

# Improving Face Tracking with 2D Template Warping

Rick Kjeldsen  
IBM T.J. Watson Research Center  
PO Box 704  
Yorktown Heights, NY 10598  
fcmk@us.ibm.com

Aya Aner  
Department of Computer Science  
Columbia University  
New York, New York  
aya@cs.columbia.edu

## Abstract

*Tracking the face of a computer user as they look at various parts of the screen is a fundamental tool for a variety of perceptual user interface applications. The authors have developed a simple but surprisingly robust tracking algorithm based on template matching and applied it successfully in several ways. This paper describes extensions to that algorithm, which improves performance at large facial rotation angles. The method is based on pre-distorting the single training template using 2D image transformations to simulate 3D facial rotations. The method avoids many of the problems associated with using a complex 3D head model. It is robust to variations in the environment and well suited to use in practical applications in typical computing environments.*

## 1. Introduction

As cameras transition from multi-media peripherals to true user interface devices, one challenge will be to develop a set of basic functions which can be applied in various ways to perform user tasks. In the same way a scroll bar can be applied to many tasks in a standard WIMP interface, so we will need a widget set for the vision-based user interface "devices" of the future.

Accurately tracking the location of the user's face as they work and look around the screen is one such widget. It is useful for a variety of tasks, including allowing the user to identify areas on the screen by pointing with their nose, allowing head gestures such as nodding or shaking, determining what the user is attending to, determining if a user is speaking to or gesticulating at the computer, or as a precursor to tasks such as gaze direction or expression classification.

One reason that precise knowledge of the location of the face in an image is useful is because it can serve as a

surrogate for facial orientation. As a user turns their head, it rotates about a point near the back of the skull where the spine attaches (see bottom rightmost coordinate system in figure 1). Therefore as the user aims their face at different locations, even locations relatively close to one another, the location of the face in a frontal image will translate significantly. We have taken advantage of this characteristic in a system that allows the use of facial aiming as a pointing device. By passing the location of the users face through a suitable transfer function, they can easily identify locations on the screen by rotating their face to aim at it. Using facial location, rather than facial orientation, has the advantage of simplicity, allowing the system to operate reliably in real time.

The focus of this paper is not the pointing application itself, rather the fast and accurate face tracking algorithms which support it, specifically, improvements to those algorithms which allow it to work in the presence of large facial rotation angles. The details of how facial location is applied to various tasks are beyond its scope.

### 1.1. Constraints

The face tracker's intended use as an interface device implies several constraints that make the task easier. We can assume that the user is standing or sitting in front of a personal display screen at a comfortable reading distance and that the camera is on or near the display. It is also safe to assume that the environment is relatively stable over time, such as an indoor office environment. Within these boundaries, however, the tracker must be robust enough to withstand different lighting conditions and geometric configurations.

The intended application also implies several criteria any solution must meet. It must be able to track only the user's face when other faces may be present in the background. In order to support real time interactions, the method must process each frame in 100 ms or less, preferably in the 30

ms range, in order to leave sufficient CPU time available for other applications.

One characteristic of many tracking algorithms is that they rely on historical information about position and movement of the target. In our experience, this leads to difficulties in determining when the algorithm has lost the target and then delays while finding it again. Therefore, we have gravitated toward algorithms that can function well in the absence of historical information, generally finding the face anew each frame. Finally, it is important that the approach use only the simplest training procedures. Experience testing similar applications on end users has indicated that anything more than a single, simple training step (e.g. "Look at the dot and click here") meets with a negative response.

## 1.2. Approaches to face tracking

Many approaches have been suggested to find and track faces in images. General-purpose face finders, such as [2] and [14] do not require training for each user, but can not differentiate between the user and other faces in the image and generally run too slowly for real-time response.

Another approach has been to combine several tracking techniques in a single system. McKenna and Gong [4] describe an interesting approach which combines fast motion-based tracking and precise neural net-based face detection to segment and track faces from video sequences. Tao, Lopez and Huang [15] present a system which tracks the location of the face and facial features using, in part, a probability network encoding the spatial relationships between facial features. Approaches such as these involve considerable system complexity, which is at odds with our goal to operate in real time and concurrently with other applications in a typical computing environment.

