pre-processing

In Figure 8.3, processing module 1 simulates the processing and memory resources of the master robot module in the organism. In other words, module 1 acts as the main processing modules and it initiates the visual information processing task. This processing module is assigned three main tasks that are, Organism locomotion, Vision based obstacle avoidance and Harris features

based object detection. To perform the organism locomotion efficiently, this processing module is required to process the information from vision sensor at a high rate. For this purpose, it implements a very light weight vision based obstacle avoidance technique which is described in detail in Chapter 4. To ensure that enough processing time was allocated for the organism locomotion, this processing module executed a very light weight features extraction technique, that is Harris features in this case, to detect the object in the images. To reduce the execution time of the Harris feature extraction technique, the resolution of the image was set to QVGA (i.e., 320x240 pixels). If the object was detected close to the vision sensor (or the robot organism), then a window was defined around the detected Harris features and the object was extracted in the low resolution image. This is shown in Figure 8.3, next to the sub-tasks of module 1. The extraction of object using Harris features is also shown in Figure 8.4a. To extract the object, the window image defined around the detected Harris features, is identified by the red coloured boundary. On the other hand, if the object was detected far (i.e., at a distance greater than 55cm) from the vision sensor, then the Harris features were detected in the low resolution image, but the object image was extracted from the high resolution VGA image (i.e., 640x480 pixels) as shown in Figure 8.4b. This kept the algorithm execution time low as the Harris features were extracted in low resolution image, and at the same time the object was provided with its detailed information as it was extracted from a high resolution image. The extracted object was then transferred to the processing module 2 for further processing. Extracting object from a high resolution image also helped when the second processing module tried to classify the object and extracted the SURF features for this purpose. The SURF feature extraction algorithm was used to detect salient features in the images. The detection of these salient features was only possible when the object's details were present in the image and for this purpose, the extraction of far lying objects from a high resolution image was necessary.

8.3.2 Module 2: Object Classification

When an object image was sent by module 1 to module 2, then the processing module 2 was required to classify the received object. For classification purposes, it required a library of visual words. One of the modules in the organism

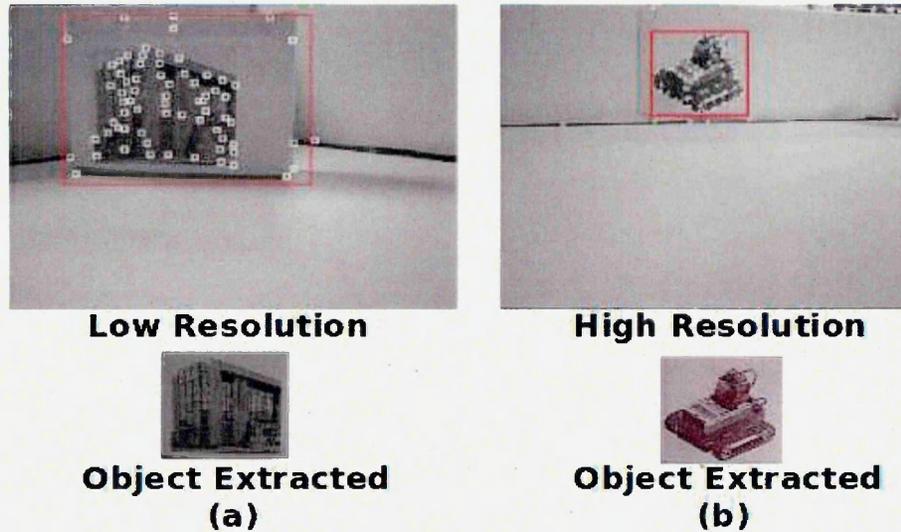


Figure 8.4: (a) Low resolution image for close object. (b) High resolution image for far object.

held the SURF features of the target objects. In this case, module 4 hold these SURF features. Module 4 processed the SURF feature and generated the library of visual words (or visual vocabulary). This library was provided to the module 2, so that it could classify the objects, it received from module 1. The library of visual words was provided in the form of the histograms for different objects and the centroids of the clustered SURF features space, as described in Section 8.3.4. Once module 2 received this visual vocabulary from module 4 (as shown in Figure 8.3), then it was able to classify the objects successfully. For object classification, module 2 was required to extract the SURF features of the received object's image. For this purpose, it executed the SURF feature extraction algorithm which was highly optimised for the Blackfin processor architecture, and is described in detail in Chapter 6. For the SURF algorithm, the Blackfin processor specific optimisation was performed as addressed in [99]. The extracted SURF features were then compared with centroids of the clustered features space. The feature space was clustered into fifty classes by module 4, so fifty centroids (one centroid for each cluster) were provided to module 2, apart from the histograms. It was decided to cluster the SURF feature space into fifty classes, because with fifty classes, the processing module took less time to cluster the feature space and also provided sufficient classification accuracy. After comparing the SURF features with the centroids of the clustered feature space, module 2 generated a histogram with fifty bins. In histogram, each bin

represented the number of features which fall into a corresponding cluster of the SURF feature space. This histogram representation of the features is called bag-of-visual-words [117] and it can be used to perform scene or object classification [116] and also to facilitate robot localisation and mapping [118] in the field of computer vision and robotics. Finally to perform classification, the Sum of Absolute Difference (SAD) of the generated histogram was computed with reference to all the histograms included in the library of visual words. Figure 8.5 shows an example output of the SAD algorithm when there were Nineteen objects in the library. The output shows that the object is classified as object eighteen, because the generated histogram from the current object provided minimum difference with the histogram of the 18th object.

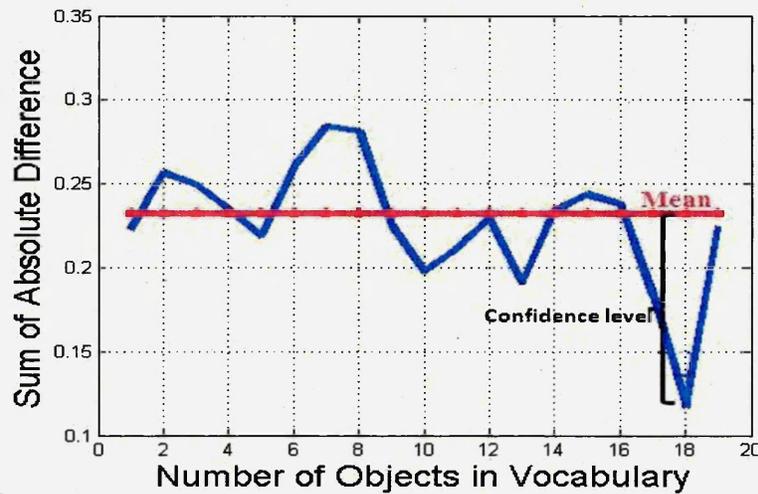


Figure 8.5: Classification probability.

In order to determine the accuracy or the confidence level of the classification results, a low computing intensive strategy was adopted. The mean value of the SAD output was computed (shown by red colour line in Figure 8.5), where the minimum difference value in the SAD output was not used in computing this mean. The difference of the minimum SAD value from this mean shows the confidence in classification. A high difference means the object is classified with a high confidence. Similarly, a low difference shows a lower confidence in the classification result. Module 2 passed this object classification ID and the confidence in classification results to the module 1 and also to the module 3 (as shown in Figure 8.3). Apart from classification results, the extracted SURF features were also transferred to the module 3 for further processing.

After receiving the classification results, depending on the level of confidence, module 1 took the appropriate actions. For example, if the organism was searching for a specific target object, then it could approach the object with the certain confidence level provided by module 2. This helped the organism to take an appropriate decision quickly, without waiting for the object recognition results from module 3.

8.3.3 Module 3: Object Recognition

Module 3 received the object classification ID, confidence in classification and object SURF features from module 2 and then it performed the object recognition task. To perform the object recognition, it also held the SURF features from all the target objects in its memory. It is to be noted that, to recognise the object, module 3 did not perform the matching of received SURF features with the SURF features of all the target objects, which was a computationally expensive task. As module 3 was provided with the object classification ID and this informed it about the identity of an expected object. So the Module 3 compared the received SURF features with the SURF features of the expected target object only. If an expected object was recognised by the module 3, then it sent the object ID confirmation (as Object Recognition Results) to the module 1, as shown in Figure 8.3. This provided organism the confirmation of the classification results generated by the module 2.

If the module 3 generated a negative confirmation results, that is the classification ID provided by module 2 was wrong, then module 3 checked the confidence level which was provided by the module 2. If the confidence level was also low then module 3 waited for the next three confirmation results. If the confidence level stays low and the next three confirmation results were also negative, then there was a high probability that the organism was viewing a new object. At this point, module 3 triggered the organism learning mechanism. It did this task by storing the SURF features of this new object in its memory for later use and at the same time, it transferred these SURF features to the module 4 which generated the new vocabulary of visual words.

8.3.4 Module 4: Vocabulary of Visual Words

This processing module was responsible for holding the organism knowledge about the target objects (i.e., the objects which the organism classified and recognised). This knowledge was represented in the form of a library of visual words which was generated using the SURF features. These SURF features were initially computed by module 2 and after going through further processing in module 3, these features arrived to module 4. There were two instances when module 4 generated the library of visual words. The first instance was when the organism was formed and only module 4 had the initial knowledge (in terms of SURF features) of target objects. At this point, the organism was not able to classify any object. Module 4 generated a library of visual words and passed this library to the module 2 for classification purposes. The second instance for the generation of library was triggered by module 3. When module 3 determined that the organism was viewing a new object, then it passed the SURF features of new object to module 4. Module 4 then learnt the new object by generating a new library of visual words.

For generating the library of visual words, module 4 clustered the complete SURF features space using KNN (K-Nearest Neighbour) algorithm. It clustered the SURF feature space into fifty clusters. If it was the first time it was clustering the SURF feature space, then in KNN algorithm, it used random centroid values for the different clusters and then found out the optimum centroid values. But if it was learning a new object, then it used the last computed centroid values as an input to the KNN algorithm and then determined the optimum values of the centroids. This helped module 4 to generate the new centroids for all the clusters in reduced time. Once the centroids for all clusters were computed, then module 4 generated a histogram for each target object in the library. As the SURF features space was partitioned into fifty clusters, so each histogram also contained fifty bins. Where each bin in the histogram represented the number of SURF features (extracted from a target object) fell into the corresponding cluster.

To explain the concept of feature space clustering and histogram generation, an example is shown in Figure 8.6. In this example, the complete feature space was divided into four clusters that is, clusters 1, 2, 3 and 4, were identified by red, blue, green and orange colours, respectively. For histogram generation,

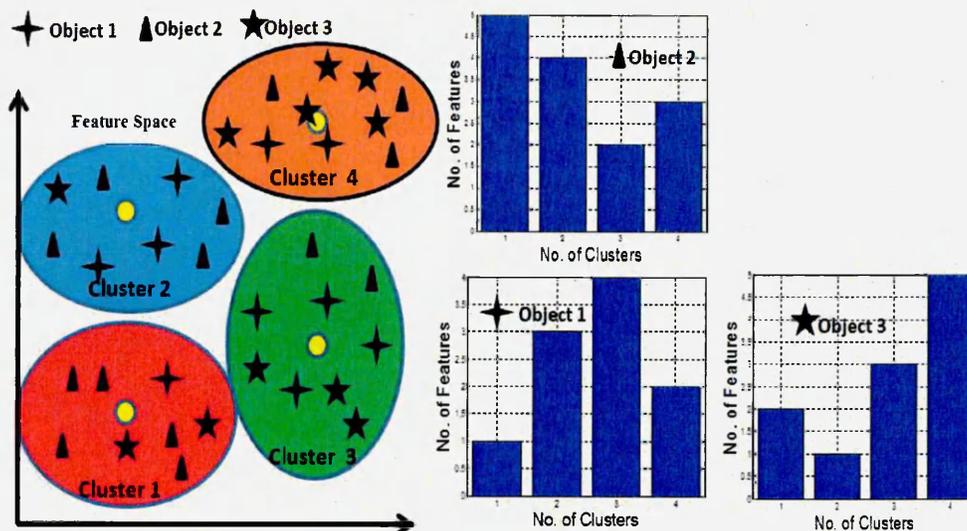


Figure 8.6: Feature space clustering and histograms generation.

three target objects were considered. The features resulting from these three target objects were identified by different signs as shown on the top left side of Figure 8.6. For object 1, the number of features falling in the clusters 1, 2, 3 and 4 were found equals to 1, 3, 4 and 2, respectively. This generated a histogram for object 1, as shown on the right side of Figure 8.6. In the histogram, the x-axis shows the total number of clusters in the feature space and y-axis represents the number of object features lying in the corresponding clusters. Similarly, depending on the number of features lying into different clusters, the histograms generated for objects 2 and 3 were also shown in Figure 8.6. The histogram representation of SURF features for target objects is called library or bag of visual words. The centroid computed by KNN algorithm, for the different clusters in the feature space, are identified by the yellow circles in Figure 8.6. These centroids of the clustered feature space, together with the generated histogram for all the target objects, were transferred to the module 2 for object classification purposes (as shown in Figure 8.3). When module 2 received the new library of visual words, it discarded the old library. Using the new library of visual words, module 2 was able to successfully classify the new object. Similarly, module 3 also started recognising the new object as it also held the SURF features of newly learned object.

8.4 Experimental Results

The performance of the modular implementation of object classification and recognition task was demonstrated using the distributed memory and processing resources of the robotic organism. A test was conducted which lasted for 300 seconds. Following the sub-tasks assignment as discussed in Section 8.3, module 4 in organism was provided with the SURF features of the target objects. In this test, module 4 was initially provided with the SURF features of 4 target objects. When the test started, in the beginning, the organism was not able to classify any object. This happened because module 4 was generating the library of visual words and module 2 needed this library to classify the objects. This is shown in Figure 8.7, where for the first ten seconds, the classification ID was zero. In Figure 8.7, the classification ID (provided by module 2) and Recognition ID (provided by module 3) are shown in blue and red colours, respectively. For the first ten seconds, when the classification ID was zero, the classification probability was also zero as shown in Figure 8.8. Then the first object was placed in front of the vision system attached to the module 1. It was classified correctly by module 2 as shown in Figure 8.7 (blue profile from 10 to 40 seconds). Similarly, module 3 also confirmed that the classified ID was correct. This is shown in Figure 8.7, where from 10 to 40 seconds, the red profile overlapped the blue profile. Later on, objects 2, 4 and then 3 were shown in sequence and they were also classified and recognised correctly. Note that, as module 4 was provided the SURF features of first four objects, which it had used for generating the visual vocabulary, so it was possible for module 2 to classify the first four objects correctly. While classifying these four objects, the classification probability computed by module 2 was also high. This is shown in Figure 8.8 where the classification probability ranged from 0.5 to 0.88 during the period from 10th to 76th seconds.

At nearly 76.5th seconds of the test, when the fifth object was shown to the vision system, as it was a new object for the organism so a mismatch in the classification and recognition ID was observed in Figure 8.7. It was mistakenly classified as object 2 and module 3 generated a false recognition result (i.e., Recognition ID is zero). In other words, module 3 rejected the classification result. During this time, the classification probability was also small, i.e., less than 0.1. When module 3 found that false classifications occurred for the

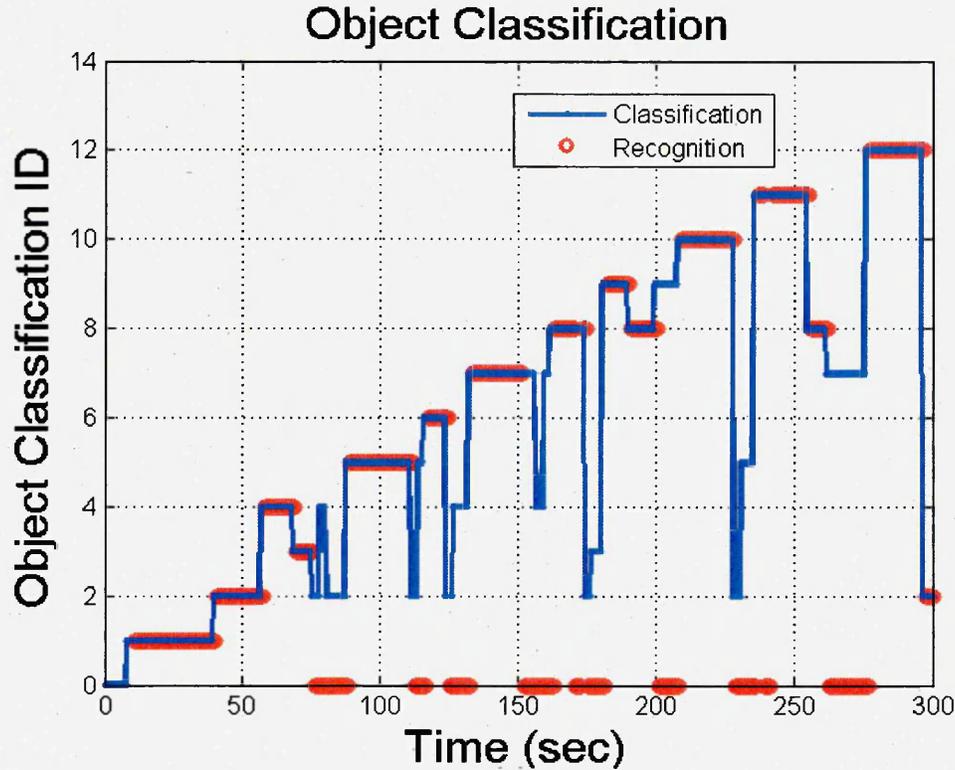


Figure 8.7: Classification and recognition ID.

consecutive three times and the classification probability remained low, then it triggered module 4 to learn the object and generated a new vocabulary of visual words. This can be observed in Figure 8.9 where the processor usage for all the processing modules is shown. It can be noticed that, nearly at 76th seconds, when module 3 triggered the organism learning mechanism, the processor usage of modules 3 and 4 was very high. To learn the fifth object, module 4 received its SURF features from module 3. Module 4 had to add the SURF features of new object into the old SURF features space and then re-cluster the whole SURF feature space using KNN algorithm for generating the library of visual words. As this was a computationally heavy operation for module 4, so it has shown 100% processor usage during this time (nearly at 76th seconds). When module 4 provided the new library of visual words to module 2, then modules 2 and 3 were able to classify and recognise object five successfully, as shown in Figure 8.7. The rise in classification probability for object 5 can also be seen in Figure 8.8. Similarly, objects 6, 7, 8, 9, 10, 11 and 12 were shown to the vision system, in sequential order, at approximately 112.5, 125, 152.5, 171, 201,

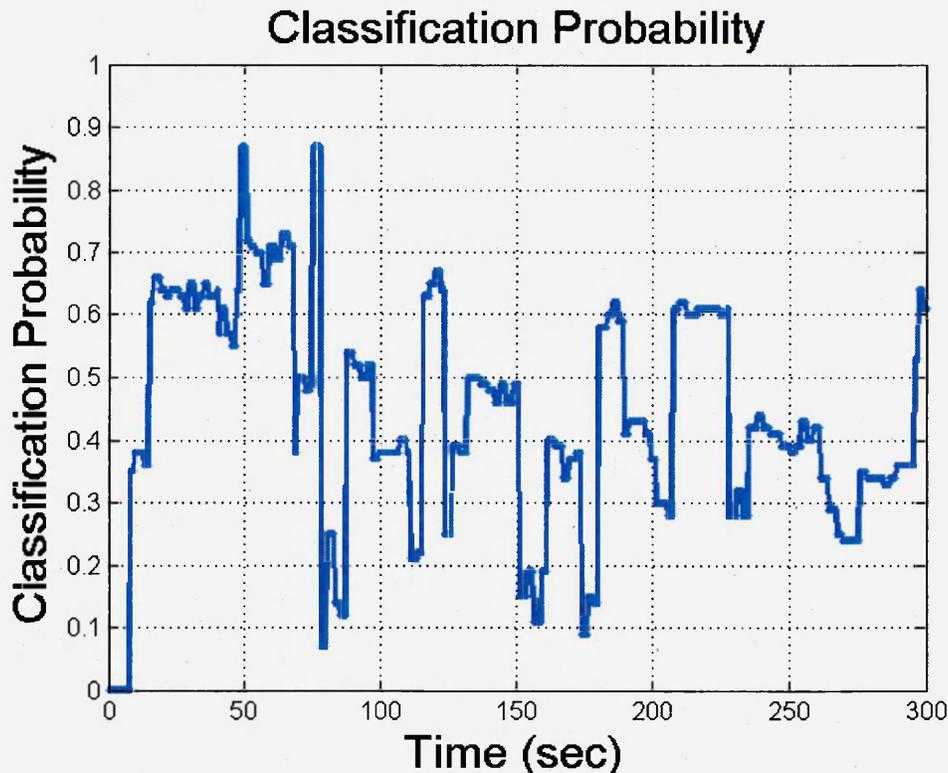


Figure 8.8: Classification probability.

229 and 263th seconds. During these times, the mismatch in the classification and recognition ID was observed and the classification probability was also found low. To learn these objects, as module 3 again triggered the module 4 to generate the new library of visual words, so a rise in processor usage for modules 3 and 4 can be seen in Figure 8.9. All these objects were successfully learnt by module 4 and this can be seen in Figure 8.7, where modules 2 and 3 have started classifying and recognising the objects IDs correctly.

From Figure 8.9 it can be seen that, on average, processor usage for modules 1, 2 and 3 were 50%, 99.95% and 40%, respectively. Whereas, module 4 was utilising the full processor performance occasionally, when it received the vocabulary update signal from the module 3. The first spike indicates the 100% processor usage of module 4 happens in the beginning 10 seconds. This is the time when the module 4 was generating the initial vocabulary of visual words. The later pulses in the module 4 processor usage were recorded when it received the new objects features to generate a new library of visual words. As processor

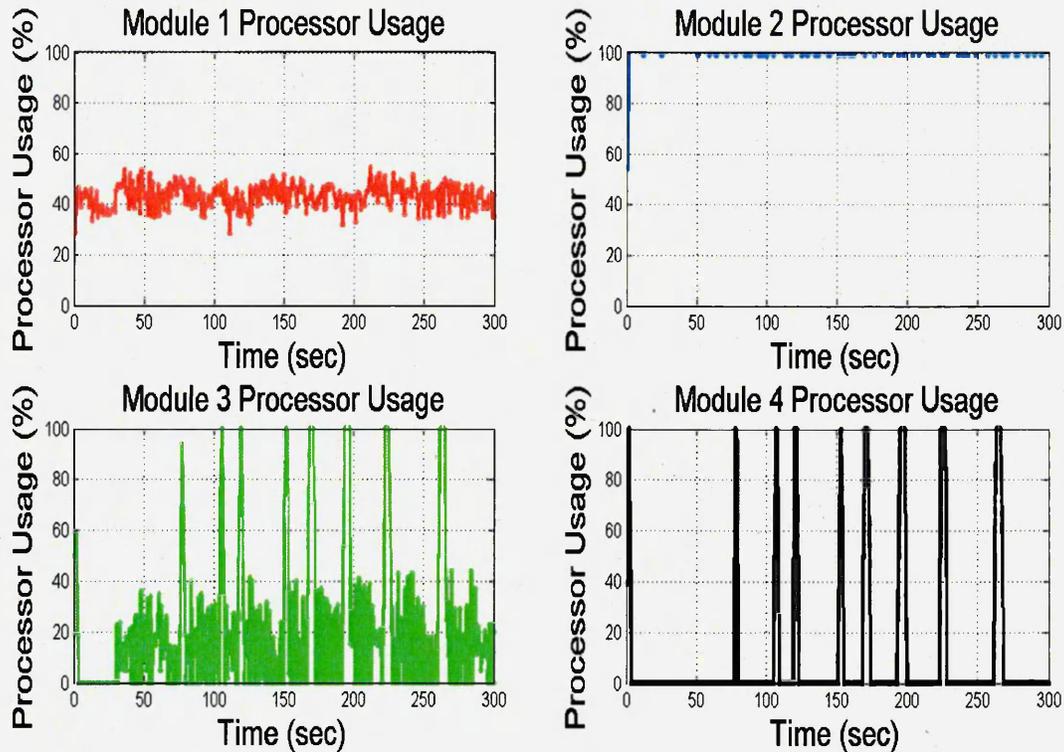


Figure 8.9: Processor usage.

usage of module 1 was 50% on average, this ensured that module 1 had enough processing resources to control the organism locomotion mechanism. Module 2 was performing the most computationally expensive tasks amongst all the modules, that is SURF feature extraction and object classification. That is why it appears to be the busiest processor. In case of module 3, similar to module 4, the rise in processor usage profile was observed when the decision to add a new object in vocabulary was taken. It is to be noted that, the second processing modules appears to be the busiest amongst all the processing module. If more processing modules are available, then some of the processing load from second processor can be shared with the other spare module. But to achieve this operation, the task assigned to second module (i.e., SURF features extraction) is required to be divided into further sub-tasks. This calls for the further modular implementation of the vision based operations. However, the processor usage profile from all the modules indicates that the distributed processing resources of the organism were utilised properly.

Similar to the processor usage, the profiles showing the memory utilisation of

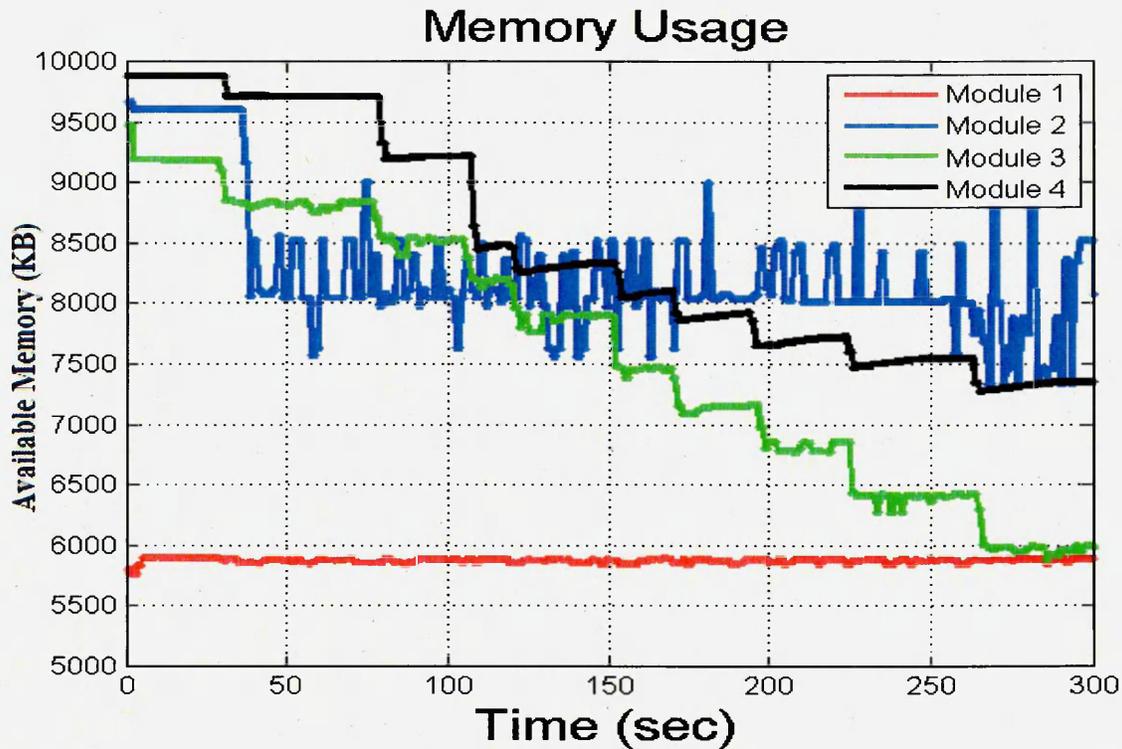


Figure 8.10: Memory usage.

the four modules during the experiment, were also recorded and it is shown in Figure 8.10. This Figure shows how much memory in KB (Kilo bytes) is available to the processor. As this research is dealing with the distributed vision processing tasks and vision processing applications utilises high amount of memory, so considering the memory usage for the distributed processing modules was important. If the same task had been performed using a single processor, then it was observed that, storing SURF features from 12 objects drops the available system memory below 900KB and the application crashes when the recognition algorithm tries to allocate more memory. It was found that, in any distributed computing application, apart from the processor utilisation, efficient utilisation of the distributed memory resources is also a very important, especially when application deals with the vision information processing. From Figure 8.10, it can be seen that, in case of modules 3 and 4, the drop in memory profile is observed during the time when new objects were added to the vocabulary. In case of module 3, as it keeps a copy of SURF features in its memory for object recognition purposes, so it utilises memory on adding every

new object. Whereas, in case of module 4, as this module adds the new object SURF features into the old SURF feature space for generating new vocabulary, so a drop in available memory was expected. In case of module 1, slight variation in the available memory was observed throughout the experiment. In case of module 2, the processing module deals with the computationally heavy SURF feature extraction algorithm. As this algorithm allocates huge amount of memory, which was de-allocated once the algorithm finished, so high variations in the available memory profile were observed (shown in Blue colour in Figure 8.10).

Ideally, the use of distributed or multi-processor computing systems shows an increase in the frequency of operations, as compared to a single processor implementation. In other words, as the distributed processing system provides a rich processing environment, so the implementation of modular vision processing application on such a system is expected to decrease the execution time of vision processing tasks as compared to a single processor system. To show the frequency of operation, the frame processing rate achieved with the four processor system, during the complete experiment, is shown in Figure 8.11. In the recorded profile, sometimes zero FPS (frames per second) was recorded. This happened because the SURF algorithm had produced too many features, which increased the computational time significantly. Or the processing modules experienced erroneous communication, which caused a delay in the completion of the feedback sequence shown in Figure 8.3. However, on average, 0.614 FPS were recorded with the modular implementation of object classification and recognition task.

A comparative result obtained, when the complete recognition task was performed on a single processor system, is also shown in Figure 8.12. It can be seen that, when features from a single object was present in the vocabulary (i.e., green profile in Figure 8.12 shows 418 SURF features for one object), then 0.63 FPS was recorded. But as the number of objects increased in the vocabulary, then the total number of features, resulting from the objects, also increased (rise in green profile in Figure 8.12). It can be noticed that, the increase in the total number of features is not strictly linear. This is because every object results in different number of features, depending upon the texture present on the object. Due to the significant increase in the total number of features,

the feature matching part of the recognition algorithm became more computationally expensive. Hence, a drop in the FPS profile was observed, as shown by blue colour profile in Figure 8.12. When this single processor system was programmed to recognise twelve objects, then the FPS was dropped to 0.13, because the number of SURF features in the vocabulary increased to 5789 (i.e., Total number of features for 12 objects represented by the green colour profile in Figure 8.12). When the single processor results were compared with the four processor system, then the four processor system increased the FPS rate by the factor of 5.

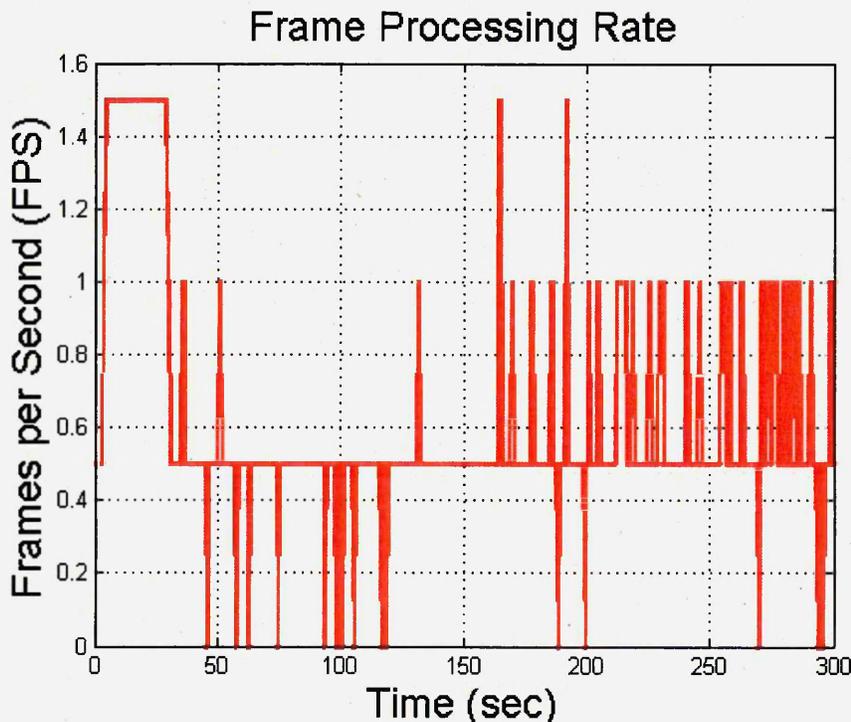


Figure 8.11: Frame per second - Four processors system.

As the proposed modular vision processing approach strongly relied on the generated classification probabilities, so an experiment was performed in which the classification probabilities were recorded for increasing number of objects in the vocabulary. The results obtained are shown in Figure 8.13. The number of objects in the vocabulary were increased from four to twenty two objects. A drop in the confidence probability was observed with an increase in the number of objects. The zoom-in image, when there were four objects in the vocabulary,

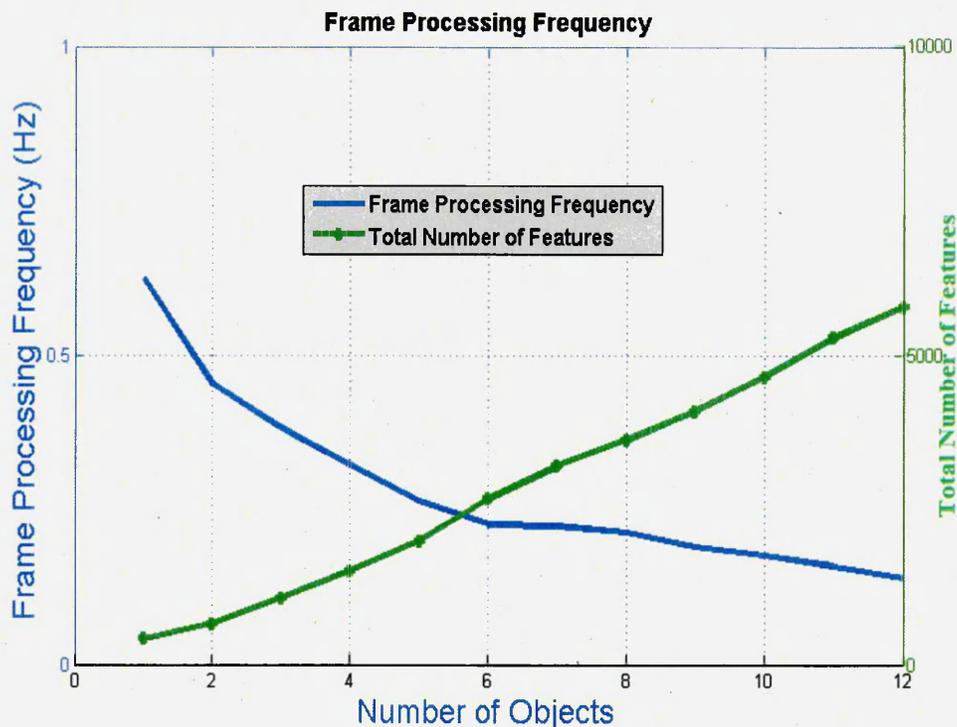


Figure 8.12: Frame per second - Single processor robot.

is shown in Figure 8.14. The classification probability for the four objects is coded in different colour bars. The classification probability ranged from 0.68 to 0.82. Similarly, when there were twenty-two objects in the vocabulary, then the classification probability generated is shown in different colour bars in Figure 8.15. The probability ranged from 0.04 to 0.21. It was observed that, when the number of objects increased in the vocabulary, then this resulted in more common features (i.e., features which were shared by more than one target objects) in the complete SURF features space. These common features could not be avoided as ignoring them made some of the target objects unrecognisable. Hence, the increase in the number of common features affected the classification results significantly and this was very prominent in the profile shown in Figure 8.13. To overcome this problem, first of all, it is necessary to identify the features which are common among different target objects, and then these features should be given low weightage when performing the object classification. To further improve the results, it is also important to identify the features which produces false results in the complete classification and

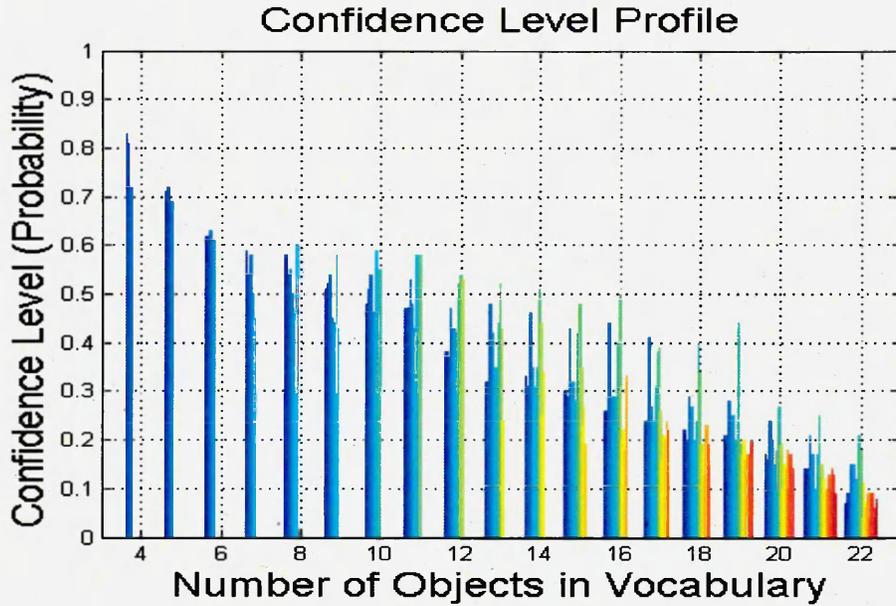


Figure 8.13: Classification probabilities with increasing number of objects in the vocabulary.

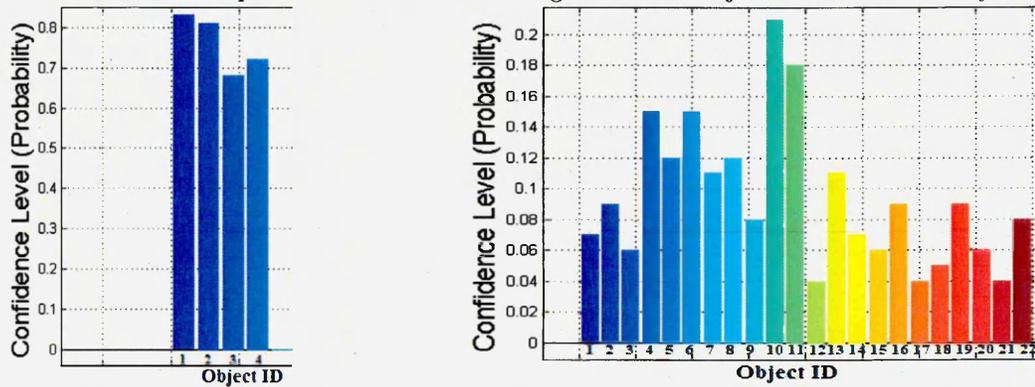


Figure 8.14: Classification probability: Vocabulary contains 4 objects.

Figure 8.15: Classification probability: Vocabulary contains 22 objects.

recognition process. Once identified, then these features should also be assigned lower weightage while performing the classification.

8.5 Performance Comparison of Distributed Modular Robotic System versus High Processing Systems

As described in Chapter 1, in swarm robotics or reconfigurable modular robotic systems, an individual robot module has very limited memory and processing

resources, but the system as a whole is considered very rich in these resources. When the modular robots use these resources collectively, they can achieve efficient performance like the other high processing robotic systems. In this Section, a detailed performance comparison is presented between the modular implementation of object classification and recognition technique on a multi-processor robotic system and a non-modular implementation of object recognition on high processing systems. For high processing systems, a Pioneer 3AT robot (shown in Figure 8.16) with Dual Core 2.26 GHz processor and 2GB RAM mounted on the on-board computer, was used for comparison. For another high processing system, a Laptop with Core-2 Duo processor and 4GB RAM was used.



Figure 8.16: Pioneer-3AT robot.

This Section is further divided into the following two sub-sections. The first sub-section briefly describes the non-modular implementation of object recognition on high processing systems. Whereas, the second sub-section presents a detailed performance comparison.

1. Object Recognition - High Processing System
2. Performance Comparison

8.5.1 Object Recognition - High Processing System

For object recognition on a high performance and single processor system, the SURF features based recognition approach was adopted. For recognition purpose, the process followed by a single processor system or robot is shown in

Figure 8.17. The SURF features extracted from all the target objects were kept in the robot's memory. As shown in Figure 8.17, the robot grabs image from vision sensor in QVGA (320x240 pixels resolution) format. The robot extracted the SURF features from the image. To determine the presence of the target object in the image, the target objects in the vocabulary are considered one by one. The SURF features of the target object are matched with the features extracted from the current image. If no feature matching was found, then the SURF features from the next target object were considered. If certain number of features were matched (i.e., the feature matching result was positive), then the matching features were processed with RANSAC (RANDOM SAMPLE CONSENSUS) algorithm to remove all outlier features. After removing the outlier features, the resultant matching features were then used for computing the Homography. Homography was computed in order to determine whether the positions of the matching features of the target object were geometrically consistent with the features of the current image. If the matching features were geometrically consistent, then the object's ID was determined and the object was recognised. On the other hand, if the geometric consistency failed then the process was repeated with the features of the next target object (until all the target objects were processed), as shown in Figure 8.17.

8.5.2 Performance Comparison

This Section presents the results obtained, when the object recognition was implemented on a modular systems (as described in Section 8.3) in comparison to high performance single processor based systems. Following the discussion in Section 8.5, for single processor system, a Pioneer-3AT high processing robot and Core-2Duo Laptop were considered. For detailed analysis, a robot with single Blackfin BF537 on-board processor was also selected. For these systems the object recognition technique discussed in Section 8.5.1, was used with increasing the number of target objects. The number of target objects was from 1 to 12. When this technique was implemented on a robot with single Blackfin processor, then the memory profile shown in Figure 8.18 was produced. This Figure shows the drop in total amount of memory (in Kilo bytes) available to the robot, when the number of target objects were increased from 1 to 12. The vertical red coloured lines shows the drop in the available memory from

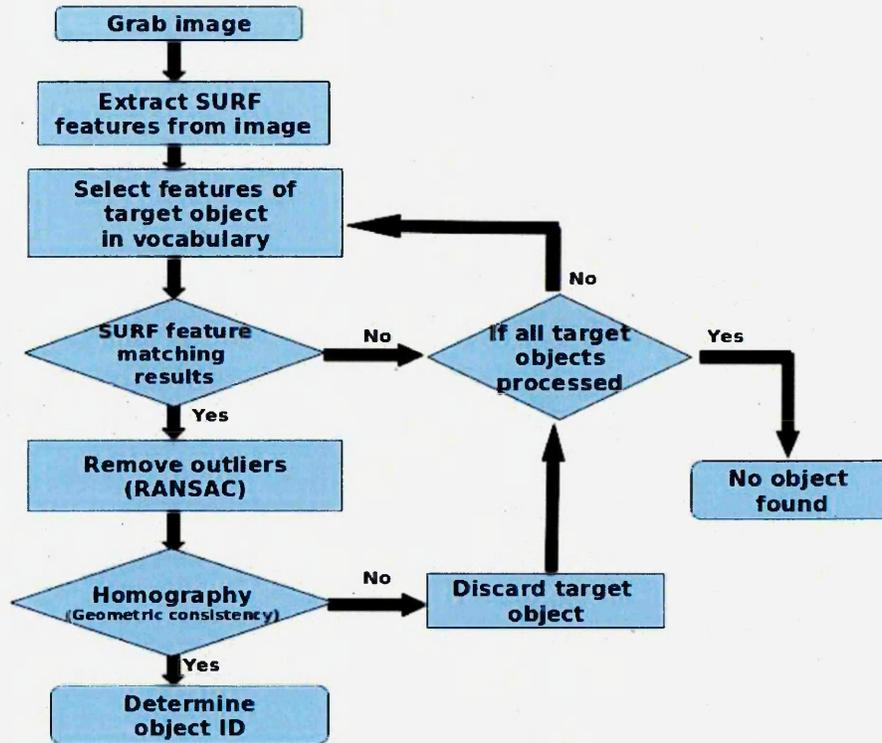


Figure 8.17: Traditional object recognition implementation - high processor system.

start to the end of test, for the corresponding number of target objects. For example, for three target objects, the initial memory available to the system was 8000-KBytes. But during the test, when the SURF features of these three target objects were loaded into the memory and feature extraction and matching techniques were executed, then the lowest available memory detected in the robot was 5200-KBytes. The green coloured points on the red bars represent the average memory which was available to the system during the tests. The significant drop in the available memory, due to the use of the SURF features, was expected. This is because, the SURF features required huge amount of memory for storage, as each feature was represented by a vector with 64 fields [98]. Figure 8.18 shows that, for 12 target objects, the lowest memory detected in the system dropped to 1300-KBytes. It was observed that, if the number of target objects exceeded by 12, then the available memory dropped below 900-KBytes and this resulted the system crash when the SURF feature extraction algorithm tried to allocate more memory for its operation.

Similar to memory usage, the processor usage recorded for the robot with single

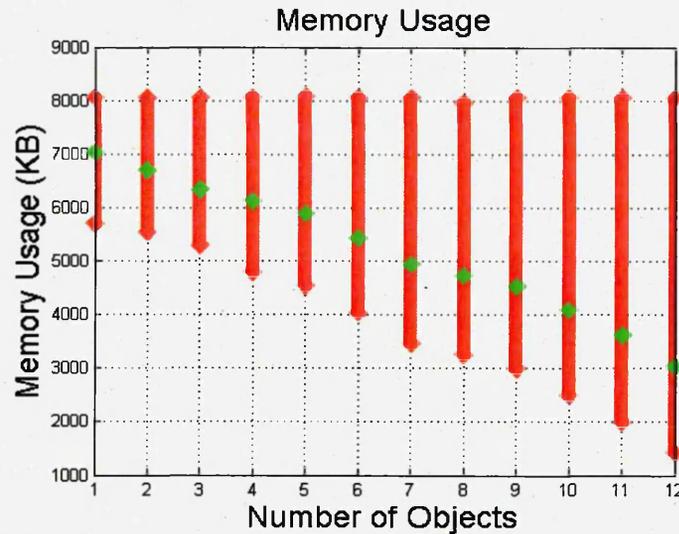


Figure 8.18: Memory usage for a robot with a single blackfin processor.

Blackfin processor, is shown in Figure 8.19. This Figure shows that, when there was only one target object, then the processor usage (on average) was 73.5%. The average processor usage exceeded 95% when the number of target objects reached up-to 12. This indicated that the processor had become increasingly occupied with increasing number of target objects.

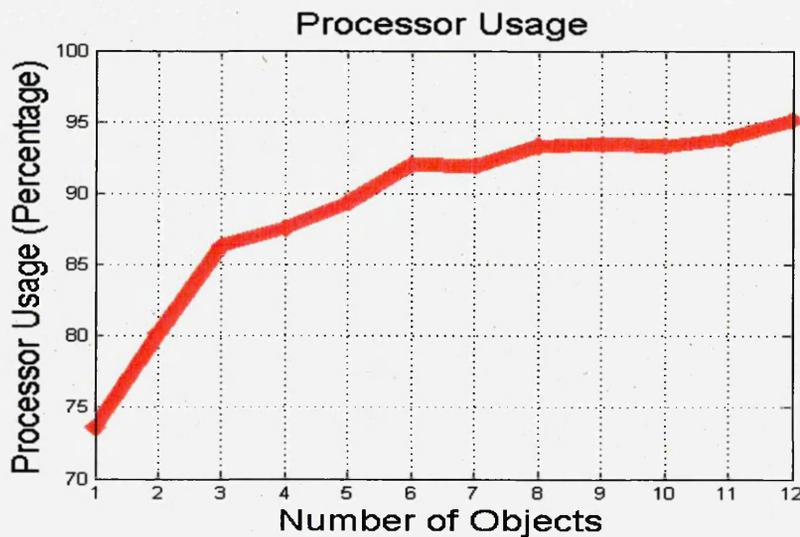


Figure 8.19: Processor usage for a robot with a single blackfin processor.

For the considered single processor systems, the average frame processing achieved, when the object recognition technique was used for increasing number of target

objects, is shown in Figure 8.20. For a robot with single Blackfin processor, when there was one target object to recognise, then the average frame processing time recorded was 1800 milli-seconds to process one image. This frame processing time was gradually increased to 7000 milli-seconds (i.e., 7 seconds), when the number of target objects reached 12. The reason for the increase in processing time, was the significant increase in the number of SURF features resulting from the target objects. The profile showing the increase in the total number of SURF features, with increasing number of target objects, is also shown in Figure 8.20 in green colour. Note that, the y-axis label on the right side of Figure 8.20 shows the scale for total number of SURF features. The total features profile is not strictly linear as every target object produces different number of features, depending upon the texture present on the target object. When the same test was performed on the Pioneer-3AT high processing robot, the frame processing time increased from 600 milliseconds (for one target object) to 1850 milliseconds (up-to 12 target objects), as shown in blue colour in Figure 8.20. Similarly, when another high processing system was used, that is a Laptop with Core-2Duo processor, then the frame processing time ranged from 250 milli-seconds to 500 milli-seconds when the target objects increased from 1 to 12.

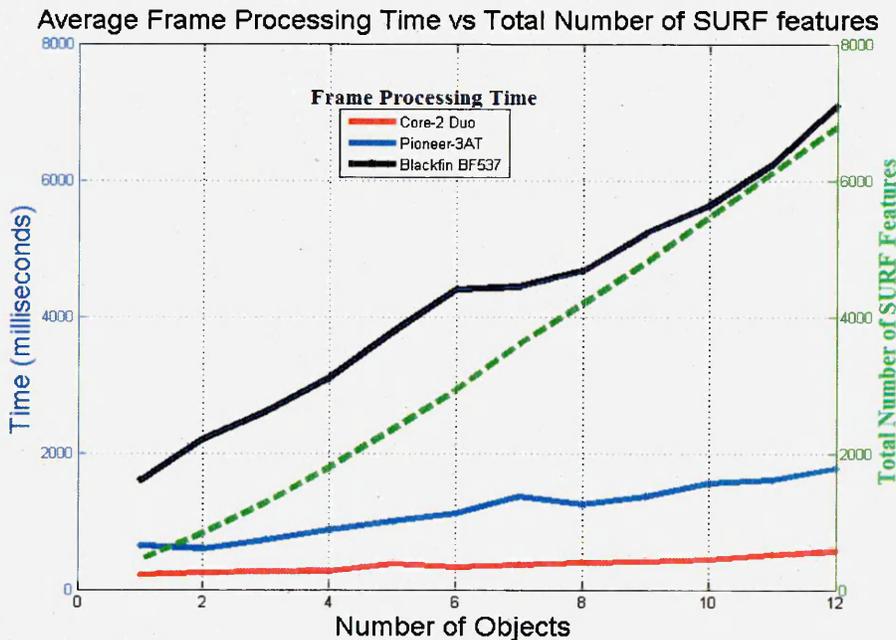


Figure 8.20: Frame processing time versus total SURF features for single processor systems.

For the performance comparison, the modular form of object recognition technique (described in Section 8.3) was implemented on a multi-processor robotic system. As four processors were used on the robot (as described in Section 8.1), where each processor was simulating the processing and memory resources of one robot module, so the memory and processor usage for the four processing modules were recorded. The memory usage for the four processing modules is shown in Figure 8.21. As discussed in Section 8.3, the processing module 3 holds SURF features to perform the feature matching process, and processing module 4 also held the complete SURF features space for generating the visual vocabulary when new objects were learnt. Therefore, a drop in the available memory profile is recorded for modules 3 and 4 as shown in blue and black coloured profile, respectively. It is to be noted that, even after adding the SURF features from 12 objects, all the processing modules still had enough memory resources available to add more objects. Whereas, for a robot with single Blackfin processor, the system did not have more memory resources available to add features for more target objects as shown in Figure 8.18. The memory usage profiles in Figure 8.21 shows that the distributed memory resources are utilised efficiently by the modular implementation of object recognition approach in distributed robotic system.

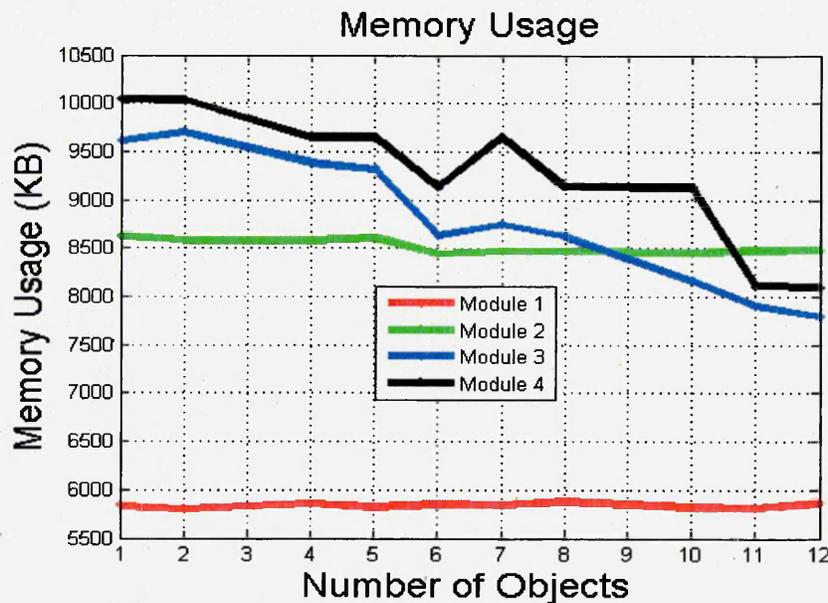


Figure 8.21: Memory usage for multi-processing system.

Similar to memory usage, the processor usage recorded for all four processing modules are also shown in Figure 8.22. From the processor usage, it can be seen that module 1 processor usage is 60% on average (shown in red colour in Figure 8.22). As module 1 is responsible for the locomotion of the modular robot, so its 60% processor usage ensured that it has enough processing resources available to process the data, from vision and other sensors, at high rate to perform the locomotion efficiently. Module 2 was responsible for the SURF features extraction task. This was a computationally very heavy task, so due to this, the average processor usage for module 2 was above 95% (green coloured profile in Figure 8.22). Module 3 performed the feature matching and recognition parts. The classification results provided the expected ID of the target object to module 3, which made the feature matching part computationally less expensive and results in the average processor usage around 70%. Module 4 was active only when it received the vocabulary update request from the module 3. So due to this, the processor was fully occupied only when it received the vocabulary update signal and clustered the complete SURF feature space to generate the new vocabulary. However, on average, the processor usage for module 4 was found to be 25%. It can be seen that, based on the assigned tasks, the distributed processing resources were utilised efficiently by the modular object recognition approach.

For comparison with the frame processing time obtained from high processing systems (as shown in Figure 8.20), the processing time achieved with modular system is shown in Figure 8.23. The profile for the corresponding total number of features in the vocabulary, is also shown in green colour. It can be seen that, the average frame processing time is 700 milli-second. The frame processing time measured with 12 target objects was found to be less than 750 milli-second. This shows a better performance in comparison to Pioneer 3-AT high processing robot, which took nearly 1850 milli-seconds to process one frame when there were 12 target objects. But in comparison to the Core-2Duo Laptop system which took 500 milli-second to process a frame, the modular system took 250 milli-seconds more. From Figure 8.23, it can be observed that the significant increase in the total number of SURF features does not affect the frame processing time in a modular system. Apart from providing competitive processing performance, the modular robotic systems also provides a

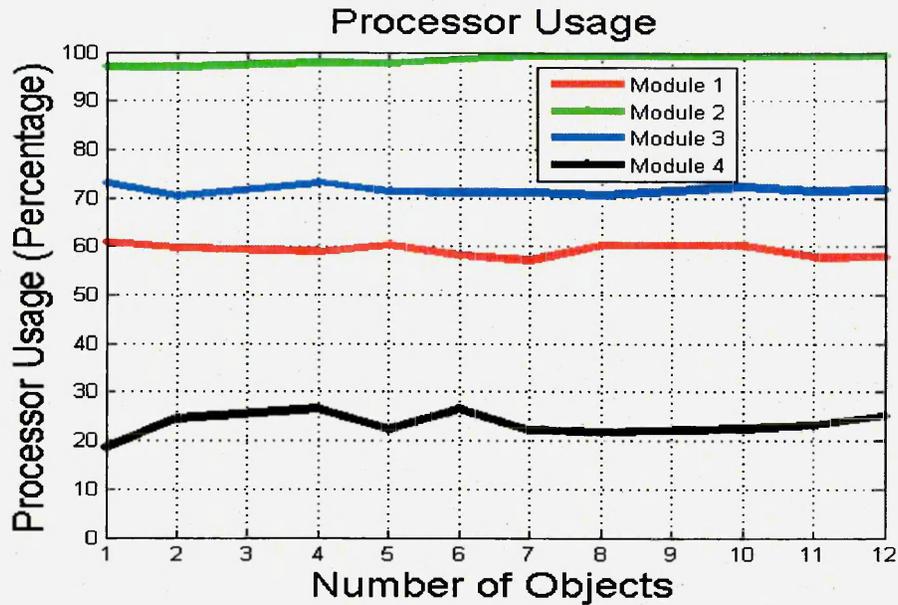


Figure 8.22: Processor usage for multi-processing system.

cost effective solution. The basic idea behind modular robotics is to keep the design of individual robot units very simple, so that the individual robot unit does not cost much and large number of robots can be produced. For example, a single Pioneer-3AT robot used in the experiment cost 6000\$ on average, whereas, the better processing performance was achieved by multi-processor robotic system, and this system cost less than 2000\$. But at the same time, the bigger size robots are also equipped with power and processing demanding sensors (e.g., laser range finders) which facilitates the robots to perform intelligent operations, and the use of such sensors is not realisable in modular robotic systems. It is also important to consider that, in case of single big size robot, any malfunction in robot will cause the mission failure. On the other hand, the redundancy in terms of number of robotic modules in modular robotic system reduces the chances of mission failure. In modular systems, if a single robotic module malfunctions, then this will not result in mission failure because another robotic module will replace the malfunctioned module.

Finally, the classification probability profile, showing the confidence in classification result, is shown in Figure 8.24. The profile for total number of SURF features in the vocabulary is also shown in green colour. The classification probability was 1 (i.e., 100% confidence level) when there were two target ob-

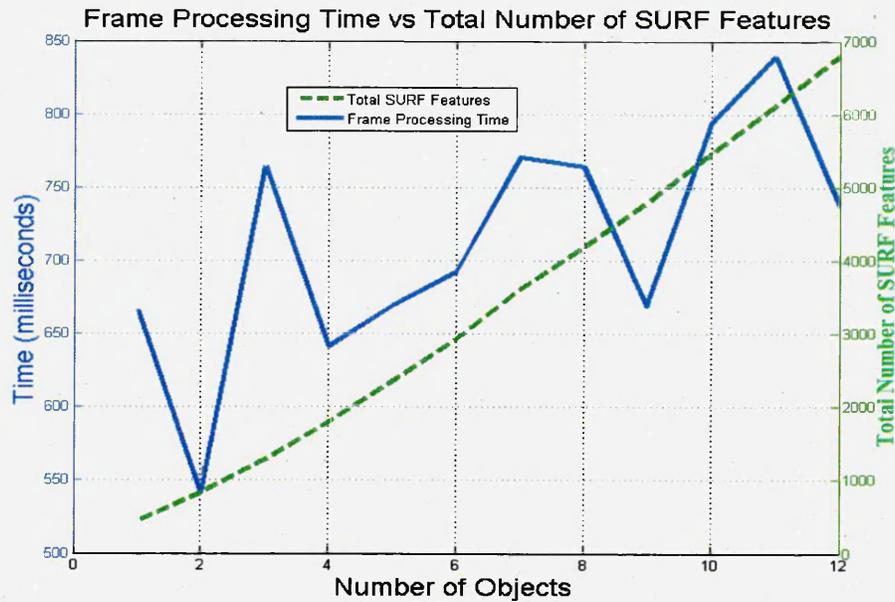


Figure 8.23: Frame processing time versus number of SURF features for modular system.

jects in the vocabulary, and it reduced to 0.82 for 3 target objects. When the number of objects exceeded by 5, the classification probability dropped below 0.7. This happened because too many common features resulted in the vocabulary which reduced the classification probability. The classification probability dropped below 0.4 when the number of target objects reached up-to 12. It was observed that, the classification accuracy highly depended upon the total number of SURF features in the vocabulary. This is very prominent in Figure 8.24, where the confidence level keeps on dropping with an increase in the total number of target objects and the SURF features in the vocabulary.

In multi-processor robot organism used in the scenarios in Chapters 7 and 8, the information from vision sensor connected to master processing module, was processed in distributed fashion. This information was processed to provide surrounding awareness to the robotic organism, and also help in performing obstacle avoidance. It is important to note that, when a robotic organism is going to be formed, then its 3D structure and overall size will be bigger than a single robotic module. The field of view provided by a single vision sensor (on master module), will not be sufficient enough to help the robotic organism to avoid colliding with the obstacles. For this purpose, the information from vision sensors present on other robotic modules, will also required to be processed.

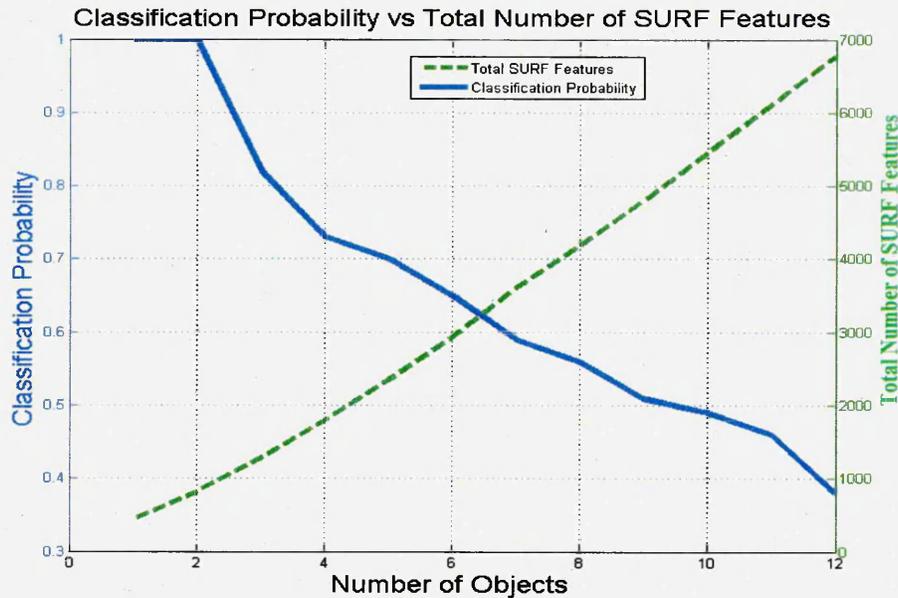


Figure 8.24: Classification probabilities versus number of SURF features for modular system.

To process that visual information, apart from performing distributed vision processing task, all the robotic modules are required to have enough processing resources to process the information from their own vision sensor. This is completely realisable from the results achieved in this research. For example, from Figure 8.22 it can be seen that, all the processing modules still have enough processing resources to perform other necessary operations.

8.6 Conclusions

In this Chapter the second distributed vision processing scenario in organism mode has been presented. This scenario addressed a modular approach to object classification and recognition, using distributed memory and processing resources of a robotic organism. The implementation of the approach using a four-processor system has been found to execute 5 times faster than a single processor system. In case of single processor system, the system crashed when the number of objects in the library, exceeded 12. On the other hand, in multi-processor robotic organism, enough memory resources will still be available, even after 12 objects has been learned by the organism. The processors usage also showed that all the processors have been utilised efficiently according to

the assigned tasks. It is shown that, as the number of objects learned by the organism increased, the classification probability showed less confidence in the classification results. This happened because more common features resulted among the target objects. In the Results Section, the robotic organism has been kept static and only images from the robot camera has been used to perform object classification and recognition tasks. However, if the objective is the organism localisation, then the presented approach can be used for landmark recognition purposes and therefore, it can effectively support the probabilistic methods (such as Partially Observable Markov Decision Process - POMDP) to track the probability distribution of the robot where-about and finally, localise the robot. The research presented in this chapter also conclude that, the collective use of processing resources in a modular robotic system can provide a performance comparable to high processing robots or systems. Similar to the organism mode scenario presented in Chapter 7, it has been concluded that, to achieve high performance from modular systems, the most challenging task is the modular implementation of the underlying technique, such that efficient use of the distributed memory and processing resources can be made.

Chapter 9

Conclusions & Future Work

9.1 Conclusions

From the research work presented in this PhD thesis, the following conclusions have been made.

- (i) A library of vision processing algorithms has been optimised for targeted embedded system. The minimum frame rate achieved with algorithms in this library, is 40 FPS, and hence, it guarantees the real time response. It has been concluded that, the efficient implementation of this library plays an important role in the development of distributed vision processing algorithms.
- (ii) A vision based obstacle avoidance technique has been developed, which provided the frame processing rate of 45 FPS. This shows the successful achievement of the second objective set in this research, and that is, the provision of a light weight obstacle avoidance algorithm to help the robots to avoid colliding with the obstacles.
- (iii) To facilitate the transition from swarm to robotic organism, a vision based docking algorithm has been implemented. This algorithm facilitates the robot to align and get close (i.e., 5cm) to the docking port. With vision based support, it is not possible to perform full docking, because when the robot tries to get very close (i.e., less than 5cm) to the docking port then the markers used to detect the docking port goes outside the robot field of view. To achieve final mechanical docking mechanism, it has been

concluded that the information from other sensor (i.e., IR sensor) will be required.

- (iv) In the swarm mode scenarios it has been shown that, the swarm of robots share novel distance vector features to efficiently generate a precise environmental map. This distance vector features are very easy to compute and not encoded with high density information, and this makes it possible to exchange these features using the wireless channel with limited bandwidth.
- (v) A novel approach to perform object recognition and localisation using distributed robotic system has been addressed. The localisation of objects requires the provision of efficient vision based recognition functionality. It has been concluded that to reduce the object recognition time on small size robots, the compiler, system and assembly level optimisation of algorithm plays an important role. The use of algorithm optimisation together with novel image pre-processing and distance based resolution techniques not only reduced the execution time (i.e., from 33 sec to 800 msec) of the algorithm, but it has also enabled the robot to recognise far lying objects. Hence, it fulfils the objective of providing robust vision based recognition functionality to the small size robot.
- (vi) From the research work presented in the swarm mode scenarios, it has been concluded that, the generation of environmental map facilitates the robots to navigate in the environment and do path planning efficiently. Similarly, recognising and localising the objects of interest in the environment also helps the robots to understand their environment. Hence, the environment mapping and objects localisation operations provides surrounding awareness to the swarm of robots, which is a major objective of this research.
- (vii) It has been concluded that the most critical aspect of the distributed vision processing is the modular implementation of the vision processing tasks. This has been shown in a organism mode scenario, where the modular implementation of object recognition, mosaics generation, and detection of objects in these mosaics is addressed. However, it has been concluded that, for obtaining effective surrounding awareness, the features of the detected objects needed to be learned by the organism.
- (viii) In organism mode scenarios it has been shown that, the addition of new object features significantly increases memory requirement. Therefore, it has

been concluded that an efficient utilisation of memory resources is critical when dealing with distributed vision processing applications. These issues are addressed in the second organism mode scenario, which quantised the objects' features space to provide fast response to object recognition. This scenario also presents a novel mechanism to learn newly detected objects in the organism. However, the drawback of the approach is reduction in the classification probability as the number of target objects increases.

- (ix) It has been concluded that the collective use of the memory and processing resources of the robotic organism can provide performance comparable to an individual less flexible robot (e.g., Pioneer-3AT) with significant higher processing capability.

9.2 Future Work

From the distributed vision processing scenarios presented in this research, number of different areas have been identified where further improvement can be made. These are described as follows:

- Using vision based docking support presented in this research, it is not possible to bring the robot closer than 5cm to the docking port. To bring the robot further close to the port so that mechanical docking operation can be performed, information from vision and IR sensors can be fused.
- To achieve further precision in environmental map generated by the swarm of robots, the use of small size laser line emitter can be made. Laser line emitted by the laser can be easily detected in the vision information, and distance to nearby objects can be determined. This distance information can be fused with the novel distance vector features, to increase accuracy of environmental maps.
- The idea behind distributed object recognition and localisation approach can be used to dynamically localise the robot positions without using information from Visual Tracking System. For this purpose, a number of target landmarks can be used in the environments and their coordinates information can be provided to the robots. When the robots recognise these target objects, then using the coordinates information of these objects, the robots can also localise their own positions.

- The distributed object recognition and visual information gathering approach can be further expanded to localise the positions of detected objects in the mosaics. To achieve this, the robot localisation information can be acquired from the Visual Tracking System, and objects can be localised with reference to robots positions. This objects localisation will facilitate the robot to navigate in the environment.
- In distributed object classification and recognition approach, it has been observed that the objects' classification probability decreases as the number of target objects increases. This occurred due to increase in the number of common features among target objects. To improve the results, the use of common features should be avoided. All the features should also be assigned different weights which depends upon the entropy associated with those features. These entropies will be assigned in such a way that, it discourages the use of those feature which has produced false classification results.
- To exhibit the organism learning behaviour, some modules in the organism acted like memory bank where the organism knowledge evolves. This can be extended to multiple modules holding the organism knowledge in a distributed fashion. This way, if one of the modules holding the organism knowledge crashes, the other modules keeping the knowledge backup replace the malfunctioned module.
- Just like the robotic research expanded from the single robot to the swarm robotic system and then to the modular or multi-robotic organism. In future, this research can be expanded to the swarm of multi-robotic organisms in which the swarm of organisms would be performing the multi-level distributed processing. That is from a distributed processing in an organism to the distributed processing in the swarm of organisms.

References

- [1] S. Kernbach, M. Szymanski, T. Schmickl and P. Corradi, "Symbiotic Robot Organisms: Replicator and Symbion Projects", *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems (PerMIS), Special Session on EU-projects, Gaithersburg, MD, USA*, Pages:62-69, August 19-21, 2008.
- [2] S. Kernbach, E. Meister, O. Scholz, R. Humza, J. Liedke, L. Ricotti, J. Jemai, J. Havlik and W. Liu., "Evolutionary Robotics: The Next Generation-Platform for On-line and On-board Artificial Evolution.", *IEEE Congress on Evolutionary Computation, Trondheim, Norway*, Pages: 1079-1086, 2009.
- [3] M. Bonani, V. Longchamp, S. Magnenat, P. Rognoz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler and F. Mondada., "The MarXbot, a Miniature Mobile Robot Opening new Perspectives for the Collective-robotic Research.", *IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan*, Pages: 4187-4193, October 18-22, 2010.
- [4] K. Hyun, Y. Lee, H. Jin, B. Hyun and D. Hwan., "Swarm Robotics: Self Assembly, Physical Configuration and Its Control.", *ICASE International Joint Conference in Bexco, Busan, Korea*, Pages: 4276-4279 , 8-21 Oct, 2006.
- [5] J. Caddy, "Ants: Monarchs of Symbiosis", <http://www.morning-earth.org/graphic-e/SymbiosisAnts.html>, 2009.
- [6] E. Tuci, R. Gro, V. Trianni, F. Mondada, M. Bonani and M. Dorigo, "Cooperation through self-assembly in multi-robot", *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 1, Issue. 2, Pages: 115-150, 2006.

- [7] Symbrion, "Symbiotic Evolutionary Robot Organisms", *7th Framework Programme Project No FP7-ICT-2007.8.2. European Communities*, URL:www.symbrion.eu/, 2008.
- [8] Replicator, "Robotic Evolutionary Self-Programming and Self-Assembling Organisms", *7th Framework Programme Project No FP7-ICT-2007.2.1. European Communities*, URL:www.symbrion.eu/, 2008.
- [9] M. Eusterbrock, "Altruism for the Win", <http://ccsbio.blogspot.co.uk/2011/02/altruism-for-win.html>, 2011.
- [10] D. Brugali and M. E. Fayad., "Distributed Computing in Robotics and Automation", *Transactions on Robotics and Automation, IEEE*, Vol. 18, Issue. 4, Pages: 409-420, August 2002.
- [11] X. Defago, "Distributed Computing on the Move: From mobile computing to cooperative robotics and nano robotics.", *In Proceedings of 1st ACM International Workshop on Principles of Mobile Computing (POMC01), Newport, RI, USA*, Pages: 49-55, 2001.
- [12] E. Menegatti and E. Pagello., "Cooperative Distributed Vision for Mobile Robots.", *Workshop on Artificial Intelligence, Vision and Pattern Recognition 7th Conference of the Italian Association for Artificial Intelligence (AI*IA)*, Pages: 75-82, 2001.
- [13] C. Ampatzis, E. Tuci, V. Trianni, A.L. Christensen and M. Dorigo, "Evolving Autonomous Self-Assembly in Homogeneous Robots", *IRIDIA-Technical Report*, 2008.
- [14] E. Tuci, R. Gross, V. Trianni, F. Mondada, M. Bonani and M. Dorigo, "Cooperation Through Self-assembling in multi-robot systems", *IRIDIA-Technical Report*, 2005.
- [15] R. Gross, F. Mondada, M. Bonani and M. Dorigo, "Autonomous Self-Assembly in Mobile Robotics", *IRIDIA-Technical Report*, 2005.
- [16] B.P. Gerkey, R.T. Vaughan and A. Howard., "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems", *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Pages: 317-323, 2003.
- [17] H.V. Balan, "Visual Object Recognition in RoboCup", *RoboCup*, 2004.

- [18] E. Menegatti and E. Pagello., "Omnidirectional distributed vision for multi-robot mapping", *In Proceedings of International Symposium on Distributed Autonomous Robotic Systems (DARS02)*, Pages: 279-288, 2002.
- [19] E. Menegatti, A. Scarpa, D. Massarin, E. Ros and E. Pagello, "Omnidirectional Distributed Vision System for a Team of Heterogeneous Robot", *Conference on Computer Vision and Pattern Recognition Workshop*, Vol. 7, Pages: 87-87, 2003.
- [20] P.G. Selfridge and S. Mahakian., "Distributed Computing for Vision: Architecture and Benchmark test", *IEEE-Transactions on Pattern Analysis and Machine Intelligence*, Vol. 7, Issue. 5, Pages: 750-755, 2009.
- [21] W. Agassounon, "Distributed Information Retrieval and Dissemination in Swarm-Based Networks of Mobile Autonomous Agents", *IEEE-Swarm Intelligence Symposium (SIS)*, Pages: 152-159, 2003.
- [22] S.S. Nestinger and H.H. Cheng, "Flexible Vision - Mobile Agent Approach to Distributed Vision sensor Fusion", *IEEE-Robotics And Automation Magazine*, Vol. 17, Pages: 66-77, 2010.
- [23] T. Matsuyama, "Cooperative Distributed Vision - Dynamic integration of Visual Perception", *Action and Communication. Proceeding KI '99 Proceedings of the 23rd Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence*, Vol. 1701, Pages: 75-88, 1999.
- [24] A. Shokripour and M. Othman, "Survey on Divisible Load Theory and its Applications", *IEEE-International Conference on Information Management and Engineering*, Pages: 300-304, 3-5 April, 2009.
- [25] A. Shokripour and M. Othman, "Categorizing DLT Researches and its Applications", *European Journal of Scientific Research*, Vol. 37, Issue. 3, Pages: 496-515, 2009.
- [26] X.L. Li, B. Veeravalli and C.C. Ko, "Distributed image processing on a network of workstations", *International Journal of Computers and Applications*, Vol. 25, Issue. 2, Pages: 136-145, 2003.
- [27] J. Fernandez, R. Guerreroa, N. Mirandaa and F. Piccolia, "A distributed image processing function set for an image mining system", *Mecanica Computacional (artculo completo)*. Alberto Cardona, Mario Storti, Carlos Zuppa. (Eds.), Pages: 2895-2906, 2008.

- [28] Z. Shen, J. Luo, G. Huang, D. Ming, W. Ma and H. Sheng, "Distributed computing model for processing remotely sensed images based on grid computing", *Information Sciences*, Vol. 177, Issue. 2, Pages: 504-518, 15 January 2007.
- [29] R. Vidal, O. Shakernia and S. Sastry, "Following the Flock: Distributed Formation Control with Omnidirectional Vision-Based Motion Segmentation and Visual Servoing.", *IEEE-Robotics And Automation Magazine*, December, 2004.
- [30] E. Menegatti and E. Pagello, "Cooperation Issues and Distributed Sensing for Multirobot Systems", *IEEE-Special issue: Multirobot systems*, Vol 94, Issue. 7, Pages: 1370-1383, July 2006.
- [31] P. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint and M.D. Gini, "Communication Strategies in Multi-Robot Search and Retrieval: Experiences with Mindart", *Proceedings of 7th International Symposium Distributed Autonomous Robotic Systems*, Pages: 301-310, 2007.
- [32] L. Panait and S. Luke, "A Pheromone-based Utility Model for Collaborative Foraging", *IEEE-Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-agent Systems*, Vol. 1, Pages: 36-43, 2004.
- [33] S.Y. Harmon and D.W. Gage, "Protocols for robot communications: Transport and content layers", *Proceedings of 1980 International Conference on Cybernetics and Society*, Pages: 1090-1097, 1980.
- [34] V.R. Lesser and D.D. Corkill., "Functionally Accurate, Cooperative Distributed Systems", *IEEE-Transactions on Systems, Man and Cybernetics*, Vol. 11, Issue. 1, Pages: 81-96, 1981.
- [35] R. Wirz, J. Sales, R. Marn, E. Cervera, U. Witkowski, J.V. Mart and L. Nomdedeu, "End-to-End Congestion Control Protocols for Robot Swarms using Ad-hoc Wireless Networks", *Proceedings of the RISE 2008. Robotics for Risky Interventions and Surveillance of the Environment, Publications de la Universitat Jaume I, Castell. Isbn: 978-84-8021-645-6.*, 2008.
- [36] D.L. Johnson and A. Lysko, "Comparison of MANET routing protocols using a scaled indoor wireless grid", *Mobile Networks and Applications*, Vol. 13, Issue. 1-2, Pages: 82-96, 2008.

- [37] C. Agüero, J.M. Canas, M. Ortuno and V. Matellan, "Design and Implementation of an Ad-Hoc Routing Protocol for Mobile Robots", *Turk Journal of Electronic Engineering*, Vol. 15, Issue. 2, 2007.
- [38] S. Jia, Y. Hada, G. Ye and K. Takase, "Distributed Tele care Robotic Systems Using Corba as a Communication Architecture", *Proceedings of the 2002 IEEE. International Conference on Robotics And Automation. Washington, DC*, Vol. 2, Pages: 2202-2207, 2002.
- [39] J. Yool, S. Kim and S. Hong., "The Robot Software Communications Architecture (RSCA): QoS-Aware Middle-ware for Networked Service Robots", *SICE-ICASE International Joint Conference in Bexco, Busan, Korea*, Pages: 330-335, 2006.
- [40] A. Saxena and J. Michels, "High speed obstacle avoidance using monocular vision and reinforcement learning", *Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany*, Pages: 593-600, 2005.
- [41] J. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors", *IEEE Journal of robotics and automation*, Vol. 4, Issue. 2, Pages: 213-218, 1988.
- [42] K.S. Pratt, "Smart sensors for optic flow, obstacle avoidance for mavs in urban environments", *ISSRT, IROS 2007-MAV Workshop San Diego, CA*, 2007.
- [43] K. Souhila and A. Karim, "Optical flow based robot obstacle avoidance", *InTech-International Journal of Advanced Robotic Systems*, Vol. 4, Issue. 1, Pages: 13-16, 2007.
- [44] W.M. Shen and P. Will, "Docking in self-reconfigurable robots", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, Pages: 1049-1054, 2001.
- [45] S. Kornienko, O. Kornienko, A. Nagarathinam and P. Levi., "From real robot swarm to evolutionary multi-robot organism", *IEEE Congress on Evolutionary Computation*, Pages: 1483-1490, 2007.
- [46] R. Gro, M. Dorigo and M. Yamakita, "Self-assembly of mobile robots - From swarm-bot to super-mechano colony", *Intelligent Autonomous System (IAS) 9, Tokyo, Japan*, 2006.

- [47] C. Ampatzis, E. Tuci, V. Trianni, A.L. Christensen and M. Dorigo, "Evolving Self-Assembly in Autonomous Homogeneous Robots: Experiments with Two Physical Robots", *Artificial Life*, Vol. 15, Issue. 4, Pages: 465-484, 2009.
- [48] S. Murata, K. Kakomura and H. Kurokawa, "Docking Experiments of a Modular Robot by Visual Feedback", *IEEE/RSJ-International Conference on Intelligent Robots and Systems*, Pages: 625-630, 2006.
- [49] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan and C.J. Taylor, "Towards robotic self-reassembly after explosion", *IEEE/RSJ-International Conference on Intelligent Robots and Systems*, Pages: 2767-2772, 2007.
- [50] D. Hhnel, W. Burgard, D. Fox, K. Fishkin and M. Philipose, "Mapping and Localization with RFID Technology", *IEEE International Conference on Robotics and Automation. Proceedings. ICRA*, Vol. 1, Pages: 1015-1020, 2004.
- [51] H. Lim and Y. S. Lee, "Real-time single camera SLAM using fiducial markers", *ICROS-SICE International Joint Conference, Fukuoka International Congress Center, Japan*, Pages: 177-182, 18-21 Aug. 2009.
- [52] B. Sohn, J. Lee, H. Chae and W. Yu, "Localization system for mobile robot using wireless communication with IR landmark", *RoboComm '07 Proceedings of the 1st international conference on Robot communication and coordination*, Article No. 6, Pages: 61-66, 2007.
- [53] S. Panzieri, F. Pascucci, R. Setola and G. Ulivi, "A low cost vision based localization system for mobile robots", *In 9th Mediterranean Conference on Control and Automation, Dubrovnik, Croatia*, 2001.
- [54] K.J. Yoon and I.S. Kweon, "Landmark design and real-time landmark tracking for mobile robot localization", *Proceedings of SPIE*, Vol. 4573, 2002.
- [55] Andrew C. Rice, Robert K. Harle and Alastair R. Beresford, "Analysing fundamental properties of marker-based vision system designs", *In Pervasive and Mobile Computing*, Vol. 2, Issue. 4, Pages: 453-471, 2006.
- [56] P. Furgale, J. Anderson and J. Baltes., "Real-Time Vision-Based Pattern Tracking Without Predefined Colors", *In Proceedings of the Third Interna-*

- tional Conference on Computational Intelligence, Robotics and Automation, Singapore, 2005.*
- [57] R.K. Harle, A.C. Rice and A.R. Beresford, "Cantag: an open source software toolkit for designing and deploying marker-based vision systems", *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Pages: 10-21, 2006.
- [58] A. Mutka, D. Miklic, I. Draganjac and S. Bogdan, "A low cost vision based localization system using fiducial markers", *Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea, Vol. 17, Issue. 1*, Pages: 9528-9533, 2008.
- [59] M. Tabuse and D. Nakai, "Mobile Robot Navigation using SURF features", *ACS'10 Proceedings of the 10th WSEAS international conference on Applied computer science*, Pages: 276-279, 2010.
- [60] K. Welke, P. Azad and R.D. Dillmann, "Fast and Robust Feature-based Recognition of Multiple Objects", *International Conference on Humanoid Robots, 6th IEEE-RAS*, Pages: 264-269, 2007.
- [61] T. Goedeme, D.N. Instituut and J.D. Nayerlaan, "Traffic sign recognition with constellations of visual words", *International conference on informatics in control, automation and robotics - ICINCO*, Vol. 1, Pages: 222-227, 2008.
- [62] H. Bay, T. Tuytelaars and L.V. Gool, "SURF speeded up robust features", *Computer Vision and Image Understanding (CVIU)*, Vol. 110, Pages: 346-359, 2008.
- [63] D. Asanza and B. Wirtzner., "Improving feature based object recognition in service robotics by disparity map based segmentation", *International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ*, Pages: 2716-2720, 2010.
- [64] M. Cummins and P. Newman, "FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model", *Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel*, Pages: 3-10, 2010.
- [65] G. Chrysanthakopoulos and G. Shani, "Augmenting Appearance-Based Localization and Navigation using Belief Update", *In Proceedings of AA-MAS 2010*, Vol. 2, Pages: 559-566, 2010.

- [66] M. Ullah, A. Pronobi, B. Caputo, J. Luo, P. Jensfelt and H.I. Christensen, "Towards Robust Place Recognition for Robot Localization", *IEEE/International Conference on Robotics and Automation, ICRA*, Pages: 530-537, 2008.
- [67] C. Cadena, D.G. Lopez, F. Ramos, J.D. Tardos and J. Neira, "Robust Place Recognition with Stereo Cameras", *International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ*, Pages: 5182-5189, 2010.
- [68] R.C. Rodrigues and S.R. Pellegrino., "An Experimental Evaluation of Algorithms for Aerial Image Matching", *Electronic Engineering and Computer Department. IWSSIP 2010 - 17th International Conference on Systems, Signals and Image Processing*, 2010.
- [69] L. Juan and O. Gwun, "A Comparison of SIFT, PCA-SIFT and SURF", *International Journal of Image Processing (IJIP)*, Vol. 3, Issue. 4, Pages: 143-152, 2009.
- [70] B.J.A. Krose, N. Vlassis, R. Bunschoten and Y. Motomura, "A probabilistic model for appearance-based robot localization", *In First European Symposium on Ambience Intelligence (EUSAI)*, Pages: 264-274, 2001.
- [71] G. Dunteman, "Principal Components Analysis", *Sage University paper series on quantitative application in the social sciences (Series No. 07-069)*, Beverly Hills: Sage, Vol. 69, 1989.
- [72] D. Sabatta, D. Scaramuzza and R. Siegwart, "Improved Appearance-Based Matching in Similar and Dynamic Environments using a Vocabulary Tree", *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA*, Pages: 1008-1013, 2010.
- [73] Y. Shen, J. Liu and D. Xin, "Environment map building and localization for robot navigation based on image sequences", *Journal of Zhejiang University Science A*, Vol.9, No.4, Pages: 489-499, 2008.
- [74] D.F. Wolf and A.Y. Hata, "Outdoor Mapping Using Mobile Robots And Laser Range Finders", *Conference of Electronics, Robotics and Automotive Mechanics*, Pages: 209-214, 2009.
- [75] Y.D. Kwon and J. Lee, "A Stochastic Map Building Method for Mobile Robot using 2-D Laser Range Finder", *Journal of Autonomous Robots*, Vol.7 No.2, Pages: 187-200, 1999.

- [76] P. Biber, H. Andreasson, T. Duckett and A. Schilling, "3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol.4, Pages: 3430-3435, 2004.
- [77] A. Howard, "Multi-Robot Mapping using Manifold Representations", *IEEE International Conference on Robotics and Automation(ICRA)*, Vol.4, Pages: 4198-4203, 2004.
- [78] L.J. Latecki, R. Lakaemper and N. Adluru, "Multi Robot Mapping Using Force Field Simulation", *Journal of Field Robotics*, Vol.24, Pages: 747-762, 2007.
- [79] A. Leon, R. Barea, L.M. Bergasa, E. Lopez, E. Ocana and D. Schleicher, "Multi-robot SLAM and Map Merging", *Journal of Physical Agents*, Vol.3, Pages: 171-176, 2009.
- [80] S. Kernbach, H. Hamann and J. Stradner, "On Adaptive Self-Organization in Artificial Robot Organisms.", *The First International Conference on Adaptive and Self-adaptive Systems and Applications*, Pages:33-43, 2009.
- [81] S. Kernbach, M. Szymanski, T. Schmickl and P. Corradi, "Symbiotic Robot Organisms: Replicator and Symbion Projects.", *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems (PerMIS), Special Session on EU-projects*, Gaithersburg, MD, USA. August 19-21, Pages:62-69, 2008
- [82] M. Dorigo, D. Floreano, L. Maria Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A.L. Christensen, A. Decugniere, G. Di Caro, F. Ducatelle, F. Ferrante, A. Froster, J.G. Martinez, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M.A. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Rtoraz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A.E. Turgut and F. Vaussard. "Swarmanoid: A novel concept for the study of heterogeneous robotic swarms.", *IEEE Robotics Automation Magazine*, July 2011
- [83] X. Defago, "Distributed Computing on the Move: From mobile computing to cooperative robotics and nano robotics.", *In Proceedings of 1st ACM International Workshop on Principles of Mobile Computing (POMC01)*, Newport, RI, USA, Pages: 49-55, 2001

- [84] D. Brugali and M. E. Fayad, "Distributed Computing in Robotics and Automation.", *IEEE, Transactions on Robotics and Automation*, Volume 18, No.4 Pages: 409-420, 2002.
- [85] V. Kumar, G. Bekey and A. Sanderson, "Chapter 7: Networked Robots.", *Automation Magazine*, Volume 12, No. 2, Pages: 73-80, 2006.
- [86] M. Yim, W.M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins and G.S. Chirikjian, "Modular Self-Reconfigurable Robot System.", *IEEE Robotics and Automation Magazine*, ISSN 1070-9932, Volume 14, No. 1, Pages: 43-52, 2007
- [87] Y. Zhang, M. Yim, C. Eldershaw, D. Duff and K. Roufas, "Scalable and Reconfigurable Configurations and Locomotion Gaits for Chain-type Modular Reconfigurable Robots", *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2003 July 16 - 20; Kobe; Japan, Volume 2, Pages: 893-899, 2003
- [88] T. Fukuda and S. Nakagawa, "Dynamically Reconfigurable Robotic System.", *Proceedings of the IEEE International Conference On Robotics and Automation*, Volume 3, Pages: 1581-1586, 1988.
- [89] Surveyor, "Functional detail of surveyor robot", <http://www.surveyor.com>, 2010.
- [90] Surveyor, "Analog devices blackfin bf537 processor data sheet", <http://www.bluetechnix.at>, 2009.
- [91] Bluetechnix, "Hardware user manual for BF537E", <http://www.bluetechnix.at>, 2009.
- [92] Analog Devices., " Embedded processor ADSP-BF537 datasheet", www.analog.com, 2008.
- [93] Analog Devices, "Ad1836 data sheet", www.analog.com, 2009.
- [94] Analog Devices, "ADSP-BF537 Blackfin processor hardware reference rev. 2.0.", www.analog.com, 2007.
- [95] M.H. Hongche Liu, T. Hong and R. Chellappa, "Accuracy vs. efficiency trade-offs in optical flow algorithms", *Computer Vision and Image Understanding*, Vol. 72, Issue. 3, Pages: 271-286, 1998.
- [96] K.P. Horn and B.G. Schunck., "Determining optical flow", *Artificial Intelligence*, Vol. 17, Pages: 185-203, 1981.

- [97] G. Bradski and A. Kaehler, "Computer Vision with the OpenCV Library", *Learning OpenCV, O'Reilly Media*, Pages: 576, 2008.
- [98] C. Evans, "Notes on the OpenSURF Library", *CSTR-09-001, University of Bristol*, 2009.
- [99] D. Katz, T. Lukasiak and R. Lukasiak, "Enhance Processor Performance in Open-Source Applications", *Analog Dialogue*, Vol. 39, Issue. 2, Pages: 1-4, 2005.
- [100] Bluetechnix, "Hardware reference manual for Blackfin Evaluation Board EVAL-BF5xx", <http://www.bluetechnix.at>, 2009.
- [101] Bluetechnix, "Hardware reference manual for Blackfin Extender Board EXT-BF5xx-Camera", <http://www.bluetechnix.at>, 2009.
- [102] R.C. Gonzalez and R.E. Woods, "Digital Image Processing (2nd Edition)", *Prentice Hall, Isbn / ASIN: 0201180758*, 2007.
- [103] M. Simon, S. Behnke and R. Rojas, "Robust Real Time Colour Tracking", *4th International Workshop on RoboCup (Robot World Cup Soccer Games and Conferences), Lecture Notes in Computer Science*, Pages: 239-248, 2000.
- [104] LinkSys, "LinkSys Smart Wi-Fi Routers and Access Point", www.linksys.com/, 2011.
- [105] Logitech, "Logitech WebCam Pro 9000", <http://www.logitech.com/>, 2011.
- [106] Bluetechnix, "Hardware reference manual for Blackfin Experimental Board - EXT-BF5xx-EXP", <http://www.bluetechnix.at>, 2009.
- [107] OmniVision, "Ov7670 VGA Camera Sensor Product Brief", www.ovt.com, 2011.
- [108] Flexi Cables., "Omni Vision Camera Samsung D600 Flexi Cables.", www.samsung-cables.supaprice.co.uk, 2010.
- [109] Surveyor Corporation, "Surveyor SRV-1 Open Source Mobile Robot Base and Charger.", , www.surveyor.com/SRV_info.html, 2011.
- [110] Lantronix, "Lantronix: MatchPort b/g an Embedded Wireless Device Server for Wireless Networking Solution", www.lantronix.com, 2011.
- [111] Ryan Mechatronics., "Navigation Board M3 User Manual for Surveyor SRV-1 Robot", www.ryanmechatronics.com, 2010.

- [112] J. C. Moser., "Town ant workers following an artificial pheromone trail", http://www.srs.fs.usda.gov/idip/spb/i/photos_ants.html, 2004.
- [113] A. Ramisa, S. Vasudevan, D. Scaramuzza, R. Opez and L. Antaras, "A Tale of Two Object Recognition Methods for Mobile Robots", *6th International Conference on Computer Visions Systems, Springer Verlag, Santorini*, Vol. 5008, Pages: 353-362, 2008.
- [114] D. Lowe, "Distinctive image features from scale-invariant key points", *International Journal of Computer Vision*, Vol. 60, Issue. 2, Pages: 91-110, 2004.
- [115] J.Y. Bouguet, "Camera Calibration Toolbox for Matlab", <http://www.vision.caltech.edu/bouguetj/calibdoc/>, 2010.
- [116] J. Yang, Y.G. Jiang, Hauptmann, G. Alexander and N. Chong-Wah, "Evaluating bag-of-visual-words representations in scene classification", *Proceedings of the international workshop on multimedia information retrieval*, Pages: 197-206, 2007.
- [117] J.R. Uijlings, A.W.M. Smeulders and R.J.H. Scha, "Real-time bag of words, approximately", *Proceedings of the ACM International Conference on Image and Video Retrieval*, Pages: 61-68, 2009.
- [118] D. Filliat, "A visual bag of words method for interactive qualitative localization and mapping", *IEEE International Conference on Robotics and Automation*, Pages: 3921-3926, 2007.
- [119] Analog Devices, "Fast Floating-Point Arithmetic Emulation on Blackfin Processors", *Analog Devices: Technical notes on using Analog Devices DSPs, processors and development tools*, 2007.
- [120] J. Canny, "A Computational Approach To Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, Issue. 6, Pages: 679-698, 1986.