Increasing the Density of Active Appearance Models*

Krishnan Ramnath
ObjectVideo, Inc.

Simon Baker
Microsoft Research

Iain Matthews
Weta Digital Ltd.

Deva Ramanan
UC Irvine

Abstract

Active Appearance Models (AAMs) typically only use 50-100 mesh vertices because they are usually constructed from a set of training images with the vertices hand-labeled on them. In this paper, we propose an algorithm to increase the density of an AAM. Our algorithm operates by iteratively building the AAM, refitting the AAM to the training data, and refining the AAM. We compare our algorithm with the state of the art in optical flow algorithms and find it to be significantly more accurate. We also show that dense AAMs can be fit more robustly than sparse ones. Finally, we show how our algorithm can be used to construct AAMs automatically, starting with a single affine model that is subsequently refined to model non-planarity and non-rigidity.

1. Introduction

Active Appearance Models (AAMs) [8] are deformable models of the human face. AAMs have been used successfully in a wide variety of applications from head pose estimation, face recogntion, and expression recognition [16], to lip-reading [19] and gaze estimation [14].

The triangulated mesh in an AAM typically only has around 50-100 vertices because AAMs are usually constructed from a collection of training images with the AAM mesh *hand-labeled* [8] on them. Besides the fact that hand-labeling is laborious, it is very difficult to accurately label corresponding points in largely textureless regions such as the cheeks. In practice, AAM meshes are best defined on the few facial landmarks that are easy to locate.

In this paper we propose an algorithm to increase the density of an AAM, initially assuming a sparse AAM has been constructed in the usual manner [8]. On the highest level, our algorithm iterates 3 steps: (1) building the AAM, (2) refitting the AAM to the training images thereby updating the training meshes, and (3) refining the AAM.

The third of these steps is divided into three subparts. (3a) Consists of adding more vertices to increase the mesh density. (3b) Refines the mesh connectivity using image-consistent triangulation [20]. Even when combined, however, (3a) and (3b) are insufficient to build a more accurate AAM. The piecewise affine warps in an AAM can be thought of as modeling 3D planes in the world. Since neither steps (3a) nor (3b) change the locations of the original sparse mesh in the training data, the implicit 3D shape and motion of these vertices is not changed. The main insight in our algorithm is to augment (3a) and (3b) with step (3c) which refines the AAM shape modes and the corresponding mesh vertex locations in the training data to minimize the model reconstruction error in a similar manner to [3]. This optimization allows the implicit 3D model to change shape and move differently; e.g. the cheeks to protrude, etc.

As the AAM is refined, our algorithm builds an increasingly accurate model of appearance variation effects such as illumination variation and the appearance and disappearance of facial structures such as eyes and teeth. In steps (2)

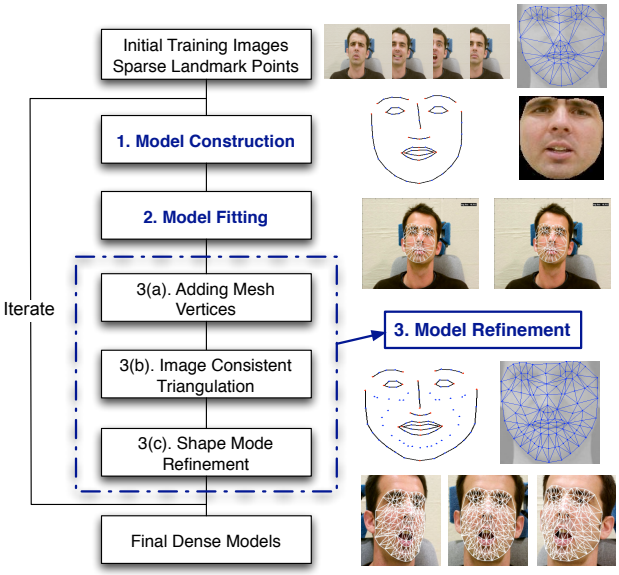


Figure 1. An overview of our algorithm. The algorithm is initialized using a set of sparse hand-labeled mesh points. The algorithm then iteratively: (1) builds an AAM, (2) refits it to the training images, and (3) refines the AAM. AAM refinement is split into three steps: (3a) the mesh is subdivided to add vertices, (3b) the mesh connectivity is refined using image-consistent re-triangulation [20], and (3c) the AAM shape modes and corresponding mesh vertex locations are optimized to minimize the model reconstruction error.

*Research conducted at Carnegie Mellon University.

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

and (3c), the appearance variation model is used in a similar way that models of appearance variation have been used in face tracking algorithms [5, 10, 9]. The key difference in our algorithm is that the model of appearance variation is repeatedly updated as the model is refined.