A system with similar constraints is described in [3]. Here Niyogi and Freeman describe a system that tracks a driver's head in real time to monitor where she is looking. They classify the pose of the head by training a decision tree that uses PCA to determine the direction of maximal variation at each branch. As such it can handle a wide variation in head pose, but is limited in the granularity of poses it can classify.

## 1.3. Tracking algorithm overview

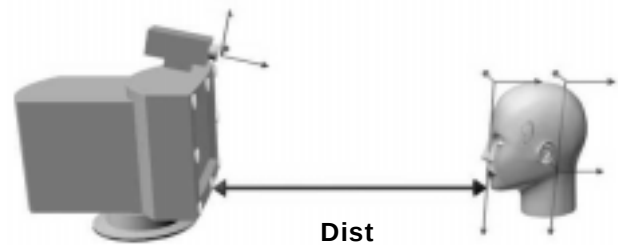
In order to satisfy the constraints outlined above, we have been exploring an approach based on matching a single facial template. The approach requires a simple training step each time the user begins to use the system. Training consists of asking the user to look at the center of the screen and press a button. Their face is located using slow but accurate face finding techniques (1 to 2 seconds).

An image of the face bounded by the eyebrows, edges of the eyes and bottom of the mouth is saved as our match template and the tip of the nose is identified.

During tracking, the face is located in each new image by finding the region which best matches the template. Our match function has varied, but currently finds the minimum sum of absolute differences between the red channel of each pixel in a region and the face template.

The details of the error function are not important, however, as our experience shows any reasonable alternative seems to give good results.

This tracking algorithm tracks the users face very well under most circumstances, finding the face as the user aims their nose at different locations around a typical screen with near 100% accuracy. It runs at 15-30 fps on a standard 300MHz Pentium with an inexpensive USB (Universal Serial Bus) camera.



**Figure 1: Geometry of Camera and User. The rightmost bottom coordinate system is the rotation point of the head. The leftmost upper coordinate system corresponds to the camera plane. We simulate user looking at eight locations on screen, shown by white dots.**

Being appearance-based, however, the reliability of the tracker begins to fall off when the image of the user's face changes significantly from the training view. This can occur for a variety of reasons, including significant changes in lighting and radical facial expressions. In our experience, however, the only failure mode that significantly impacts usability is when the user turns their head beyond about 25 degrees from the original orientation. This occurs when users aim their face at or beyond the edge of a large screen. In the face pointing application we find that this seems to be a natural urge, especially with novice users, perhaps to try to get the pointer to the edge of the screen faster. In these circumstances the template tends to match best somewhere else on to the users face or neck. As soon as the face returns to a more frontal view, the algorithm recovers.

In the remainder of this paper we will describe how we have attempted to improve performance of the face tracker as the angle of the user's face increases. Our approach is to pre-distort the training image to match the expected

orientation of the user's face. The next section describes the warping algorithm we use to distort the training image. Section 3 describes how the distorted templates are used. In Section 4 we then present the results of tests which show that the pre-distortion significantly improves behavior of the face tracker at large rotation angles.

## 2. Facial pre-distortion

In order to improve tracking performance as the user turns their head, we use a simple facial model and some assumptions about the environment to create a set of images that better correspond to how the rotated facial image actually appears. This pre-distortion algorithm uses two-dimensional warps to simulate rotation of a simple three-dimensional model of the face. Our "face model" consists of a plane containing the lips, chin, cheeks, eyes and forehead, and a three-sided pyramid for the nose. Since we are not concerned with the sides of the face, this model has proved sufficiently accurate. Indeed, in the region of the face we are most concerned with, it is more accurate than some simple 3D facial models, such as the elliptical head model of Pappu and Beardsley [8], because, except for the nose, the facial features very nearly lie on a plane. While better looking results could likely be obtained using a full 3D model such as [6], or a more refined 2D model such as was used in [12] and [13], for our purposes the additional accuracy is not worth the added complexity.

The goal of the pre-distortion is to improve the match score, which is not the same as creating realistic looking images. For our needs, the facial features must be located in approximately the right locations and their relative sizes must be approximately right, but details such as the exact shape of the nose can be distorted with little effect on match score. In order to get good results from 3D models, the original image must be accurately mapped onto the model, which is a non-trivial task, and the model must correspond well to the true shape of the user's face. Our simple model gives results sufficiently accurate to improve tracking without the complexity of more elaborate techniques.

