Eye Gaze Correction with Stereovision for Video-Teleconferencing

Ruigang Yang, *Member*, *IEEE*, and
Zhengyou Zhang, *Senior Member*, *IEEE*

Abstract—The lack of eye contact in desktop video teleconferencing substantially reduces the effectiveness of video contents. While expensive and bulky hardware is available on the market to correct eye gaze, researchers have been trying to provide a practical software-based solution to bring video-teleconferencing one step closer to the mass market. This paper presents a novel approach: Based on stereo analysis combined with rich domain knowledge (a personalized face model), we synthesize, using graphics hardware, a virtual video that maintains eye contact. A 3D stereo head tracker with a *personalized* face model is used to compute initial correspondences across two views. More correspondences are then added through template and feature matching. Finally, all the correspondence information is fused together for view synthesis using view morphing techniques. The combined methods greatly enhance the accuracy and robustness of the synthesized views. Our current system is able to generate an eye-gaze corrected video stream at five frames per second on a commodity 1 GHz PC.

Index Terms—Stereoscopic vision, eye-gaze correction, model-based tracking, head tracking, pose determination.

1 Introduction

VIDEO teleconferencing, a technology enabling communicating with people face-to-face over remote distances, does not seem to be as widespread as predicted. Among many problems faced in video-teleconferencing, such as cost, network bandwidth, and resolution, the lack of eye-contact seems to be the most difficult one to overcome [10].

In a typical desktop video-teleconferencing setup, the camera and the display screen cannot be physically aligned, as depicted in Fig. 1. A participant looks at the image of the remote party displayed on the monitor, but not directly into the camera, while the remote party "looks" at her through the camera, thus she does not appear to make eye contact with the remote party. Research [14] has shown that if the divergence angle (α) between the camera and the display is greater than five degrees, the loss of eye-contact is noticeable. If we mount a small camera on the side of a 21-inch monitor, and the normal viewing position is at 20 inches away from the screen, the divergence angle will be 17 degrees, well above the threshold at which the eye-contact can be maintained. Under such a setup, the video loses much of its communication value and becomes ineffective compared to telephone, resulting in unnatural and even awkward interaction.

Several systems have been proposed to reduce or eliminate the angular deviation by using special hardware. They make use of half-silvered mirrors or transparent screens with projectors to allow the camera to be placed on the optical path of the display. A brief review of these hardware-based techniques has been given in [6]. The expensive cost and the bulky setup prevent them from being used in a ubiquitous way. In our work, we address the eye-contact problem by synthesizing videos as if they were taken from a camera behind the display screen, thus establishing natural eye-contact between video-teleconferencing participants without using any kind of special hardware.

Our approach involves three steps: pose tracking, view matching, and view synthesis. We use a pair of calibrated stereo cameras and a personalized face model to track the head pose in 3D. One camera is mounted on the upper edge of the display, while the other is on the lower edge of the display. This type of wide baseline configuration makes stereo matching very hard. The vertical stereo setting significantly reduces ambiguities in matching facial features. The use of strong domain knowledge (a personalized face model) and that of a stereo camera pair greatly increase the robustness and accuracy of the 3D head pose tracking, which in turn helps stereo matching. The stereo camera pair also allows us to match parts not modeled by the face model, such as the hands and shoulders, thus providing wider coverage of the subject. Finally, the results from the head tracking and stereo matching are combined to generate a virtual view. Unlike other methods that only "cut and paste" the face part of the image, our method generates natural looking and seamless images, as shown in Fig. 2. We believe our proposed approach advances the state of the art in the following ways:

- By combining a personalized model and a stereo camera pair, we exploit a priori domain knowledge with added flexibility from the stereo cameras.
- By exploiting both temporal and spatial coherence, we present a novel head tracking algorithm that operates in real time and maintains highly accurate full 3D tracking, even under difficult conditions such as partial occlusions or nonrigid motions.
- By combining a comprehensive set of view matching criteria, we match as many features as possible, including salient point features and object silhouettes. These features are not restricted by the model we use, allowing us to create seamless and convincing virtual views.

We believe the quality of our synthesized views is among the best of previously published methods.