An additional benefit of our algorithm is that it can help decide how dense an AAM needs to be. The algorithm can be terminated when the reduction in reconstruction error is too small to merit the introduction of more vertices.

We first evaluate our algorithm quantitatively with a set of ground-truth data using a form of hidden markers. In the computation of dense 3D Morphable Models [6], optical flow is used to compute dense correspondence. We therefore compare with a number of optical flow algorithms, including the overall best performer [7] in a recent evaluation of optical flow algorithms [4]. We find that none of the optical flow algorithms yields any noticable improvement in the accuracy of hidden marker correspondence prediction. On the other hand, our densification produces a significant improvement. We also perform comparisons to show improvement in fitting robustness and present a number of tracking results to qualitatively illustrate our algorithm.

Finally, we show how our algorithm can be used to construct AAMs automatically, starting with a rigid planar model of the face that is subsequently refined to model the non-planarity and the non-rigid components. Our results are a significant improvement over previous unsupervised AAM construction algorithms such as [3].

2. Background: AAMs

2.1. Construction

Active Appearance Models (AAMs) are usually constructed from a set of training images with the AAM mesh vertices hand-labeled on them [8]. The training mesh vertices are first aligned with Procrustes and then Principal Component Analysis (PCA) is used to build a 2D linear model of shape variation [8]. The 2D shape $\mathbf{s} = (x_1, y_1, \ldots, x_M, y_M)^{\mathrm{T}}$ can be represented as a base shape $\mathbf{s}_0$ plus a linear combination of m shape vectors $\mathbf{s}_j$:

$$\mathbf{s} \;=\; \mathbf{s}_0 + \sum_{j=1}^{m} p_j \, \mathbf{s}_j \tag{1}$$

where the coefficients p_j are the shape parameters. The AAM model of appearance variation is obtained by first warping all the training images onto the mean shape and then applying Principal Component Analysis on the shape normalized appearance images [8]. The appearance of an AAM $A(\mathbf{u})$ can be represented as a mean appearance $A_0(\mathbf{u})$ plus a linear combination of l appearance vectors $A_j(\mathbf{u})$:

$$A(\mathbf{u}) \;=\; A_0(\mathbf{u}) + \sum_{j=1}^{l} \lambda_j \, A_j(\mathbf{u}) \tag{2}$$

where the coefficients λ_j are the appearance parameters and $\mathbf{u}$ runs over all pixels in the base mesh. For notational convenience we denote the set of pixels inside the mesh $\mathbf{s}_0$.

An AAM model instance with shape parameters p_j and appearance parameters λ_j can then be generated by warping the appearance $A(\mathbf{u})$ from the base mesh $\mathbf{s}_0$ to the model shape mesh $\mathbf{s}$. In particular, we define a piecewise affine warp from $\mathbf{s}_0$ to $\mathbf{s}$ and denote it as $\mathbf{W}(\mathbf{u};\mathbf{p})$. Following [18], we also incorporate the 2D similarity transformation that is used to normalize the shape [8] into $\mathbf{W}(\mathbf{u};\mathbf{p})$.

2.2. Fitting

Fitting an AAM to an input image I can be posed [18] as minimizing the following non-linear criterion:

$$\sum_{\mathbf{u}\in\mathbf{s}_0} \left[A_0(\mathbf{u}) + \sum_{j=1}^{l} \lambda_j A_j(\mathbf{u}) - I(\mathbf{W}(\mathbf{u};\mathbf{p})) \right]^2 \tag{3}$$

with respect to the 2D shape $\mathbf{p}$ and appearance λ_j parameters. In [18] it was shown that the optimization in Equation (3) can be solved at over 200 frames per second using the inverse compositional algorithm [2].

3. AAM Densification

We now describe our algorithm to increase the density of an AAM. See Figure 1 for a flow diagram. In the outer loop, our algorithm iterates three steps: (1) Model Construction (see Section 2.1), (2) Model Fitting (see Section 2.2) and (3) Model Refinement (see below.) We refine the model in three ways: (3a) the mesh is subdivided to add vertices. (3b) the mesh connectivity is refined using image-consistent re-triangulation [20], and (3c) the AAM shape modes and corresponding mesh vertex locations are optimized to minimize the model reconstruction error. We now describe each of these steps in turn.

(3a) Adding Mesh Vertices

The first step in the iterative refinement process is to add more mesh vertices. There are a number of ways to choose a mesh triangle and the location within the triangle to add the points. We adopt a simple but effective way to ensure that we end up with similarly sized triangles. At each iteration, we choose the mesh triangle with the longest edge. A new point is then added on the mid-point of this edge. Choosing the longest edge in this way avoids the creation of "long thin" triangles. Figure 2 illustrates the addition of two points to the mesh. We chose to maintain symmetry and add a pair of points simultaneously to both halves of the face mesh at each step. Maintaining symmetry is optional but we found it reduced the tendency to overfit the data and also results is more visually appealing models.

