Real-time 3D Face Recognition using Line Projection and Mesh Sampling

RODRIGUES, Marcos <http://orcid.org/0000-0002-6083-1303> and ROBINSON, Alan

Real-Time 3D Face Recognition using Line Projection and Mesh Sampling

M.A. Rodrigues and A. Robinson
GMPR Geometric Modelling and Pattern Recognition Research Group
Sheffield Hallam University, Sheffield, UK
{*M.Rodrigues, A.Robinson*}*@shu.ac.uk*

***Abstract*—The main contribution of this paper is to present a novel method for automatic 3D face recognition based on sampling a 3D mesh structure in the presence of noise. A structured light method using line projection is employed where a 3D face is reconstructed from a single 2D shot. The process from image acquisition to recognition is described with focus on its real-time operation. Recognition results are presented and it is demonstrated that it can perform recognition in just over one second per subject in continuous operation mode and thus, suitable for real time operation.**

Keywords-3D face recognition; 3D reconstruction; 3D post-processing; sampling-based recognition; real time recognition

I. INTRODUCTION

It is often stated that 3D facial recognition has potential advantages over 2D methods [2, 6] as a number of limitations can be overcome including lighting variations and viewpoint dependency [9, 10]. Moreover, 3D data can provide high accuracy on describing surface features such as cheeks and nose through curvature descriptors [11]. This paper describes a real-time, fully automatic 3D face recognition system: from 2D eye tracking and image capture to 3D reconstruction, to 3D post-processing and recognition. It is noted that recognition is based on geometry alone: once an image is reconstructed in 3D, no texture information is used in the recognition process.

3D facial recognition algorithms based on geometry alone have been described with recognition rates up to 97% [4, 13] for selected models with low levels of noise. Furthermore, for special cases of high-density meshes with minimum noise accuracy of 100% are reported [12]. In those experiments, a person was enrolled once in the database from a standard frontal view and the verification models were reconstructed from different structured light images in which the person could be facing slightly left or right – it seems unlikely that such performance would not be possible using only 2D recognition methods. However it is important to stress the role of relative noisy free data in those experiments.

The availability of 3D models and the format in which they are presented are not convenient for research aiming at fast recognition rates. While the Face Recognition Grand Challenge FRGC [7] has allowed the wider research community to test recognition algorithms from standard 2D and 3D databases, a severe limitation is that it was not designed to cater for real-time requirements. The FRGC database is standardized such that an application can load pre-formatted data for feature extraction and recognition. 3D data were reconstructed from human subjects taken from a frontal, but arbitrary view point and, given that these are large files containing the structure of the vertices in 3D, this rules out the possibility of testing algorithms in a real-time scenario. Therefore, while 3D data were profitably used to test recognition algorithms in the FRGC, the process does not represent a natural way in which 3D facial recognition systems are to be deployed. This paper presents a contribution towards solving real-time issues in 3D face recognition.

The works of [5, 13,14, 16, 17] have described structured light methods for fast 3D reconstruction from line projection. While those or alternative structured light methods (such as fringe processing) or stereo vision can be used, there are prescribed steps that need to be performed in order to achieve a fully automatic 3D face recognition system:

- 2D pre-processing: face and eye tracking; image filtering; image correspondence (stereo) or projection pattern detection (structured light methods)
- 3D post-processing: generation of 3D point cloud and mesh triangulation; noise removal; 3D hole filling; mesh smoothing (optional); mesh subdivision (optional); pose normalization; feature extraction
- Enrolment and recognition: features are enrolled in a database for subsequent identification (one-to-many) or verification (one-to-one) recognition using appropriate algorithms

These steps are described as follows. Section II describes 2D image processing and Section III the required 3D operations. Section IV deals with pose normalization in 3D, Section V describes a sampling method, sections VI and VII present experimental results. Finally conclusions are presented in Section VIII.

II. AUTOMATIC FACE AND EYE TRACKING IN 2D

Our scanner has three major components: a near-infrared (NIR) projector capable of projecting a pattern of sharp lines that remain in focus over a large distance up to 5m. Two CMOS cameras are used one operating in the visible and another in the NIR spectrum. A beam splitter is placed in front of the cameras such that both cameras see the same portion of the world. Face and eye tracking are performed in the visible spectrum and, when the image satisfies given constrains (see below) the NIR projector is switched on and the image is taken with the NIR camera. The NIR image contains the projected

lines forming a pattern of stripes that are then processed to recover the 3D structure of the face.

The Intel's Microcomputer Research Lab has developed a highly optimized computer vision library that is fine-tuned for the underlying processor type. The processor type is automatically detected and the optimized functions can run from 2 to 8 times faster than equivalent optimized C functions (Bradski and Pisarevsky, 2000). Intel libraries come with built-in routines for real-time face detection based on Haar-like features. A great advantage of the Intel libraries is that it is possible to train and use a cascade of boosted classifiers for rapid object detection for any arbitrary object, not only for faces [1, 18].