As will be discussed in more detail in Section 2, several attempts have been made to address the eye-contact problem using a single camera with a face model [4],[3], or using dense stereo matching techniques [11], [7]. With a single camera, we found it difficult to maintain both the real-time requirement and the level of accuracy we need with head tracking. Existing model-based monocular head tracking methods either use a simplistic model so they could operate in real time, but produce less accurate results, or use some sophisticated model and expensive processing to yield highly accurate results, but take at least several seconds to compute. A single-camera configuration also has difficulties to deal with occlusions. Considering these problems, we decided to adopt a stereo configuration. The important epipolar constraint in stereo allows us to reject most outliers without using expensive robust estimation techniques, thus keeping our tracking algorithm both robust and simple enough to operate in real-time [16]. Furthermore, two cameras usually provide more coverage of the scene.

One might raise the question of why we do not use a dense stereo matching algorithm. We argue that, first, doing dense stereo matching on a commodity PC in real time is difficult, even with today's latest hardware. Second, and most importantly, dense stereo matching is unlikely to generate satisfactory results due to camera placement. If we put the cameras on the opposite edges of the display, given the normal viewing distance, we have to converge the cameras toward the person sitting in front of the desktop, and such a stereo system will have a large baseline. That makes stereo matching very difficult; even if we were able to get a perfect matching, there would still be a significant portion of the subject which is occluded in one view or the other. Alternatively, if we put the cameras close to each other on the same edge of the monitor frame, the occlusion problem is less severe, but generalization to new distant views is poor because a significant portion of the face is not observed. After considering various aspects, we have decided to put one camera on the upper edge of the display and the other on the lower edge, and follow a model-based stereo approach to eye-gaze correction.

The remaining of this paper is organized as follows: We introduce some previous work about eye-gaze correction in Section 2. In Section 3, we present a high-level overview of our proposed setup and methods. Three major steps of our methods,

- *R. Yang is with the Department of Computer Science, University of Kentucky, Lexington, KY 40506-0195. E-mail: ryang@cs.uky.edu.*
- *Z. Zhang is with Microsoft Research, One Microsoft Way, Redmond, WA 98052. E-mail: zhang@microsoft.com.*

Manuscript received 17 Sept. 2002; revised 6 Aug. 2003; accepted 4 Nov. 2003. Recommended for acceptance by L. Quan.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 117393.

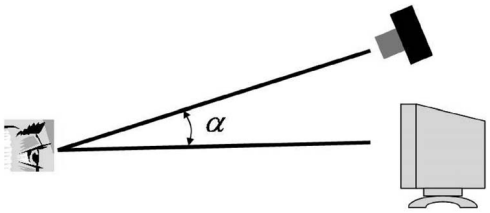

Fig. 1. Camera-screen displacement causes the loss of eye-contact.

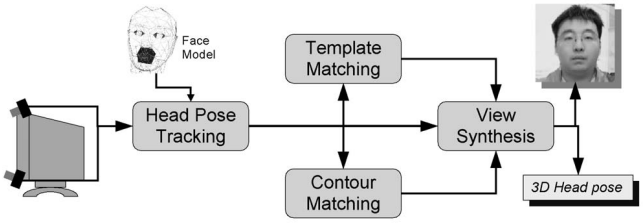

Fig. 3. The components of our eye-gaze correction system.

stereo tracking, view matching, and view synthesis, are presented in Section 4, 5, and 6, respectively. Then, we present some results from real sequences in Section 7 and conclude in Section 8. Due to space limitation, the reader is referred to [16], [15] for full technical details.

2 Related Works

We are not the first researchers attempting to create eye-contact using computer vision and computer graphics algorithms. Ott et al. [11] proposed to create a virtual center view given two cameras mounted on either side of the display screen. Stereoscopic analysis of the two camera views provides a depth map of the scene. Thus, it is possible to "rotate" one of the views to obtain a center virtual view that preserves eye contact. Similarly, Liu et al. [7] used a trinocular stereo setup to establish eye contact. In both cases, they perform dense stereo matching without taking into account the domain knowledge. While they are generic enough to handle a variety of objects besides faces, they are likely to suffer from the vulnerability of brute-force stereo matching. Furthermore, as discussed in the previous section, we suspect that direct dense stereo matching is unlikely to generate satisfactory results due to the constraint of camera placement imposed by the size of the display monitor.

Cham and Jones, at Compaq Cambridge Research Laboratory [1], approached this problem from a machine learning standpoint. They first register a 2D face model to the input image taken from a single camera, and then morph the face model to the desired image. The key is to learn a function that maps the registered face model parameters to the desired morphed model parameters [4]. They achieve this by nonlinear regression from a large number of registered-morphed parameter pairs which are obtained from training data. As far as we know, their research is still in a very early stage, so it is not clear if this approach is capable of handling dramatic facial expression changes. Furthermore, they only deal with the face part of the image—the morphed face image is superimposed on the original image frame, which sometime leads to errors near the silhouettes due to visibility changes.

The GazeMaster project at Microsoft Research [3] uses a single camera to track the head orientation and eye positions. Their view synthesis is quite unique in that they first replace the human eyes in a video frame with synthetic eyes gazing in the desired direction, then texture-map the eye-gaze corrected video frame to a generic rigid face model rotated to the desired orientation. The synthesized photos they published look more like avatars, probably due to the underlying generic face model. Another drawback is that, as noted in their report, using synthetic eyes sometime inadvertently changes the facial expression as well.

3 System Overview

Fig. 3 illustrates the block diagram of our eye-gaze correction system. We use two digital video cameras mounted vertically, one on the top and the other on the bottom of the display screen. They are connected to a PC through 1394 links. The cameras are calibrated using the method described in [17]. We choose the vertical setup because it provides wider coverage of the subject and higher disambiguation power in feature matching. Matching ambiguity usually involves symmetric facial features such as eyes and lip contours aligned horizontally. The user's personalized face model is acquired using a rapid face modeling tool [8]. Both the calibration and model acquisition require little human interaction, and a novice user can complete each task within five minutes.

With a calibrated camera pair and a 3D face model, we are able to correct the eye-gaze using the algorithm outlined below:

1. Background model acquisition.
2. Face tracker initialization.
3. For each image pair, perform
   - background subtraction,
   - temporal feature tracking in both images,
   - updating head pose,
   - correlation-based stereo feature matching,
   - stereo silhouette matching, and
   - hardware-assisted view synthesis.

Currently, the only manual part of the system is the face tracker initialization, which requires the user to interactively select a few markers in the first frame. We are currently working on automatic initialization.

The tracking subsystem includes a feedback loop that supplies fresh salient features at each frame to make the tracking more stable under adversary conditions, such as partial occlusions and facial expression changes. Furthermore, an automatic tracking recovery mechanism is also implemented to make the whole system even more robust over extended period of time. Based on the tracking information, we are already able to manipulate the head pose by projecting the live images onto the face model.

(a) (b) (c)

Fig. 2. Eye-gaze correction: (a) and (c) were taken from the stereo cameras mounted on the top and bottom sides of a monitor while the person was looking at the screen. (b) is a synthesized virtual view that preserves eye-contact. Note that the eye gaze in the virtual view appears to be looking forward, as desired.

However, we also want to capture the subtleties of facial expressions and other foreground objects, such as hands and shoulders. So, we further conduct correlation-based feature matching and silhouette matching between the stereo images. All the matching information, together with the tracked features, is used to synthesize a *seamless* virtual image that looks as if it were taken from a camera behind the display screen. In the next three sections, we will explain three major steps in the gaze correction process: stereo head tracking, stereo matching, and view synthesis.

4 Stereo 3D Head Pose Tracking

We use a personalized face model in our head pose tracking. The problem can be stated as follows:

1. *Given* 1) a set of triplets $S = \{(\mathbf{p}, \mathbf{q}, \mathbf{m})\}$ at time t, where $\mathbf{p}$ and $\mathbf{q}$ are, respectively, points in the upper (first) and the lower (second) camera images, and $\mathbf{m}$ is their corresponding point in the face model, and 2) a pair of images from the stereo cameras at time $t+1$.
2. *Determine* 1) $S' = \{(\mathbf{p}', \mathbf{q}', \mathbf{m})\}$ at time $t+1$, where $\mathbf{p}'$ and $\mathbf{q}'$ are the new positions of $\mathbf{p}$ and $\mathbf{q}$, and 2) compute the head pose so that the projections of $\mathbf{m}$ in time $t+1$ are $\mathbf{p}'$ and $\mathbf{q}'$ in the stereo image pair, respectively.