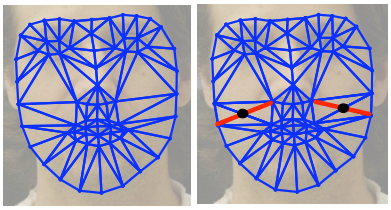


Figure 2. A pair of images showing the mesh before and after two new mesh points have been added to the longest edges. The newly added points and edges are highlighted. Note that adding a new vertex causes two adjacent triangles to split.

One extension of this algorithm might be to explore other heuristics to choose where to add the new points such as choosing the triangle with the largest average coding error, and trying to place points on structural discontinuities. However, it should be noted that as the mesh gets more and more dense, the choice of a specific heuristic becomes less important as there are vertices close to any point on the face.

(3b) Image-Consistent Re-Triangulation

The next step is to refine the mesh connectivity using image-consistent re-triangulation[20]. We look at each pair of adjacent triangles in turn and flip the common edge (i.e. replace the edge with the other diagonal edge in the quadrilateral defined by the two trianlges.) We look at the RMS model reconstruction error:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{\mathbf{u}\in\mathbf{s}_0}\left[A_0(\mathbf{u})+\sum_{j=1}^{l}\lambda_j^i A_j(\mathbf{u})-I^i(\mathbf{W}(\mathbf{u};\mathbf{p}))\right]\right]^2} \tag{4}$$

across the training data to determine whether the flip resulted in a reduction in reconstruction error or not. If the flip did result in a reduction in reconstruction error, we perform the flip on the mesh and proceed to the next pair of triangles. The current appearance model for estimating the reconstruction error is obtained by fitting the AAM to the image data. In Figure 3 we show the mesh before and after performing image-consistent re-triangulation. Note that again we enforce symmetry, for the same reasons as above.

(3c) Shape Mode Refinement

Even when combined, steps (3a) and (3b) are insufficient to build a significantly more accurate AAM. The piecewise affine warps in an AAM can be thought of as modeling 3D planes in the world. Since neither steps (3a) nor (3b) change the locations of the original sparse mesh in the training data, the implicit 3D shape and motion of these vertices is not changed. In order to allow the implicit 3D shape of the AAM to change and new shape deformations to be possible, the shape modes must be refined.

In step (2) of our algorithm, the model is refit to the training data, but the motion of the mesh vertices is limited to the shape subspace of the face model. To allow the shape modes to change when the model is rebuilt the next time step (1) is applied, we must repeat the refitting in step (2) but allow the mesh vertices to move outside the current shape subspace. We can then possibly learn new deformations that better explain the training imagery. In particular, we perform a model fit similar to the one described in Section 2.2 except that we replace the shape modes with identity bases that span the entire 2D space:

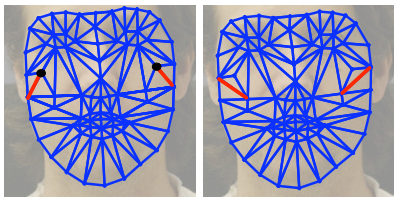


Figure 3. A pair of images showing the mesh before and after performing an image-consistent re-triangulation of the mesh [20]. The symmetric pair of edges that were flipped are highlighted.

$$[\mathbf{s}_1 \dots \mathbf{s}_{2M}] = \begin{pmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \end{pmatrix}_{2M\times 2M} \tag{5}$$

where M is the number of mesh vertices. This optimization is very high dimensional and ambiguities may appear in regions with insufficient texture (a form of generalized aperture problem.) In order to avoid this problem, we regularize this optimization with two priors. The first is a smoothness constraint on newly added vertices. The second is a constraint that the initial sparse vertices cannot move too far from the input (hand-marked) locations (to avoid the model collapsing to a single pixel in each training image.)

The smoothness prior encourages newly added points to be not too far away from their initial position relative to the vertices of the triangle that they were added to. Denote the vertices of a triangle which has had a vertex added to it by $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, and denote the added vertex $\mathbf{v}_4$. The smoothness prior is given by:

$$\|\mathbf{v}_1 + \lambda\,(\mathbf{v}_2 - \mathbf{v}_1) + \mu\,(\mathbf{v}_3 - \mathbf{v}_1) - \mathbf{v}_4\|^2 \tag{6}$$

summed over all newly added mesh vertices $\mathbf{v}_4$. The λ and μ coefficients are the barycentric coordinates of the added vertex $\mathbf{v}_4$ with respect to the triangle $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$.