Since eye detection is not built-in into OpenCV 1.0 (it is now included in OpenCV 2.0), we trained boosted classifiers with examples of left and right eye and negative images. The general problem with such detection techniques is the number of false positives. For instance, on any image there could be various detected faces and some might not be real faces. The same problem happens with eye detection; the routines normally detect more eyes than there are in the scene.

In order to solve this problem a number of constraints are defined: first, there should be only one face detected in the image and the face width must be larger than a certain threshold (300 pixels in our case); second, there should be only one left and only one right eye detected in the image, and these must be within the region of interest set by the face detection; third, the position of the face and eyes must not have moved more than a set threshold since last detection (10 pixels in our case) so to avoid inconsistent shots caused by rapid motion.

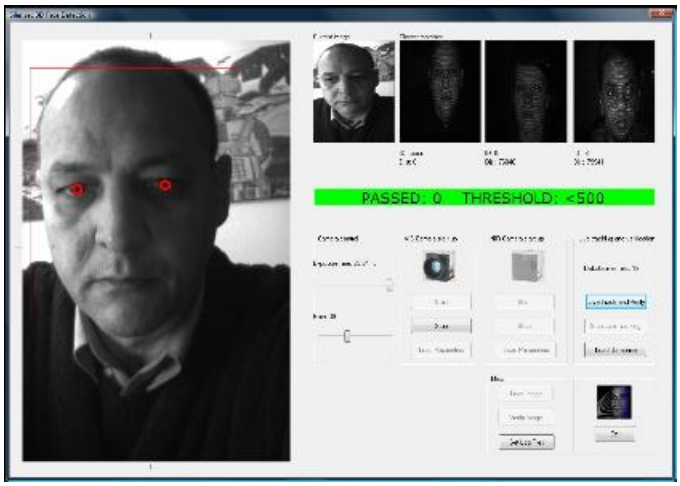

Figure 1. Automatic face and eye tracking

Figure 1 shows the system's interface; it is continuously tracking and detecting (possibly multiple) faces and eyes, but only when the above conditions are satisfied a shot is taken. In this way, the system is apparently idle until someone places their face in front of the camera. We have extensively tested Intel's face detection in connection with our own eye detection and it works remarkably well in real-time. Before a shot is taken a near-infrared line pattern is projected onto the subject. The result is that we now have a structured light 2D image enabling 3D reconstruction and, from eye tracking, we know the position of the eyes in 2D from which we can know their 3D counterparts.

The next step in the process is to apply 2D image filters on the image that contains the stripe patterns (pictured left) namely a median filter followed by a weighted mean filter. This enables the detection of the stripe patterns in the image. Given that we know the geometry of the camera and projector, by knowing the stripe indices we can now fully reconstruct in 3D by trigonometry. Details of the process have been published in [16].

III. AUTOMATIC PRE- AND POST-PROCESSSING IN 3D

3D reconstruction is achieved by mapping the image space to system space (camera + projector) in a Cartesian coordinate system. We have developed a number of successful algorithms to deal with the mapping as described in [5, 16]. Once this mapping is achieved, a 3D point cloud is calculated and the output is triangulated using the connectivity of the vertices as depicted in Figure 2.

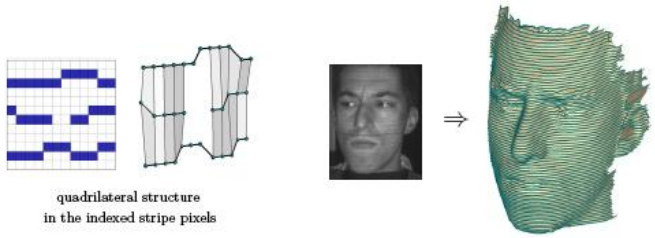

Figure 2. Point cloud and triangulation from the detected stripe pattern in 2D

Once the surface shape has been modeled as a polygonal mesh, a number of 3D post-processing operations are required: hole filling, mesh subdivision, smoothing, and noise removal. There are several techniques that can be used to fill in gaps in the mesh such as the ones discussed in [19, 20, 21]. From the techniques we considered, we tend to focus on three methods namely bilinear interpolation, Laplace, and polynomial interpolation. We found that the most efficient method for real-time operation is bilinear interpolation [12].

The next step is mesh subdivision that, depending on the recognition algorithm to be used may or may not be required. Our research indicates that for the sampling based method described in Section V mesh subdivision is advisable. We use a polynomial interpolation of degree 4 across the stripe patterns and this increases the mesh density while making features more discernible. This is demonstrated in Figure 3, where in the

subdivided mesh (pink) the region around the eyes and lips are better delineated.

