Automatic Projector Display Surface Estimation Using Every-Day Imagery

Ruigang Yang and Greg Welch

Department of Computer Science

Univeristy of North Carolina at Chapel Hill

Abstract

Projector-based display systems have been used in computer graphics for about as long as the field has existed. While projector-based systems have many advantages, a significant *disadvantage* is the need to obtain an accurate analytical model of the mechanical setup, including the external parameters of the projectors, and a description of the display surface. We introduce a new method for the latter. Instead of employing some form of *imperceptible structured light* that requires non-trivial infrastructure, we continually observe images of whatever graphical content is being projected, to refine an ongoing estimate for the display surface geometry. In effect we enjoy the benefits of the high signal-to-noise ratio of "structured" light, but do not get to choose the structure. The approach is robust and accurate, and can be realized with commercial off-the-shelf components. And although we do not demonstrate this, it can be extended to include continual estimation of other system parameters that vary over time. The method can be used with a variety of projector-based displays, for scientific visualization, trade shows, entertainment, or the *Office of the Future*.

CR Categories and Subject Descriptors: 1.4.1 [Image Processing and Computer Vision]:Digitization and Image Capture—Imaging geometry; Scanning; 1.4.8 [Image Processing and Computer Vision]: Scene Analysis—Range data; Shape

Additional Keywords and Phrases: Computer Vision, Image Processing, Image-Based Rendering, Shape Recognition

Introduction

Technological and economic improvements are helping to make projector-based display systems increasingly a viable option for applications such as large-scale scientific visualization, simulation, or entertainment. Example systems include the CAVE™ [1], the ReActor Room (and similar systems) by Trimensions, the *Office of the Future* [2], the Princeton Display Wall [3], and the Stanford Information Mural [4]. Beyond permanent fixtures, such display systems are often used for portable visualization, for example at conferences or trade shows. On a much more grand scale, newer and more powerful light projectors are increasing opportunities to turn large physical structures into temporary projector display surfaces. For example, during the millennium celebration in Egypt, the Pyramids were used as display surfaces for dynamic imagery.

While projector-based systems offer many advantages over other display options for many applications, a significant *disadvantage* is the need to obtain an accurate analytical model of the mechanical setup, including the external parameters of the projectors, and a description of the display surface. The problem is that the display surface is not an integral part of a single device, and therefore it must be initially characterized, and periodically monitored.

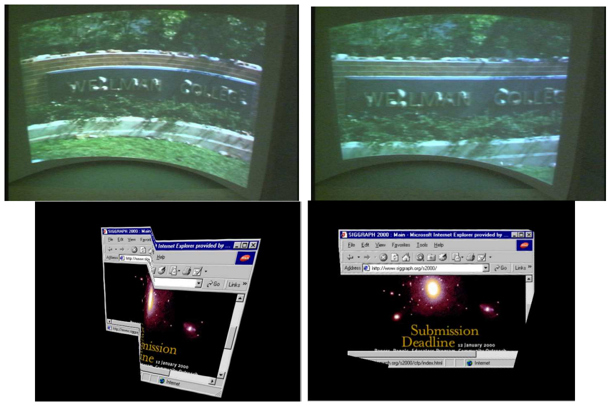

Figure 0. (From left to right, top to down) (a) A image is directly projected on a curved surface; (b) The image is pre-warped based on the display surface estimation. The bending artifact was corrected; (c) A desktop window is severely distorted due to a sharp discontinuity on the display surface; (d) The view

after correction.

We present an iterative approach to automatically determine the display surface geometry, without human intervention, unobtrusively and continuously while the system is being used for real work. We use cameras in a closed-loop fashion to automate the process. Given the physical relationship between projector and a camera, and a crude estimate of the display surface geometry, we iteratively refine the estimate based on image-based correlation between the known projector image, and the observed camera image. Specifically we use a Kalman filter to estimate the length of a (parametric) ray from each projector pixel. The result is a complete 3D description of the surface, allowing one to modify the projected imagery so that it appears correct from any given viewpoint [5]. Some experiments results are shown in Figure 0.

It is interesting to consider the inherent appropriateness of this approach for display surfaces. Typically, finding feature correspondence using correlation techniques is less reliable for images or regions that lack high-frequency content. However, for our particular application, it is OK to miss measurement opportunities in such a region because if there are no problematic features for the system to observe, there are none for the human to observe either. When there are noticeably distorted features, the user will see them, but so will the system, which can then account for them by adjusting the estimate of the display surface. Given a sufficient variation of the projected image contents over time, the system eventually converges on the actual display surface geometry. Because our method is non-intrusive, the calibration process can always been running to maintain an optimal calibration while the system is being used for real work. Our simulation results (described later) predict a high degree of accuracy, and our actual implementation appears to agree.