The pre-distortion algorithm is given a single frontal view of the user, a bounding box around the eyebrows and mouth, and the location of the tip of the nose. It is given a coarse description of the camera location, in terms of above, beside or below the screen. In order to keep the training process simple, it makes some basic assumptions regarding the user's position with respect to the screen and the size of the user's head, but it has no detailed model of the environment or the shape of the user's head. In order to avoid camera calibration, we use a representative value for camera focal length. These assumptions can be tuned for individual situations, but in practice we have not seen a noticeable change in performance when we do.

Using this simple facial model and the texture data from the single frontal training image, we create eight views of the face corresponding to cases when the user is looking at different edges of the screen: top, bottom, right, left and the four corners. We generate the images in a two step process, first tilting the facial plane, then rotating the nose pyramid. We will describe each process separately.

### 2.1. Facial plane rotation

First consider the tilting of the facial plane. Since the user is asked to look at the center of the screen when the training image is taken, we can assume that in the training image the facial plane is approximately parallel to the screen and that a line perpendicular to the facial plane and passing through the nose intersects the center of the screen, as shown in Figure 1. We have observed that minor variations from these conditions do not affect performance. We draw a rectangle in the facial plane bounded by the user's eyebrows, lower lips and outer eye edges. Because the image plane is in general not parallel to the facial plane, this will project into a quadrilateral in the image. We use perspective projection, using our assumed camera focal length  $d$ , to determine that quadrilateral. For each corner  $i$  of the 3D rectangle  $(Rx_i, Ry_i, Rz_i)$ , the corresponding quadrilateral corner is:

$$Qx_i = Rx_i / (Rz_i / d)$$

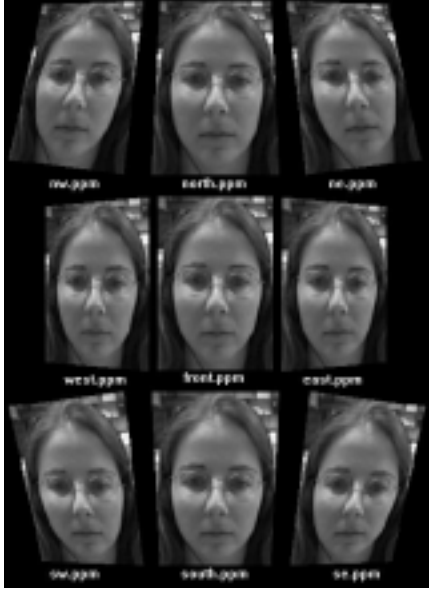
$$Qy_i = Ry_i / (Rz_i / d)$$

We then rotate the facial plane by the desired amount and translate the rectangle around the facial features into the image plane creating a second quadrilateral.

Using Heckbert's method [1], we compute the perspective transformation matrix between these quadrilaterals. We first scale the quadrilaterals so that they match the size of the front view image to be warped, that is, convert from inches to pixels. To compute the transformation between two quadrilaterals  $q1$  and  $q2$ , we compute a transformation matrix from a square to  $q1$ , find its adjoint and multiply this with the matrix to transform a square to  $q2$ . We then compute the inverse transformation and apply it using backward mapping from the rotated view to the frontal view quadrilaterals.

If we apply the resulting perspective transformation matrix only on the facial bounding box of the training image, we end up with template of different size and shape for each direction, which would add significant complexity to the matching process. Instead the face warp is applied on a larger portion of the training image and the result is cropped to a rectangle.

We compute where the rectangle should be cropped with the same perspective transformation matrix used above. This process is surprisingly complex, however, because the final result must contain the majority of the facial features,



**Figure 2: 2D quadrilaterals after perspective projection transformation**

but must not contain background (non-face) pixels, as they will significantly affect the match score. If we simply apply the perspective transformation to the facial bounding box to generate a quadrilateral that surrounds the facial features in the rotated image and then take the bounding box of that quadrilateral, compound rotations (e.g. the corner images of Figure 2) often end up with background pixels in the template. Instead we do the following. For simple rotations (around the X or Y axis only), we apply the transformation on the middle points between x and y coordinates of upper, lower, left and right corners of the bounding box. For compound rotations we use the coordinates of the corners that transform into the quadrilateral's inner corners, so that the final rectangle gets thinner (Figure 4).