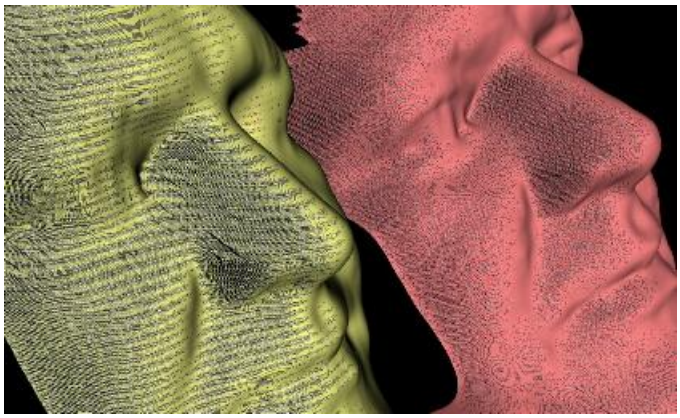

Figure 3. Nonsubdivided mesh (left) and subdivided (right)

We use two smoothing techniques namely moving average and Gaussian smoothing. Moving average is performed through cycling across and then along the stripes. The average of every three points is estimated and the middle point is replaced by the average. In this way the boundary vertices remain anchored. Gaussian smoothing is iteratively estimated by considering 8 neighbours for each vertex. A convolution mask of size 3x3 is applied and the centre vertex is perturbed depending on the values of its neighbours. The difference between each neighbour and the centre vertex is recorded and averaged as ΔV. A multiplier factor is provided (L) which determines how much of this average should be used to correct the value of the centre vertex; i.e., it defines the momentum of the error correction. We use L=0.9 and the number of iterations is set to maximum 35. In each iteration cycle i, the centre vertex V is corrected by $V_i = V_{i-1} + L\Delta V$. The effects of smoothing are depicted in Figure 4. It is clear that Gaussian smoothing (orange model) has considerable limitations regarding the perceived quality of the mesh. The moving average algorithm (cyan) seems to work considerably better. By running a Gaussian followed by a moving average slightly improves the model (magenta) especially around the lip area, which becomes more pronounced.

Figure 4. The effects of smoothing: white, the original model; orange, Gaussian smoothing; cyan, moving average; magenta, Gaussina followedd by moving average

Noise removal is mainly concerned with the region around the eyes, as considerable amount of noise exist due to eyelashes and unwanted reflections. A natural solution would be to replace the vertices in the eye by a spherical surface centred somewhere behind the face model. By experimentation, we chose the centre of the sphere at a position 40mm behind the face model in the same Z-axis as the centre of each eye. An elliptical mask is marked centred on each eye, and all vertices within the elliptical surface have their values replaced by their spherical counterparts. This however, resulted in unnatural looking models. A second solution, which is conceptually simpler, is to punch an elliptical hole centred at each eye and then fill in the holes with bilinear interpolation. This has proved to work well for the face models as shown in Figure 5.

Figure 5: Comparative analysis of noise removal around the eyes. The white model shows the original mesh; red: fitting an elliptical mask around the eyes and filling with a spherical surface centered at 40mm behind the model; yellow: punching an elliptical hole around the eyes; green: filling the elliptical hole with bilinear interpolation.

Figure 6. Triangulated mesh (left), with color and black and white texture mapping (centre) and with original stripe pattern (right)

Finally, texture mapping can be overlaid on the mesh either by using the same striped image or by an identical image in the visible spectrum without stripe information. This is visualized in Figure 6, which shows from left to right the triangulated model (gold), with color and black and white texture, and with

the original projected stripe pattern. This can be useful for instance if one wishes to integrate 2D and 3D face recognition algorithms.

IV. Pose Normalization

Before we can proceed to feature extraction (by mesh sampling) and recognition, all models in the database need to be brought to a standard pose. The standard pose needs to be consistent, i.e., it should be possible to evaluate for all models. We have chosen the pose depicted in Figure 7 below where the origin is placed at the tip of the nose.

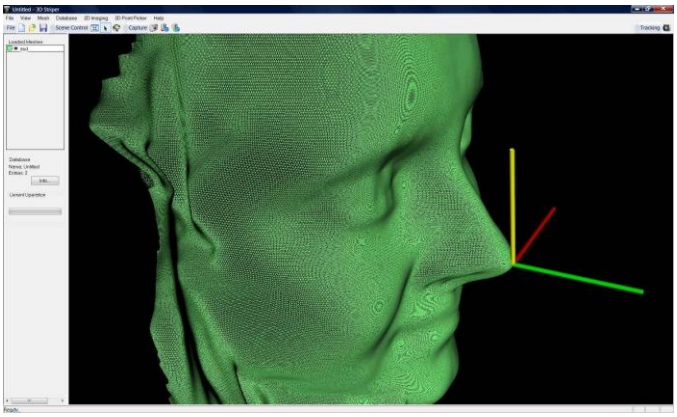

Figure 7. The standard pose with the origin at the tip of the nose

In this pose, the X-axis (red) is aligned with the position of the eyes, the Y-axis (yellow) forms a constant angle of $\pi/10$ with a point located on the front between the two eyes, and the Z-axis points away from the model such that all depths are negative. The algorithm to achieve this standard pose is described as follows (given that we know the position of the eyes (E_1 and E_2) in 3D:

1. Estimate the angle $ß_1$ in the XY-plane between E_1 and E_2
2. Centered on E_1 rotate the mesh around the Z-axis by angle $ß_1$: Rot(z, $ß_1$)
3. Estimate the angle $ß_2$ in the YZ-plane between E_1 and E_2
4. Centered on E_1 rotate the mesh around the Y-axis by angle $ß_2$: Rot(y, $ß_2$)
5. Find the intersection point on the mesh (above the eyes, on the front) of the arcs centered on E_1 and E_2 with radius 0.75 of the inter-ocular distance. Mark this point as F
6. Find the tip of the nose. This is defined as the highest point on the mesh below eye positions within a search threshold of one inter-ocular distance. Mark this point as T
7. Estimate the angle $ß_3$ described by F, T, and the Y-axis
8. Centered on T, rotate the mesh around the X-axis by $(\pi/10 — ß_3)$: Rot(x, $\pi/10 — ß_3$)
9. After this rotation, the highest point on the mesh defining T might have slightly changed. Repeat steps 6, 7 and 8 until $(\pi/10 — ß_3)$ is below a set threshold.

V. Mesh Sampling

Once the model assumes the standard pose, we have developed and tested a recognition algorithm based on extracting a set of features from sampling the mesh. First we define a region on the face from which sampling points are to be taken. The region is cropped in 3D analogous to cropping a 2D image: we define two horizontal planes (top and bottom) and two vertical planes (left and right) setting the limits for cropping the 3D structure. All calculations are performed in 3D and if a point lies within the boundaries of the four cropping planes it is marked as valid, otherwise it is marked as invalid. The result is a cropped face structure, or a face mask from which a fixed number of points are sampled. We chose to sample 900 points (30x30). These points form the feature vector that uniquely characterizes a face and it is used for recognition. Figure 8 below shows the original face structure in blue and the sampled mask in white. Note that the sampled mask has its corner points removed. While this may not be strictly necessary it may increase recognition rates as it has been observed that points in the four corners are more susceptible to noise.

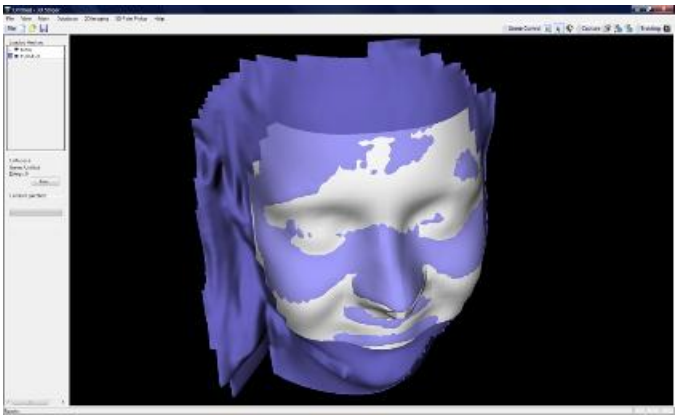

Figure 8. Sampling method: a mask (white) is cut from the model (blue). The mask contains 900 points whose depths are used for recognition

The starting point from cropping and sampling is the position of the two eyes in 3D. This is known from 2D eye tracking, from which corresponding 3D vertices can be determined. The idea is to cut the mesh in 3D half a distance between the eyes to the left and right of each eye, and 1/3 of the distance to the top and 2/3 of the distance to the bottom. Recalling that the mesh has been normalized so the origin is at the tip of the nose and the reference distance is the inter-ocular distance $E_2—E_1$ where (E_1, E_2) are the locations or the two eyes in 3D. A rectangular sampling mask is defined through four cropping planes as follows.

- $\prod_{\text{TOP}}$: a plane parallel to XZ-plane defined at point (0, $1.3(E_2 — E_1)$, 0)
- $\prod_{\text{BOTTOM}}$: a plane parallel to XZ-plane defined at point (0, $-0.66(E_2 — E_1)$, 0)
- $\prod_{\text{LEFT}}$: a plane parallel to YZ-plane defined at point ($-(E_2 — E_1)$, 0, 0)
- $\prod_{\text{RIGHT}}$: a plane parallel to YZ-plane defined at point ($(E_2 — E_1)$, 0, 0)

It has been observed that noisy regions at the boundaries of the cropped face present a serious problem. To illustrate this, consider Figure 9. The green models show both sides of the cropped face. When the shot was taken, the person was looking to their right, assuming a similar pose to the green model on the right. A clear view of that side of the face is available while the other side is partially occluded. Stripes are projected horizontally on the face and because some stripes stretch further than others, it is possible to fill in a number of holes in the occluded area, but this is never a perfect job, as it can be seen on the green model on the left.