Our approach has the following key advantages:

- **Self-calibrating**. Once started, no human intervention is needed.
- **Continuous and unobtrusive**. Close-loop continuous calibration that does not affect the projected image quality. When there are visible problems it corrects them, when there are not, it does nothing.
- **Robust**. We use a Kalman Filter (minimal mean stochastic estimator) to optimally weight the measured correlation, with a relatively conservative tuning to reduce the likelihood of a negative impact from a false correlation.
- **Minimal equipment**. No need for high-speed cameras or projectors, or specialized image processing hardware
- **Flexible setup**. The cameras must be rigid but can be located relatively casually with respect to the projectors. The only restriction is that what they cannot “see” they cannot be used to calibrate.
- **Stochastic framework**. Because the framework is in place, other parameters can be added to the list of elements to be estimated. For example, internal projector parameters could be estimated using techniques similar to [6].

Our goal is to improve the setup and maintenance of conventional projector-based display systems, and to further enable the rendering of perspectively corrected imagery on more unusual surfaces [2, 5].

Related Work

We could categorize different calibration methods based on two orthogonal criteria, passive vs. active, and online vs. off-line. Active methods usually emit explicit energy into the display environment to aid in the estimation of the surface properties, while passive methods use only existing energy in the environment, such as light. Off-line calibration methods usually would interfere with the normal operation of the system. The normal operation has to be interrupted if an off-line calibration procedure has to be performed. On-line calibration methods can be used while the system is in normal use. Based on these criteria, existing calibration methods can be categorized in table 1.

	On-line	Off-line
Passive	Stereo	Mechanical alignment
Active	Imperceptible structured light	Laser scan, Structured light

Table 1. Different Calibration Method

Most commercial systems use a passive off-line calibration method. They use precise electromechanical setup to ensure that projectors and display surfaces are complied with the specification, such as perpendicular projection to a planar surface at a given distance. Such setup is usually bulky and expensive, and sometime impossible to implement due to space restriction.

In some setup, the projectors are casually placed. In order to generate perspectively corrected imagery, the exact display surface geometry needs to be acquired. Active off-line method such as Laser range scan or stereo from structured light can be use. The major disadvantage of these methods is that they interfere with the normal operation. If the system needs to be re-calibrated due to various changes, such as drifts or changes in the setup, it has to be shut down first to perform the calibration.

In [2], they proposed a new on-line active calibration method called imperceptible structured light. They use special engineered digital light projector that is able to turn light on and off at a very high rate (over 1000 Hz). This projector projects image bit-plane by bit-plane. Two of the 24 bit-planes are reserved to insert one structured light pattern and its complement. Because the switching is so fast, human eye is unable to distinguish between the bit-plane showing structured light pattern and the next one that shows its complement. So what a human sees is a normal image with slightly extra shade of gray. However, a synchronized camera with a shutter speed faster than one bit plane's duration is able to see only the structured light pattern. With the help of these structured patterns, the display surface geometry can be accurately acquired. Because this method hides the patterns within the normal imagery, it can be used online while people are using the system for every day work. There are two major disadvantage of this method; first, it sacrificed the image quality, only 22 of the 24 bit-planes is used to display the normal image, the two reserved bit-planes adds an extra gray level to the entire image, both the color number and the contrast is reduced; secondly, it requires a special digital light projector that is not available on the market now.

Passive stereo algorithm requires no special hardware and is very easy to implement. However, finding correspondence between two images based on correlation is known to be unreliable. So no commercial system to our knowledge has been using stereo algorithm to calibrate their display sub-system.

The algorithm we proposed in the next section is a passive online method.

Algorithm

We define the display surface as a triangular mesh in the projector space. Each vertex (*Ver*) in the mesh corresponds to a pixel(*Z*) in the projector's image plane. *Z* is called a feature point. Feature points are uniformly distributed over the projector's image plane. The density of feature points depends on the surface continuity and the computational budget. The more irregular the surface is, the higher density is required, at the cost of longer computing time. The center of projection of the projector (O_p) and *Z* defines a projection ray, *Ver* can only move along this projection ray. Figure 1 shows the constraint between different points.