The second prior penalizes points that move too much from their initial hand specified locations:

$$\|\mathbf{s}_0 + \sum_{j=1}^{m} p_j\,\mathbf{s}_j - \hat{\mathbf{s}}\|^2 \tag{7}$$

where the mean shape $\mathbf{s}_0$, 2D shape parameters $\mathbf{p}$ and the eigenvectors $\mathbf{s}_j$ are all defined over the number of initial vertices and $\hat{\mathbf{s}}$ are the initial hand-labeled vertex locations.

The final optimization criterion is a combination of the terms in Equations (3), (6) and (7) and is given by:

$$\sum_{i=1}^{N}\sum_{u\in s_0}\left[I^i\left(\mathbf{W}\left(\mathbf{u};\mathbf{p}^i\right)\right)-\left(A_0\left(\mathbf{u}\right)+\sum_{j=1}^{l}\lambda_j^i A_j\left(\mathbf{u}\right)\right)\right]^2$$

$$+K_1\cdot\sum_{i=1}^{N}\|\mathbf{s}_0+\sum_{j=1}^{m}p_j^i\,\mathbf{s}_j-\hat{\mathbf{s}}^i\|^2+$$

$$K_2\cdot\sum_{i=1}^{N}\sum_{j=1}^{M2}\|\mathbf{v}_1^{ij}+\lambda^j\left(\mathbf{v}_2^{ij}-\mathbf{v}_1^{ij}\right)+\mu^j\left(\mathbf{v}_3^{ij}-\mathbf{v}_1^{ij}\right)-\mathbf{v}_4^i\|^2 \tag{8}$$

where there are $i = 1,\ldots,N$ training images I^i, $j = 1,\ldots,M2$ new vertices have been added, λ^j, μ^j are the Barycentric coordinates of the j^{th} new vertex in the mean shape, $\mathbf{v}_4^{ij}$ is the location of the j^{th} new vertex in image i, and $(\mathbf{v}_1^{ij}, \mathbf{v}_2^{ij}, \mathbf{v}_3^{ij})$ are the vertices of the triangle to which the j^{th} new vertex was added in image i. Equation (8) is optimized using the inverse compositional algorithm with priors [1]. The weights K_1 and K_2 are currently chosen by trial and error.

Termination Criterion

An additional benefit of our algorithm is that it can help decide how dense an AAM needs to be to explain all the training data with a certain degree of fidelity. In particular, a natural termination criterion is to threshold the reduction in the image coding error (first term in Equation (3)).

4. Experimental Results

There are two ways to initialize our algorithm. One way is to start with an existing sparse AAM and then increase the mesh density. In Sections 4.1, 4.2 and 4.3 we present results for this case. We can also automatically construct a dense AAM using the output of a blob tracker as initialization. In Section 4.4 we present results using this approach.

4.1. Quantitative Evaluation

In this section we present quantitative comparisons to demonstrate the improved accuracy of our algorithm in building dense models. We collected high resolution data with hidden markers, manually locate the markers, and use the resulting correspondences as ground-truth. See Section 4.1.1. We then evaluate an AAM using the dense correspondence between the training images implied by the locations of the training mesh vertices. We use this correspondence to warp all the hidden markers into the mean shape and then evaluate how consistent they are.

We compare dense models constructed with our algorithm against those estimated using optical flow. Optical flow is used to build dense models in an analogous manner to [6]. For any given AAM density, the optical flow is used to predict the locations of the added mesh vertices in the training images. A dense AAM is then constructed in the usual manner using the extra vertex locations. We compare our algorithm with 4 different optical flow algorithms [12, 17, 21, 7] one of which [7] was the top overall performer in a recent evaluation of flow algorithms [4].

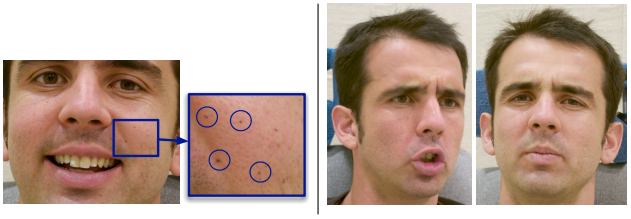


Figure 4. Left: An example of the high resolution images used to generate the ground-truth. The hand-marked ground-truth points on the face are highlighted using dark circles. Right: Two examples of down sampled images. Notice that the ground-truth points are almost invisible in the down sampled images.