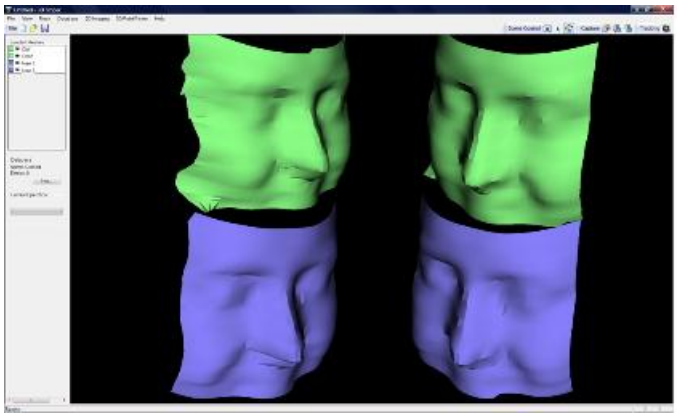

Figure 9. Boundary noise in cropped face models. The noisy side of the geen model (top left) is corrected by point reflection, resulting in the blue model

The solution to this problem is to reflect the good points on one side of the mesh replacing the noisy ones on the other side. Only a few points are required to be reflected and the decision as to how many points and which side of the face needs to be reflected is made depending on whether the mesh has been rotated clockwise or anti-clockwise in the Y-axis (yellow axis in Figure 7) and by how much. For the sampling mask 30x30, these are suggested limits by experimentation for N (the number of reflected points):

- If rotation angle $0.00 < \text{abs}(ß_2) <= 0.10$, N=3
- If rotation angle $0.10 < \text{abs}(ß_2) <= 0.15$, N=6
- If rotation angle $0.15 < \text{abs}(ß_2) <= 0.20$, N=7
- If rotation angle $0.20 < \text{abs}(ß_2)$, N=8

Finally, if $ß_2 > 0$, reflect points from right to left of the YZ-plane, otherwise from left to right. The result of reflecting such points is shown in the blue model on the left of Figure 9: its initial state was the green model above. It clearly becomes a better-defined model.

Since the starting point for cropping the face in 3D is the position of the eyes, it is a natural question to ask to what precision eyes need to be detected. In Figure 10 the white mask was cropped from the eye locations as detected by eye tracking. A difference of 10 pixels in an image of size 1392x1040 was added to each eye location, effectively making the distance between the two eyes 20 pixels wider. A wider eye distance means a larger cropped face in 3D; this is shown in the blue model of Figure 10. However, note that the feature vector is constructed from the depth of the 900 sampled points, so the actual difference in depth between the white and blue cropped faces is negligible (<0.1%) and should not impair recognition as shown in the next section by analyzing any misrecognition.

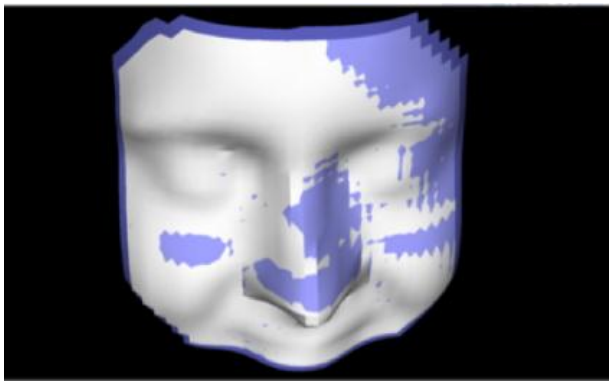

Figure 10. Sensitivity to variations of 20 pixels in eye location

VI. RECOGNITION RESULTS

A version of the eigenface method described in [22] was used for recognition where a face is represented by a vector of size 900x1, that is, the depth of each sampled point. Given that we wanted to test the feasibility of the method for real time operation, we performed identification (one-to-many) and extracted statistics from the log file. Four databases were created with 20, 40, 80, and 200 entries. For enrollment, 2 models of each person were used: one frontal and another looking to the side (either right or left) as in the example shown in Figure 11:

Figure 11. Examples of enrolment shots

If in the example of Figure 12, if the identity of the person were "person 01", the model in the database would be enrolled as `p01s1`, `p01s2`, `p01s3`, `p01s4` (person 01 shot 1, person 01 shot2, person 01 shot 3, person 01 shot 4). It is important to stress that in the tests reported here we have not tried to select noise-free models: all models were selected at random so some models contain high levels of noise. Recognition results are summarized in Table I for the closest match in the database.

