

Object recognition and real-time tracking in microscope imaging

WEDEKIND, Jan, BOISSENIN, Manuel, AMAVASAI, Balasundram P., CAPARRELLI, Fabio and TRAVIS, Jon R.

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

<http://shura.shu.ac.uk/3738/>

This document is the author deposited version. You are advised to consult the publisher's version if you wish to cite from it.

Published version

WEDEKIND, Jan, BOISSENIN, Manuel, AMAVASAI, Balasundram P., CAPARRELLI, Fabio and TRAVIS, Jon R. (2006). Object recognition and real-time tracking in microscope imaging. In: Proceedings of the 2006 Irish Machine Vision and Image Processing Conference (IMVIP 2006). Dublin, Vision System Group, 164-171.

Copyright and re-use policy

See <http://shura.shu.ac.uk/information.html>

Object Recognition and Real-Time Tracking in Microscope Imaging

J. Wedekind, M. Boissenin, B.P. Amavasai, F. Caparrelli, J. Travis

MMVL, Materials and Engineering Research Institute

Sheffield Hallam University,

Pond Street,

Sheffield S1 1WB

{J.Wedekind,B.P.Amavasai,F.Caparrelli,J.R.Travis}@shu.ac.uk,

Manuel.Boissenin@student.shu.ac.uk

30.6.2006

Abstract

As the fields of micro- and nano-technology mature, there is going to be an increased need for industrial tools that *enable the assembly and manipulation of micro-parts*. The feedback mechanism in a future micro-factory will require computer vision.

Within the **EU IST MiCRoN** project, a computer vision software based on *Geometric Hashing* and the *Bounded Hough Transform* to achieve **recognition of multiple micro-objects** was implemented and successfully demonstrated. In this environment, the micro-objects will be of variable distance to the camera. Novel automated procedures in biology and micro-technology are thus conceivable.

This paper presents an approach to estimate the pose of multiple micro-objects with up to four degrees-of-freedom by using focus stacks as models. The paper also presents a formal definition for Geometric Hashing and the Bounded Hough Transform.

Keywords: object recognition, tracking, Geometric Hashing, Bounded Hough Transform, microscope

1 Introduction

Under the auspices of the European *MiCRoN* [MiCRoN consortium, 2006] project a system of multiple micro-robots for transporting and assembling μm -sized objects was developed. The micro-robots are about 1 cm^3 in size and are fitted with an interchangeable toolset that allows them to perform manipulation and assembly. The project has developed various subsystems for powering, locomotion, positioning, gripping, injecting, and actuating. The task of the *Microsystems & Machine Vision Lab* was to develop a real-time vision system, which provides feedback information to the control system and hence forms part of the control loop.

Although there are various methods for object recognition in the area of computer vision, most techniques which have been developed for microscope imaging so far do not address the issue of real-time. Most current work in the area of micro-object recognition employs 2-D recognition methods (see *e.g.* [Begelman et al., 2004]) sometimes in combination with an auto-focussing system, which ensures that the object to be recognised always stays in focus.

This paper presents an algorithm for object recognition and tracking in a microscope environment with the following objectives: Objects can be recognised with *up to 4 degrees-of-freedom, refocussing is not required, tracking is performed in real-time.*

The following sections of this paper will provide a formalism to *Geometric Hashing* and the *Bounded Hough Transform*, how they can be applied to a pre-stored focus stack, and how this focus stack can be used to recognise and track objects, the results, and finally we draw some conclusions.

2 Formalism

In applying Geometric Hashing and the Bounded Hough Transform to micro-objects, recognition and tracking with three degrees-of-freedom is first developed. Later this is expanded to four degrees-of-freedom.

2.1 Geometric Hashing

[Forsyth and Ponce, 2003] provides a complete description of the Geometric Hashing algorithm first introduced in [Lamdan and Wolfson, 1988]. Geometric Hashing is an algorithm that uses geometric invariants to vote for feature correspondences.