Rotation of the facial plane takes care of the spatial distortion of the major facial features. On a horizontal rotation, the eyes become closer, one cheek becomes more prominent than the other and the angle of the mouth and the eyes change (Figure 2). In vertical rotation the chin or forehead becomes more prominent and the distance between eyebrows and mouth changes.

The image of the nose, however, is not moved sufficiently far as it most seriously violates the planar assumption. Thus the final issue in creating the pre-distorted images is to tilt the nose pyramid.

## 2.2. Nose warping

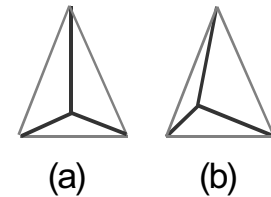
Using only facial plane rotation does not give a significant improvement in match performance. As noted in [9], one of the most important changes in facial appearance

as the face rotates is the change in appearance of the nose. We therefore add an additional step to our pre-distortion algorithm to approximate the change in appearance of the nose as the face rotates.

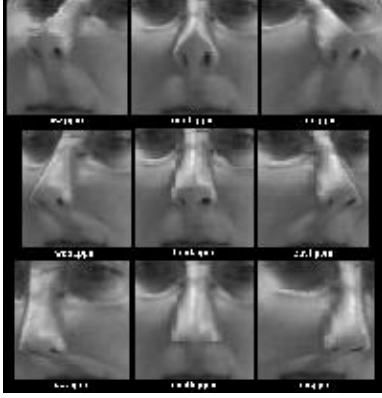
Conceptually, we model the nose as a three-sided pyramid, which we tilt along with the plane. In practice, we simulate this 3D transformation with a 2D warp by modeling the pyramid as a triangle with three smaller triangles inside it (Figure 3). The triangle is scaled based on the size of the facial bounding box and placed on the template such that the intersection point of the inner triangles lies on the tip of the nose. To simulate the 3D rotation of the pyramid, the intersection point of the three internal triangles is moved to the new nose tip location, distorting all three triangles as shown in Figure 3b. The pixel data from the original triangles is then warped into the new shape. In other words, when a triangle expands, we distribute the original pixel values appropriately through new triangle and fill in the missing data by interpolating between surrounding pixels. When a triangle contracts, we combine several pixel values into one. In order to determine the amount to move the nose tip, we perform a straightforward geometric calculation of the displacement of the nose tip using the assumed environmental geometry.

## 3. Tracking with the warped templates

During tracking, the facial template is matched at each pixel displacement in a region around the base face location. This base location can be the original training location or can be computed from other image features, such as the location of the shoulders. The displacement with the best match score is taken as the current location of the face. Since displacement from the base location is assumed to occur only from facial rotation, each match attempt corresponds to some amount of facial rotation. Therefore, our original thought was to use a unique warped image for each match attempt. In other words, if we were matching at a location  $x+i, y+j$ , where  $x, y$  is the base facial location, we would use as a match template the image warped by the rotation implied by offset  $i, j$ .



**Figure 3: Nose Warping. Left figure shows original triangular model with direction of warp.**



**Figure 4: Original template (center) and eight pre-distorted versions**

Unfortunately this approach proved impractical because of the difficulty of registering the warped image with the true facial rotation.

Since matching with the original image performs very well for small facial rotations, the pre-distorted templates are only needed at larger rotations. Therefore, we instead settled on using a small number of warped images corresponding to the maximal facial rotations we expected to see, and using these only in the periphery of the match region. At training time we create eight warped images corresponding to the maximal facial rotations in each of the eight primary directions. We divide the search region into nine sub-regions. In the large center region we use the original training image. In each of the outer sub-regions we use the warped face image (Figure 4) corresponding to the rotation direction implied by the offset from the base location. The boundaries between the regions have been determined empirically.

## 4. Experiments and results

To test the effect of the facial pre-distortion algorithm, we began with a face tracking system based on matching only the original training image as described in Section 1.3. As described earlier, this system tracked the face very well for small rotations, but tended to fail as the facial rotation reached the very edges of the screen. We then integrated the facial pre-distortion algorithm as described in Section 3. The result was a significant decrease in the number of times the tracking algorithm lost the face as the user looked toward and past the edges of the screen. Two example shots from this system are shown in Figure 5. The white rectangle shows the location where the original training template matched best. The gray rectangle shows where the pre-distorted template had its best match score. Above each image we show the pre-distorted template for that direction. While this type of test provides the most accurate

demonstration of the performance improvement, the results are difficult to quantify.