In performing identification (one-to-many) we saved to a spreadsheet the results of the 10 closest matches so we can reason abut setting thresholds for verification. Obviously that for different sizes of databases the threshold varies and the correct threshold to minimize FAR can only be found by

experimentation. As an example of determining the correct threshold for verification, Table II shows the sorted results for 3 subjects only where correct identification is colored green. It is clear that some models are more distinctive than others and over the database the top two results point to the right identity. A reasonable threshold for verification for this database would thus be a distance of 72. This is the worse case scenario meaning that for any person performing verification on this database if the distance to the claimed model is 72 or less it would be considered a pass.

TABLE I. RECOGNITION RESULTS (CLOSEST MATCH)

Sampling Mask Method		
Number of Entries in DB	Correctly Identified	Percentage
20	20/20	100
40	40/40	100
80	80/80	100
200	192/200	96

TABLE II. IDENTIFICATION (ONE-TO-MANY)

Real Identity: p01		**Real Identity: p02**		**Real Identity: p03**	
Found in DB	Shortest distances	Found in DB	Shortest distances	Found in DB	Shortest distances
p01s2	43	p02s1	68	p03s2	19
p01s1	53	p02s2	72	p03s1	22
p01s3	69	p01s1	85	p05s1	92
p01s4	71	p01s3	88	p05s2	102
p02s2	108	p04s1	88	p03s3	111
p02s1	121	p01s2	93	p04s2	135
p05s2	139	p04s2	96	p04s1	138
p04s1	147	p01s4	106	p03s4	143
p04s2	155	p05s2	111	p02s1	146
p05s4	155	p05s4	112	p02s2	154

The sampling method presented here is also shown to be robust to variations in expression as illustrated in Figure 12, which shows a typical example of one-to-many identification. The input model (by definition unseen) has its closest match in person 49 shot 3, which is the correct model with a smiley face. The database has 200 entries for 50 distinct subjects and a measure of the distance (74) between input model and closest match is also displayed.

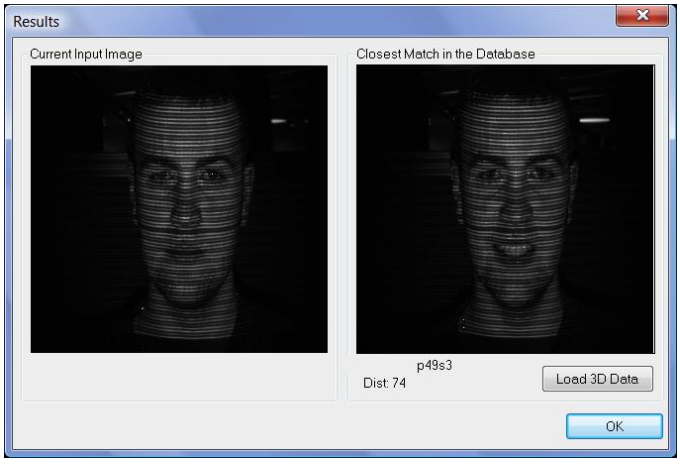

Figure 12. An example of identification (one-to-many)

The next examples depicted in Figures 13 and 14 are on one-to-one verification. We extended the verification procedure to also retrieve the closest match in the database. The threshold for verification was set at 150, based on information from a table similar to the one depicted in Table II. Figure 13 on the left shows the (unseen) input model. The input claims to be the person displayed in the middle (which is enrolled in the database). The claim is accepted as the distance is smaller than the threshold and a green banner shows "PASSED". The third image on the right is the actual closest match in the database when compared to the input model. Note that the two enrolled models (centre and right) do not smile. Figure 14 shows the input model on the left claiming to be the person in the middle. The claim is not accepted as the distance is greater than the threshold and a red banner with the message "FAILED: TOO DISSIMILAR" is displayed. Note that this time the closest match in the database is not the same as above (given small differences in eye detection) but still the correct person.

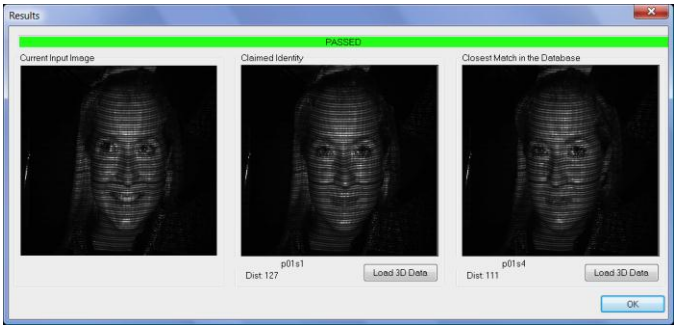

Figure 13. An example of passed verification (one-to-one)