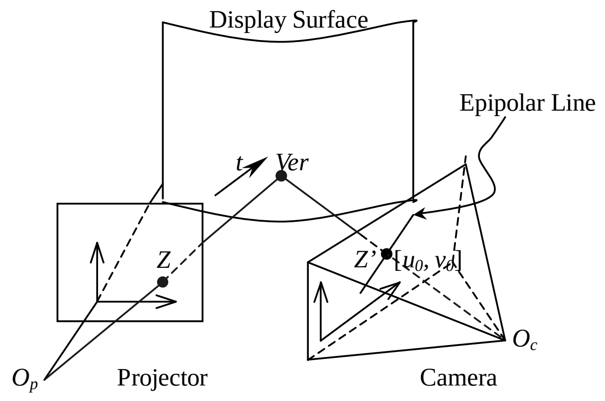

Figure 1. Point *Ver* is a termination point on the projection ray from O_p though *Z*. It is uniquely decided by a parametric value *t*. *Ver*'s projection (*Z'*) on the image plane is limited by the epipolar constraint.

From [7], we know that for a given 3 × 4 projection matrix (M_{proj}), and a feature point *Z* on the image plane, if we rewrite M_{proj} *as* $M_{proj} = [p \quad \tilde{p}]$, where *p* is a 3 × 3 matrix, $\tilde{p}$ is a 3 ×1 vector, *Ver* can be computed by the following formula:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = p^{-1}(-\tilde{p} + t\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ 1 \end{bmatrix}) \qquad \textbf{Equation 1}$$

where *t* is a parametric value, and $Z = [\tilde{u}, \tilde{v}]^T$, its position on the projector's image plane..

When a feature point *Z* is projected on the display surface, its projection *Z'* can be found by a camera. If we know the projection matrix of the camera (M_{cam}), we could solve for *t* using the traditional stereo algorithm. Though stereo algorithm is simple, it only uses a single observation, thus it is very vulnerable to noise or false matches. So instead, we use a Kalman filter, a minimum variance stochastic estimator, to estimate the parametric value *t* iteratively. A basic introduction to the Kalman filter can be found in Chapter 1 of [8] and [9].

With multiple observations over time, the chance of a false match is greatly reduced so the robustness of the algorithm is improved. The Kalman filter also provides a predication of where the *Z* will be projected. We use this predication to search for match in a smaller area while the stereo algorithm usually has to search the entire epipolar line. When an actual match is found, the *difference* between the predication and the actual match was corrected.

In our Kalman filter model, for every *Z*, we have a measurement $[u_0, v_0]^T$ -- its projection on the display surface observed by the camera, we want to solve for *t*, the parametric value that determines the 3D position of *Ver*. Assuming the display surface is static, the followings are our system equations that govern the estimate process. Because the perspective projection is not linear, we have to use Extended Kalman filter (EKF).

$$\begin{aligned} t_{k+1}^{-} &= t_k \\ [u, v, s]^T &= M_{cam}[x \quad y \quad z \quad 1]^T \qquad \textbf{Equation 2} \\ [U, V]^T &= [u_0, v_0]^T + H(T_k - T_k^{-}) \end{aligned}$$

where *H* is the Jacobian matrix of partial derivatives of measurement function with respect to *t*,

$$H = \begin{bmatrix} \frac{\partial(u/s)}{\partial t} & \frac{\partial(v/s)}{\partial t} \end{bmatrix}^T .$$

We denote P_k as the estimated error covariance, R_k is the measurement variance, the time update equations are

$$t^-_{k+1} = t_k$$
$$P^-_{k+1} = P_k + Q$$

Equation 3

And our measurement update equations are

$$K_k = P^-_k H_k (H_k P^-_k H^T_k + \begin{bmatrix} R_k & 0 \\ 0 & R_k \end{bmatrix})^{-1}$$
$$t_k = t^-_k + K_k ([u_0, v_0]^T - [{}^u\!/_s, {}^v\!/_s]^T)$$
$$P_k = (I - K_k H_k) P^-_k$$

Equation 4

After the Kalman filter update, the 3D position of *Ver* is updated with the new t_k using equation 1.

Our algorithm starts with a very rough estimate of the display surface -- every feature point has the same initial *t* set to 0.5, then randomly refines the estimation iteratively. In each iteration, we perform the following operation:

1. Capture of display image and the projected imagery
2. Select a subset of feature points to find their corresponding image in projected image using correlation.
3. Kalman Filter update of these selected feature points.
4. 2D Delaunay Triangulation to update the mesh.

