GEOMETRY-ASSISTED STATISTICAL MODELING FOR FACE MOSAICING

Xiaoming Liu and Tsuhan Chen

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, U.S.A.
xiaoming@andrew.cmu.edu tsuhan@cmu.edu

ABSTRACT

The modeling of facial appearance has many applications. This paper proposes an approach to generating a statistical face model based on video mosaicing. Unlike traditional video mosaicing, we use the geometry of a face to improve the mosaicing result. Given a face sequence, each frame is unwrapped onto certain portion of the surface of a sphere, as determined by spherical projection and the minimization procedure using the Levenberg-Marquardt algorithm. A statistical model containing a mean image and a number of eigenimages, instead of only one image template, is used to represent the face mosaic. Good experimental results have been observed.

1. INTRODUCTION

Human face modeling is a hard problem due to large variations in human faces, such as expression, pose, illumination, etc., that need to be modeled. Because face modeling has many applications in face recognition, detection, tracking, animation and content-based video retrieval, many researchers have proposed different approaches to dealing with this problem.

Statistical modeling is one type of approach to face modeling. Principal Component Analysis (PCA) [1], and Linear Discriminant Analysis (LDA) [1] are common methods in this category. While these modeling methods perform reasonable well for certain variations, such as face expression, they might not be able to provide a good model when there is too much variations in face images, for example, an image sequence showing different poses of human faces. Researchers have noticed this problem and proposed different ways to overcome it. In [2], face shape is tracked and normalized to the frontal position before performing recognition. In order to model large expression variations, Liu et al. [3] proposed to calculate the pixel motion between images first, followed by applying PCA to motion vectors, which provides better recognition performance than applying PCA directly to the pixel intensity. Chowdhury et al. [4] utilized a generic model to construct a 3D face model from a video sequence.

As motivated by the research on video mosaics [5] and fingerprint mosaicing [6], we propose to model the face appearance by constructing a mosaic from a video sequence of the face at various poses. Traditionally, the pose variation is very difficult to model. We propose to use the geometry of the face to improve the mosaicing result. By modeling the human head with a sphere, in our method each face image is treated as the projection of the sphere's certain portion on the image plane. As shown in Figure 1, given a face sequence $\mathbf{F}_1, \mathbf{F}_2, \Lambda\ \mathbf{F}_n$, which is cropped from the whole video frame by the face tracker [7], we can "unwrap" each face image onto the surface of the sphere, which has a α and β coordinate system, by using the geometric matching algorithm. In the meantime, a statistical model composed of a mean image and a number of eigenimages, is trained by using the "unwrapped" images. Our statistical model is essentially based on the basic idea of PCA, which also includes a mean and a few eigenimages.

In literary, there are a few papers proposed for face mosaicing [8][9]. Compared to them, our method has two novelties. One is that instead of using the cylindrical projection, we use the spherical projection, which works better with the head motion in both horizontal and vertical directions. The other is that while traditional mosaicing algorithms usually only result in one template image, our method generates one statistical model with both the mean image and eigenimages. Usually one template image can take care of the pose variation by patching different images together. And the eigenimages can be used to model other types of variations, such as illumination, expression and the variation due to the mismatch between the true 3D face geometrical model and the spherical model. Thus our method can provide a much rich statistical model comparing to using only one template image.

The paper is organized as follows. In Section 2, we introduce the proposed geometric matching algorithm. Section 3 introduces the missing data modeling algorithm. In Section 4, we show the modeling result by using our algorithm. The conclusion and future work are presented in Section 5.

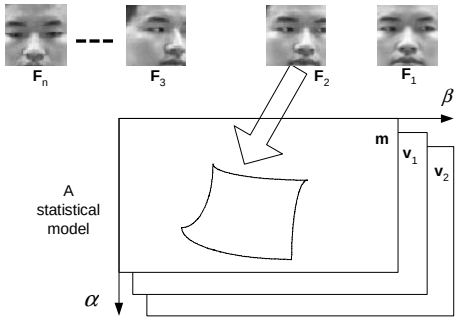

Figure 1 System overview

2. GEOMETRIC MATCHING ALGORITHM

In this section, we introduce how to "unwrap" one face image onto certain portion of the sphere. The spherical projection and the estimation of matching parameters using the Levenberg-Marquardt algorithm [10] are presented.

2.1. Spherical projection

As shown in Figure 2, there are three coordinate systems: the 2D image plane space QUV, the 3D space $OXYZ$, and the sphere's

surface space $O\alpha\beta$. Spherical projection describes the correspondences between the image plane space QUV and the surface space $O\alpha\beta$.

