

# PARAMETRIC FACE MODELING AND AFFECT SYNTHESIS

Satya Prakash Mallick and Mohan Trivedi

University of California, San Diego,  
Department of Electrical and Computer Engineering,  
Computer Vision and Robotics Research.

## ABSTRACT

In this paper, we present a fast algorithm for automatically generating a 3D model of a human face and synthesizing affects on it. This work is a step towards low bandwidth virtual Tele-presence using avatars. The idea is to generate a 3D model of a person's face and animate different emotions on the face at a remote observer's end, thus eliminating the need to send raw video. This research is part of a larger project, which deals with development of novel sensory systems and interfaces to enhance safety of a driver. The proposed algorithm uses a stereo pair of images of a driver's face and creates a 3D mesh model by deforming and texture mapping a predefined 3D mesh.

## 1. INTRODUCTION

This research was motivated by the need to convey a vehicle driver's state to a remote observer. The bigger project [1], of which this research is a part, consists of a system for driver affect recognition and display at remote observers end, on a hand held device, like a cell phone. The display could be at various levels of details: 1) A two bit emoticon display 2) an avatar display 3) a raw video display. Bandwidth is a major bottleneck in sending raw video. On the other hand emoticons look unrealistic. Avatars are able to represent different emotions while using extremely low bandwidth. The avatar in our implementation consists of 185 points and one texture, which are sent to the remote observer only once. Facial affects (emotions) can be reconstructed at the remote observer's end. The algorithm is motivated by [2], but we have made several variations to make it fast and more applicable for use in an intelligent car. The algorithm uses two views of a person's face acquired using one of the methods described later.

## 2. FACE MODELING

In this section, we describe our method for generating a 3D model of a person's face, using a pair of face images, in details. A human face lacks texture, and hence it is almost impossible to acquire 3D data of the face using dense correspondence. Hence, we use sparse stereo to reliably recover the depth of a few corner points (point features). These points are used to change the parameters of a predefined face model. Each parameter deforms the predefined mesh in a specific way. For

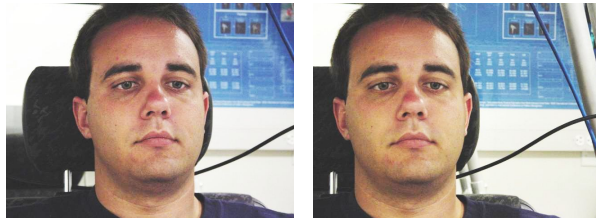


Fig. 1. Sample input images.

example, we have parameters to change the shape of nose, size of forehead, position of eyebrows etc. Greater number of parameters ensures better reconstruction, but at the same time requires more computation time. The number of parameters was deliberately kept low to make the algorithm fast.

### 2.1. Camera calibration and image acquisition

The two input images can be acquired using a stereo rig. Alternatively a single camera can be moved perpendicular to its optical axis so that corresponding points are on the same scanline. A third alternative is to take two images using a stationary single camera, while the person moves his/her head. This is a more general case, because the parallel axis geometry is not maintained, and we need to go through an additional step of estimating the fundamental matrix relating the two views. The internal parameters of the camera(s) are found using MATLAB's "Camera Calibration Toolbox" due to Jean-Yves Bouquet.

### 2.2. Face segmentation using skin tone

To segment out the face from the images, we use a skin tone based approach as suggested in [3]. Their empirical findings show that the variation in skin tone is much less in color than in intensity. Let,

$$r = R/(R + G + B) \quad (1)$$

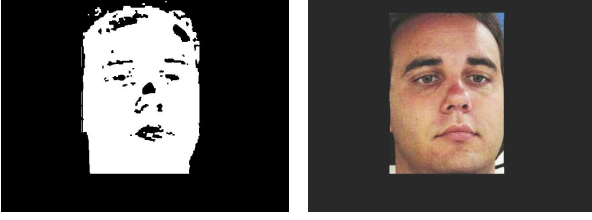
$$g = G/(R + G + B) \quad (2)$$

where, R, G, and B, are the red, green and blue components of color. The skin tones are represented by bivariate gaussian distributions in the chromatic color space represented by (r,g).