We let this process run as long as the system is turned on. Notice that in the time update equation, we added a very small amount of process variance to compensate for possible drifts in the system. If we set *Q* to zero, the Kalman filter will not update its estimate of the display surface after it has converged. With this added process variance, it is still able to adjust itself, though very slowly, towards changes due to drift or other factors, even after convergence.

Selection of Feature Points

Due to computational constraints, we cannot compute the entire feature set within in one iteration, instead we select a subset of feature points in one iteration. The selection process has two parts, sequential selection, and distance-based selection. In sequential selection, we first define a list of feature points, and then permute the list. In each iteration, a number of consecutive points in the permuted list are selected. So that every point has exactly equal possibility of being updated.

In distance-based selection, we want to identity *possible* outlying points and correct them as soon as possible. We found that outlying points are usually far away from the correct points in 3D. So we use a selection process based on Euclidean distance. We define a maximum neighborhood distance (MND). For every feature point (*Z*) that has been updated, we find its closest neighbor (Z_n) that also has been updated at least once, if the distance between *Z* and Z_n is greater than MND, this *Z* is considered as a point with higher uncertainty and added to the selected point list. One may argue that this distance-based selection imposes an assumption of the display surface geometry -- no two neighbor points can be farther than MND, but in fact, this selection only tries to identify *possible* outlying points. If a point with high uncertainty turns out to be a correct one indeed, it will converge to that position in subsequent updates. In practice, we set the MND to be twice the distance between two neighboring feature points with the initial estimate (t = 0.5). We found this MND works well in practice.

Predicative Pattern Match

Once we have a list of selected feature points, we want to find out its corresponding points in the camera image; With the current parametric value *t* and the estimated error variance P_k, we compute the closest point (Z_{min}) and the furthest point (Z_{max}), where Z_{min} is computed with *t - sqrt(* P_k *)*, and Z_{max} is computed with *t* + *sqrt(* P_k *)*; The Z_{max} and Z_{min} are projected back to the camera image plane. They form a rectangular bounding box on the image plane. One diagonal line of the bounding box is the epipolar line. The search only needs to be performed in the bounding box. The estimated error variance P_k will gradually decrease as the Kalman filter slowly converges. Consequentially, the search area will become smaller and smaller.

For each selected feature point, we use a 16x16 block around it as the correlation template. Matrox Imaging Library (MIL) is used to perform the pattern matching within the specified bounding box in the camera image. It can return a match with subpixel accuracy. In some cases, there will be multiple matches returned by the MIL, all are within the bounding box. In such a case, we computed the mean and the standard deviation of these matches, if the standard deviation is greater than the measurement variance R_k, the entire match set is discarded. Otherwise, their mean is used as the final result.

Since MIL's template matching routine searches the entire area, some time it will return a matching that is not on the epipolar line. This is probably due to two factors; first, the calibration error of the projector and camera's external and internal parameters; secondly, there is electronic noise during the process of digitizing analog videos. If such deviated result is encountered, we compute its distance to the epipolar line (one diagonal line of bounding box); if it is greater than R_k, this match is discarded.

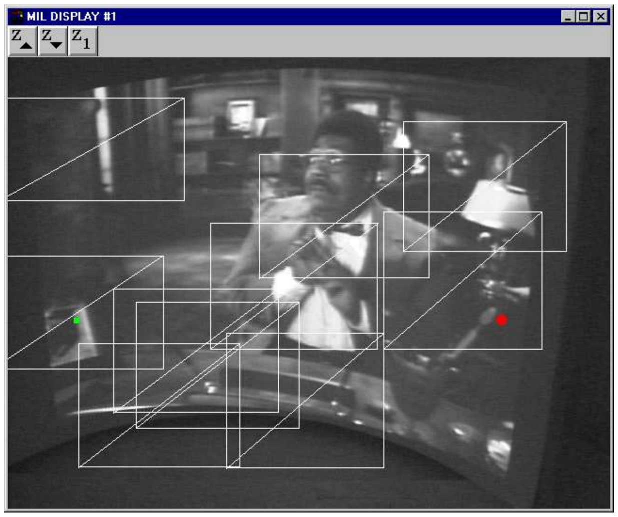

Figure 2. A image captured by the camera. The bounding box is superimposed, its diagonal is the epipolar line. The highlighted green square indicates an accepted match, while the red circle indicates a rejected one because it is too far away from the epipolar line.