We use the KLT tracker to track feature points $\mathbf{p}, \mathbf{q}$ from time t to $t+1$ [13]. Note that there is one independent feature tracker for each camera, thus we need to apply the epipolar constraint to remove any stray point. After we have removed all the stray points that violate the epipolar constraint, we update the head pose, represented by a 3×3 rotational matrix $\mathbf{R}$ and a 3D translation vector $\mathbf{t}$, so that the sum of reprojection errors of $\mathbf{m}$ to $\mathbf{p}'$ and $\mathbf{q}'$ is minimized.

After the head pose is determined, we replenish the matched set S' by adding more good feature points selected using the criteria in [13]. We must be careful not to add feature points in the nonrigid parts of the face, such as the mouth region. To do so, we define a bounding box around the tip of the nose that covers the forehead, eyes, and nose region. Any good feature points outside this bounding box will not be added to the matched set. However, they will be used in the next stereo matching stage to be discussed in Section 5.1. The replenish scheme greatly improves the robustness of the tracking algorithm. Our experiments have shown that our tracker can keep tracking under large head rotations and dramatic facial expression changes, and also that it is much more robust than monocular tracker. The interested reader is referred to [16] for details.

5 Stereo View Matching

The result from the 3D head pose tracking gives a set of good matches within the *rigid* part of the face between the stereo pair. To generate convincing and photo-realistic virtual views, we need to find more matching points over the entire foreground images, especially along the contour and the nonrigid parts of the face. We incorporate both feature matching and template matching to find as many matches as possible. During this matching process, we use the reliable information obtained from tracking to constrain the search range. In areas where such information is not available, such as hands and shoulders, we relax the search threshold, and then apply the disparity gradient limit (defined in [12]) to remove false matches.

To facilitate the matching (and, later, view synthesis in Section 6), we rectify the images using the technique described in [9] so that the epipolar lines are horizontal.

5.1 Feature Matching Using Correlation

For unmatched good features in the first (upper) image, we try to find corresponding points, if any, in the second (lower) image by template matching. We use normalized correlation over a 9×9 window to compute the matching score. The disparity search range is confined by existing matched points from tracking, when available. Matched features are further filtered using the disparity gradient limit, i.e., if a feature's disparity gradient with regard to a tracked point is greater than a certain threshold, we will discard this feature. Details can be found in [15].

Note that, unlike temporal tracking, we consider at this stage all good features including points on the mouth and eye regions. This is because the two cameras are synchronized, and facial deformation does not affect the epipolar constraint in stereo matching.

5.2 Contour Matching

Template-matching assumes that corresponding image patches share some similarity and are distinct from their neighbors. This assumption may be wrong at object contours, especially at occluding boundaries. Yet, object contours are very important cues for view synthesis because the lack of matching information along object contours will result in excessive smearing or blurring in the synthesized views. So, it is necessary to include a module that extracts and matches the contours across views in our system.

The contour of the foreground object can be extracted after background subtraction. It is approximated by polygonal lines using the Douglas-Poker algorithm [2]. The control points (vertices) on the contour, $\mathbf{v}_i$, are further refined to subpixel accuracy using the "snake" technique [5]. Once we have two polygonal contours, we use the dynamic programming technique (DP) to find the global optimal matching between them. Due to space limitation, we leave out the details here and refer the interested reader to [15]. In summary, contour matching outputs a list of matching line segments across two views; important features such as face silhouettes and the outlines of waving hands are matched.

6 View Synthesis

From the previous tracking and matching stages, we have obtained a set of point matches and line matches that can be used to synthesize new views. Note that these matches contain not only the modeled face part, but also other foreground part such as hands and shoulders. This is yet another advantage of our stereovision-based approach. We could obtain a more complete description of the scene geometry beyond the limit of the face model. Treating these matches as a whole, our view synthesis methods can create seamless virtual imagery.