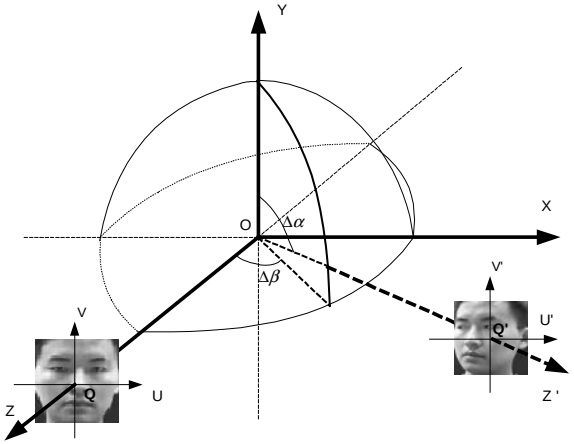

Figure 2 Spherical projection for faces.

One assumption is made for our algorithm. That is, the input face sequences are captured under the orthographic camera model, which is a reasonable assumption in the case where the camera is far from the subject. Also we constrain that the first face image in a sequence is frontal view. We have the following relation between one point on the image plane, $[u,v]^T$, which is also in the image $\mathbf{F}$, and one point in the surface space, $\mathbf{p}=[\mathbf{p}_x,\mathbf{p}_y,\mathbf{p}_z]^T$, which is in the unwrapped image $\mathbf{S}$.

$$\begin{aligned}\mathbf{p}_x &= r\sin(\alpha)\sin(\beta)\\ \mathbf{p}_y &= r\cos(\alpha)\\ \mathbf{p}_z &= r\sin(\alpha)\cos(\beta)\end{aligned} \quad (1)$$

$$\begin{aligned}u &= \mathbf{p}_x\\ v &= \mathbf{p}_y\end{aligned} \quad (2)$$

After the first frame is projected back into the $O\alpha\beta$ space by using the inverse formulations of (1) and (2), it occupies the center region of the $O\alpha\beta$ space.

Now starting from the second image in the sequence, we need to find its corresponding position in the $O\alpha\beta$ space, which is parameterized by $[\Delta\alpha,\Delta\beta]^T$ in our algorithm. That is, if we rotate the YZ plane by $-\Delta\alpha$ degree and the XZ plane by $-\Delta\beta$ degree, on the Z' axis there will be a new image plane space $Q'U'V'$ capturing the side view of human faces. These rotations can be formulated as:

$$\mathbf{p}' = \mathbf{R}_y(-\Delta\beta)\mathbf{R}_x(-\Delta\alpha)\mathbf{p} \quad (3)$$

$$\mathbf{R}_x(\Delta\alpha)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\Delta\alpha) & -\sin(\Delta\alpha)\\ 0 & \sin(\Delta\alpha) & \cos(\Delta\alpha)\end{bmatrix}$$

$$\mathbf{R}_y(\Delta\beta)=\begin{bmatrix}\cos(\Delta\beta) & 0 & \sin(\Delta\beta)\\ 0 & 1 & 0\\ -\sin(\Delta\beta) & 0 & \cos(\Delta\beta)\end{bmatrix}$$

where $\mathbf{R}_x(\Delta\alpha)$ and $\mathbf{R}_y(\Delta\beta)$ are the rotation matrices with respect to the X and Y axis respectively. Now the spherical projection is defined as given the original image $\mathbf{F}$ and $[\Delta\alpha,\Delta\beta]^T$, to determine the unwrapped image $\mathbf{S}'$. To accomplish this, for each pixel $\mathbf{p}'$ inside $\mathbf{S}'$, we can find its correspondence in the original image $\mathbf{F}$ by inverting (3):

$$\mathbf{p} = \mathbf{R}_x(\Delta\alpha)\mathbf{R}_y(\Delta\beta)\mathbf{p}' \quad (4)$$

Then bilinear interpolation will be used to calculate the pixel intensity at $\mathbf{p}$, which is filled in as the pixel intensity of $\mathbf{p}'$.

In practice, the spherical projection might not be computational efficient because each pixel in the unwrapped image needs to find its corresponding in the original image. To solve this problem, we approximated the mapping function with a triangular mesh. That is, the unwrapped image is represented as a set of triangles; for the vertexes of these triangles, we derive their corresponding positions in the input image. For the pixels inside each triangle, affine transformation is applied using the scan-line algorithm. The goal of this approximation is to speed up the spherical projection while not affecting the modeling, which will be discussed further in Section 4.