2.1.1 Preprocessing-Stage

Let the triplet $\vec{p} := (t_1, t_2, \theta)^\top \in P$ be the pose of an object and $P := \mathbb{R}^3$ the pose-space. $\dim(P) = 3$ is the number of degrees-of-freedom. The object can rotate around one axis and translate in two directions. It can be found using a set $M \subset \mathbb{R}^{d+1}$ of homogeneous coordinates denoting 2-D ($d = 2$) feature points (here: edge-points acquired with a Sobel edge-detector) as a model

$$M := \{\vec{m}_i = (m_{i,1}, m_{i,2}, 1)^\top \mid i \in \{1, 2, \dots\}\} \quad (1)$$

First the set of geometric invariants $L(M)$ has to be identified. A geometric invariant of the model is a feature or a (minimal) sequence of features such that the pose of the object can be deduced if the location of the corresponding feature/the sequence of features in the scene is known.

Consider the example in fig. 1. In this case the correspondence between a two feature points $\vec{s}_1, \vec{s}_2 \in S$ in the scene $S \subset \mathbb{R}^{d+1}$ and two feature points $\vec{m}_1, \vec{m}_2 \in M$ of the model M would reveal the pose \vec{p} of the object. Therefore feature tuples can serve as geometric invariants.

In practice only a small number of feature tuples can be considered. A subset of $M \times M$ is selected by applying a minimum- and a maximum-constraint on the distance between the two feature points of a tuple (\vec{l}_1, \vec{l}_2) . Hence in this case, L is defined as $L(A) = \{(\vec{l}_1, \vec{l}_2) \in A \times A \mid g_u \leq \|\vec{l}_1 - \vec{l}_2\| \leq g_o\}$ for any set of features $A \subseteq \mathbb{R}^{d+1}$.

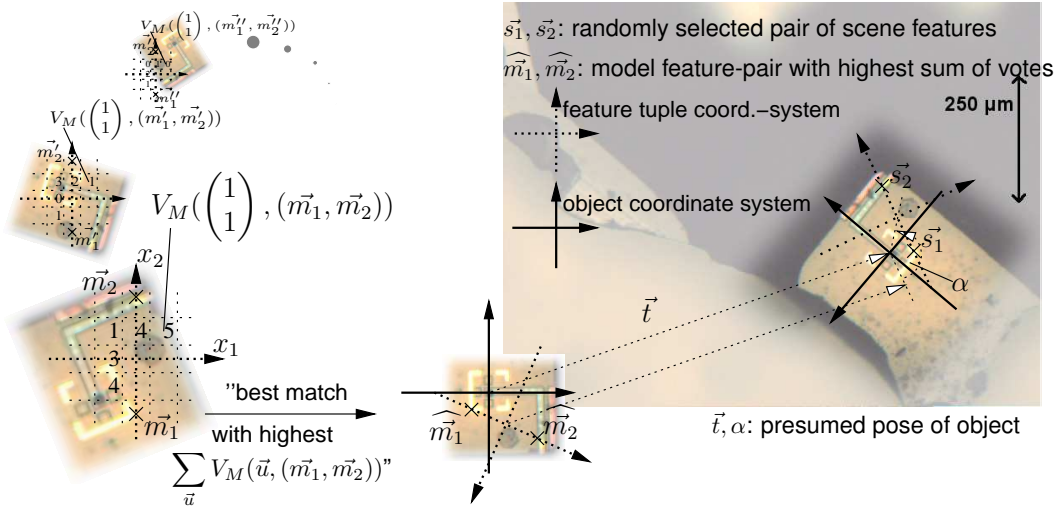


Figure 1: Geometric Hashing to locate a syringe-chip (courtesy of IBMT, St. Ingbert) in a microscope image (reflected light) allowing three degrees-of-freedom

Geometric Hashing provides a technique to establish the correspondence between the geometric invariants $(\vec{m}_1, \vec{m}_2) \in L(M)$ and $(\vec{s}_1, \vec{s}_2) \in L(S)$ where $S, M \subset \mathbb{R}^{d+1}$.