We use graphics hardware to synthesize virtual views in real time. In our view synthesis scheme, we treat every pair of line match, which is obtained from contour matching in Section 5.2, as two pairs of point matches, i.e., the end points of one line match the end points of the other line based on the epipolar geometry. Together with the point matches from tracking and template matching, we create a 2D triangular mesh using Delaunay triangulation in the first camera's image space. We then offset each vertex's coordinate by its disparity modulated by a view morphing factor c_m, i.e., $[u_i', v_i'] = [u_i + c_m d_i, v_i]$. c_m controls the exact view position. It is usually between 0 and 1, whereas a value of 0 corresponds exactly to the first camera view, and a value of 1 corresponds exactly to the second camera view. Any value in between represents a virtual viewpoint somewhere along the path from the first camera to the second.

The offset mesh is then fed to the graphics hardware with two sets of texture coordinates, one for each camera image. The two images are blended together with a varying weight depending on the view morphing factor c_m. Note that all the images and the mesh are in the rectified coordinate space. We need to set the viewing matrix to the inverse of the rectification matrix to "unrectify" the resulting image to its normal view position. Thus, correct views with desired eye gaze can be synthesized on the graphics hardware in real time.

Regarding the background, it is very difficult to obtain a reliable set of matches since the baseline between the two views is very large, as can be observed in the two original images shown in Fig. 2. In this work, we do not attempt to model the background at all, but we offer two solutions. The first is to treat the background as unstructured, and add image boundary as matches. The result will be ideal if the background has a uniform color; otherwise, it will be fuzzy as shown in the synthesized view shown in Fig. 2. The second solution is to replace the background by anything appropriate. In that case,

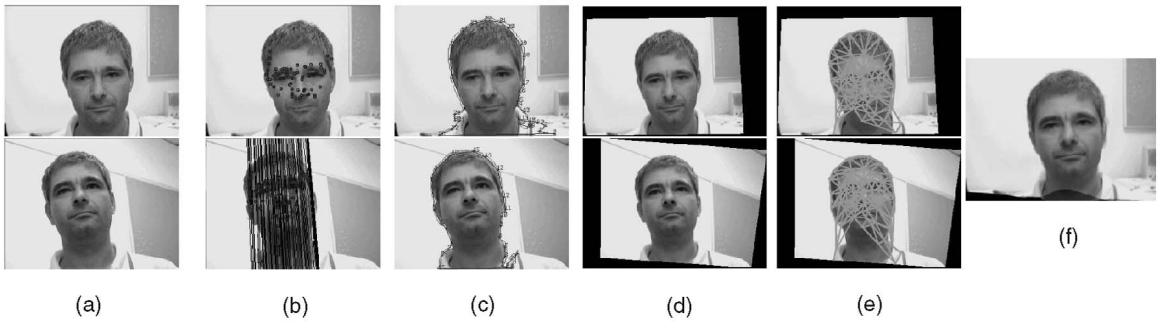

Fig. 4. Intermediate results of our eye-gaze correction algorithm. (a) The input image pair. (b) Tracked feature points with epipolar line superimposed. (c) Extracted foreground contours. (d) Rectified images for stereo matching. (e) Delaunay triangulation over matched points. (f) Final synthesized view (uncropped).

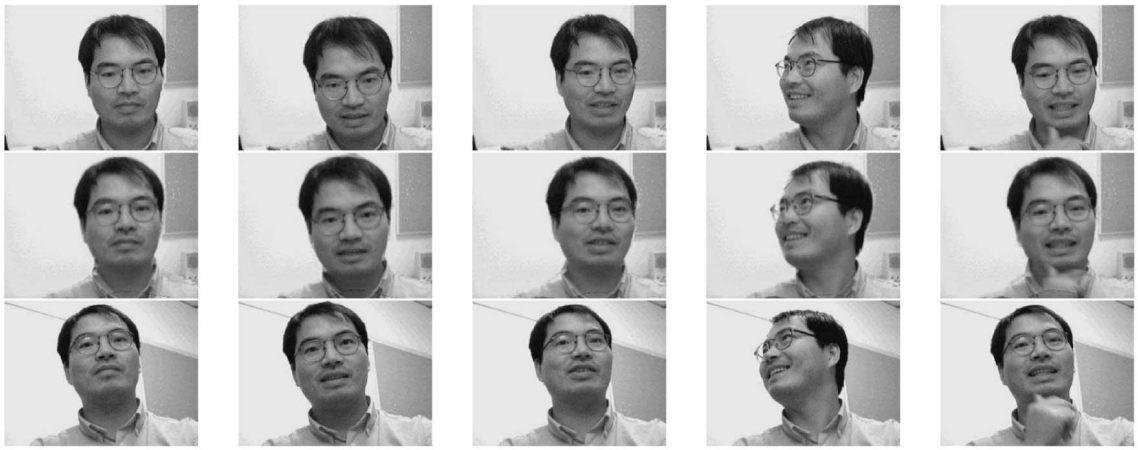

Fig. 5. Sample results from sequence A. The top and bottom rows show the images from the top and bottom cameras. The middle row displays the synthesized images from a virtual camera located in the middle of the real cameras. The frame numbers from left to right are 1, 51, 220, 272, and 1,010.

view synthesis is only performed for the foreground objects. In our implementation, we overlay the synthesized foreground objects on the image from the first camera. The results shown in the following section were produced in this way.

7 Experiment Results

We have implemented our proposed approach and tested with several sets of real data. Very promising results have been obtained. We will first present a set of sample images to further illustrate our algorithm, and then we will show some more results from different test users. For each user, we built a personalized face model using a face modeling tool [8]. This process, which takes only a few minutes and requires no additional hardware, only needs to be done once per user. All the parameters in our algorithm are set to be the same for all the tests.

Fig. 4 shows the intermediate results at various stages of our algorithm. It starts with a pair of stereo images in Fig. 4a; Fig. 4b shows the matched feature points, the epipolar lines of feature points in the first image are drawn in the second image. Fig. 4c shows the extracted foreground contours: the light-colored one (typically a few pixels far away from the "true" contour) is the initial contour after background subtraction while the dark-colored one is the refined contour using the "snake" technique. In Fig. 4d, we show the rectified images for template matching. All the matched points form a mesh using Delaunay triangulation, as shown in Fig. 4e. The last image (Fig. 4f) shows the synthesized virtual view. We can observe that the person appears to look down and up in the two original image, but looks forward in this synthesized view.

During our experiments, we captured several sequences with a resolution of 320×240 at 30 frames per second. Our current implementation runs at four to five frames per second on a 1 GHz PC. The results shown here are computed with our system in a "step-through" mode. Except the manual tracker initialization performed once at the beginning of the test sequences, the results are computed automatically without any human interaction.

Fig. 5 shows some synthesized views from sequence A. Note the large disparity changes between the upper and lower camera images, making direct template-based stereo matching very difficult. However, our model-based system is able to accurately track and synthesize photo-realistic images under the difficult configuration, even with partial occlusions or oblique viewing angles.

Sequence B is even more challenging, containing not only large head motions, but also dramatic facial expression changes and even hand waving. Results from this sequence, shown in Fig. 6, demonstrated that our system is both effective and robust under these difficult conditions. Nonrigid facial deformations, as well as the subject's torso and hands, are not in the face model, yet we are still able to generate seamless and convincing views, thanks to our view matching algorithm that includes a multitude of stereo matching primitives (features, templates, and curves). Templates matching finds matching points, as many as possible, in regions where the face model does not cover, while contour matching preserves the important visual cue of silhouettes.

8 Conclusions

In this paper, we have presented a software scheme for maintaining eye contact during video-teleconferencing. We use model-based stereo tracking and stereo analysis to compute a partial 3D description of the scene. Virtual views that preserve eye contact are then synthesized using graphics hardware. In our system, model-based head tracking and stereo analysis work hand in hand to provide a new level of accuracy, robustness, and versatility that neither of them alone could provide. Experimental results from real sequences have demonstrated the viability and effectiveness of our proposed approach.

Fig. 6. Sample results from sequence B. The upper and lower rows are the original stereo images, while the middle rows are the synthesized ones. The triangular face model is overlayed on the bottom images. From left to right and from top to bottom, the frame numbers are 159, 200, 400, 577, 617, 720, 743, and 830.

While we believe that our proposed eye-gaze correction scheme represents a large step toward a viable video-teleconferencing system for the mass market, there is still plenty of room for improvements, especially in the stereo view matching stage. We have used several matching techniques and prior domain knowledge to find as many good matches as possible, but we have not exhausted all the possibilities. We believe that the silhouettes in the virtual view could be more clear and consistent across frames if we incorporate temporal information for contour matching. Furthermore, there are still salient curve features, such as hairlines and necklines, that sometimes go unmatched. We are investigating a more advanced curve matching technique.

REFERENCES

[1] T.J. Cham, S. Krishnamoorthy, and M. Jones, "Analogous View Transfer for Gaze Correction in Video Sequences," *Proc. Int'l Conf. Automation, Robotics, Control and Vision,* 2002.

[2] D.H. Douglas and T.K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature," *Canadian Cartographer,* vol. 10, no. 2, pp. 112-122, 1973.

[3] J. Gemmell, C.L. Zitnick, T. Kang, K. Toyama, and S. Seitz, "Gaze-Awareness for Videoconferencing: A Software Approach," *IEEE Multimedia,* vol. 7, no. 4, pp. 26-35, Oct. 2000.

[4] M. Jones, "Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes," *Int'l J. Computer Vision,* vol. 29, no. 2, pp. 107-131, Aug. 1998.

[5] M. Kass, A. Witkin, and D. Terzopoulos, "Snake: Active Contour Models," *Int'l J. Computer Vision,* vol. 1, no. 4, pp. 321-331, 1987.

[6] R. Kollarits, C. Woodworth, J. Ribera, and R. Gitlin, "An Eye-Contact Camera/Display System for Videophone Applications Using a Conventional Direct-View LCD," *SID Digest,* 1995.

[7] J. Liu, I. Beldie, and M. Wopking, "A Computational Approach to Establish Eye-Contact in Videocommunication," *Proc. Int'l Workshop Stereoscopic and Three Dimensional Imaging (IWS3DI),* pp. 229-234, 1995.

[8] Z. Liu, Z. Zhang, C. Jacobs, and M. Cohen, "Rapid Modeling of Animated Faces From Video," *J. Visualization and Computer Animation,* vol. 12, no. 4, pp. 227-240, 2001.

[9] C. Loop and Z. Zhang, "Computing Rectifying Homographies for Stereo Vision," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 125-131, June 1999.

[10] L. Mhlbach, B. Kellner, A. Prussog, and G. Romahn, "The Importance of Eye Contact in a Videotelephone Service," *Proc. 11th Int'l Symp. Human Factors in Telecomm.,* 1985.

[11] M. Ott, J. Lewis, and I. Cox, "Teleconferencing Eye Contact Using a Virtual Camera," *Proc. Conf. Human Factors in Computing Systems,* pp. 119-110, 1993.

[12] S. Pollard, J. Porrill, J. Mayhew, and J. Frisby, "Disparity Gradient, Lipschitz Continuity, and Computing Binocular Correspondance," *Robotics Research: Proc. Third Int'l Symp.,* O.D. Faugeras and G. Giralt, eds., vol. 30, pp. 19-26, 1986.

[13] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 593-600, June 1994.

[14] R.R. Stokes, "Human Factors and Appearance Design Considerations of the Mod II PICTUREPHONE Station Set," *IEEE Trans. Comm. Technology,* vol. 17, no. 2, Apr. 1969.

[15] R. Yang and Z. Zhang, "Eye Gaze Correction with Stereovision for Video-Teleconferencing," *Proc. European Conf. Computer Vision,* pp. 479-494, 2002.

[16] R. Yang and Z. Zhang, "Model-Based Head Pose Tracking With Stereovision," *Proc. Fifth IEEE Int'l Conf. Automatic Face and Gesture Recognition,* May 2002.

[17] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 11, pp. 1330-1334, 2000.