Non-Metric Image-Based Rendering for Video Stabilization

Chris Buehler Michael Bosse Leonard McMillan
MIT Laboratory for Computer Science
Cambridge, MA 02139

Abstract

We consider the problem of video stabilization: removing unwanted image perturbations due to unstable camera motions. We approach this problem from an image-based rendering (IBR) standpoint. Given an unstabilized video sequence, the task is to synthesize a new sequence as seen from a stabilized camera trajectory. This task is relatively straightforward if one has a Euclidean reconstruction of the unstabilized camera trajectory and a suitable IBR algorithm.

However, it is often not feasible to obtain a Euclidean reconstruction from an arbitrary video sequence. In light of this problem, we describe IBR techniques for non-metric reconstructions, which are often much easier to obtain since they do not require camera calibration. These rendering techniques are well suited to the video stabilization problem. The key idea behind our techniques is that all measurements are specified in the image space, rather than in the non-metric space.

1. Introduction

In this paper, we consider the problem of video stabilization. We define an unstabilized video to be an image sequence that exhibits unwanted perturbations in the apparent image motion. The goal of video stabilization is to remove these perturbations while preserving the dominant motions in the image sequence.

Most video destabilization is due to physical motions of the camera. Thus, many solutions to the problem involve hardware for damping the motion of the camera, such as a Steadicam rig or gyroscopic stabilizers. This equipment works well in practice, but it is very expensive. Recently, many consumer grade video cameras have been equipped with video stabilization features. However, these methods are often not good enough to stabilize gross motions of the camera.

Because of these reasons, software solutions are attractive. We propose a software video stabilization algorithm based on an image-based rendering (IBR) approach. The basic premise of our approach is simple. Assume that an ideal IBR algorithm exists that can generate perfect novel views from some set of known reference images. Assume further that the unstabilized camera trajectory (i.e., camera positions and orientations) is also known. Then, video stabilization can proceed in two steps:

1. Remove unwanted motions from the known camera trajectory through some sort of filtering or smoothing procedure.

2. Using the ideal IBR algorithm, render a new image sequence along the stabilized camera trajectory.

Unfortunately, the two above assumptions are rather restrictive. First, determining cameras' positions and orientations (i.e., recovering a Euclidean reconstruction of a scene) is a difficult problem, especially for arbitrary video sequences. In light of this, we have tailored our approach to work with non-metric scene reconstructions. By working with non-metric reconstructions, our approach is applicable to uncalibrated video sequences.

Second, there is no ideal IBR algorithm that can generate any desired view. However, there are algorithms that work well when the desired viewpoints are reasonably close to the reference viewpoints, which should be the case in video stabilization.

However, many IBR algorithms assume a known Euclidean reconstruction of the scene, making them not directly applicable to our problem. To develop a non-metric IBR algorithm, we examine three features that we feel a useful IBR algorithm should have. Briefly, these features are correspondence between pixels in the reference images, interpolation between multiple reference images, and virtual navigation in the scene. By implementing these features under non-metric assumptions, we derive a generalized, non-metric IBR algorithm. Our new algorithm is applicable to the video stabilization problem and, of course, other problems as well. The key idea behind our technique is that all measurements are specified in the image space, rather than in the non-metric space.

1.1. Previous Work

In fact, many previous software approaches to video stabilization have been "non-metric" algorithms. That is, they assume little to no knowledge of the actual three-dimensional camera motion, and instead they work to minimize image space motions directly. A common approach is to estimate a dominant planar homography that stabilizes a large planar region in the video [10] or to use simple 2D translations to lock onto an object.

One basic problem with these approaches is that pure image transformations often do not correspond to reasonable camera transformations. For example, stabilizing a video by using pure translation of the video frames is equivalent to varying the camera's principal point, which is unlikely to be the true cause of the destabilization. Homography-based schemes do a better job, but they only stabilize planar scenes or rotational camera motions. Highly non-planar scenes or translational camera motions are not stabilized.