4.1.1 Ground-Truth Data Collection

We collected high-resolution face data using a 6 megapixel Canon EOS SLR camera. We obtained facial ground-truth data using a form of hidden markers on the face. See [22, 4] for two different ways of embedding hidden ground-truth. We mark a number of very small black dots on the face with a fine tip marker. We then hand locate the markers in the high resolution images. Figure 4 shows one such high resolution image with ground-truth points marked on it along with a zoomed in version highlighting the ground-truth point locations. The input data to all algorithms consists of all the high resolution images (3072 x 2040) down sampled to one fourth their size (768 x 510.) The ground-truth points are no longer visible in these low resolution images and hence do not influence the working of the algorithms. Two down sampled images are shown in Figure 4.

Note that we use only a single person's ground-truthed data for the quantitative comparisons. The notion of corresponding points is not well defined across different people. We cannot estimate where a point on the face of person A should correspond to a point on the face of person B. Also note that we cannot use range data to help with this process since the important aspect of the ground-truth is the non-rigid mapping from frame to frame.

4.1.2 Optical Flow Computation

Optical flow can be particularly hard when the motion is large. In our case, the head moves around quite a bit in the input image. Our densification algorithm keeps track of where the original head locations were and so implicitly

(a) (b) (c)

Figure 5. (a) An example of the mesh used to warp input images onto the mean shape for computing optical flow. The face mesh is extended to eliminate boundary effects for optical flow algorithms. (b) The original input images to our algorithm. It is difficult for optical flow algorithms to work on these images with varying head locations. (c) The two images from (b) warped onto the mean shape using the mesh from (a). By warping the images onto the mean shape we reduce the search space to 4–5 pixels.

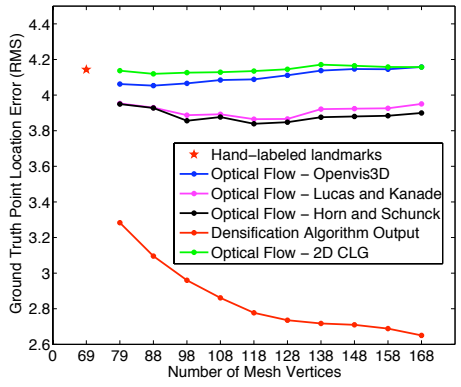
avoids this large search space. We provide equivalent information to the optical flow algorithms by warping all the input images into the coordinate frame of the mean face for the initial sparse model. This means that the maximum flow for all of the images is of the order of 4–5 pixels, well within the search ranges of most optical flow algorithms. Another issue that can cause difficulty for optical flow algorithms is boundary effects. We avoid this by also warping a boundary region around the face. We present examples of the face mesh and the original and warped images in Figure 5. Observe that the warped images are closer to each other and hence makes it easier for the optical flow algorithms.

4.1.3 2D Ground-Truth Points Prediction Results

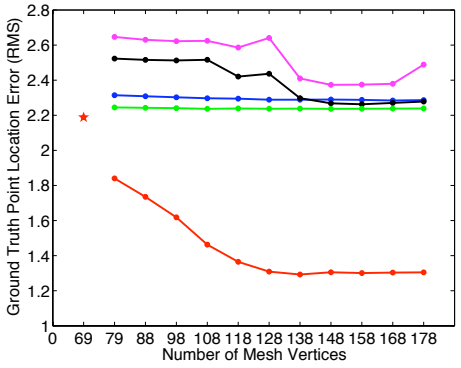
We compare our algorithm with four optical flow algorithms: (1) Horn-Schunck [12], (2) Lucas-Kanade [17], (3) diffused connectivity (Openvis3D) [21], and (4) combined local and global (CLG) [7]. The fourth of these algorithms is the best overall performer in a recent evaluation of optical flow algorithms [4]. We use the authors implementation of (1), the OpenCV implementations [13] of (2) and (3), and Stefan Roth's implementation of (4) [4].

To evaluate the optical flow algorithms, we first compute AAMs using the output of the flow algorithms. For any given size of AAM, we use the optical flow to generate estimates of where the extra vertices are in the training images. We then use the computed AAMs to predict the correspondence between the hidden-marker ground-truth points in the training images by warping each point into the other images using the AAM vertex locations and piecewise affine interpolation. Once we have the predicted locations of the ground-truth points in all images we compute the RMS spatial error between the predicted and the actual ground-truth point locations. We compare the results of the flow algorithms with the corresponding results for the dense AAMs constructed using our algorithm.