To apply Geometric Hashing a function

$$t : \begin{cases} L(\mathbb{R}^{d+1}) & \rightarrow \mathbb{R}^{(d+1) \times (d+1)} \\ (\vec{l}_n) & \mapsto T((\vec{l}_n)) \end{cases} \quad (2)$$

is chosen which assigns an affine transformation matrix $T((\vec{l}_n))$ to a geometric invariant $(\vec{l}_n) := (\vec{l}_1, \vec{l}_2, \dots, \vec{l}_n)$ in $L(M) \subset L(\mathbb{R}^{d+1})$ or $L(S) \subset L(\mathbb{R}^{d+1})$. The affine transformation inverses the transformation which is designated by the sequence of features $(\vec{l}_1, \vec{l}_2, \dots, \vec{l}_n)$. E.g. in this case t must fulfil

$$\forall \vec{p} \in P, (\vec{l}_1, \vec{l}_2) \in \mathbb{R}^3 \times \mathbb{R}^3 : t((\mathcal{R}(\vec{p}) \cdot \vec{l}_1, \mathcal{R}(\vec{p}) \cdot \vec{l}_2)) = \mathcal{R}(\vec{p}) \cdot t((\vec{l}_1, \vec{l}_2))$$

$$\text{where } \mathcal{R} \begin{pmatrix} t_1 \\ t_2 \\ \theta \end{pmatrix} := \begin{pmatrix} \cos(\theta) & -\sin(\theta) & t_1 \\ \sin(\theta) & \cos(\theta) & t_2 \\ 0 & 0 & 1 \end{pmatrix} \quad (3)$$

Furthermore t must conserve the pose-information, *i.e.* $\dim(\text{aff}(t(L(\mathbb{R}^{d+1})))) = \dim(P)$ where $\text{aff}(X)$ is the affine hull of X .

Note that \vec{l} and \vec{l}' are homogeneous coordinates of points, and for simplification we can use $l_3 = l'_3 = 1$. Using this, a possible choice for t is given by

$$t((\vec{l}, \vec{l}')) = \frac{1}{\sqrt{(l'_1 - l_1)^2 + (l'_2 - l_2)^2}} \begin{pmatrix} l'_2 - l_2 & l_1 - l'_1 & 0 \\ l'_1 - l_1 & l'_2 - l_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -\frac{1}{2}(l_1 + l'_1) \\ 0 & 1 & -\frac{1}{2}(l_2 + l'_2) \\ 0 & 0 & 1 \end{pmatrix} \quad (4)$$

The chosen transformation $t((\vec{m}_1, \vec{m}_2))$ maps the two feature points \vec{l} and \vec{l}' on the x_2 -axis as shown in figure 1.

Let h be a quantising function for mapping real homogeneous coordinates of feature positions to whole-numbered indices of voting table bins of discrete size Δs :

$$h : \begin{cases} \mathbb{R}^{d+1} & \rightarrow \mathbb{Z}^d \\ \vec{x} & \mapsto \vec{u} \text{ where } u_i = \left\lfloor \frac{x_i}{x_{d+1} \Delta s} + \frac{1}{2} \right\rfloor, i \in \{1, 2, \dots, d\} \end{cases} \quad (5)$$

[Blayvas et al., 2003] offers more information on how to choose the bin size Δs properly. Note that $x_{d+1} = 1$ since h is going to be applied to homogeneous coordinates of points only.

First a voting table $V_M : \mathbb{Z}^d \times L(M) \rightarrow \mathbb{N}_0$ for the model M is computed (see alg. 1)¹. In practice V_M only needs to be defined on a finite subset of \mathbb{Z}^d , while $L(M)$ is finite if M is.

Algorithm 1: Creating a voting table offline, before doing recognition with the Geometric Hashing algorithm[Forsyth and Ponce, 2003]

Input: Model $M \subset \mathbb{R}^{d+1}$

Output: Voting table $V_M : \mathbb{Z}^d \times L(M) \rightarrow \mathbb{N}_0$

/* Set all elements of V_M to zero */

$V_M(\cdot, \cdot) \mapsto 0$;

foreach geometric invariant $(\vec{m}_n) = (\vec{m}_1, \vec{m}_2, \dots, \vec{m}_n) \in L(M)$ **do**

foreach feature point $\vec{m}' \in M$ **do**

 /* Compute index of voting table bin */

$\vec{u} := h(t((\vec{m}_n)) \cdot \vec{m}')$;

 /* Add one vote for the sequence of features (\vec{m}_n) */

$V_M(\vec{u}, (\vec{m}_n)) \mapsto V_M(\vec{u}, (\vec{m}_n)) + 1$;

end

end

$(t((\vec{m}_1, \vec{m}_2)) \cdot \vec{m}')$ is the position of \vec{m}' relatively to the geometric invariant $(\vec{m}_1, \vec{m}_2) \in L(M)$. This relative position is quantised by h and assigned to \vec{u} . $V_M(\vec{u}, (\vec{m}_1, \vec{m}_2))$ is the number of features residing in the bin of the voting table with the quantised position \vec{u} relative to the geometric invariant $\vec{m} \in L(M)$.