2.2. Estimation of matching parameters

Essentially the geometric matching algorithm is a minimization procedure. The objective is to minimize the difference between the unwrapped image $\mathbf{S}'(\alpha',\beta')$ and the statistical model $\Pi=\{\mathbf{m},\mathbf{V}\}$, which consists of the mean $\mathbf{m}$ and k eigenvectors $\mathbf{V}=\{\mathbf{v}_1,\mathbf{v}_2,\Lambda,\mathbf{v}_k\}$:

$$J = \left\| vec(\mathbf{W}) \circ (vec(\mathbf{S}') - \mathbf{m} - \mathbf{V}\mathbf{c}) \right\|^2 = \sum_{\alpha'=0,\beta'=-180}^{\alpha'=180,\beta'=179} e^2 \quad (5)$$

$$\mathbf{c} = (\mathbf{V}^T diag(\mathbf{W}^2)\mathbf{V})^{-1}(diag(\mathbf{W}^2)\mathbf{V})^T(vec(\mathbf{S}') - \mathbf{m}) \quad (6)$$

where $\circ$ refers to the element-wise multiplication, $diag()$ generates a matrix whose diagonal element is the input vector, $vec()$ is to take the vector representation of a matrix by scanning it line by line, and $\mathbf{c}$ is the eigen-coefficient of $\mathbf{S}'$ with respect to the mosiacing model. $\mathbf{W}$ is the weighting map for $\mathbf{S}'$, which combines the weighting information from two sources. One is the weighting map for the original input image $\mathbf{F}$, $\mathbf{W}_f(u,v)$. It describes the reliability of each pixel in the input image, and thus specifies the local weighting information. The other, $\mathbf{W}_g(\alpha',\beta')$, is to model the global weighting information for the whole $O\alpha\beta$ space. For both of them, we assign them as a gaussian weighting map with higher weights in the center and lower weights toward the boundary. The combined weighting map, $\mathbf{W}$, takes both the local and global information into consideration.

We adopt the Levenberg-Marquardt algorithm to find $[\Delta\alpha,\Delta\beta]^T$ that minimizes (5). This algorithm requires the computation of the partial derivatives of e with respect to the unknown parameters $[\Delta\alpha,\Delta\beta]^T$, for example:

$$\frac{\partial e}{\partial \Delta\alpha} = \frac{\partial e}{\partial \mathbf{S}'}\left(\frac{\partial \mathbf{S}'}{\partial \alpha'}\left(\frac{\partial \alpha'}{\partial \mathbf{p}_x'}\frac{\partial \mathbf{p}_x'}{\partial \Delta\alpha} + \frac{\partial \alpha'}{\partial \mathbf{p}_y'}\frac{\partial \mathbf{p}_y'}{\partial \Delta\alpha}\right)\right)$$

$$+\frac{\partial \mathbf{S}'}{\partial \beta'}\left(\frac{\partial \beta'}{\partial \mathbf{p}_x'}\frac{\partial \mathbf{p}_x'}{\partial \Delta\alpha}+\frac{\partial \beta'}{\partial \mathbf{p}_y'}\frac{\partial \mathbf{p}_y'}{\partial \Delta\alpha}\right)) \quad (7)$$

where $\frac{\partial \mathbf{S}'}{\partial \alpha'}$ and $\frac{\partial \mathbf{S}'}{\partial \beta'}$ are the image intensity gradients of $\mathbf{S}'$ at (α', β'). With these partial derivatives, we can calculate an approximate Hessian matrix **A** and the weighted gradient vector **b** [10]. Then the parameters can be updated by

$$\left[\hat{\Delta\alpha}, \hat{\Delta\beta}\right]^T = (\mathbf{A}+\lambda\mathbf{I})^{-1}\mathbf{b} \quad (8)$$

This algorithm consists of the following steps:

1. Assign the initial value for $[\Delta\alpha, \Delta\beta]^T$.
2. Compute $\mathbf{S}'$ and **W.**
3. Compute the error e as in (5) and the intensity gradient on $\mathbf{S}'$, computer the partial derivative of e with respect to $[\Delta\alpha, \Delta\beta]^T$, and compute **A** and **b**.
4. Linearly update $[\Delta\alpha, \Delta\beta]^T$ by $\left[\hat{\Delta\alpha}, \hat{\Delta\beta}\right]^T$ calculated in (8).
5. Evaluate (5) using the updated parameters and check whether the error J decreases; if not, increase λ as described in [10], and compute a new $\left[\hat{\Delta\alpha}, \hat{\Delta\beta}\right]^T$.
6. Continue the iteration until the parameters converge or a fixed number of steps have been finished.