In our algorithm, we do not perform the rectification of the camera image. Rectification is widely used in stereo algorithm. It is a two-dimensional transformation that transfers the epipolar lines parallel to the image scan lines. So that the search for correspondence is limited to the scanlines. We do not do so because the number of points we compute in each iteration is at most at the order of ten, rectification would cost more time than the speedup it brings in the search phase. We use MIL to perform template matching, its hierarchy search algorithm is very fast so that there is virtually no time difference whether the search area is a line or a box. Plus, we still check the result returned by MIL to see if it is within the epipolar line with some tolerance.

Rendering Correct Image

We use the technique described in [2] to render perspectively corrected image. To do this, we first need to create a triangular mesh. We first implemented a scan-line based triangulation routine to create a complete mesh and let it deform as its vertices' 3D position is being updated. In our experiments, we found that this approach created some very noticeable distortions if there is a "hole" in the mesh. The "hole" can be defined as one not-yet-updated point surrounded by updated points. So we have to perform triangulation in run-time. Assuming there is no self-intersection of the display surface, the triangulation can be performed in projector's screen space. 2D Delaunay triangulation algorithm is much easier to implement and more robust than its 3D counterpart.

Since rendering is not our primary focus, we did not go to great lengths to achieve fast rendering speed. We just wrote a bare bone OpenGL program that was enough to demonstrate that the surface estimate was correct. This can be seen in video. A faster computer would do it. Further more, we believe that in most applications, the rendering and the capture of depth information is decoupled.

Experiment Results

We implemented our algorithm under Windows NT environment. We initially developed and tested our algorithm in simulation, in which we performed some well-controlled experiments, and then went to a real system. The difference between the simulator and the real setup is that in the real setup, we have to find the external and internal parameters of the camera and the projector. We first present our result in the simulator, then the result in a real setup. To make our result more convincing, in our simulation, we used the real external and internal parameters of the camera and the projector found in the real setup. So all of our experiments have the same setup, the projector is about 1000 mm away from the display surface, and the camera is 600 mm upper right to the projector, pointing at the display surface.

In our simulation, we set Q (process noise) to 1e-10, measurement variance R to 9 (3*3 pixel), and the estimated error covariance P_k to 0.5^2. The density of the feature points is 40 x 30. We performed two experiments, one with a planar display surface with discontinuity, the other with a curved concave surface. During the estimation process, we happened to use a canned image sequenced recorded using a DV camcorder. In practice, this sequence would be the ongoing stream of whatever every-day imagery the user was displaying. We let the system run about 45 minutes in each experiment, the accuracy of our result is shown in Table 2.

	Mean Error(mm)	Max. Error[1](mm)
Planar Surface	2.41	6.78
Curved Surface	1.39	5.23

Table 2. Accuracy of the Simulation

The estimated surfaces are shown in figure 3 and 4 respectively. Figure 0 (c) and (d) are simulated pictures based on the estimate of the planar surface.

[1] In both of our experiments, we found a few outlies (five or six), all of them were on the boundary. They are at least 100 mm away from their neighbors, so they can easily be identified by the distance-based selection routine described earlier. The result shown here does not take into account the points selected by distance-based selection routine.

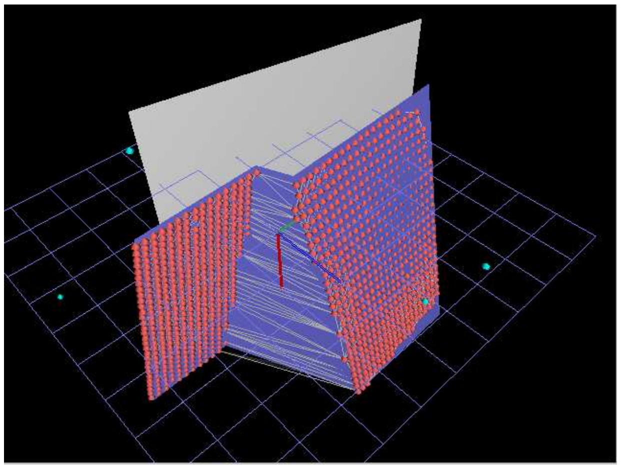

Figure 3. Planar surface simulation. Blue surface is the actual surface; red dots are the estimated feature points. Light blue dots are selected outlying points detected by our distance-based heuristic.