We present the results for two different people in Figure 6. We plot the RMS ground-truth prediction error vs the



Person 1



Person 2

Figure 6. A comparison of the algorithms in terms of their ability to predict the correspondence between the hidden markers. On the x-axis we plot the number of mesh vertices in the AAM (each iteration of our algorithm adds 10 mesh points). On the y-axis we plot the RMS ground-truth point prediction error. Our AAM densification algorithm clearly outperforms the flow algorithms.

number of mesh points in the AAM. The results show that our densification algorithm results in an AAM that is better able to predict the ground-truth data the more vertices are used; i.e. the AAM gets more accurate the more mesh vertices are used. On the other hand, the performance of all four of the optical flow algorithms barely improves as the density of the AAM is increased. This means that the locations of the extra vertices in the training data that are predicted using the optical flow is no better than that which would be obtained by interpolating the sparse mesh using the piecewise affine warp; i.e. the optical flow algorithms add little new information over the hand-labeled vertices. Since none of the flow algorithms gives any substantial improvement, their relative performance is somewhat random and essentially depends on the various sources of noise.

4.1.4 3D Ground-Truth Points Prediction Results

We now perform comparisons similar to the ones in the previous section to evaluate the 3D consistency of the corre-

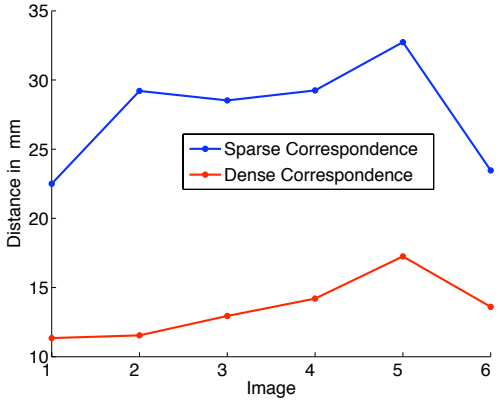


Figure 7. The distance of the triangulated 3D ground truth points from the 3D mesh plane for each 3-frame. The values were computed for six 3-frames. The smallest triangle in which the ground-truth point lies in 2D was computed. The distance was computed between the triangular plane (formed by the 3D mesh vertices) and the corresponding 3D ground truth points. The average distance across images for the sparse correspondence (68 mesh points) is **27.84 mm** whereas for the dense correspondence (168 mesh points) it is **13.625 mm**.

spondence computed by our algorithm with respect to the ground-truth. In this case we evaluate our algorithm on trinocular stereo data. We repeat the experimental setup described in Section 4.1.1 except that now we have a stereo rig with calibrated cameras [15]. We use the initial sparse correspondence (the input to our algorithm) and the dense correspondence from our algorithm and triangulate them to obtain 3D point locations. We also triangulate the 2D ground-truth points to obtain 3D ground-truth points.

We compare the 3D fidelity of the sparse and the dense correspondences by computing the distance of each 3D ground-truth point from the corresponding triangular plane comprised of the sparse and dense mesh vertices. We find that the ground-truth points are closer (in the depth direction) to the dense triangular mesh planes than the sparse ones, indicating that our densification algorithm generated mesh vertices with higher 3D fidelity. We include the results of our 3D quantitative comparison in Figure 7.

4.2. Fitting Robustness

In Figure 8 we show quantitative results to demonstrate the increased robustness of our dense AAMs. In experiments similar to those in [18], we generated 1800 test cases (20 trials each for 90 images) by randomly perturbing the 2D shape model from a ground-truth obtained by tracking the face in video sequences and allowing the algorithms to converge. The 2D shape and similarity parameters obtained from the dense AAM tracks were perturbed and the perturbations were projected on to the ground-truth tracks of the sparse AAMs. This ensures that the initial perturbation is

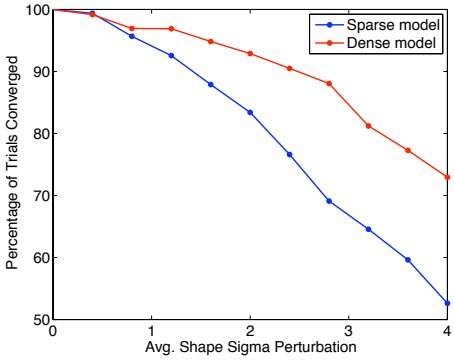


Figure 8. Fitting robustness results comparing the sparse AAM (68 vertices) and the dense AAM (168 vertices). An increase in frequency of convergence is obtained using the dense AAM.

a valid starting point for all algorithms. We then run each algorithm (one using the dense AAM and the other with the sparse AAM) from the same perturbed starting point and determine their convergence by computing the RMS error between the mesh location of the fit and the ground-truth mesh coordinates. The algorithm is considered to have converged if the RMS spatial error is less than 2.0 pixels. The magnitude of the perturbation is chosen to vary on average from 0 to 4 times the 2D shape standard deviation. The perturbation results were obtained on the trinocular stereo data (Section 4.1.1) for each of the three camera views and the average frequency of convergence is reported in Figure 8.

The results show that the dense AAM converges to ground truth more often than the sparse AAM. The increased robustness of the dense AAM may be surprising given its apparent increased flexibility. But note that both the sparse and dense AAMs have the same number of shape modes. The increased robustness of the dense AAM is because it is a better (more compact) coding of the training face images, and has a refined appearance model. Also note that since both the sparse and the dense AAMs have the same number of parameters that are optimized during the fit, the dense AAM fitting is as fast as the sparse AAM fitting. The additional overheads such as in computing the affine warp for composition hardly affect the speed of fitting which is dominated by dot-producting the error image with a steepest descent image for each shape parameter [18].

4.3. Face Tracking

In Section 4.1 we used single-person data to allow a quantitative comparison with ground-truth. Our algorithm can of course be applied to data of any number of people. In this section we present a qualitative evaluation of the tracking ability of the dense AAM constructed using our algorithm. We collect tracking data of five different subjects using a video camera and use our algorithm to compute a

Figure 9. Although to obtain quantitative results in Figure 6 we had to build AAMs for a single person, our algorithm can naturally be applied to data of multiple people. It is just not possible to generalize the procedure in Figure 6 to multiple people. Here we show a few frames of a dense multi-person AAM being used to track five different people. See *face_track.mov* for the complete sequences.

dense multi-person AAM. We then use the dense AAM thus computed to track test data with varied facial expressions across multiple subjects. The AAM tracker is initialized by hand. We find that the dense AAM can be used to reliably track the test data that we presented. In particular, the dense model generalizes well to unseen expressions. The video `face_track.mov` illustrates the tracking. A few snap shots of the tracking video are presented in Figure 9.

4.4. Application to Blob Tracker Output

We now illustrate how our densification algorithm can be used to perform unsupervised AAM construction. Our algorithm is different from previous automatic construction algorithms such as [3] in two important ways: 1) Our algorithm does not need a hand specified mesh; the mesh topology is computed by our algorithm and 2) Our algorithm works far better because of the progressive model refinement. The results obtained by previous authors [3] were fairly limited and mostly consisted of simple rigid motions. In comparison, we apply our densification algorithm to face data with substantial non-rigid deformations.

We used a blob tracker based on a skin color model to detect and track the faces in the video sequences. We use the output of this blob tracker (a roughly face shaped affine-warped planar grid) as the initialization to our algorithm.

To illustrate the computed AAM, we used them to track the faces in two different video sequences. The two video sequences were captured under different illumination conditions. A few snaps shots from the tracking video are shown in Figures 10 and 11. The complete tracking sequences are included in `auto1.mov` and `auto2.mov`. Note that the AAMs used in these sequence are computed in a completely unsupervised manner with absolutely no hand-labeling.

5. Conclusion

In this paper, we have addressed the task of increasing the density of AAMs. Although we work with AAMs, it is possible that the same ideas could be re-applied to the 3D range scan and texture map data used to build more accurate 3DMMs [6]. We also restricted to frontal models. It is possible that the algorithm in this paper can be extended to also model occlusion using the algorithm in [11]. Is is also possible that it can be applied to other data besides human faces.

Our algorithm operates by iterating AAM construction, fitting, and refinement. Refinement itself consists of three steps: adding mesh vertices, re-triangulation, shape mode refinement. Note that any one of these steps on its own would not be sufficient. If we just refine the shape modes, the AAM has no more representational power and does not get any more dense. Similarly, if we just re-triangulate with image-consistent re-triangulation, the representational power does not change much. The average size of the planar triangular elements of the mesh cannot change much and so the AAM has very little more power to model faces more accurately. If we just add vertices, then we can only add them on the planar faces already implicit in the model. It is the combination of the three techniques that gives our algorithm its power. An additional benefit is that our algorithm provides a measure that can be used to decide when to stop increasing the mesh density. Finally, besides being used to increase the density of of a hand constructed AAM, our algorithm can also be used to automatically construct an AAM using no hand-labels by starting from a single planar model of a face obtained with a blob tracker.

Acknowledgements

We would like to thank Stefan Roth for providing the implementation of Bruhn *et al.* [7] that was used in [4].

References

[1] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 years on: Part 4. Technical Report CMU-RI-TR-

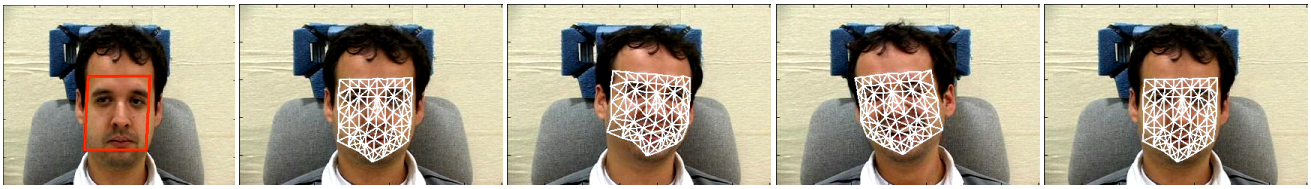


Figure 10. Leftmost image: An example of the rigid tracker inputs to our AAM construction algorithm. Right: A few frames of a face being tracked with an AAM computed in a fully unsupervised manner with no hand labeling. Instead, the AAM was built by running our densification algorithm initialized with the blob tracker. Note that the tracking is fairly accurate especially the mesh region around the mouth deforms well according to the change in expression.

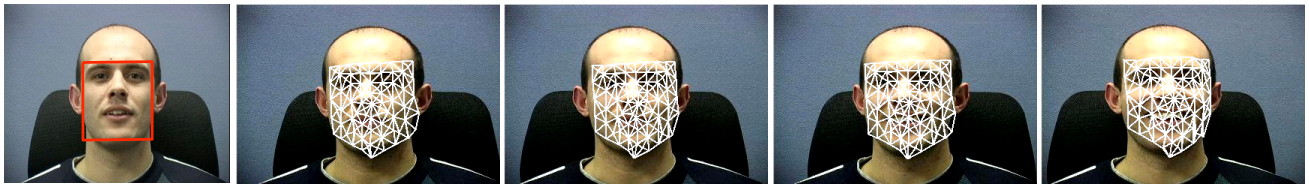


Figure 11. Leftmost image: A second example of the rigid tracker inputs to our AAM construction algorithm. Right: A few frames of a face being tracked with an AAM computed in a fully unsupervised manner with no hand labeling. Tracking with an automatically constructed AAM is far harder task than using a hand-constructed AAM. These results should be compared with the fairly more limited ones in [3].

04-14, Carnegie Mellon University Robotics Institute, 2004.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 56(3):221 – 255, 2004.

[3] S. Baker, I. Matthews, and J. Schneider. Automatic construction of active appearance models as an image coding problem. *PAMI*, 26(10):1380–1384, 2004.

[4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. Database and evaluation methodology for optical flow. In *ICCV*, 2007.

[5] M. Black and Y. Yacoob. Tracking and recognising rigid and non-rigid facial motions using local parametric models of image motion. In *ICCV*, pages 374–381, 1995.

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, pages 187–194, 1999.

[7] A. Bruhn, J. Weickert, and C. Schnorr. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. *IJCV*, 61(3):211–231, 2005.

[8] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *PAMI*, 23(6):681–685, 2001.

[9] D. DeCarlo and D. N. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *IJCV*, 38(2):99–127, 2000.

[10] I. A. Essa and A. Pentland. Facial expression recognition using a dynamic model and motion energy. In *ICCV*, pages 360–367, 1995.

[11] R. Gross, I. Matthews, and S. Baker. Active appearance models with occlusion. *Image and Vision Computing*, 24(6):593–604, 2006.

[12] B. Horn and B. Schunck. Determining Optical Flow. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, April 1980.

[13] Intel Corporation. Intel Open Source Computer Vision Library. http://opencvlibrary.sourceforge.net, 2005.

[14] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade. Passive driver gaze tracking with active appearance models. In *Proceedings of the 11th World Congress on Intelligent Transportation Systems*, 2004.

[15] J.-Y.-Bouquet. Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc, 2005.

[16] A. Lanitis, C. Taylor, and T. Cootes. Automatic interpretation and coding of face images using flexible models. *PAMI*, 19(7):742–756, 1997.

[17] B. Lucas and T. Kanade. An iterative image registration technique with application to stereo vision. In *DARPA Image Understanding*, pages 121–130, 1981.

[18] I. Matthews and S. Baker. Active Appearance Models revisited. *IJCV*, 60(2):135–164, 2004.

[19] I. Matthews, T. Cootes, A. Bangham, S. Cox, and R. Harvery. Extraction of visual features for lipreading. *PAMI*, 24(2):198–213, 2002.

[20] D. D. Morris and T. Kanade. Image-consistent surface triangulation. *PAMI*, 1:332 – 338, 2004.

[21] A. Ogale and Y. Aloimonos. Shape and the stereo correspondence problem. *IJCV*, 65(1):147–162, 2005.

[22] M. F. Tappen, E. H. Adelson, and W. T. Freeman. Estimating intrinsic component images using non-linear regression. In *CVPR*, volume 2, 2006.