When we obtain $[\Delta\alpha, \Delta\beta]^T$ for the input face image, we can unwrap it to generate $\mathbf{S}'$ using (4). In the meaning time, the weighting map **W** for $\mathbf{S}'$ is also calculated. Given $\mathbf{S}'$ and **W**, we can use them to refine the existing statistical model Π by retraining. As we know, for different face poses, different weighting map will be generated. Since we treat the pixels with zero weights as missing data, we will introduce how to retrain the model using missing data modeling techniques in the next section.

3. MISSING DATA MODELING

Many researchers have proposed different methods for missing data modeling [11][12]. As motivated by the research in error concealment for video sequences [13], we propose the following algorithm to train a statistical model for data with missing pixels.

First, given the unwrapped image $\mathbf{S}_1', \mathbf{S}_2', \Lambda\, \mathbf{S}_n'$ and their corresponding weighting map $\mathbf{W}_1, \mathbf{W}_2, \Lambda\, \mathbf{W}_n$, we compute the mean by

$$\mathbf{m} = \left(\frac{1}{\sum_{i=1}^{n} vec(\mathbf{W}_i)}\right) o \sum_{i=1}^{n} (vec(\mathbf{W}_i)\, o\, vec(\mathbf{S}_i'))$$

where division refers to the element-wise operation. Second, for each pixel of $\mathbf{S}_i'$ $(1 \le i \le n)$, let its intensity equal to the corresponding one in the mean if its weight is zero. So the idea is to assume the missing pixels to be the mean. Finally we apply the traditional PCA algorithm to obtain k eigenimages, $\mathbf{v}_1, \mathbf{v}_2, \Lambda\, , \mathbf{v}_k$.

4. EXPERIMENTAL RESULTS

In the experiments, we collect a face sequence with the subject showing different poses while fixing the camera. This sequence has 58 frames in total; each has the size of 44 by 44 pixels. We also know the height and the width of the head are around 60 by 60 pixels.

If we apply the traditional PCA algorithm to model the variations in this sequence, we obtain the mean image and the first two eigenimages in Figure 3. As we can see, all three images look burred and messy. However, by taking this sequence as input to our algorithm, we are able to generate the face mosaic model as shown in Figure 4. Two eigenimages are used in training our statistical model. The input sequence, the unwrapped image, the mean and two eigenimages are shown in each column. We evenly choose eight time instants among the whole sequence and show them on eight rows. The resulting statistical model with the mean and eigenimages shows identity information about this subject. Since our geometric matching algorithm works well, there is no obvious blurring artifacts in the resulting mosaic.

As we expected, the spherical projection is slow and the registration of one input image takes 6 seconds on PC. We also implemented the fast projection with a triangular mesh, which reduces the time to be less than 1 second and generate almost the same modeling results as the conventional projection.

Figure 3 The result of applying PCA on the testing sequence (mean, 1st eigenimage, 2nd eigenimage)

5. CONCLUSIONS

This paper proposes an approach to generating a statistical face model based on video mosaicing. Unlike traditional video mosaicing, we use the geometry of a face to improve the mosaicing result. Given a face sequence, each frame is unwrapped onto certain portion of a sphere's surface, which is determined by the spherical projection and the minimization procedure with the Levenberg-Marquardt algorithm. A statistical model containing the mean image and a number of eigenimages, instead of only one image template, is used to represent the face mosaic. Good experimental results have been observed on face sequences by using our algorithm.

We constrain the input sequence to our algorithm only includes the face potion. However, it could be removed by adding the position and size of human head as the unknown parameters. Thus using the same framework, given one video frame, we need to solve 5 parameters (2 for rotation, 2 for position and 1 for size) instead of 2 parameters. One large benefit with this extension is that we can do face tracking and modeling simultaneously. Another future direction is that, in Section 3, the missing data modeling algorithm retrains the model whenever there is a new input image, which makes the training process becoming slower as the input sequence getting

longer. Thus we need an incremental algorithm to update the existing model. This makes all the updating procedures finish in the same "unit" time. Also, given one face image, we can perform face recognition by measuring the difference between the unwrapped image and the mosaicing model.

6. ACKNOWLEDGEMENT

This work is supported in part by Under Secretary of Defence for Acquisition, Technology and Logistrics (USD(AT&T)) and Combating Terrorism Technology Support Office, Technical Support Working Group (TSWG).

7. REFERENCES

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Second edition. John Wiley & Sons. Inc., New York, 2001.