IBR stabilization methods potentially avoid these problem because they can allow for virtual camera navigation in the scene. Thus, the virtual camera can be moved on a smooth path, removing the video destabilization at its source and resulting in a stable video for all image regions. IBR methods can also merge information from multiple video frames to synthesize a completely new view, so they are not limited to simple image warping.

There have been many different proposed IBR algorithms. Many algorithms represent scenes as calibrated images with associated polygons [4] or depth maps [2, 13]. New images are produced by warping the depth map to a new view and (possibly) blending colors from the original images. Some algorithms use dense pixel correspondences rather than explicit depth maps [1]. Other algorithms use much less scene structure in favor of larger numbers of reference images [7, 11]. These "light field" rendering techniques blend multiple reference images together using clever interpolation techniques and consequently require less geometric information.

All of the above IBR algorithms assume a Euclidean reconstruction of the scene. In most cases, the problem of virtual camera navigation (discussed further in Section 2) necessitates a Euclidean assumption. However, some IBR algorithms have been developed to work in a non-metric setting. Good examples of these are view morphing [14], joint view triangulation [12], and earlier work with fundamental matrix representations [6].

2. Three IBR Features

A basic goal of image-based rendering is to generate plausible, novel views of a real scene from a set of known views. A useful IBR algorithm generally has to address three problems: correspondence, interpolation, and navigation. Often, the solutions to these problems assume a Euclidean scene reconstruction.

Correspondence Most IBR algorithms represent correspondence between pixels in the reference images. The correspondence may be explicit, as in the case of flow fields or warping fields, or it may be implicit, as in the case of polygonal geometry or depth maps. In any case, the correspondence defines how pixels move when the virtual camera moves. Fortunately, pixel correspondence is generally not difficult to represent in both metric and non-metric scenes.

Interpolation Many IBR algorithms, especially those that work with multiple reference views, include some type of interpolation or blending of reference views. Often, this interpolation requires a Euclidean reconstruction of the scene. For example, in view-dependent texture mapping [4], images are blended based on the angles between rays. In the light field and lumigraph algorithms, the blending factors are determined by distances measured in two planes defined in the scene. Unfortunately, angles and distances in non-metric spaces are not meaningful.

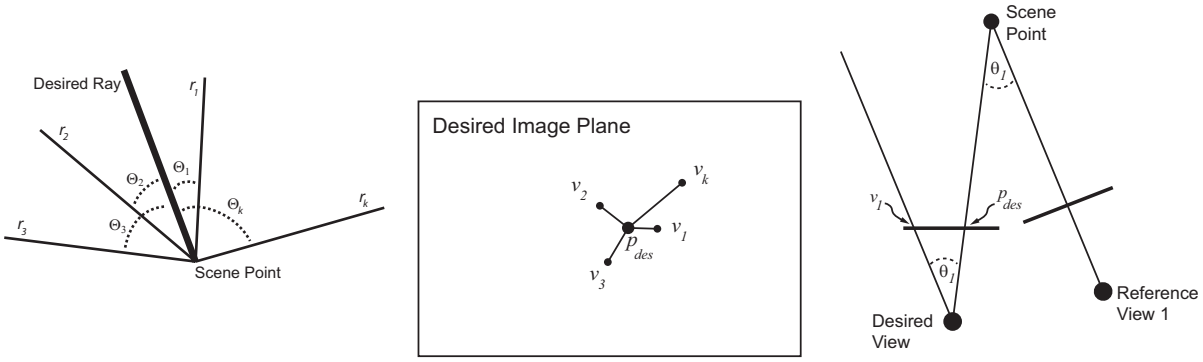
Navigation Generally, an IBR algorithm should allow virtual camera navigation. This requirement is often the most difficult to satisfy in a non-metric setting. In a Euclidean setting, camera positions can be specified, for example, by setting a translation vector, a rotation matrix, and a field-of-view. In a non-metric setting, the unknown intrinsic and extrinsic camera parameters can not be nicely separated in this way. Camera position, orientation, and field-of-view must be specified in a different way, or constrained in such a way as to be realizable.

3. A Non-metric IBR Algorithm

We can obtain a useful non-metric IBR algorithm by modifying an existing IBR algorithm such that correspondence, interpolation, and navigation can be achieved despite a non-metric scene. We choose to modify a recent algorithm called "unstructured lumigraph rendering" (ULR) [3] because it is designed for rendering from dense sets of images that are arranged in an unstructured manner, such as those found in an unstabilized video sequence. The key points of the ULR algorithm are that correspondence is specified by polygonal geometry, navigation is specified by moving a virtual camera in Euclidean space, and interpolation is based on the angles between rays.

We begin by assuming that a projective reconstruction of a scene is known, and that it has been obtained from a set of corresponding point features. We represent the scene as a collection of 3×4 projection matrices P_i and 4×1 structure points M_j. The projective reconstruction specifies only that

$$m_{ij} \doteq P_i M_j,$$



(a) (b) (c)

Figure 1. Color interpolation strategies for image-based rendering. (a) In the ULR algorithm, colors along rays r_i are blended together based on the angular distance from the desired ray. (b) In our algorithm, we propose blending reference colors based on the distance of the vanishing points v_i from the projection of the scene point p_{des}. (c) The vanishing point tells which ray in the desired view is parallel to the reference ray. If the reference ray is parallel to the desired ray (i.e., they are the same ray) then the vanishing point equals p_{des} and the distance is zero.

where m_{ij} is a point feature (represented $(u, v, 1)^T$) in image i, and $\doteq$ denotes equality up to scale. Note that a projective reconstruction is uniquely defined up to an arbitrary projective transformation T. That is, we can transform the projection matrices and the structure points with T and arrive at an equivalent projective reconstruction:

$$m_{ij} \doteq (P_i T^{-1})(T M_j) = P'_i M'_j.$$

This unknown projective transformation makes it impossible to compare angles and distances in the non-metric space. However, we can see that the *projections* of quantities into the image space are unaffected by T. Thus, we design our IBR algorithm to operate in image space whenever possible.

Correspondence The ULR algorithm uses correspondence based on polygons, which are simply piecewise planar patches. We also define correspondence in our algorithm using piecewise planar patches. Given some desired view with projection matrix P_{des}, we project the structure points M_j into that view. We then Delaunay triangulate these image points, which results in a tessellation of the image plane. For each triangle T_k in the tessellation, we compute the plane Π_k that passes through the three structure points defining the vertices of the triangles. Using Π_k, we compute the planar homography H_{ki} that maps pixels in the desired view to pixels in reference view i:

$$H_{ki} \doteq P_i P_k^+,$$

where P_k^+ is a 4×3 inverse projection matrix that maps pixels in the desired view onto the plane Π_k [5]. The homographies H_{ki} establish a correspondence between pixels in the desired view and the reference views. An example tessellation is shown in Figure 2.

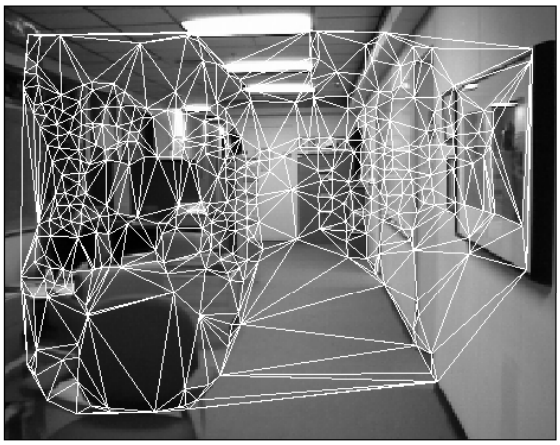


Figure 2. An example triangular tessellation that we use for pixel correspondence.

Interpolation For each individual pixel in the desired view, the ULR algorithm blends between corresponding pixels from multiple reference views. Given a set of corresponding pixels p_i in the reference views, the final color of the pixel p_{des} in the desired view is computed as a weighted average of these reference pixel colors. The colors are inversely weighted based on the angular difference between the reference viewing rays and the desired viewing ray (see Figure 1a). In fact, the actual angles are not important; rather, it is the relationships between angles (e.g., the ordering of angles) that matter most. We exploit this fact in our algorithm.

Consider measuring relative angle sizes using the following metric defined in image space. We compute the vanishing points v_i of all the corresponding reference rays as

seen in the desired view. The distance $d_i = \|v_i - p_{des}\|$ is then a measure of the angular distance between the desired viewing ray and the reference viewing ray (see Figure 1b). This measure behaves enough like the true angle (i.e., smaller angles have smaller measures) to compute interpolation weights. The diagram in Figure 1c demonstrates how this construction works.

Of course, computing vanishing points requires knowledge of the plane at infinity Π_∞, which is unknown in a projective reconstruction. If we could compute the plane at infinity (i.e., upgrade the reconstruction from projective to affine) then we could use vanishing points to measure relative angle magnitudes. In fact, is is not critical to have the true plane at infinity to use this angle measure. In developing our algorithm, we have found that it suffices to use a plane that satisfies cheirality constraints [8] to approximate plane at infinity. Using such a plane, which we call $\tilde{\Pi}_\infty$, results in a quasi-affine reconstruction of the scene [8].

Navigation Navigation is perhaps the most difficult IBR problem when dealing with non-metric spaces. In a projective setting, the projection matrix of a novel view can be computed by specifying the desired image locations of the structure points. However, this method of navigation is not very intuitive and may lead to non-rigid camera motions or other improbable motions. Nevertheless, we have found that this type of navigation is, in fact, well-suited to video stabilization. First, we smooth the observed feature motion in the original unstabilized image sequences. Then, we compute a sequence of stabilized projection matrices that best reproduce the smoothed features. We discuss the details of this feature-smoothing procedure in the following section.

4. Stabilizing Video

The first step in our video stabilization procedure is to obtain a projective reconstruction of the video sequence. In our implementation, we compute an initial solution using a projective factorization technique [16]. Then we refine the solution using standard robustified bundle adjustment techniques [17]. For long image sequences, we break the solution up into overlapping sub-sequences of frames and compute independent solutions for each sub-sequence. We then compute pairwise projective transformations to map all of the independent solutions into a common projective frame. While this method does not always give a globally consistent solution, it is sufficient for the video stabilization task.

Next, we upgrade the reconstruction to a quasi-affine reconstruction by approximating a plane at infinity. We have found that the procedure detailed in [9] provides a suitable plane for this purpose. Given this plane $\tilde{\Pi}_\infty$, we transform our projection matrices and structure points with the corresponding 4×4 projective transformation to convert to a quasi-affine reconstruction. Once we do this, we can decompose our projection matrices into two parts: $P_i = (P_{i,3\times3} \mid p_i)$. From this, we can compute the approximate infinity homography relating any two images as

$$\tilde{H}_{i,j,\infty} \doteq P_{j,3\times3} P_{i,3\times3}^{-1}.$$

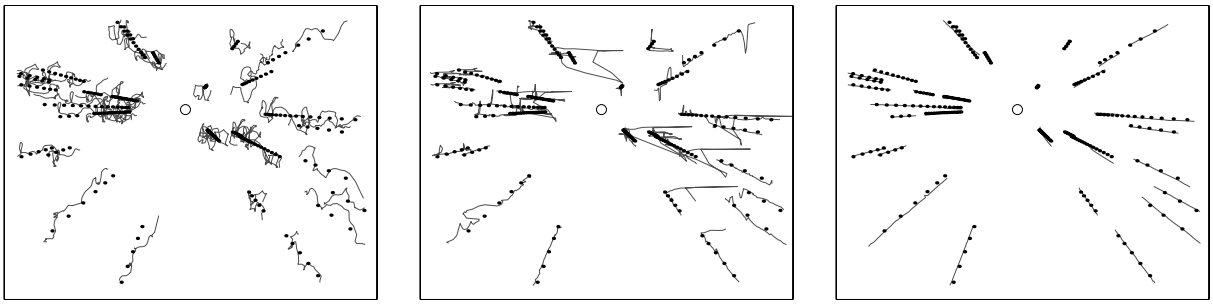
This homography is used for computing approximate vanishing points.

4.1. Feature Smoothing

Once we have a quasi-affine reconstruction of the original video sequence, we then proceed to compute a sequence of stabilized projection matrices P_i'. We do this by first computing target locations for the features in the stabilized sequence. Then we compute the stabilized projection matrices by running a nonlinear optimization of the original unstabilized projection matrices that minimizes the reprojection error of the structure points from the desired feature locations. This second optimization runs efficiently since we hold the structure points constant and allow only the projection matrices to change.

Computing the stabilized target feature locations is the most difficult step. One way to stabilize the locations of the features is to smooth out the entire track of the feature, using a spline, and then remap the original features to points on the new track. This technique works well for smoothing general motion with few assumptions. However, we have found it advantageous when we can apply a prior model to the motion of the desired feature locations. Using a motion model helps insure that the new projection matrices correspond to rigid motions. Some possible motion models include linear translation, fronto-parallel translation, circular motion about a fixed target, and simultaneously moving and zooming the camera while keeping the apparent size of a scene object constant. We describe the linear motion model with constant velocity in detail: the stabilized cameras should move at a constant velocity along a straight line while looking in the same direction. In this case, it is well-known that all image features move radially to (or from) a point called the focus of expansion.

To use a linear motion model, we first estimate (or specify by hand) the focus of expansion, which may be in or out of the field of view, and fit radial lines to the initial feature tracks. Next we map the unstabilized features to points on the radial lines. Often, the original features do not map well onto the lines (see Figure 3a), but in any case we just map the original features to the closest point on the line. Next, we redistribute these points along the length of the line according to a constant velocity model: points should move with time toward (or away from) the focus of expansion. More specifically, the projection of a point moving



(a) (b) (c)

Figure 3. Three iterations of our feature smoothing procedure. The initial feature tracks are drawn as solid lines, and the desired radial features are shown as dots. The desired focus of expansion is marked with a circle. (a) Before optimization. (b) After one iteration. (c) After all iterations.

with constant velocity should fit a log function of the form $a \log |t - t_0| + b$, where t is a time index and a, b, and t_0 are unknown parameters that we optimize for each point.

After we determine the desired feature locations, we run the optimization to compute new projection matrices. The solution converges with only one or two iterations of refitting the target features and optimizing the projection matrices (see Figure 3b,c), and the projection matrices then produce a stabilized video sequence when used with our IBR algorithm.

4.2. Rendering

As the final step, we render the new image sequence corresponding to the new projection matrix sequence P'_i. We have implemented our non-metric IBR algorithm off-line in a ray-tracing fashion. For each desired view, the renderer first computes the triangular tessellation of the visible structure points.[1] Then, for each pixel in the desired view, the renderer computes the corresponding pixels in the reference views using the planar homographies H_{ki}. Each reference pixel is assigned a weight according to the vanishing point distances, which are computed using the homographies $\tilde{H}_{i,j,\infty}$. To compute these weights, we follow the approach in [3]. Specifically, we take the k closest reference pixels (in terms of vanishing point distance) and linearly convert those distances into weights such that the largest distance is equal to zero: $w_i = 1 - d_i/d_k$, where $1 \leq i \leq k$, and the distances increase with i. We then normalize the w_i such that they all add to one. The output color of the pixel is taken to be the weighted sum of the reference pixel colors. We typically use $k = 4$ reference images at each pixel, and we use a maximum of 11 reference images for each desired image.

[1] A structure point is deemed visible in a desired view if it is visible in the corresponding original view.

5. Results

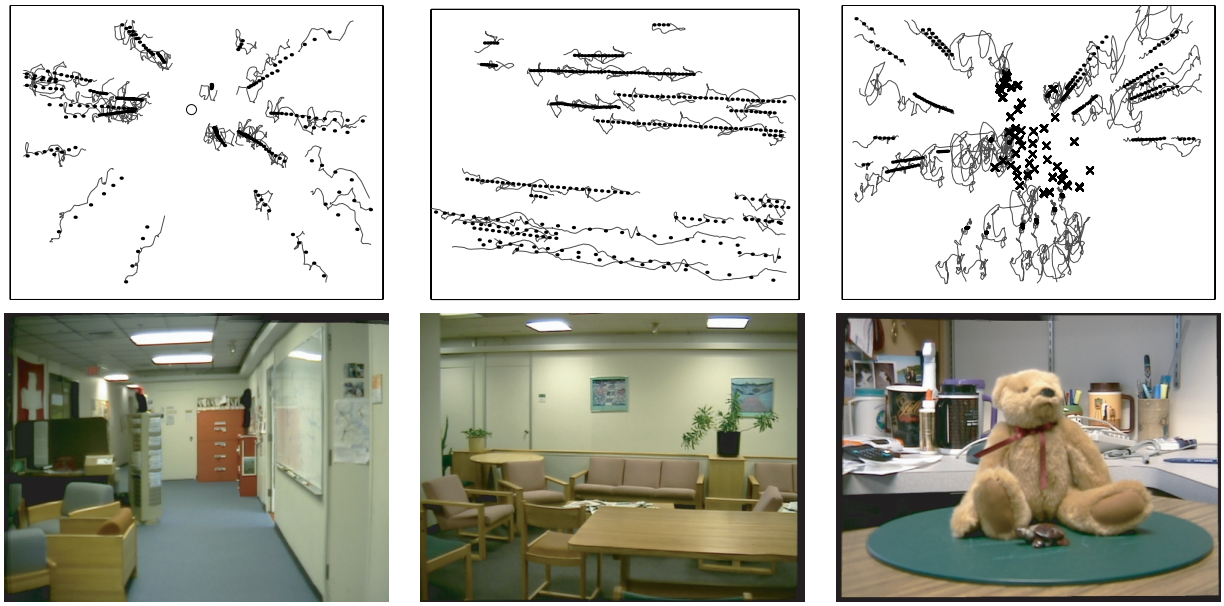
We have tested our video stabilization technique on a variety of video sequences. All of our video sequences are acquired with a hand-held video camera that has had its own video stabilization features disabled. All other automatic features of the camera have been enabled. In the first two examples we use a constant velocity linear motion model with the focus of expansion in and out of the field of view, respectively. In the third example we use a simultaneous moving and zooming motion model: the camera moves forward while zooming out to keep the size of the foreground object constant. This motion imitates a cinematographic effect made popular by Alfred Hitchcock. Note that both the time varying intrinsic (focal length) and the extrinsic camera parameters are smoothed.

In Figure 4 we have shown plots of feature motion as they evolve over time. The tracks drawn with solid lines are the original features that were tracked with automatic feature tracking software [15]. The tracks drawn in dotted lines are the target features as seen from the stabilized camera trajectory. The images below the track plots are representative frames from the stabilized sequence. Note that the track plots depict point features tracked over time and not linear features in a single image.

6. Conclusions and Future Work

In this paper, we have demonstrated a technique for stabilizing video sequences using non-metric image-based rendering techniques. We have suggested that useful IBR algorithms for this purpose should be able to do three things: correspond points between reference images, interpolate between reference images, and navigate a virtual camera in the scene. In fact, these three properties are generally useful in most IBR applications.

Unfortunately, most IBR algorithms handle these three issues using the assumption that a Euclidean reconstruc-



(a) (b) (c)

Figure 4. Smoothed feature tracks and rendered frames from our algorithm. Original features are solid, and stabilized features are dotted. (a) The motion is forward, so the focus of expansion, marked with a circle, is in the image. (b) The motion is mostly horizontal, so the focus of expansion is to the left. (c) This sequence varies the camera focal length to simulate a "Hitchcock" zoom. The sequence is stabilized with the motion model that the bear (marked with X's) remains fixed and that the background moves radially.

tion of the scene is available. Our algorithm requires only a quasi-affine reconstruction, which is a projective reconstruction that has been upgraded with a plane that satisfies cheirality constraints. Empirical results suggest that the particular choice of the plane does not greatly impact the appearance of the output images.

One main drawback of our approach is the difficulty of fitting motion models to complex motions. We have demonstrated results with two simple motion models, but further research is needed into more complex models and how to fit them to observed data. One approach might be to express a complex motion as a superposition of simpler motions.

Also, since our algorithm blends together multiple reference images, ghosting or double images may appear if there are fast moving objects in the original sequence. A more sophisticated algorithm for detecting moving objects and interpolating them differently could reduce these artifacts.

References

[1] Avidan, S. and Shashua, A., "Novel view synthesis in tensor space," *CVPR '97*, 1997.

[2] Chen, S. E. and Williams, L., "View Interpolation for Image Synthesis," *SIGGRAPH '93*, pp. 279-288, 1993.

[3] Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M., "Unstructured Lumigraph Rendering." *SIGGRAPH 2001*, pp. , 2001.

[4] Debevec, P., Taylor, C., Malik. J., "Modeling and Rendering Architecture from Photographs," *SIGGRAPH '96*, pp. 11-20, 1996.

[5] Faugeras, O., Luong, Q. T., and Papadopoulo, T., *The Geometry of Multiple Images* MIT Press, 2001.

[6] Faugeras, O. and Laveau, S., "Representing Three-Dimensional Data as a Collection of Images and Fundamental Matrices for Image Synthesis," *Proceedings of ICPR94*, pp. 689-691, 1994.

[7] Gortler, S., Grzeszczuk, R., Szeliski, R., and Cohen, M., "The Lumigraph, " *SIGGRAPH '96*, pp.43-54, 1996.

[8] Hartley, R., "Cheirality," *ICCV '98*, 1998.

[9] Hartley, R., Hayman, E., de Agapito, L., and Reid, I., "Camera Calibration and the Search for Infinity," *ICCV '99*, 1999.

[10] Irani, M., Rousso, B., and Peleg, S., "Recovery of Ego-Motion Using Image Stabilization," *CVPR '94*, pp. 454-460, 1994.

[11] Levoy, M. and Hanrahan, P., "Light Field Rendering," *SIGGRAPH '96*, pp. 31-42, 1996.

[12] Lhuillier, M. and Quan, L., "Image Interpolation by Joint View Triangulation," *CVPR '99*, 1999.

[13] McMillan, L. and Bishop, G., "Plenoptic Modeling: An Image-Based Rendering System," *SIGGRAPH '95*, pp. 39-46, 1995.

[14] Seitz, S. and Dyer, C., "Physically-Valid View Synthesis by Image Interpolation," *Workshop on Representation of Visual Scenes*, 1995.

[15] Tomasi, C. and Shi, J., "Good Features to Track," *CVPR '94*, pp. 593-600, 1994.

[16] Triggs, B., "Factorization Methods for Projective Structure and Motion," *CVPR '96*, 1996.

[17] Triggs, B., McLauchlan, P., Hartley, R. and Fitzgibbon, A., "Bundle Adjustment — A Modern Synthesis", *Vision Algorithms: Theory and Practice*, Corfu, Greece, pp.298-373, 1999.