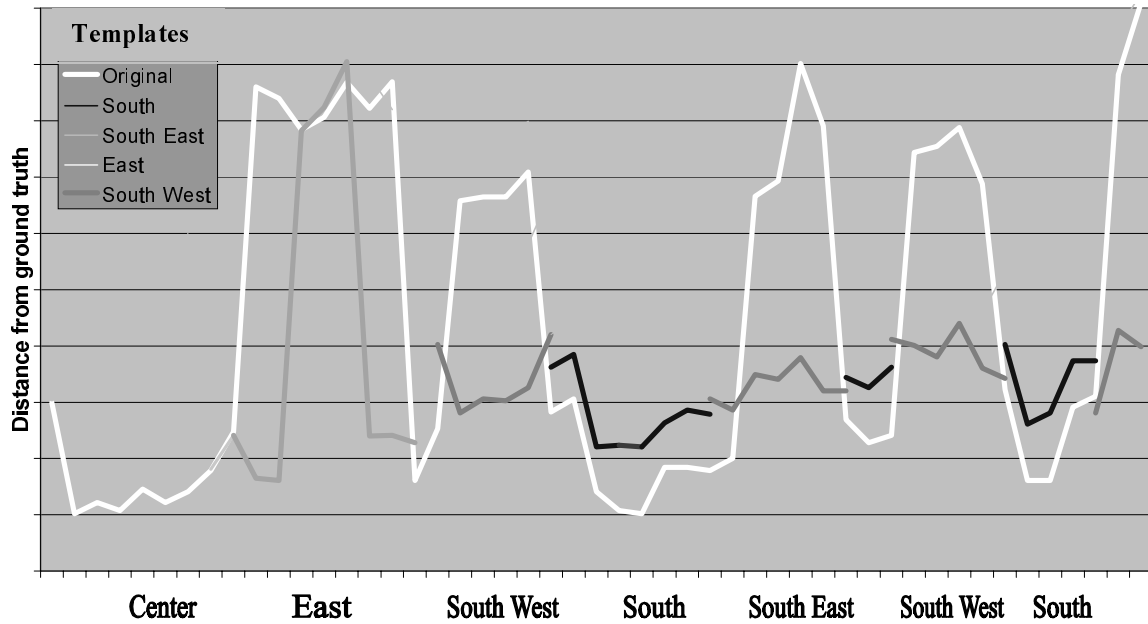
In order to get more concrete data on the performance improvement provided by the pre-distortion algorithm, we performed a more controlled experiment. We enlisted eight different users, asking them to sit at a comfortable working distance from a 19" monitor. We then recorded video sequences as they rotated their head in a zigzag scanning of the screen. We ensured that they scanned far enough above, below and to the sides that the original tracking algorithm would have trouble. For each user we also took a training image and created eight pre-distorted templates where the rotation of the face would bring it to the edge of the screen.

We sampled every fifth frame of the videos to give us a set of representative test images. It should be pointed out that the important metric is not the match score itself, but rather the location at which the best match occurs. Therefore, for each image, we computed ground truth in the form of the location of a specific point on the user's face. Finally, we computed the location in each image where each template had its best match score and computed the distance between this location and the ground truth.

Figure 6 shows an example of the performance of various templates across a few scans by one of the users. The vertical axis is the distance of the best template match location from ground truth. The white line shows the performance of the original template and the gray lines show the performance of the pre-distorted templates that are appropriate for the direction the user is looking. Notice that the original template does well when the user is looking toward the center of the screen, but fails as the user looks to the edges (periodic peaks in the graph). The pre-distorted images nearly always did better than the original image in



**Figure 5: Running example of our tracker. The white rectangle shows where the original template matched, the gray rectangle shows where the rotation template matched. The original and the rotated templates are shown above.**



**Figure 6: Performance of the original and pre-distorted templates during facial scanning.**

the region where they are appropriate. Only in the first East scan did the distorted template fail. Here the user had turned their head so far that it was nearly in profile.

Table 1 summarizes the performance for several users. Over 70% of the time, using the appropriate warped template gave a match location closer to ground truth than using the original. Only 28% of the time did the original template do better. Often, however, one template or the other is slightly closer to ground truth, but both are close enough to be considered an accurate match. We therefore set a distance threshold above which we considered the match to have failed. This distance was set subjectively, but we attempted to be realistic in the sense that matches within the threshold would generally be usable and those outside would generally be unusable. We then examined those frames where the original template gave a failed match. In 33 of those cases the warped templates gave a good match. In comparison, there were no cases when the warped template failed, but the original template gave a good match. We should note that with our test data it was not meaningful to record the absolute number of times the templates failed. At the end of each scan the user's face was often rotated so far beyond the edge of the screen that both templates failed to match, as in the east case of figure 6.

## 5. Summary and conclusions

We have presented a method of tracking the location of the user's face as they look around the screen in a typical computing environment. The method is based on template

matching from a single training image and therefore begins to fail at head rotations that are likely to be encountered under normal conditions. We focused on an extension to the method to improve its performance with large face rotations, based on pre-distorting the template using generic knowledge of facial geometry and the environment.

Since the method uses 2D warping rather than elaborate 3D models, it gains speed and reliability and avoids the need for elaborate training or calibration techniques. In spite of its simplicity, we have shown that it significantly improves the performance of the face tracking system with a variety of users.

One significant shortcoming of the technique is that it does not account for changes in the appearance of the face under rotation that are due to directional lighting (e.g. [5]).

Future work will address this issue.

**Table 1: Results gathered from several users looking at different locations on the screen**

|                                                                       |     |
|-----------------------------------------------------------------------|-----|
| Warp template did better than original template                       | 72% |
| Original template did better than warp template                       | 28% |
| Successful matching of warped templates when original template failed | 33  |
| Successful matching of original template when warped templates failed | 0   |

## References

- [1] P. Heckbert. Fundamentals of Texture Mapping and Image Warping. MS thesis, Dept. of EECS, Univ. of California at Berkeley, Technical Report No. UCB/CFD89/516, June 1989
- [2] H.A. Rowley, S. Baluja and T. Kanade. Human Face Detection in Visual Scenes. School of Computer Science, Carnegie Mellon University Technical Report No. CMU--CS--95--158R, November, 1995
- [3] S. Niyogi and W. T. Freeman. Example based head tracking. In *Proceedings 2nd International Conference on Automatic Face and Gesture Recognition*, Killington, VT, USA, 1996.
- [4] S. McKenna and S. Gong., Tracking Faces. In *Proceedings 2nd International Conference on Automatic Face and Gesture Recognition*, Killington, VT, USA, 1996.
- [5] P.W. Hallinan. A Low-Dimensional Representation of Human Faces For Arbitrary Lighting Conditions. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 995-999, Seattle, WA, 1994.
- [6] A.C. Aitchison and I. Craw. Synthetic Images of Faces – An Approach to Model-Based Face Recognition. In *Proc. British Machine Vision Conference*, pages 226-232, 1991.
- [7] P.G. Schyns and H. H. Blthoff. Conditions For Viewpoint Dependent Face Recognition. A.I. Memo No. 1432, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.
- [8] R. Pappu and P. A. Beardsley. A Qualitative Approach to Classifying Gaze Direction. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, April 1998.
- [9] A. Pentland, B. Moghaddam and T. Starner. View-Based and Modular Eigenspaces for Face Recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 84-91, Seattle, WA, 1994.
- [10] Q. Chen, H. Wu, T. Fukumoto and M. Yachida. 3D Head Pose Estimation without Feature Tracking. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, April 1998
- [11] D. J. Beymer. Face Recognition Under Varying Pose. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 756-761, Seattle, WA, 1994. OR:
- [12] D. J. Beymer and T. Poggio. Face Recognition From One Example View. In *Proceedings of the fifth International Conference on Computer Vision*, 1995.
- [13] T. Vetter. Learning novel views to a single face image. In *Proceedings 2nd International Conference on Automatic Face and Gesture Recognition*, Killington, VT, USA, 1996.
- [14] K. Sung and T. Poggio. Example-based Learning for View-based Human Face Detection. MIT AI Laboratory Technical Report No. A11521/CBCL112, 1994.
- [15] H. Tao R. Lopez and T. Huang. Tracking Facial Features Using Probabilistic Network. *IEEE Workshop on Face and Gesture Recognition*, 1998.