2.1.2 Recognition-Stage

A random pair of features (\vec{s}_1, \vec{s}_2) is picked from the Sobel-edges of the scene-image. All other features of the scene are mapped using the transform $t((\vec{s}_1, \vec{s}_2))$ (see alg. 2). The accumulator a is used to decide where both features are located on the object and whether they are residing on the same object at all.

On success, sufficient information to calculate the pose of the object is available. The pose $\vec{p} = (t_1, t_2, \theta)^\top$ of the object can be calculated using:

$$\mathcal{R}(\vec{p}) = t((\vec{s}_1, \vec{s}_2))^{-1} t((\widehat{\vec{m}}_1, \widehat{\vec{m}}_2)) \quad (6)$$

2.2 Bounded Hough Transform

As Geometric Hashing alone is too slow to achieve real-time vision, a tracking algorithm based on the Bounded Hough Transform[Greenspan et al., 2004] was employed. Thus after a micro-object has been located, it can be tracked in consecutive frames with much lower computational cost.

¹ $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$

Algorithm 2: The Geometric Hashing algorithm for doing object recognition[Forsyth and Ponce, 2003]

Input: Set of scene features $S \subset \mathbb{R}^{d+1}$
Output: Pose \vec{p} of object or failure
 Initialise accumulator $a : L(M) \rightarrow \mathbb{N}_0$;
 Randomly select a geometric invariant $(\vec{s}_n) = (s_1, s_2, \dots, s_n)$ from $L(S)$;
foreach feature points $\vec{s}^t \in S$ **do**
 /* Compute index \vec{u} of voting table bin */
 $\vec{u} := h(t((\vec{s}_n)) \cdot \vec{s}^t)$;
 foreach $(\vec{m}_n) \in L(M)$ **do**
 /* Increase the accumulator using the voting table */
 $a((\vec{m}_n)) \mapsto a((\vec{m}_n)) + V_M(\vec{u}, (\vec{m}_n))$;
 end
end
 /* Find accumulator bin with maximum value */
 $(\widehat{\vec{m}}_n) := \underset{(\vec{m}_n) \in L(M)}{\operatorname{argmax}} (a((\vec{m}_n)))$;
if $a((\widehat{\vec{m}}_n))$ is bigger than a certain threshold **then**
 /* $t((\vec{s}_n))^{-1} t((\widehat{\vec{m}}_n))$ contains suggested pose of object.
 Back-project and verify before accepting the hypothesis[Forsyth and Ponce, 2003] */
else
 /* Retry by restarting algorithm or report failure */
end

2.2.1 Preprocessing-Stage

The basic idea of the Bounded Hough Transform is to transform the positions of all features $\vec{s} \in S$ to the coordinate-system defined by the object's previous pose \vec{p} . If the speed of the object is restricted by r_1, r_2, \dots (i.e. $|p'_i - p_i| \leq r_i, \vec{r} \in \mathbb{R}^{\dim(P)}$) and the change of pose is quantised by q_1, q_2, \dots (i.e. $\exists k \in \mathbb{Z} : p'_i - p_i = k q_i, \vec{q} \in \mathbb{R}^{\dim(P)}$), the problem of determining the new pose $\vec{p}' \in P$ of the object is reduced to selecting an element $\widehat{\vec{d}} := \vec{p}' - \vec{p}$ from the finite set $D \subset P$ of pose-changes

$$D := \{ \vec{d} \in P \mid \forall i \in \{1, \dots, \dim(P)\} : |d_i| \leq r_i \wedge \exists k \in \mathbb{Z} : d_i = k q_i \} \quad (7)$$

Fig. 2 illustrates how the Bounded Hough Transform works in the case of two degrees-of-freedom ($\vec{p} = (t_1, t_2)^\top$). The hough-space of pose-changes D is limited and therefore only the features residing within a small local area of M can correspond to the scene-feature $\vec{s} \in S$. Each possible correspondence

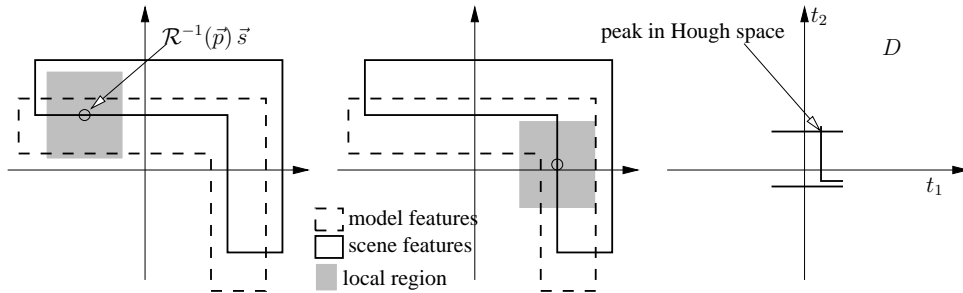


Figure 2: Bounded Hough Transform with 2 degrees-of-freedom

votes for one pose-change (in the general case it may vote for several different pose-changes). As one can see in fig. 2, accumulating the votes of two scene-features already can reveal the pose-change of the object.

First a voting table H_M is computed as shown in alg. 3. In practice H_M only needs to be defined on a finite subset of \mathbb{Z}^d while D is finite.

The functions $C : \mathbb{R}^{d+1} \rightarrow P$ and $W : \mathbb{R}^{d+1} \rightarrow \mathbb{R}_0^+$ are required to cover H_M properly. In the case of two degrees-of-freedom one can simply use $C(\vec{m}) = \{\vec{0}\}$ and $W(\vec{m}) = 1$ if the quantisation of the

Algorithm 3: Initialising voting table offline, before doing tracking using the Bounded Hough Transform algorithm

Input: Model $M \subset \mathbb{R}^{d+1}$, ranges \vec{r} , quantisation \vec{q}
Output: Voting table $H_M : \mathbb{Z}^d \times D \rightarrow \mathbb{N}_0$
 /* Set all elements of H_M to zero */
 $H_M(\cdot, \cdot) \mapsto 0;$
foreach pose difference vector $\vec{d} \in D$ **do**
 foreach feature point $\vec{m} \in M$ **do**
 foreach pose difference vector $\vec{c} \in C(\vec{m})$ **do**
 /* Compute index of voting table bin */
 $\vec{u} := h(\mathcal{R}(\vec{d} + \vec{c}) \cdot \vec{m});$
 /* Update votes for pose-change \vec{d} */
 $H_M(\vec{u}, \vec{d}) \mapsto H_M(\vec{u}, \vec{d}) + W(\vec{m});$
 end
end
end

translation in D does not exceed the bin-size (i.e. $q_i \leq \Delta s$).

In the case of three degrees-of-freedom ($\vec{p} = (t_1, t_2, \theta)^\top$) the *density* of the votes depends on the features distance from the origin (radius). If the radius is large, several bins of H_M may have to be increased. If the radius is very small, the weight of the vote should be lower than 1 as the feature cannot define the amount of rotation unambiguously. Therefore in the general case C and W are defined as follows

$$C(\vec{m}) := \{ \vec{c} \in P \mid \forall i \in \{1, \dots, \dim(P)\} : |c_i| \leq \frac{q_i}{2} \left\| \frac{\delta \mathcal{R}(\vec{x})}{\delta x_i} \vec{m} \right\| \wedge \exists k \in \mathbb{Z} : c_i = k \Delta s \} \quad (8)$$

$$W(\vec{m}) = \prod_{i=1}^{\dim(P)} \min \left(1, \left\| \frac{\delta \mathcal{R}(\vec{x})}{\delta x_i} \vec{m} \right\| \right) \quad (9)$$

In the case of three degrees-of-freedom C and W are defined using

$$\left(\left\| \frac{\delta \mathcal{R}((t_1, t_2, \theta)^\top)}{\delta t_1} \vec{m} \right\|, \left\| \frac{\delta \mathcal{R}((t_1, t_2, \theta)^\top)}{\delta t_2} \vec{m} \right\|, \left\| \frac{\delta \mathcal{R}((t_1, t_2, \theta)^\top)}{\delta \theta} \vec{m} \right\| \right)^\top = \begin{pmatrix} m_3 \\ m_3 \\ \sqrt{m_1^2 + m_2^2} \end{pmatrix} \quad (10)$$

Note that \vec{m} is a homogeneous coordinate of a point and therefore $m_3 = 1$.

2.2.2 Tracking-Stage

The tracking-stage of the Bounded Hough Transform algorithm is fairly straightforward. All features of the scene are mapped using the transform $\mathcal{R}(\vec{p})^{-1}$ defined by the previous pose \vec{p} of the object (see alg. 4). The accumulator b is used to decide where the object has moved or whether it was lost.

2.3 Four Degrees-of-Freedom

In practice the depth information contained in microscopy images can be used to achieve object recognition and tracking with four degrees-of-freedom. Recognition and tracking with four degrees-of-freedom is achieved by using two sets of competing voting tables $\{V_{M_1}, V_{M_2}, \dots\}$ and $\{H_{M_1}, H_{M_2}, \dots\}$, which have been generated from a focus stack of the object. Figure 3 shows an artificial focus stack of the text-object ‘‘Mimas’’, which is being compared against an artificial image, which contains two text-objects.

The voting tables for recognition can be stored in a single voting table V_M^* if an additional index for the depth is introduced. Furthermore during tracking only a subset of $\{H_{M_1}, H_{M_2}, \dots\}$ needs to be considered as the depth of the object can only change by a limited amount. In practice an additional index for change of depth is introduced, and a set of voting tables $\{H_{M_{1,2}}, H_{M_{1,2,3}}, H_{M_{2,3,4}}, \dots\}$ is created from images of neighbouring focus-layers. During tracking only a single voting table in this set needs to be considered.

Algorithm 4: The Bounded Hough Transform algorithm for tracking objects

Input: Set of scene features $S \subset \mathbb{R}^{d+1}$, previous \vec{p} of object
Output: Updated pose \vec{p} of object or failure
Initialise accumulator $b : D \rightarrow \mathbb{N}_0$;
foreach feature point $\vec{s} \in S$ **do**
 /* Compute index \vec{u} of voting table bin */
 $\vec{u} := h(\mathcal{R}(\vec{p})^{-1} \vec{s})$;
 foreach vector of pose-change $\vec{d} \in D$ **do** */
 /* Increase the accumulator using the voting table */
 $b(\vec{d}) \mapsto b(\vec{d}) + H_M(\vec{u}, \vec{d})$;
 end
end
/* Find accumulator bin with maximum value */
 $\hat{\vec{d}} = \underset{\vec{d} \in D}{\operatorname{argmax}} (b(\vec{d}))$;
if $b(\hat{\vec{d}})$ is bigger than a certain threshold **then**
 /* $\vec{p} = \vec{p} + \hat{\vec{d}}$ is the suggested pose of the object */
else */
 /* Report failure */
end

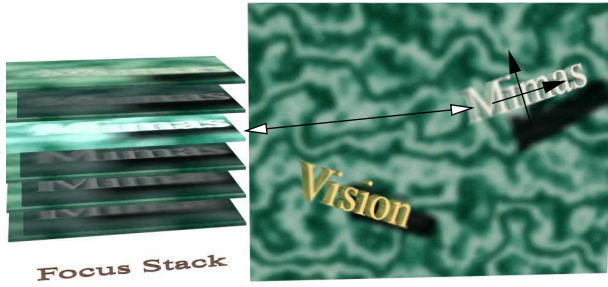


Figure 3: Geometric Hashing with four degrees-of-freedom

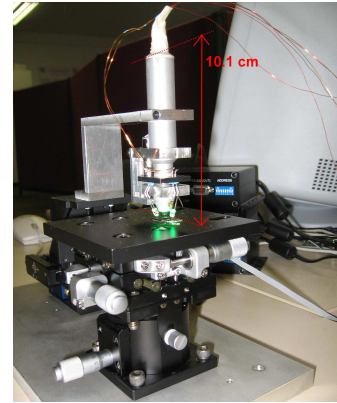


Figure 4: Test environment

3 Results

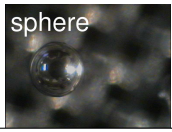
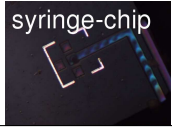
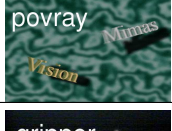


In order to observe the tools and micro-objects, a custom built micro-camera was developed and mounted on a motorised stage (see fig. 4). The micro-camera has an integrated lens system and a built-in focus drive that allows the lens position to be adjusted. The field of view is similar to that obtained from a microscope with low magnification (about $0.8 \text{ mm} \times 0.5 \text{ mm}$ field of view).

The test environment (see fig. 4) allows the user to displace a micro-object using the manual translation stage. The task of the vision-system is to keep the micro-object in the centre of the image and in focus using the motorised stage.

Figure 5 shows a list of results acquired on a 64-bit AMD processor with 2.2 GHz. The initialisation time for the voting-tables has not been included as they are computed offline. First recognition using geometric hashing was run on 1000 frames. The *recognition rate* indicates the percentage of frames, when the object was recognised successfully. In a second test tracking was applied to 1000 frames. The last column in the table shows the corresponding improved frame-rate. In both tests the graphical visualisation was disabled (which saves 0.013 seconds per frame). To require less memory for V_M and H_M , recognition and tracking are performed on down-sampled images. The disadvantage is that the resulting pose-estimate for the micro objects is coarser.

The recognition rate can be increased at the expense of allowing more processing time. However in reality a low recognition rate is much more tolerable than a low frame-rate. Furthermore recognition is only required for initial pose-estimates, when new objects are entering the scene. As the tracking-

Figure 5: Results for object recognition with Geometric Hashing in a variety of environments

video	resolution (down-sampled)	time per frame (recognition)	stack size	degrees-of-freedom	recognition-rate	time per frame (tracking)
 sphere	384×288	0.20 s	7	(x, y, z)	88%	0.020 s
 syringe-chip	160×120	0.042 s	10	(x, y, z, θ)	87%	0.016 s
 povray	384×288	0.27 s	16	(x, y, z, θ)	88%	0.025 s
 gripper	384×288	0.072 s	14	(x, y, z, θ)	88%	0.018 s
 gripper & capacitor	192×144	0.32 s	9 1	(x, y, z, θ) (x, y, θ)	35% 45%	0.022 s
dry run (load frames only)	384×288	0.0081 s	-	-	-	-

rate (the complement of the recognition-rate) always is near 100%, a low recognition rate does not necessarily affect the overall performance of the system.

The recognition rates are particularly low when the object is small, when the object has few features, when there is too much clutter in the scene image, or when multiple objects are present in the scene. The reason is that the final feature (or feature-tuple), which leads to a successful recognition of the object, needs to reside on the object. Furthermore both features of a feature-tuple need to reside on the same object. If all corresponding features to the features of the model M are present in the scene S , the probability of randomly selecting a suitable sequence of features is $(|M|/|S|)^n$.

The focus stack must not be self similar. For example the depth of the micro-capacitor in fig. 7 cannot be estimated independently because a planar object which is aligned with the focused plane will have the same appearance regardless whether it is moving upwards or downwards.

The grippers displayed in fig. 6 and 7 show a rough surface due to the etching step in the gripper’s manufacturing process. From a manufacturing point of view it would be desirable to smooth out this “unwanted” texture. This surface texture however led to the best of all results because it is rich with features.

As both recognition and tracking are purely combinatorial approaches, the memory requirements for the algorithms are high. In the case of the video showing the micro-gripper and the micro-capacitor, 130 MByte of memory was required for the tracking- and 90 MByte for the recognition-algorithm. State-of-the-art algorithms like RANSAC (see [Fischler and Bolles, 1981]) use local feature context so that less features are required. RANSAC in combination with Linear Model Hashing also scales better with number of objects[Shan et al., 2004].

In Geometric Hashing, it is only feasible to compute V_M from a small subset of $M \times M$. By considering only a part of M , one can reduce the size of H_M in a similar fashion. Experimentally H_M was initialised only from features fulfilling $\|\vec{m}_i\|_{q_3} \geq 1$ without affecting the tracking performance.

4 Conclusion

The presented algorithm was applied successfully in a variety of environments: the micro camera environment as shown in figure 4, reflected light microscope environment, and transmitted light microscope

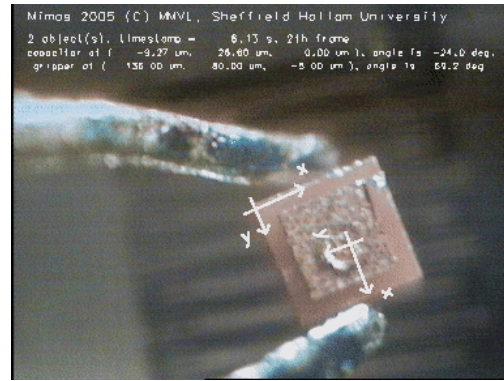
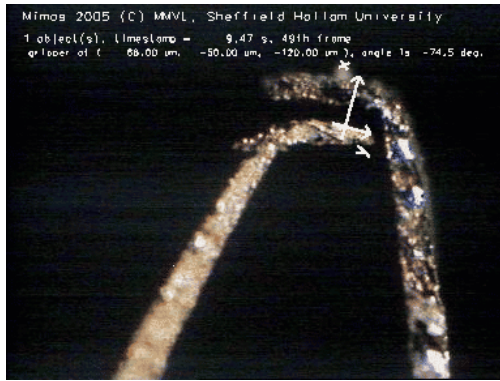


Figure 6: Micro camera image of gripper with superimposed pose estimates (courtesy of SSSA, Sant' Anna) with superimposed pose estimates for gripper and capacitor

environment.

According to [Breguet and Bergander, 2001] the future micro-factory will most probably require automated assembly of micro-parts. The feedback mechanism for the robotic manipulators could be based on computer vision. A robust computer vision system which allows real-time recognition of micro-objects with 4 or more degrees-of-freedom would be desirable.

The algorithm presented in this paper has been implemented using the computer vision library of the *Microsystems & Machine Vision Lab* called **Mimas**, which has been under development and refinement for many years. The library and the original software employed in the MiCRoN-project are available for free at <http://vision.eng.shu.ac.uk/mediawiki/> under the terms of the LGPL.

References

- [Begelman et al., 2004] Begelman, G., Lifshits, M., and Rivlin, E. (2004). Map-based microscope positioning. In *BMVC 2004*. Technion, Israel.
- [Blayvas et al., 2003] Blayvas, I., Goldenberg, R., Lifshits, M., Rudzsky, M., and Rivlin, E. (2003). Geometric hashing: Rehashing for bayesian voting. Technical report, Computer Science Department, Technion Israel Institute of Technology.
- [Breguet and Bergander, 2001] Breguet, J. M. and Bergander, A. (2001). Toward the personal factory? In *Microrobotics and Microassembly III, 29-30 Oct. 2001*, Proc. SPIE - Int. Soc. Opt. Eng. (USA), pages 293–303.
- [Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–95.
- [Forsyth and Ponce, 2003] Forsyth, D. A. and Ponce, J. (2003). *Computer Vision: A modern Approach*. Prentice Hall series in artificial intelligence.
- [Greenspan et al., 2004] Greenspan, M., Shang, L., and Jasiobedzki, P. (2004). Efficient tracking with the bounded hough transform. In *CVPR'04: Computer Vision and Pattern Recognition*.
- [Lamdan and Wolfson, 1988] Lamdan, Y. and Wolfson, H. J. (1988). Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision, Dec 5-8 1988*, pages 238–249. Publ by IEEE, New York, NY, USA.
- [MiCRoN consortium, 2006] MiCRoN consortium (2006). Micron public report. Technical report, EU IST-2001-33567. http://wwwipr.ira.uka.de/~sefried/MiCRoN/PublicReport_Final.pdf.
- [Shan et al., 2004] Shan, Y., Matei, B., Sawhney, H. S., Kumar, R., Huber, D., and Hebert, M. (2004). Linear model hashing and batch ransac for rapid and accurate object recognition. volume 2 of *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 121–8.