A face color distribution can be represented by a Gaussian model  $N(m, \Sigma^2)$  where,  $m = (\bar{r}, \bar{g})$  and

$$\bar{r} = \frac{1}{N} \sum_{i=1}^N r_i \quad \bar{g} = \frac{1}{N} \sum_{i=1}^N g_i \quad (3)$$

This research is sponsored by UC Discovery Grants and the Daimler Chrysler Corporation. The authors would also like to thank their colleagues in the CVRR Laboratory for valuable assistance and cooperation.



**Fig. 2.** The left image shows the skin-mask and the right image shows the bounding box obtained using the skin-mask.

$$\Sigma = \begin{bmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{bmatrix} \quad (4)$$

Using a threshold on the probability, we can separate the images into skin and non-skin regions. We call the binary image thus formed the “skin-mask”. The biggest connected blob in the skin-mask is the face. Smaller connected components are obviously noise, and can be removed by first adding the mask values along the rows and then along the columns followed by a threshold. Thus, we get a bounding box in which the face lies. Refer to fig. 2 for results.

### 2.3. Corner detection and matching

Corner detection and matching is the most crucial and error prone step of the algorithm. We use Harris corner detection algorithm [4] to detect corners in the two images.

We detect and match the corners only in regions which are white pixels in the skin-mask image. We use two corner matching algorithms to match corners. The first algorithm is based on Singular Value Decomposition (SVD) due to Pilu [5] and the second one is a variation of the corner detection by Zhang et al [6] based on relaxation. The result of corner matching is shown in fig. 3.

The SVD based method builds on Scott and Longuet-Higgins algorithm. The *proximity matrix* is changed to take into account the similarity between images patches around the corners. Let  $I$  and  $J$  be two images with corner points  $I_i (i = 1 \dots m)$  and  $J_j (j = 1 \dots n)$ . The SVD based algorithm is described below:

1. Calculate the *proximity matrix*  $G$ :

$$G_{ij} = \frac{(C_{ij} + 1)}{2} e^{-r_{ij}^2 / 2\sigma^2} \quad (5)$$

where,

$r_{ij} = \|I_i - J_j\|$  = The euclidian distance between the corners.

$C_{ij}$  = Normalized cross-correlation between the  $W \times W$  patches around the two corners.

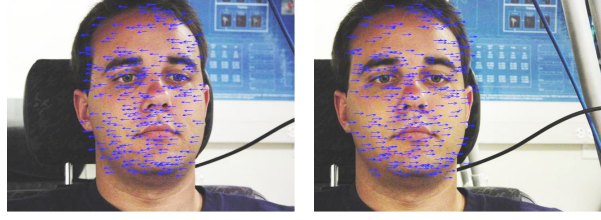
$\sigma^2$  = Variance.

2. Perform the SVD of  $G$ :

$$G = TDU^T$$

3. Convert  $D$  to a new matrix  $E$  by converting all diagonal elements of  $D$  to 1. Calculate a new matrix  $P$  such that:

$$P = TEU^T$$



**Fig. 3.** The start of the arrows represent the position of the corners in the image and the end of the arrows show the position of the same corner in the other image. The arrows look parallel indicating good matches.

Corner  $I_i$  matches corner  $J_j$  if  $P(i, j)$  is maximum in its row as well as its column.

We do not describe the relaxation based method proposed by Zhang [6] because of space constraints. A brief description of our version of the algorithm is given below:

1. **Select candidate matches:** For each corner in one image, select a set of candidate matches in the other image, based on the normalized cross correlation between a  $W \times W$  image patches centered at the two points.
2. **Calculate strength of candidate matches:** Calculate the strength of matches based on the premise that if  $(I_i, J_j)$  is a good match, then, in a small neighborhood  $\mathcal{N}$  of  $I_i$ , there would be other good matches  $(I_k, J_l)$ , such that,  $I_k$  would be transformed to  $J_l$  in a similar way as  $I_i$  is transformed to  $J_j$ .
3. **Disambiguate matches:** We use the “winner-take-all” strategy to disambiguate matches instead of a more elegant technique proposed by Zhang [6] to make the algorithm fast.  $(I_i, J_j)$  is considered to be a valid match if among all candidate matches of  $I_i$ , it has the greatest strength with  $J_j$  and vice-versa.

### 2.4. Extraction of depth information

If parallel axis geometry is used, then the extraction of depth is simple. Let  $(x_l, y)$  be the position of a corner in the first image and  $(x_r, y)$  be the position of the corresponding corner in the second image. Let  $f$  be the focal length(s) of the camera(s) and  $B$  be the baseline. Then, the depth of the point in 3D corresponding to the corner is given by:

$$Z = \frac{fB}{x_l - x_r}, \quad X_l = \frac{Zx_l}{f}, \quad Y_l = \frac{Zy}{f}$$

However, if we do not use parallel axis geometry, we need to estimate the depth using epipolar geometry. The fundamental matrix relating the two views is calculated using the eight-point algorithm [7]. The estimated fundamental matrix is used to recover the rotation ( $R$ ) and translation ( $T$ ) relating the two camera positions. Once  $R$  and  $T$  are calculated, the depths can be recovered using the following equations:

$$Z_r = R_3^T (p_l - \hat{T}) \quad (6)$$

$$Z_l = f_l \frac{(f_r R_1 - x_r R_3)^T \hat{T}}{(f_r R_1 - x_r R_3)^T p_l} \quad (7)$$

where,

$$\hat{T} = \frac{T}{\|T\|}, \quad R = \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix}, \quad p_l = \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix}$$

$f_l$  and  $f_r$  are the focal lengths of the left and right cameras.  $(x_l, y_l, 1)$  is the homogeneous coordinate of a corner in the left image.

## 2.5. Coordinate transformation

We are dealing with two coordinate systems: 1) the coordinate system in which the predefined mesh resides and 2) the coordinate system in which the reconstructed corners reside. The number of reconstructed corners is in the order of 400-500 points and the number of points on the predefined mesh is 185. The idea is to select 185 points out of the 400-500 corner points which closely match the 185 points of the predefined mesh. However, in order to be able to define closeness of points in one mesh to points in another mesh, they must be in the same coordinate system. We use the algorithm given in [8] to find the rotation( $R$ ), translation ( $T$ ) and scale( $s$ ) relating the coordinate systems. Five corresponding points on the face are hand-clicked in the two images and the depths of the five points are recovered using the calibration information of the cameras. The five points are: the two ends of the lips, the tip of the nose, and the inner ends of the eye. The coordinates of these five points in the predefined mesh are known. Hence, we can recover the rotation ( $R$ ), translation ( $T$ ), and scale ( $s$ ) relating the two coordinate systems. The scale recovered using this method is not reliable. Using the  $R$  and  $T$  values, the coordinates of the predefined mesh are projected onto the image plane of the left camera. The amount of scaling required for the projected mesh to touch the bounding box containing the segmented face gives us the scale ( $s$ )

## 2.6. Solving 3D correspondence

The  $R$ ,  $T$  and  $s$  values calculated above are used to transform the 3D coordinates of the corners to the coordinates of the predefined mesh. We call this set of transformed corner points  $P$ . The predefined mesh is made of 185 points. We call this set of points  $P_{mesh}$ . For each point in  $P_{mesh}$ , we find a point in  $P$ , which corresponds most closely with it, using Hungarian linear partitioning algorithm. We call this set of points  $N_{mesh}$ .  $N_{mesh}$  is a mesh that is obtained using actual data and  $P_{mesh}$  is the predefined mesh. The next step is to change the parameters of  $P_{mesh}$  so that it matches  $N_{mesh}$ .

## 2.7. Parameter Estimation

The predefined mesh can be deformed by changing a few parameters of  $P_{mesh}$ . Each parameter defines a particular attribute of the face; like the shape and size of the nose, placement of the eyebrows, size of forehead etc. Each parameter has an upper bound and a lower bound. The constraint ensures that we do not overfit the noisy  $N_{mesh}$ . Let,  $\hat{N}_{mesh}$  is the estimate of  $N_{mesh}$  by deformation of  $P_{mesh}$ . Therefore,

$$\hat{N}_{mesh} = P_{mesh} + \sum_{i=1}^n \alpha_i d_i \quad -1 \leq \alpha \leq 1 \quad (8)$$



**Fig. 4.** The left image shows the predefined mesh and the right image shows the deformed mesh.

where,

$\alpha_i$  = degree of deformation of the  $i^{th}$  parameter.

$d_i$  = maximum deformation of the  $i^{th}$  parameter.

Hence, the cost function that we need to minimize is given by:

$$C = \|\hat{N}_{mesh} - N_{mesh}\|^2 = \sum_{i=1}^n C_i \quad (9)$$

Cost function  $C$  can be broken down into smaller independent cost functions  $C_i$ 's, each of which is much easier to solve. We could do this because the parameters are independent of each other. For example, the size of the nose can be thought to be independent of the shape of the forehead. This also reduces the complexity of the problem. We use least squares to solve for  $\alpha_i$ s which reduce the cost functions  $C_i$ s. In the actual implementation, we use the following formulation for least squares. All  $\alpha_i$ s are estimated separately, independent of each other. Let,

$n_{mesh}^i$  = Set of all points in  $N_{mesh}$ , which are affected by the deformation  $d_i$ .

$p_{mesh}^i$  = Set of all points in  $P_{mesh}$ , which are to be deformed by the deformation  $d_i$ .

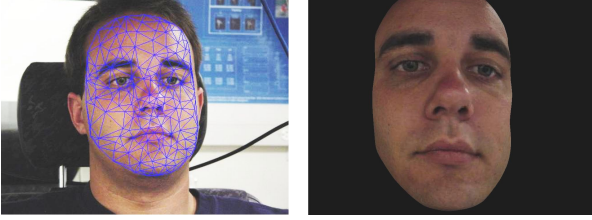
For example, if  $d_i$  is the maximum deformation of the nose a human being can have( Note that all  $d_i$ s are empirically predefined values). Then,  $n_{mesh}^i$  is the set of all points in  $N_{mesh}$  which are part of the nose and  $p_{mesh}^i$  is the set of all points in  $P_{mesh}$  which are part of the nose. Then we can rewrite equation (8) as:

$$\begin{aligned} n_{mesh}^i &= p_{mesh}^i + \alpha_i d_i \\ \Rightarrow n_{mesh}^i - p_{mesh}^i &= \alpha_i d_i \end{aligned}$$

This becomes a simple least squares problem to solve. Any estimate of  $\alpha_i > 1$  is clamped to 1 and any estimate of  $\alpha_i < -1$  is clamped to -1. An example of mesh deformation is shown in fig. 4.

## 2.8. Texture mapping

We use a single image to texture map the deformed mesh  $\hat{N}_{mesh}$ .  $\hat{N}_{mesh}$  is projected onto the left image and the texture enclosed within a triangle formed by the vertices of the projected mesh is mapped onto the corresponding 3D triangle in  $\hat{N}_{mesh}$ . The simple texture mapping technique described above results in very realistic 3D models. View dependent texture maps can be generated using many texture images of the persons face [2][9]. However, we feel that our simple method is computationally less expensive and the compromise in quality is not very discernible.



**Fig. 5.** The left image shows the back projection used for texture mapping, and the right image shows the 3D model of the face.



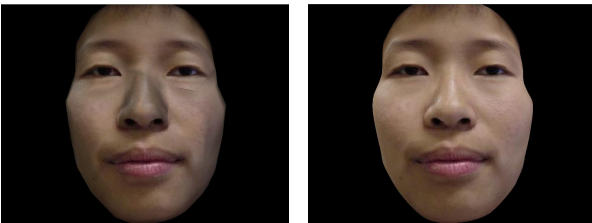
**Fig. 6.** Results shown people with different race, gender and head shapes.

### 3. AFFECT SYNTHESIS

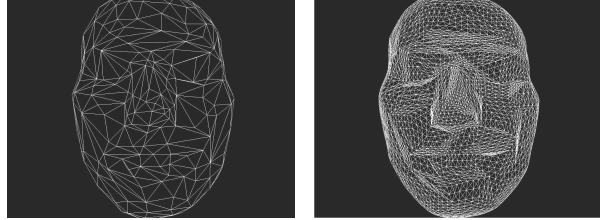
Emotions can be synthesized by a non-linear transformation of the vertices of the mesh. In our implementation, we first generated six meshes corresponding to the six emotional states, by manually moving the vertices of the predefined mesh. We calculate the percentage deformation of each vertex of the predefined mesh required to generate a particular emotion. The same emotion can be generated on the new mesh by moving each vertex of the mesh so as to bring about the same percentage deformation. Intermediate emotions can be generated by interpolating the coordinates of each vertex between two emotional states.

#### 3.1. Smoothing using splines

In both face modeling and animation we are deforming a smooth mesh. The resulting deformation may sometimes result in a faceted appearance which can look quite unrealistic. To overcome this problem, we use splines to smoothen the deformed mesh. In our implementation, we are using the in-built spline based smoothing in the VRML browser called “Cortona” by Parallel Graphics. The results are shown in fig. 7. There is a noticeable improvement at the nose, the cheekbones and the region between the eyebrows.



**Fig. 7.** The left image shows the results obtained before smoothing and the right image shows the results after smoothing using splines.



**Fig. 8.** The left image shows the deformed mesh and the right image shows the dense mesh obtained using splines.



**Fig. 9.** Results of affect synthesis are shown in the left image. The emotions shown are “Neutral”, “Happy”, “Sad”, “Fear”, “Disgust” and “Anger”. Results of generation of intermediate emotions using interpolation are shown in the right image.

### 4. CONCLUSION

In this paper we have presented a very fast and efficient algorithm for automatic 3D face modeling and animation. Our main goals were to achieve high speed, realistic 3D modeling of faces and extremely compact representation of various emotional states using the 3D face model. We have tried the face-modeling algorithm on people of different gender, race, color and face shapes with reasonable success.

### 5. REFERENCES

- [1] Joel McCall, Satya P. Mallick, and Mohan Trivedi, “Real-time driver affect analysis and tele-viewing system,” *IEEE Intelligent Vehicle Symposium*, June 2003.
- [2] Zhengyou Zhang, Zicheng Liu, Dennis Adler, Michael F. Cohen, Erik Hanson, and Ying Shan, “Robust and rapid generation of animated faces from video images,” *MSR-TR-2001-101*, 2001.
- [3] Yang, W. Lu, and A. Waibel., “Skin-color modeling and adaptation.” *ACCV*, vol. 2, pp. 687–694, 1998.
- [4] C. Harris and M. Stephens, “A combined corner and edge detector,” *Proc. 4th of Alvey Vision Conference*, pp. 189–192, 1988.
- [5] M. Pilu, “A direct method for stereo correspondence based on singular value decomposition,” *International Conference of Computer Vision and Pattern Recognition*, 1997.
- [6] Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong., “A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry,” *Artificial Intelligence Journal*, vol. 78, pp. 87–119, October 1995.
- [7] R. I. Hartley, “In defence of the 8-point algorithm,” *5th International Conference on Computer Vision*, pp. 1064–1070, June 1995.
- [8] B. Horn, “Closed-form solution of absolute orientation using unit quaternion,” *Journal of Optical Society*, pp. 4(4),629–642, April 1987.
- [9] Frederic Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David Salesin., “Synthesizing realistic facial expressions from photographs.” *SIGGRAPH*, 1988.