[2] G.J.Edwards, C.J.Taylor, T.F.Cootes, "Learning to Identify and Track Faces in Image Sequences", Int. Conf. on Face and Gesture Recognition 1998, pp. 260-265, 1998.

[3] X. Liu, T. Chen, B.V.K. Vijaya Kumar, "Face Authentication for Multiple Subjects Using Eigenflow", *Pattern Recognition*, Volume 36, Issue 2, February 2003, pp. 313-328.

[4] A. Roy Chowdhury, R. Chellappa, R. Krishnamurthy and T.Vo, " 3D Face Recostruction from Video Using A Generic Model ", Int. Conf. on Multimedia and Expo, 2002.

[5] Szeliski R., "Video Mosaics for Virtual Environments", *IEEE Computer Graphics and Applications*, Vol.16, No.2, pp. 22-30, March 1996.

[6] A. K. Jain and A. Ross, "Fingerprint Mosaicking", *Proc. of ICASSP*, Orlando, Florida, May 13-17, 2002.

[7] F. J. Huang and T. Chen, Tracking of Multiple Faces for Human-Computer Interfaces and Virtual Environments. IEEE Intl. Conf. on Multimedia and Expo, New York, July 2000.

[8] Chen D., State A. and Banks D. Interactive shape metamorphosis. In 1995 symposium on interactive 3D graphics, ACM SIGGRAPH, pp. 43-44, 1995.

[9] Lee, Y., Terzopoulos, D., and Waters, K. Realistic Modeling for Facial Animation, in SIGGRAPH 95 conference proceedings, ACM SIGGRAPH, pp. 55-62, 1995.

[10] W.H. Press et al., Numerical Recipes in C: The Art of Scientific Computing, 2nd edition, Cambridge Univeristy Press, Cambridge, England, 1992.

[11] De la Torre, F. and Black, M. J., Robust principal component analysis for computer vision, Int. Conf. on Computer Vision, Vancouver, 2001.

[12] Danijel Skocaj, Weighted Incremental Subspace Learning. The proceeding of European workshop on Cognitive Vision, Zürich, Switzerland, 19-20 September 2002.

[13] T. P.-C. Chen and T. Chen, "Second-Generation Error Concealment for Video Transport over Error Prone Channels", Wireless Communications and Mobile Computing, Oct. 2002.

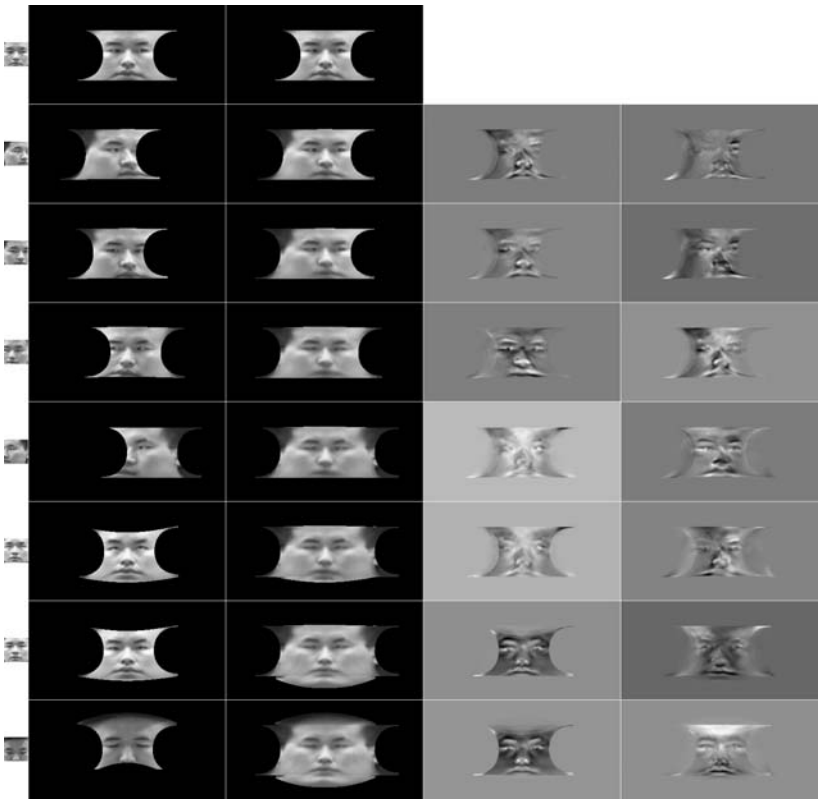

Figure 4. Experimental result (column 1:testing sequences, column 2: unwrapped image, column 3: mean, column 4: 1st eigenimage, column 5: 2nd eigenimage)