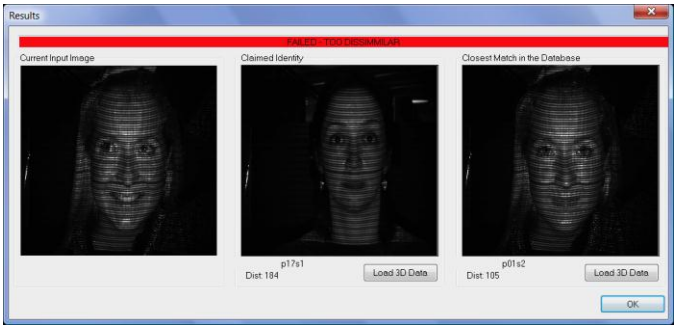

Figure 14. An example of failed verification (one-to-one)

Concerning the misclassifications in Table I, the reasons for these are clear and have to do with camera calibration issues and noise in the eye region. It has been established that, on data collection, the camera focus was changed between enrollment and verification causing the scale of the models to change. Effectively, calibration is thrown out by a change of focus. In addition, noise in the eye region has resulted in inaccurate pose normalization. This is illustrated in Figure 15 in which the green model is the enrolled one and the wire mesh is the verification model. It is clear that a large number of points at the boundaries differ substantially and this resulted in a closest match to a different model in the database. The lessons from these misclassifications are twofold: first, it

is necessary to run the subdivision algorithm, as subdivision tends to reduce noise around the eyes (all models were generated without subdivision). Second, if the camera focus is changed for whatever reason during data collection, the camera needs to be recalibrated before proceeding – this seems obvious and had we spotted these problems before enrolling and verifying, uncalibrated models would have been removed from the database and we would probably have gotten perfect recognition results for these tests.

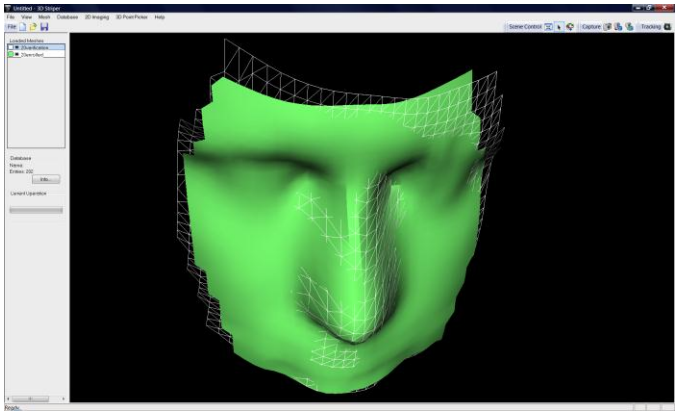

Figure 15. A failed identification due to scale and noise

VII. Real time performance

The algorithms were tested working together from eye tracking to 3D reconstruction and recognition including logging the results with time stamps onto an HTML file. We used a Sony Vaio computer, Intel Core 2 Duo, 2.4GHz, 4GB memory. It has been shown that the methods presented here lend themselves to real-time operation as, from the moment the eye tracking algorithms lock on the eyes, it takes only 1second 200millisencods to perform the following operations:

Take a shot in 2D
Run image filters (median and weighted mean filters)
Detect the stripes in the image
Convert the stripes into a 3D point cloud
Triangulate the point cloud
Perform hole filling on the mesh
Determine the location of eyes in 3D
Punch a hole in the eyes, fill with bilinear interpolation
Find tip of the nose
Normalize pose with origin at the tip of the nose
Determine the cropping planes and crop the mesh
Replace noisy points by reflection
Sample the cropped mesh to 900 points
Search the database for the closest match
Display results of recognition on screen
Save to log file: time stamp, current image, closest match
Continue eye tracking and repeat the sequence

This is indeed a remarkable performance given the high number of operations and complexity of the required functions. Thread functions were implemented where required for improved performance and there is scope for further optimization by expanding thread granularity. In a recent trial at Heathrow Airport (London, UK) the requirements were set at maximum 10seconds per subject for 3D face recognition. Our solution is one order of magnitude better than those requirements and thus, can be realistically applied where high throughput 3D face recognition is desired.

VIII. Conclusion

This paper has presented methods for real-time 3D face recognition from face and eye tracking in 2D to fast 3D reconstruction, to feature extraction by mesh sampling, to identification and verification. Based on geometry alone, the reported recognition accuracy is excellent and there is the potential to achieve 100% recognition for small databases (200 entries was the maximum size used). The misclassified cases were traced back to camera calibration issues.

It is important to stress that the objective of the paper was not to compare the recognition method based on sampling the mesh with other recognition methods reported in the literature that use FRGC data. We argued in the introduction that FRGC data are not suitable for real-time operation because video stream data with face and eye tracking are required to demonstrate real-time capabilities by generating 3D data on the fly. We thus use our own purpose built 3D scanner in conjunction with fast 3D reconstruction algorithms. We have demonstrated in this paper that the process from 2D tracking to 3D recognition takes only 1second 200milliseconds per subject and thus, can be used in a real-time scenario given the speed and accuracy of the 3D recognition.

Future work includes designing and implementing a method for fully automatic camera calibration. We are also pursuing research on mesh description using partial differential equations (PDEs) and developing recognition methods that use PDE information. Furthermore, methods to compress the 3D mesh are required to enable network-based 3D recognition systems. Research on these issues is under way and will be reported in the near future.

References

[1] Adolf, F. (2003). How-to build a cascade of boosted classifiers based on Haar-like features. http://lab.cntl.kyutech.ac.jp/kobalab/nishida/opencv/OpenCV ObjectDetection HowTo.pdf.

[2] Bowyer, K.W., K. Chang, and P. Flynn (2004). A Survey Of Approaches To Three-Dimensional Face Recognition, Int Conf on Pattern Recognition (ICPR), 358–361.

[3] Bradski, G.R and V. Pisarevsky (2000). Intelapos' Computer Vision Library: applications in calibration, stereo segmentation, tracking, gesture, face and object recognition. Computer Vision and Pattern Recognition. Proceedings. IEEE Conference on Volume 2, 796 – 797.

[4] Brink,W. (2008). 3D Face Scanning and Alignment for Biometric Systems, PhD Thesis, Sheffield Hallam University.

[5] Brink, W., A. Robinson, M. Rodrigues (2008). Indexing Uncoded Stripe Patterns in Structured Light Systems by Maximum Spanning Trees, British Machine Vision Conference BMVC 2008, Leeds, UK, 1–4 Sep 2008.

[6] Cook, J., C. McCool, V. Chandran, and S. Sridharan (2006). Combined 2D/3D Face Recognition Using Log-Gabor Templates, Advanced Video and Signal Based Surveillance, IEEE Conference on, pp. 83, 2006 IEEE

Intl Conf on Advanced Video and Signal Based Surveillance (AVSS'06).

[7] FRGC, (2005). The Face Recognition Grand Challenge, http://www.frvt.org/FRGC/

[8] GMPR, 2009. Geometric Modelling and Pattern Recognition Video Demos at http:// www.shu.ac.uk/ research/meri/gmpr/videos.html

[9] Gordon, G. (1992). Face recognition based on depth and curvature features. Computer Vision and Pattern Recognition (CVPR), 108–110.

[10] Medioni, G. and R.Waupotitsch (2003). Face recognition and modeling in 3D. IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG 2003), 232–233.

[11] Hesher, C., A. Srivastava, and G. Erlebacher (2003). A novel technique for face recognition using range images. 7th Int Symposium on Signal Processing and Its Applications).

[12] Rodrigues, M.A, and A.Robinson (2009). Novel Methods for Real-Time 3D Facial Recognition. ATINER 5th Int Conf on Computer Sc and Info Sys, Athens, Greece, 27-30 July 2009.

[13] Rodrigues, M.A. A. Robinson,W. Brink (2008). Fast 3D Reconstruction and Recognition, in New Aspects of Signal Processing, Computational Geometry and Artificial Vision, 8th WSEAS ISCGAV, Rhodes, 2008, p15–21.

[14] Rodrigues, M.A., A. Robinson, W. Brink (2007). 'Issues in Fast 3D Reconstruction from Video Sequences, Lecture Notes in Signal Science, Internet and Education, Proceedings of 7th WSEAS International Conference on MULTIMEDIA, INTERNET and VIDEO TECHNOLOGIES (MIV '07), Beijing, China, September 15-17, 2007, pp 213–218.

[15] Rodrigues, M.A., A. Robinson, L. Alboul, W. Brink (2006). 3D Modelling and Recognition, WSEAS Transactions on Information Science and Applications, Issue 11, Vol 3, 2006, pp 2118–2122.

[16] Robinson, A., L. Alboul, and M. Rodrigues (2004). Methods for indexing stripes in uncoded structured light scanning systems. Journal of WSCG, 12(3) 371–378, February 2004.

[17] Robinson, A., M.A. Rodrigues, L. Alboul (2005). Producing Animations from 3D Face Scans,Game- On 2005, 6th Annual European GAME-ON Conference, De Montfort University, Leicester, UK, Nov 23–25, 2005.

[18] Seo, N. (2009). Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features), http://note.sonots.com/ SciSoftware/haartraining.html

[19] Tekumalla, L.S., and E. Cohen (2004). A hole filling algorithm for triangular meshes. tech. rep. University of Utah, December 2004.

[20] Wang, J. and M. M. Oliveira (2003). A hole filling strategy for reconstruction of smooth surfaces in range images. XVI Brazilian Symposium on Computer Graphics and Image Processing, pages 11–18, October 2003.

[21] Wang, J. and M. M. Oliveira (2007). Filling holes on locally smooth surfaces reconstructed from point clouds. Image and Vision Computing, 25(1):103–113, January 2007.

[22] Turk, M.A. and A. P. Pentland (1991). Face Recognition Uisng Eigenfaces. *Journal of Cognitive Neuroscience* **3** (1): 71–86.