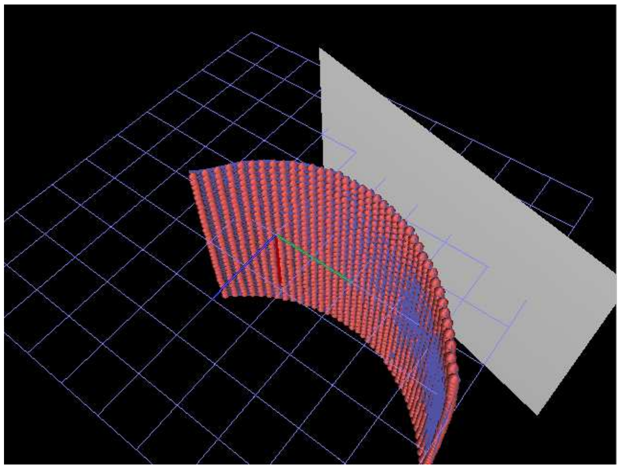

Figure 4. A bird-eye view of the curved surface simulation.

In our real setup, there is no ground truth we can compare our result with. The estimated surface in the real setup is shown in figure 5. Figure 0 (a) and (b) show the difference between an uncorrected view and a corrected view. As they show, the distortion due to non-planar display surface was corrected. More of the results can be found in the video.

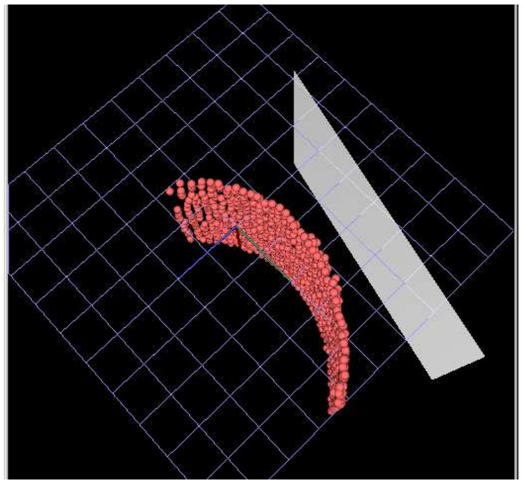

Figure 5. The estimate of a curved display surface after we run our algorithm for over a half hour in a real setup.

Conclusions

Beyond large display systems, we are excited by the growing prospect of graphical imagery displayed on real surfaces around us [2]. We believe that our approach to surface estimation provides an important piece of the puzzle. The approach is accurate, robust, and can be implemented in practice with reasonably common components and minimal infrastructure.

Beyond the algorithmic improvements we present here, we look forward to improved hardware. For example, some day "smart projectors" with built-in cameras will be common, enabling automatic adjustments beyond simple keystone correction. Some day graphics engines will support more efficient rendering onto non-planar (and non-rectangular) surfaces, and maybe will even support automatic view-dependent correction.

References

[1] Curz-Neira, Carolina *et al*., 1993 Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE, *SIGGRAPH Conference Proceedings*, Annual Conference Series, Addison-Wesley, July 1993.

[2] Raskar, R. *et al*., 1998, The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. *SIGGRAPH Conference Proceedings*, Annual Conference Series, Addison-Wesley, July 1998.

[3] Scalable Display Wall. http://www.cs.princeton.edu/omnimedia/

[4] Greg Humphreys and Pat Hanrahan, 1999, A Distributed Graphics System for Large Tiled Display, in the *Proceeding of IEEE Visualization 1999*.Oct. 1999.

[5] R. Raskar, M, Cutts, G. Welch, and W. Stürzlinger, 1999, Efficient Image Generation for Multiprojector and Multisurface Display, in the *Proceedings of the Ninth Eurographics Workshop on Rendering*, (Vienna, Austria), June 1998

[6] Azarbayejani, Ali, and Alex Pentland, Juen 1995, Recursive Estimation of Motion, Structure, and Focal Length. *IEEE Trans Pattern Analysis and Machine Intelligence*, June 1995, 17(6).

[7] O. Faugeras, Three-Dimensional Computer Vision: A Geometric Viewpoint,Cambridge, Massachusetts; MIT Press, 1993.

[8] Maybeck, Peter S. 1979. *Stochastic Models, Estima-tion, and Control*, Volume 1, Academic Press, Inc.

[9] Greg Welch and Gary Bishop, 1997. *An Introduction to Kalman Filter.* Technical Report, TR 95-041, Dept. of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC