Multi-view Spatial and Temporal Interpolation for Dynamic Event Visualization

Sundar Vedula

CMU-RI-TR-99-15

Thesis Proposal

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

June 1999

Abstract

The problem of visualizing dynamic real-world events on a digital computer has been a challenging one so far. We present a framework and algorithm to model and visualize these events by creating virtual images from synthetic cameras located at arbitrary viewpoints, at arbitrary time instants. We discuss Geometry, Radiance, and Motion Map (GRAMM) as an image based representation of shape, texture, and instantaneous motion of a scene. A GRAMM allows us to create virtual images, by interpolating between sets of images taken at finitely sampled positions in location and in time. This interpolation requires algorithms for estimation of 3-D scene motion, image based interpolation, and imposition of multi-camera geometric constraints. Applications include visualization of complex surgical procedures, entertainment, or training, where the physical locations and/or image capture rates of cameras are constrained, but there is a need to accurately reconstruct the appearance from any position, at any time instant.

Contents

1 Introduction

Events occurring in the world around us are dynamic. Reconstructing and visualizing a general, dynamically changing event in a computer is a challenging task, mainly because of the complexity of the geometry, large number of degrees of freedom, and lack of obvious representations.

A first step towards visualization of such a changing scene is the ability to recreate what the scene would have looked like, from an arbitrary viewpoint, at an arbitrary time during its occurrence. With this capability, one can imagine a user at a desktop computer adjusting the position and orientation of a *virtual* camera, and moving a slider representing time in the event. The user can then look at what the event would have appeared like, had there really been a camera capturing the action at that particular place and time. Such a tool would be invaluable as a visualization aid, because humans are extremely good at perceiving the underlying representation of a scene from a single or small number of such images.

This capability potentially has a large number of applications. Imagine being able to fly around a basketball court, watching the game from any angle you choose. The ability to change one's viewpoint at will adds a whole dimensionality to the immersive effect. Similarly, virtual renderings can be used in training and simulation applications, or even as dynamic scene content for video games. Movie production would be much easier, as camera angles for shots can be decided after rather than during recording.

1.1 Problem Definition

We are interested in physical events that can be modeled as general time-varying scenes. For particular domain applications, it is sometimes possible to make simplifying assumptions about the scene, such as conformity to a parametrized or articulated model. Apart from physical limitations on workspace, we make no use of domain knowledge - for purposes of modeling, the event itself is equivalent to a scene with a few free-form surfaces arbitrarily changing shape and moving in time. In addition, we wish to avoid interfering with the event; hence only video cameras, rather than any active imaging method are used to capture the action.

Therefore, using a set of video sequences from different positions, our goal is to be able to visualize a dynamic event by accurately reconstructing its appearance from an arbitrary position, at any time. Figure 1 shows an example of dynamic event with, a player bouncing a basketball. Images captured at different points in time from the same viewpoint are aligned vertically, while those captured at the same time from different positions are aligned horizontally. Camera position is shown along one dimension for simplicity - it is actually a multi-dimensional parameter, with as many degrees of freedom as a camera's intrinsic and extrinsic calibration parameters.

Clearly, reconstructing the appearance from positions other than that of the input cameras requires spatial interpolation between images. Similarly, view synthesis for a time at which no real images were taken involves temporal interpolation across images captured at

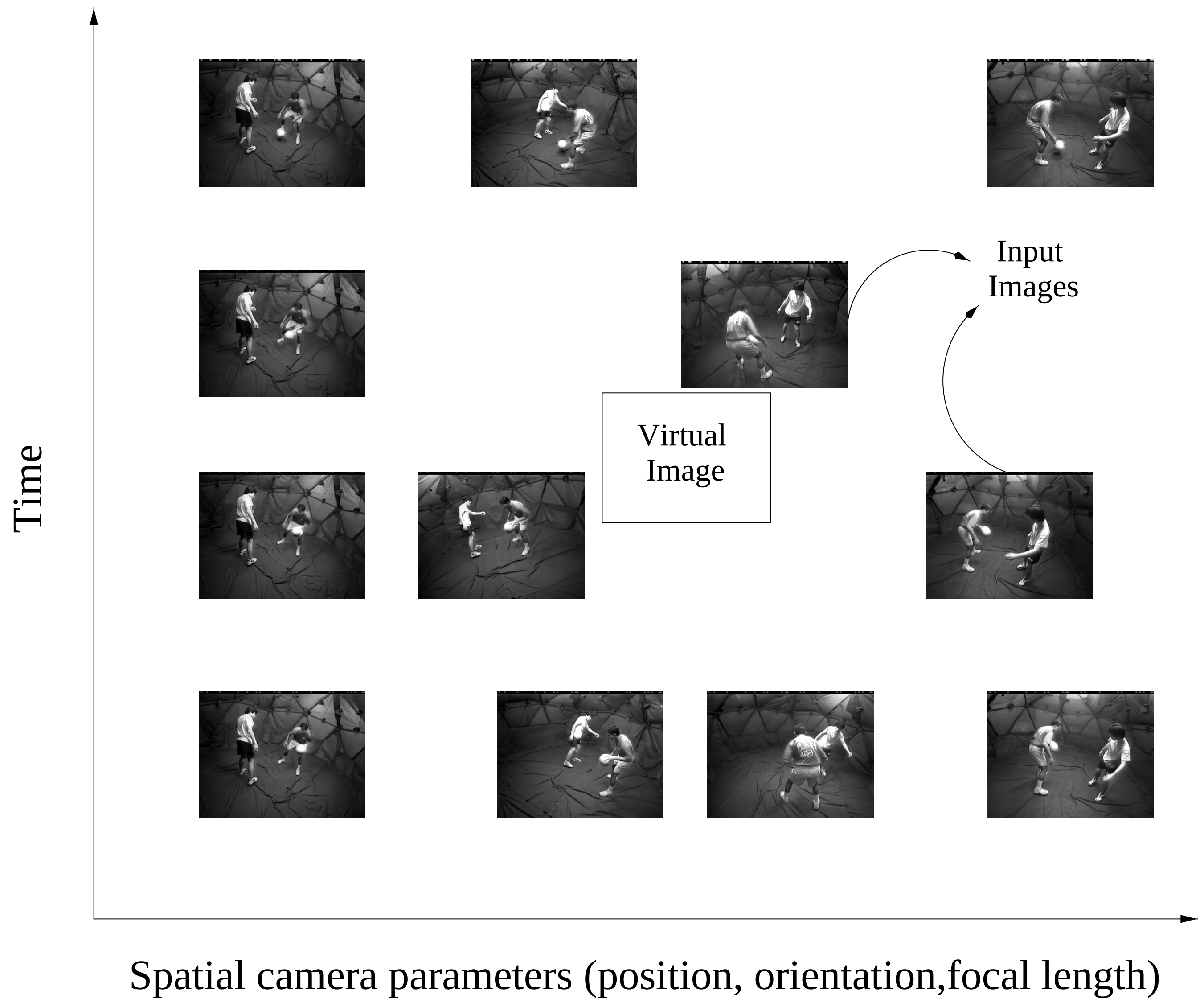

Time
Input Images
Virtual Image
Spatial camera parameters (position, orientation,focal length)

Figure 1: View interpolation in the spatial and temporal domains. Shown are sampled images, which are taken from synchronized video sequences, captured from various angles. While time is one dimensional, the spatial parameters determining each image are position, orientation, focal length, and other

different instants in time. Thus, our problem can be summarized as:

Reconstruction of the appearance of a dynamically changing physical event at any time instant during its occurrence, from an arbitrary point of view by interpolating across a set of images captured from different positions, at different times.

Reconstruction of appearance involves four main components: accurately estimating the geometrical relationship between cameras and the scene and between themselves, estimation of scene structure, estimation of scene motion, and interpolation in space and time for creation of the virtual image.

Most existing algorithms focus on recovering scene structure at a single time instant, often in a human-assisted system, and then using image based rendering methods to reconstruct a new view. However, our interest is in modeling the spatial and temporal appearance variation of a non-parametric, general dynamic event. The fact that human intervention on a large amount of data is impossible, and that we deal with a multi-camera system capturing images of a large workspace makes the problem more challenging.

Image based interpolation requires point to point correspondences between sets of images. Stereo and Optical flow are two sets of techniques for establishing correspondences across pairs of images separated in space, and time respectively. There is a need for a single, consistent framework in which the relationship between these two sets of information can be combined, for a general non-rigid scene.

Once the relationship between correspondences is known, there still remains the task of creating the virtual image, given the spatial relationships between them (image correspondences at the same time instant), and a measure of scene motion (scene flow). While current image based rendering algorithms partly address the view interpolation problem at a particular time instant, they lack several necessary properties demanded by an application seeking smooth variation of image appearance.

1.2 Summary of Proposed Approach

Spatial correspondences between images captured at the same time instant are computed by multi-baseline stereo. These stereo depth maps are then volumetrically merged, and then re-projected back onto the image plane to give accurate depth maps for each camera. We propose a new representation, *Geometry, Radiance, and Motion Map (GRAMM)* to store the depth maps and captured images. In addition, we propose an algorithm for *Scene Flow*, to compute the instantaneous motion of every point in the GRAMM (each point comes from one spatial correspondence). Thus, we have a motion field in 3-D that allows us to temporally interpolate all points in the GRAMM. We then show how this GRAMM is used to visualize the scene from any arbitrary virtual camera position, with this reconstruction having certain desirable properties.

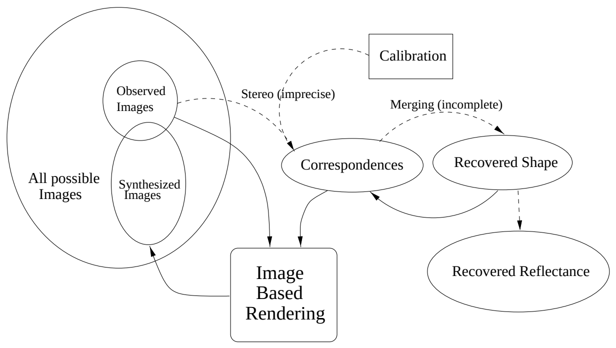


Figure 2: The Image reconstruction pipeline for a fixed time instant

1.3 Related work

Most work to date has focussed on two separate sub-problems - Image Based Modeling and Rendering of static scenes, and motion analysis and structure estimation using motion-stereo. Multi-camera calibration consistency is an issue that has almost never been addressed. This is mainly due to the fact that a single camera or a binocular stereo head usually suffices for problems where either the motion of the camera or that of the scene is predominant. Figure 2 summarizes the relationship between observed and reconstructed images, scene structure, and calibration.

Image Based Rendering methods:

- View Interpolation [Chen and Williams1993] : Generate new views between two images by linearly interpolating correspondences. Does not produce physically correct views under perspective projection.
- View Morphing [Seitz and Dyer1996]: limited to two cameras, and the intermediate view position is limited to the line segment connecting the two optical centers.
- Trilinear tensor based interpolation [Avidan *et al.*1997]: used to interpolate in a projective framework between two cameras, by constructing the trilinear tensor between two cameras, and the virtual camera.
- Scene Representation as a Collection of Images and Fundamental Matrices [Laveau and Faugeras1994a] : New view synthesis is by reverse mapping, requiring a search along the epipolar line. New view needs to be specified by choosing the positions of image control points, which makes camera control somewhat awkward.
- Layered Depth Images [Gortler *et al.*1997] : Multiple intensity images are combined into a single image coordinate system, and stored with varying depths as layers. This allows McMillan's sorting algorithm to be used. Unfortunately, pixels are resampled

twice from input to output (same-view-same-image is not guaranteed), and range of motion is limited to within frustum of LDI camera.

- View-Dependent texture mapping [Debevec *et al.*1996] : Uses 3-D model obtained from model-based stereo, rather than a set of correspondences. Also, visibility can be precomputed for 3 nearest cameras, and texture mapping hardware can used for fast rendering.
- Lightfield rendering [Levoy and Hanrahan1996] and Lumigraph [Gortler *et al.*1996]: Store just a sample of the plenoptic function, by encoding appearance from a large number of input views. New views are constructed by just interpolating in ray space, without any knowledge of the underlying geometry.
- Eigen-texture method [Nishino *et al.*1998]: Use eigenspace compression on input images aligned to the recovered 3-D model. Eigenspace allows a high compression ratio since appearance changes are limited to brightness variation due to illumination only. In addition, interpolation can be done in eigenspace too.
- Object Shape and Reflectance Modeling from Observation [Sato *et al.*1997]: Object surface shape is first reconstructed from multiple laser-scanned range images. The diffuse and specular reflection parameters are then recovered, thus enabling new view synthesis using standard graphics hardware.

Motion Estimation methods:

There is a large body of work in the motion literature, that deals with some form of estimation of scene motion.

A common approach to recover three-dimensional scene motion from multiple cameras is to combine recovery of motion and stereo in an approach known as motion-stereo [Waxman and Duncan1986]. However, most motion-stereo algorithms assume that the scene is rigid. [Liao *et al.*1997] is an example of motion-stereo that considers non-rigid motion. It uses a relaxation-based algorithm to co-operatively match features in both the temporal and spatial domains, and therefore does not provide dense motion.

Another common approach is to make *a priori* assumptions about the scene, as either a deformable model [Metaxas and Terzopoulos1993] or the assumption that the motion minimizes the deviation from a rigid body motion [Ullman1984]. With such assumptions, recovery of three-dimensional non-rigid motion from a monocular view is possible. [Penna1994] is a recent survey of monocular non-rigid motion estimation, and the assumptions used to compute it.

Timestamp	Location	Radiance	Motion	Sampled direction	(Surface normal)
t	(x_1, y_1, z_1)	(r_1, g_1, b_1)	$(\frac{dx_1}{dt}, \frac{dy_1}{dt}, \frac{dz_1}{dt})$	(u_1, v_1)	$(n_{x_1}, n_{y_1}, n_{z_1})$
t	(x_2, y_2, z_2)	(r_2, g_2, b_2)	$(\frac{dx_2}{dt}, \frac{dy_2}{dt}, \frac{dz_2}{dt})$	(u_2, v_2)	$(n_{x_2}, n_{y_2}, n_{z_2})$
$t + k_1$	(x_3, y_3, z_3)	(r_3, g_3, b_3)	$(\frac{dx_3}{dt}, \frac{dy_3}{dt}, \frac{dz_3}{dt})$	(u_3, v_3)	$(n_{x_3}, n_{y_3}, n_{z_3})$
.					
.					
.					
$t + k_T$	(x_N, y_N, z_N)	(r_N, g_N, b_N)	$(\frac{dx_N}{dt}, \frac{dy_N}{dt}, \frac{dz_N}{dt})$	(u_N, v_N)	$(n_{x_N}, n_{y_N}, n_{z_N})$

Table 1: A GRAMM holds information on the geometry of a set of points, their radiances as sampled in a particular direction, their instantaneous motion, and optionally, the surface normal at each point. N is the total number of points, and T the total number of time instants.

2 Geometry, Radiance and Motion Map (GRAMM)

We define a GRAMM as a unified collection of scene points and their geometric, visual, and motion properties. For each point in the GRAMM, information on its location, sampling direction, instantaneous motion, and color is stored. Each element of the GRAMM can be thought of as a sample of the *Plenoptic function*[Adelson and Bergen1991], which refers to the pencil of rays visible at any point in space, at any time.

At any time instant, consider a small surface patch S_P. Let G be the set of all points in the GRAMM, and $P \subset G$ be the subset of points in the GRAMM that lie on or close to the surface patch S_P. Now, the radiances of each of these points (which are sampled in different directions since they come from different cameras) are samples of the plenoptic function at S_P. The radiance of each point in a particular direction is the value of the pixel in the corresponding image, assuming that image irradiance equals scene radiance [Horn1986]. Each of these points is stored as a separate entry in the GRAMM. Table 2 shows a sample GRAMM. It is also possible to store multiple radiances for each point to store the plenoptic function explicitly - but we choose to avoid that, to avoid resampling of the original data.

2.1 Creation of a GRAMM

Each point in the GRAMM is contributed by one correspondence match. A correspondence map between two cameras is equivalent to a depth map when the cameras are fully calibrated. The GRAMM is filled by just projecting pixels from depth maps of all cameras into the same world-coordinate system. Figure 3 shows an example of a GRAMM point coming from a two-camera stereo pair. Unfortunately, this creates two problems: First, the number of points becomes very large. As an example, data from 50 images, each with 640x480 pixels creates 15 million points, for just one time instant. Secondly, the results of simple stereo matching are usually very noisy, with a large number of outliers due to incorrect matches.

A possible solution to the problem of noisy matches is to use an algorithm such as Voxel

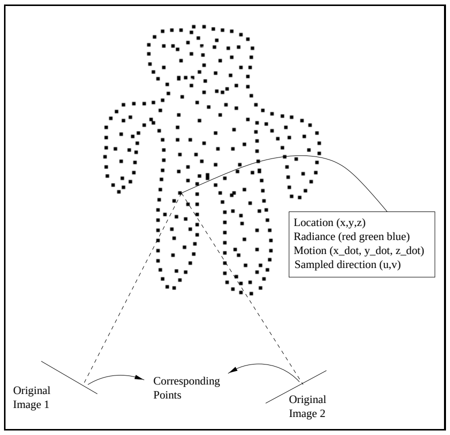


Figure 3: A sample GRAMM at one time instant. A correspondence between two calibrated cameras gives one GRAMM point. For each such point, information on its geometry, motion, and radiance are stored.

Coloring [Seitz and Dyer1997]. Voxel coloring determines occupancy of the scene in volumetric space, by ensuring consistency among all images that view a particular voxel. The center of each occupied voxel can be used as a point on the GRAMM. Another alternative is to use a space-sweep algorithm [Collins1996], where image features are back-projected onto a plane sweeping through space, as the scene is reconstructed. As a means of enforcing multi-image matching to remove outliers that individual stereo correspondence maps produce, we use multi-baseline stereo and volumetric merging [Rander *et al.*1996] to fuse the range maps into a noise free volumetric model. This recovered shape is then projected back into each of the cameras to give a much cleaner set of depth maps. Since we have an approximate estimate of the depth map, the stereo process may be repeated for increased accuracy of correspondences, as described in [Vedula *et al.*1998].

For each point in the GRAMM, we store the pixel value in the corresponding image as the radiance, in addition to the sampling direction (direction of the camera optical center relative to the GRAMM point location). We then estimate the instantaneous motion for this point. This is done using a consensus algorithm from a 2-D estimate of the motion from multiple cameras, similar to how multiple stereo estimates are integrated into a 3-D model. The algorithm is detailed in Section 3.1.

2.2 Points or polygons?

In the graphics literature, researchers have argued for the suitability of points as a modeling and display primitive. See for example [Hoppe *et al.*1992], [Szeliski and Tonnesen1992], and [Levoy and Whitted1985]. The main disadvantage of points over polygons is that scan conversion of points can leave holes in the image created, and typically more points are needed than textured polygons.

The advantage, however, is that they are faster to render and don't require topological connectivity. The latter is the main reason why we choose a point based representation in this work. If a set of points in a GRAMM are used to model the scene at one time instant, then the instantaneous motion predicts that the points move in a certain manner. However, if there is a specified connectivity between these points (as in a polygonal mesh), then this can cause badly shaped polygons, or even polygons that cross each other.

The other advantage of using such a point based representation is that the density of points at any particular part of the scene is roughly proportional to the number of cameras that are viewing it. The quality of the rendered image should, ideally, be a function of the density of points, and therefore, the number of input views. This gives us a natural framework for super-resolution analysis, if the calibration error is within a pixel, and the stereo correspondences are accurate.

3 Time Interpolation of a GRAMM

For visualizing an event at a particular time, a seemingly feasible option is to compute the spatially reconstructed image from the virtual viewpoint, at the nearest time(s) that sampled images are available. An optical flow field may then be created, to capture the variation in appearance with time, and the variation in this flow field may be used to predict the appearance at the required time. However, this is not practical because optical flow is highly noisy, and leads to holes because of the fact that extrapolation is done in the image plane, and there is no explicit modeling of occlusions.

Rather, we create a GRAMM at the desired time instant by interpolating between GRAMMs at nearby time instants. To do this, we compute an explicit 3-D motion model, and interpolate scene structure in space. The computed scene motion avoids the undesirable noise present in typical optical flow, and the computed GRAMM can then be used to reconstruct appearance.

3.1 Estimation of 3-D Scene Flow

Recall that a GRAMM contains a field for instantaneous motion of the point. We propose a new algorithm for *Three dimensional scene flow*, which we define as a three dimensional flow field containing instantaneous velocities for every point in the scene. In practice, we compute scene flow for each point on the GRAMM. This GRAMM is then used for forward prediction of the scene structure, which is required for virtual view synthesis.

Optical flow [Horn1986] is used to compute a flow field on an image plane, between two images taken from the same camera but at times t and $t + \delta t$. Conceptually, just as how 3-D volumetric merging eliminates noise in stereo correspondences, 3-D scene flow eliminates noise in individual camera optical flows.

Consider the implicit linear relation between the scene flow and optical flow on an image plane, shown in Figure 4. If $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ is the instantaneous velocity of a point from the GRAMM and $\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t}$ is the projection of that motion in the image plane of camera i, then we have the relationship

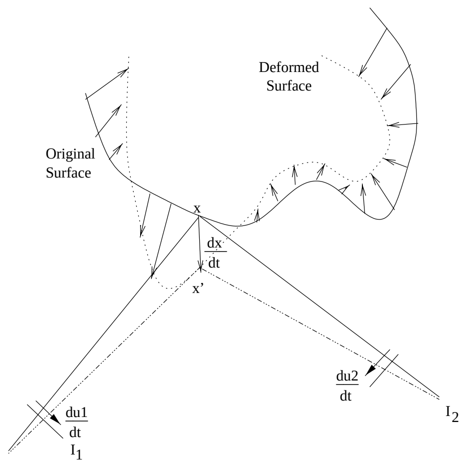


Figure 4: Relationship between Scene flow, and temporal optical flow on two image planes. $\frac{d\mathbf{x}}{dt}$ is the Scene Flow when the surface represented by the dark line deforms to the surface represented by the dotted line, while $\frac{d\mathbf{u_1}}{dt}$ and $\frac{d\mathbf{u_2}}{dt}$ are the optical flows at Images 1 and 2.

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}. \tag{1}$$

where $\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}$ is a 2x3 matrix that relates rate of change of an image projection to the rate of change of motion in the scene, and is obtained by just differentiating the camera projection matrix. This set of equations provides two linear constraints on $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$. Therefore, if we have 2 or more cameras, we can solve for $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$, by setting up the system of equations $\mathbf{Bx} = \mathbf{U}$, where:

$$\mathbf{B} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z} \\ \frac{\partial v_1}{\partial x} & \frac{\partial v_1}{\partial y} & \frac{\partial v_1}{\partial z} \\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} & \frac{\partial u_2}{\partial z} \\ \frac{\partial v_2}{\partial x} & \frac{\partial v_2}{\partial y} & \frac{\partial v_2}{\partial z} \\ . & . & . \\ . & . & . \\ \frac{\partial u_N}{\partial x} & \frac{\partial u_N}{\partial y} & \frac{\partial u_N}{\partial z} \\ \frac{\partial v_N}{\partial x} & \frac{\partial v_N}{\partial y} & \frac{\partial v_N}{\partial z} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \frac{\partial u_1}{\partial t} \\ \frac{\partial v_1}{\partial t} \\ \frac{\partial u_2}{\partial t} \\ \frac{\partial v_2}{\partial t} \\ . \\ . \\ \frac{\partial u_N}{\partial t} \\ \frac{\partial v_N}{\partial t} \end{bmatrix} \tag{2}$$

This gives us $2N$ equations in 3 unknowns, therefore for $N \geq 2$ we have an over-constrained system and can find an estimate for the scene flow. (This system of equations is degenerate if and only if the point $\mathbf{x}$ and the N camera centers are collinear). A singular value decomposition of $\mathbf{B}$ gives the solution that minimizes the sum of least squares of the error

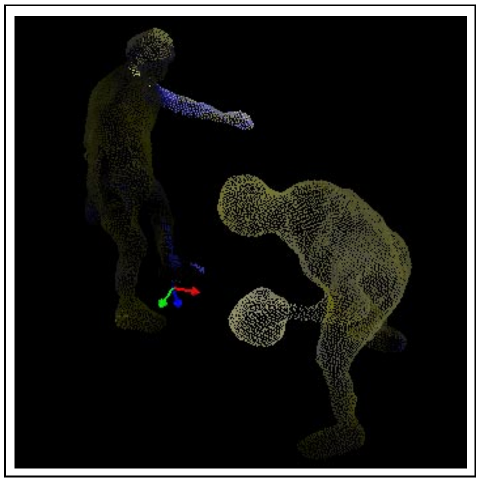


Figure 5: The computed scene flows are displayed as color-coded points, (the locations of which are obtained by projecting depth maps from 4 cameras into the scene.) The absolute values of the scene flows in the x, y, and z direction contribute to red, green, and blue colors proportionally to their magnitude. The three-dimensional motion of the person with the ball has components in all three directions and so appears an off-white color. The arm of the person at the back moves upwards so is mostly blue.

obtained by re-projecting the scene flow onto each of the optical flows. For a more detailed analysis of the problem of computing Scene Flow, see [Vedula *et al.*1999].

3.1.1 Results

We implement the above algorithm on an image sequence, that contains a player bending forward as he begins to bounce a basketball. We use optical flows from 15 different cameras. The use of this many optical flows ensures that every point at which we desire to compute scene flow is viewed by at least 2 or 3 cameras.

Figure 5 shows the computed scene flow color-coded with the absolute values of the computed flows in the x, y, and z directions mapped to red, green, and blue. This scene flow is computed for all points in a GRAMM built using 4 range images from widely spaced viewpoints. These range images are essentially depth maps obtained by projecting the volumetric model obtained by using multi-baseline stereo and volumetric merging, into 4 widely separated cameras. It is seen that the motion of the ball (in a direction with roughly equal components along all axes), and the vertical motion of the far person's left hand are the most significant (intensity is proportional to the magnitude of the scene flow).

3.1.2 Discussion

The algorithm described above does not scale very well to scenes with more clutter. Scenes with more clutter have a larger number of object discontinuities, where optical flow is often noisy. Thus, there would be larger number of outliers and the computation needs to be more robust. An even bigger problem with more objects is that the relation between optical flow and scene flow (Equation 1) holds only for those cameras that directly see the scene point. The cameras that are occluded are not treated any differently, and therefore contribute incorrectly to the computation of the scene flow.

We propose to account for invalid contributions due to outliers and occluded cameras using a least median squares approach, rather than the currently used SVD which treats all contributions equally. An alternative approach to investigate is the use of depth information from the scene points on the GRAMM, which would give us occlusion information that lets us eliminate spurious optical flow contributions.

3.2 Creating a GRAMM at the required time instant

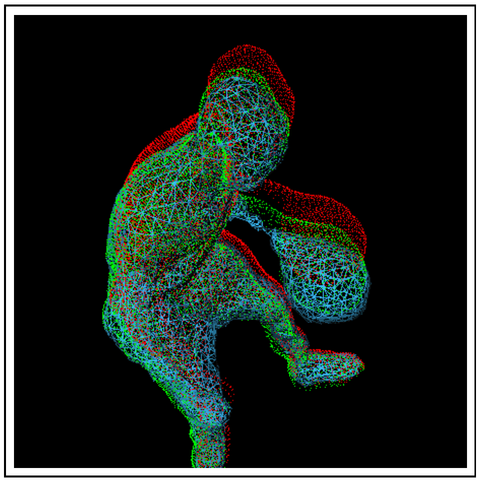


Figure 6: Computed scene flow for a dynamic scene. The GRAM at the initial time instant is shown in red, while the "flowed" (predicted) point cloud using the computed 3-D Scene flow is shown in green, and the independently computed volumetric model at the next time instant (shown as a wire-frame) is in blue.

In order to visualize the scene as it would appear from a particular viewpoint at some point in time, we first need to determine scene structure at that time instant. Since this time can be arbitrary, we are not guaranteed to have input images, and hence a GRAMM available for that time. However, we can create one by interpolating from a GRAMM at a nearby time instant, using the computed scene flow as a motion field for the motion of all points in the GRAMM. Recall that every element of a GRAMM at time t contains a point with location x and computed scene flow $\frac{dx}{dt}$.

To interpolate the GRAMM to the desired time $t + \delta t$, a simple strategy is to use

$$x_{t+\delta t} = x_t + \frac{dx}{dt}\delta t \tag{3}$$

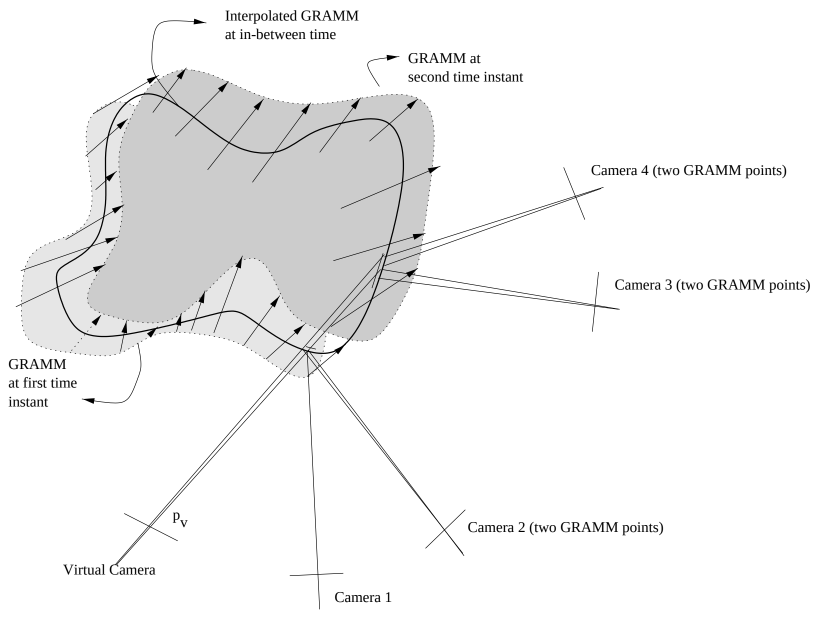


Figure 7: An interpolated GRAMM, whose points are connected with the curve. Each point in the GRAMM is projected onto the virtual image.

Results of this simple linear forward prediction are shown in Figure 6. The event involves a player bouncing a basketball. The point cloud in red shows the GRAMM at the initial time instant, $t = 1$. A simple forward prediction of the geometry using the recovered scene flow to yield the GRAMM at the next time instant results in the green point cloud. It can be seen that the motion of the points from red to green results in the points moving almost onto the "true" model, shown as a blue wire-frame. This model (the blue wire-frame) is obtained independently using the stereo and volumetric merging technique of [Rander *et al.*1996] at time $t = 2$, without any use of images at $t = 1$. This demonstrates the validity of the computed scene flow, and the linear prediction mechanism. Thus, the scene flow can be used to predict the GRAMM at a diferent time instant.

Currently, we use only one GRAMM to compute the GRAMM at the desired time by forward extrapolation. It is also possible to reverse-extrapolate from the GRAMM computed from image samples at a later time instant, or combine the two sets of information to increase the robustness of the algorithm. This will be investigated in the thesis.

4 From an Interpolated GRAMM to a Virtual Image

Once the GRAMM is temporally interpolated to the required time instant, it can be used to create an image from the virtual viewpoint.

4.1 Creating a virtual image from a GRAMM

The algorithm computes the color of each pixel in the virtual image, one by one. Consider the hypothetical scene shown in figure 7. The boundaries of the two shaded regions represent the GRAMMs at two time instants, and the arrows show the computed scene flow from one to the other. The set of points on the temporally interpolated GRAMM at the in-between time instant are all connected by the thick curve.

Let $\mathbf{P}_i$ be the 3x4 camera matrix for the i^{th} camera, and let $\mathbf{P}_v$ be the projection matrix corresponding to the virtual camera location. Consider point j of the GRAMM with location expressed in projective coordinates as $\mathbf{x}_j = (x_j, y_j, z_j, 1)^T$, that is contributed by camera i. Then, we have $\mathbf{x}_j$ projecting to $\mathbf{u}'$ in the virtual camera:

$$\mathbf{u}' = \mathbf{P}_v \mathbf{x}_j \tag{4}$$

If we are trying to reconstruct a pixel whose image co-ordinates are $\mathbf{u}$, we define G_u as the set of all $\mathbf{x}_j$ that satisfy

$$\mathbf{u} - \mathbf{u}' < \epsilon \tag{5}$$

Therefore, G_u consists of all points in the GRAMM, that project to a particular pixel with 2-D projective coordinates $\mathbf{u} = (u, v, 1)^T$ in the virtual image. In our example, G_u would contain one GRAMM entry from camera 1, and two each from cameras 2, 3, and 4. The choice of ϵ is an estimate for the point spread function of the imaging process. In practice, a spread of one pixel works reasonably well. The task now becomes computing a suitable color for the pixel at $\mathbf{u}$, from all the GRAMM points in G_u. The algorithm proceeds as follows:

1. Sort all the points in G_u in increasing order of projected depth. If multiple GRAMM points are contributed by the same camera, they can be sorted using McMillan's ordering algorithm. Otherwise, we sort them just by their projected depth values (software z-buffering).

2. Consider the set of points $S_u \subset G_u$ that correspond to the first cluster of depth values (this would eliminate the GRAMM points coming from Cameras 3 and 4 in the above example).

3. Initialize $I = 0.0$, $w = 0.0$

4. Currently, we assume that we know the surface normal for each point in the GRAMM. Let the angle between the viewing direction and surface normal be ϕ_i, and angle between the sampled ray direction and surface normal be θ_i.

5. For each point k in S_u, and separately for each color band
   $I = I + \cos\theta_j \cos\phi_j I_i$
   $w = w + \cos\theta_j \cos\phi_j$

6. Final intensity c = I/w, again for each color band.

The algorithm loops over every GRAMM element, projecting it into the virtual image. For each such projected point, the above steps are performed. Note that resampling of pixel values from the original cameras to the virtual camera is performed only once. The projection is forward, thus avoiding the need to search for matching points, as is done by [Laveau and Faugeras1994b]. The value of the confidence of the correspondence match (stereo algorithm) can be easily incorporated if available.

We propose two improvements : the surface normal is currently available because we use an explicit 3-D representation. The algorithm will be improved to not use this information. In addition, we want the property that a virtual view from the same camera position as an original view, identically yields the same image. This is possible because of the fact that there is no resampling of input data into an intermediate representation. In addition. we will also show that the degradation of image quality as one moves away from an original view is smooth.

4.2 Results

Figure 8 shows a comparison of an original image, and a rendered image from the same viewpoint. Correspondences are only available for points corresponding to those on the object, and therefore the background pixels are all black.

Figure 9 shows a rendered image from a virtual camera located very close to the ball. For comparison, the 3-D polygonal mesh of the ball is shown. This mesh itself is not used in the rendering process, thus artifacts arising from the polygonization of the mesh are avoided.

Figure 10 shows that we have arbitrary control of our virtual camera parameters. The virtual camera has a very wide field of view and therefore displays strong perspective effects. Figure 11 shows two more rendered virtual images.

4.3 Useful features of the GRAMM based rendering algorithm

- GRAMM provides a framework for interpolation of multi-image data, for view synthesis. The rendering algorithm works with GRAMMs that are interpolated between time instants, in order to create a virtual image that is interpolated in both spsace and time.

- Virtual camera control is easy, because its co-ordinates need to be specified in a Euclidean coordinate frame. (as compared to methods of projective reconstruction [Laveau and Faugeras1994b], which require creating a correspondence map between the virtual camera position and real cameras). To create a virtual *flythrough*, multiple virtual camera positions are needed and Euclidean co-ordinates ensure that the motion is smooth.

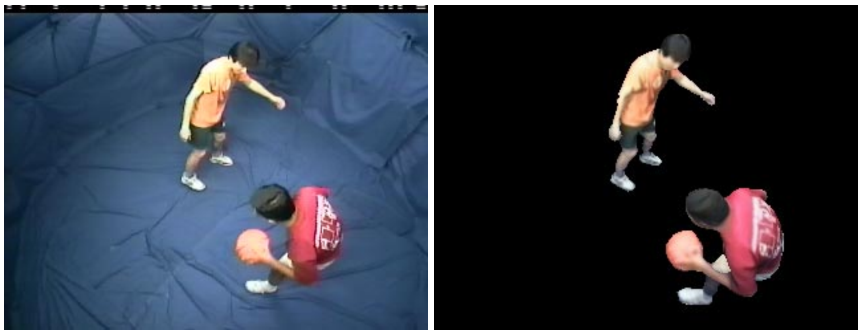

(a) Original Image (b) Rendered Image

Figure 8: Original image and Rendered image from the same viewpoint

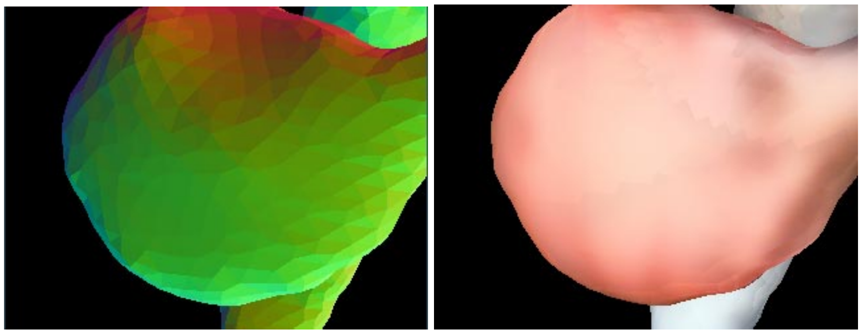

(a) Actual polygonal mesh (not used) (b) Rendered Image

Figure 9: The actual 3-D model of the ball is a polygonal mesh : Rendering using a GRAMM doesn't use a 3-D model and therefore no mesh artifacts are seen

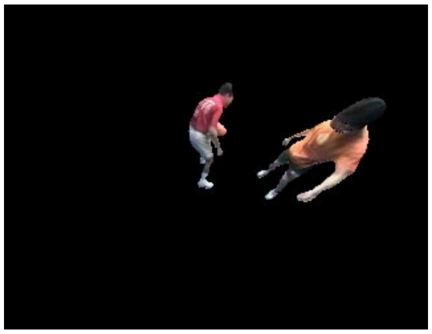
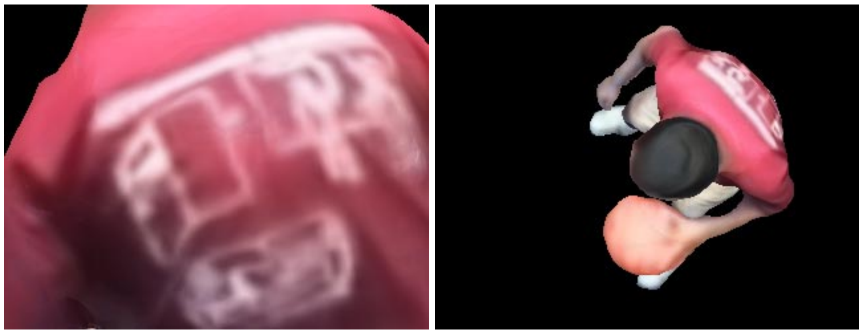


Figure 10: A view from a virtual camera with very wide field of view. Note the strong perspective foreshortening, similar to what a real camera would produce



(a) View from very close to the scene (b) Overhead view

Figure 11: Two more virtual views. Note the absence of polygonal artifacts

- While creating a virtual image, a GRAMM enables use of all available views (not just 2 or 3 views like methods based on view morphing, or interpolating the trilinear tensor). Hence, the rendered image is less likely to have holes, especially for more complex scenes.
- Since original data is preserved in a GRAMM without resampling, with a carefully chosen interpolation scheme, it is possible for the algorithm to have the property of *Same view yields same image,* meaning that virtual views rendered from the same position as a real view would reconstruct the original exactly. This is a useful property, satisfied by two-view morphing methods such as [Seitz and Dyer1996] and [Chen and Williams1993]. This will be investigated in the thesis.

5 Multiple-camera geometric constraints

5.1 Why geometric consensus?

Computing Scene flow and creating a virtual image from the GRAMM both require that data be combined from multiple images. Therefore, it is important that different cameras are geometrically in consensus about the mapping from 2-D image co-ordinates to 3-D scene co-ordinates. While camera calibration usually computes this mapping for a single camera, use of multiple cameras in a Euclidean framework requires that they are in mutual agreement as well. Figure 12 shows two images of a set of 3-D points, with one pair of corresponding points marked. When each of the cameras are calibrated independently of each other, it is seen that the inferred epipolar line (shown as solid) for a particular point can be quite far from the corresponding point. On the other hand, the epipolar line given by the fundamental matrix is far more accurate. The fundamental matrix is computed only by using measured image co-ordinates, rather than 3-D co-ordinates.

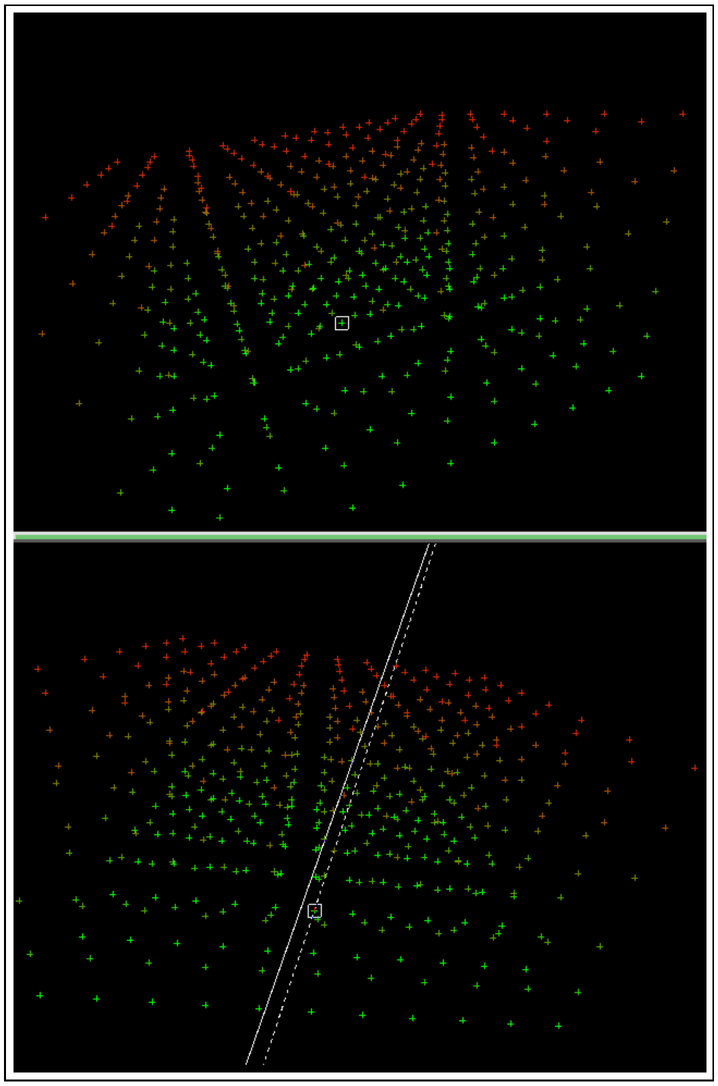


Figure 12: Two images of the set of calibration points. The solid line shows that the epipolar line predicted by two independent camera calibrations can be quite far from the true corresponding point. The dotted line is the “true” epipolar line obtained by calculating the fundamental matrix, and passes much closer to the corresponding point.

This inconsistency arises because of errors in positioning the calibration target. This is particularly a problem in a multi-camera situation such as ours, where the volume of interest is large and the number of cameras is large enough that it is impractical to calibrate them individually. Therefore, we use a target that is small enough so that it is visible from all cameras, and it is moved at marked intervals to span the workspace.

5.2 Optimization to reduce epipolar error

We propose an algorithm to optimize for Euclidean parameters, while using a non-linear least-squares algorithm to fit the recovered parameters to the local homography. Currently, we use Downhill Simplex optimization.

Suppose we have N calibration points, with (approximately) known 3-D co-ordinates. Consider two cameras with 3x4 camera matrices P and $P^{'}$. Let the i^{th} 3-D point be $\mathbf{x}_i = (x_i y_i z_i)^T$. This i^{th} point is imaged in the two cameras and has pixel locations with projective co-ordinates $\mathbf{u_i}$ and $\mathbf{u_i'}$.

Let

$$P^* = P^T(PP^T)^{-1} \tag{6}$$

be the pseudo-inverse of P. Then, the ray in space corresponding to the pixel u_i is given by

$$P^* u_i + \lambda p \tag{7}$$

where p is a vector in the null space of P. If this ray is projected into the second camera, we have a line whose parametric equation is given by

$$P^{'}(P^* u_i + \lambda p) \tag{8}$$

Performing the projective division and re-writing the above line equation in the form $ax + by + c = 0$, we have the *epipolar error* d_i for point i as

$$d_i = a u_i^{'} + b v_i^{'} + c \tag{9}$$

Thus, we can formulate the optimization problem for a particular pair of cameras as

$$\min \sum_{i=1}^{N} d_i \tag{10}$$

where the summation is over all calibration points mutually visible in both cameras.

We use the non-linear Downhill Simplex Algorithm [Nelder and Mead1965], currently minimizing only the six extrinsic parameters of the reference camera. The algorithm is initialized with the calibration parameters from the Tsai camera calibration. A simplex of 7 points, each with different values of the 6 extrinsic parameters is defined, and the algorithm proceeds by iteratively taking steps modifying the simplex in a greedy manner.

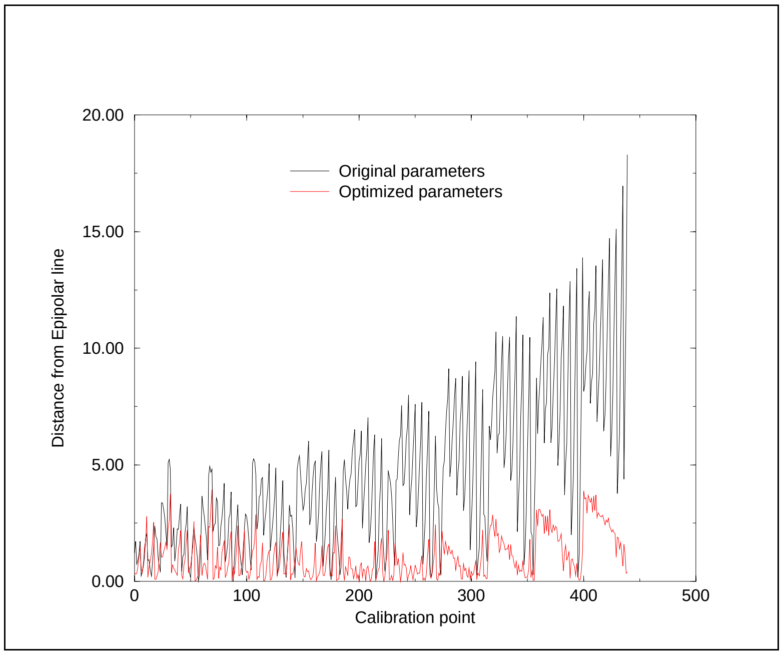


Figure 13: Downhill Simplex results : the original epipolar error is shown in white, while the error after optimization is shown in yellow.

Termination is reached when the decrease in the function value during one step is lesser than a certain tolerance. To avoid termination due to one bad step, each of the simplex points are perturbed, and the algorithm is repeated until it repeatedly converges to the same solution.

5.3 Results and Discussion

Figure 13 shows the result of the optimization, to achieve geometric consensus. For each of 440 points, the distance from the epipolar line using the initialized data is shown in black. The red curve shows the resulting error, after the optimization algorithm. There are two interesting observations : firstly, that the initial error exhibits a systematic pattern, varying by the location of the point relative to the camera. The optimization is not partial to any point, and therefore this effect is mostly eliminated. Secondly, the average error decreases from 4.56 pixels to 1.07 pixels. Optimization results on other pairs of cameras are somewhat comparable.

Ideas we propose to investigate are :

- Investigate explicit 3D error metric - Euclidean distance to epipolar line only implicitly models 3D position. We may be able to do better with explicitly modeling uncertainty in the 3D measurement.

- Currently, we are optimizing Camera A, assuming that Camera B is perfect, which is obviously not the case. Hence, we will investigate an iterative scheme, where the algorithm improves parameters of many cameras simultaneously.

- Currently, we only optimize extrinsic parameters. The strategy will be expanded to include intrinsic parameters as well.

6 Expected Contributions

I expect the following contributions from my thesis:

1. An algorithm for efficient image based rendering, which can simultaneously handle interpolation in the spatial and temporal domains. It will yield perfect reconstruction at original camera positions, and smooth degradation as it moves away.

2. Geometry, Radiance and Motion Map (GRAMM) : Image based representation for efficient and geometrically accurate spatio-temporal view interpolation. Representation encodes all the information in stereo correspondences, optical flow, and captured images in a single datastructure.

3. Algorithms for estimation of dense three-dimensional scene flow. This flow can be used to interpolate the scene spatially and temporally, by converting calculating it for all points in a GRAMM.

4. Algorithm for imposition of multi-camera geometric constraints on top of regular Euclidean calibration methods, including use of epipolar constraints to reduce positioning uncertainty of calibration object.

5. The above ideas and results will be demonstrated on data obtained by digitizing actual events using a real-time synchronized multi-camera image capture system, a significant part of whose design and engineering I am involved with.

7 Timetable

1999	Spring	Preliminary work on scene flow, Thesis Proposal
1999	Summer	Investigate and complete multi-camera consensus, Start work on temporal interpolation of GRAMM
1999	Fall	Improve Scene flow to account for outliers, complete GRAMM representation (point vs. mesh) issues
2000	Spring	Complete rendering algorithm (Same view yields same image and smooth degradation) and integrate with remaining modules
2000	Summer	Write thesis and defend

References

[Adelson and Bergen1991] E. H. Adelson and J. R. Bergen. *Computational Models of Visual Processing*, chapter 1 (The Plenoptic Function and the Elements of Early Vision). MIT Press, Cambridge, MA, 1991.

[Avidan *et al.*1997] S. Avidan, T. Evgeniou, A. Shasua, and T. Poggio. Image-based view synthesis by combining trilinear tensors and learning techniques. In *VRST '97*, pages 103–110, September 1997.

[Chen and Williams1993] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '93)*, pages 279–288, 1993.

[Collins1996] Robert Collins. A space sweep approach to true multi-image matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 358–363, June 1996.

[Debevec *et al.*1996] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH ' 96)*, pages 11–20, 1996.

[Gortler *et al.*1996] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 43–54, 1996.

[Gortler *et al.*1997] Steven J. Gortler, Li-Wei He, and Michael F. Cohen. Rendering layered depth images. Technical Report MSTR-TR-97-09, Microsoft Research Advanced Technology Division, Redmond, WA, March 19 1997.

[Hoppe *et al.*1992] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '92)*, pages 71–78, 1992.

[Horn1986] B.K.P. Horn. *Robot Vision*. McGraw Hill, 1986.

[Laveau and Faugeras1994a] Stephane Laveau and Olivier Faugeras. 3-d scene representation as a collection of images. Technical Report Technical Report RR-2205, INRIA - The French Institute for Research in Computer Science and Control, February 1994. Available from http:www.inria.fr.

[Laveau and Faugeras1994b] Stephane Laveau and Olivier Faugeras. 3-d scene representation as a collection of images and fundamental matrices. Technical Report No-2205, INRIA Sophia-Antipolis, February 1994. Available from http://www.inria.fr.

[Levoy and Hanrahan1996] Marc Levoy and Pat Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 31–42, 1996.

[Levoy and Whitted1985] Mark Levoy and Turner Whitted. The use of points as a display primitive. Technical Report UNC-CS TR85-022, Department of Computer Science, University of North Carolina, 1985.

[Liao *et al.*1997] W.-H. Liao, S.J. Aggrawal, and J.K. Aggrawal. The reconstruction of dynamic 3D structure of biological objects using stereo microscope images. *Machine Vision and Applications*, 9:166–178, 1997.

[Metaxas and Terzopoulos1993] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE PAMI*, 15(6):580–591, 1993.

[Nelder and Mead1965] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 1965.

[Nishino *et al.*1998] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. Eigen texture method - appearance compression based on 3d model. Technical Report IIS-CVL-98-102, Institute of Industrial Science, The University of Tokyo, July 1998.

[Penna1994] M.A. Penna. The incremental approximation of nonrigid motion. *CVGIP*, 60(2):141–156, 1994.

[Rander *et al.*1996] P.W. Rander, P.J Narayanan, and T. Kanade. Recovery of dynamic scene structure from multiple image sequences. In *Proc. of the 1996 Intl. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pages 305–312, 1996.

[Sato *et al.*1997] Yoichi Sato, Mark Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '97)*, 1997.

[Seitz and Dyer1996] S. M. Seitz and C. R. Dyer. View morphing. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, pages 21–30, 1996. Available from ftp.cs.wisc.edu.

[Seitz and Dyer1997] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the Image Understanding Workshop*, pages 935–942, 1997. Available from ftp.cs.wisc.edu.

[Szeliski and Tonnesen1992] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '92)*, pages 185–194, 1992.

[Ullman1984] S. Ullman. Maximizing the rigidity: The incremental recovery of 3-D shape and nonrigid motion. *Perception*, 13:255–274, 1984.

[Vedula *et al.*1998] S. Vedula, P. Rander, H. Saito, and T. Kanade. Modeling, combining, and rendering dynamic real-world events from image sequences. In *Proceedings of Fourth International Conference on Virtual Systems and Multimedia*, pages 326–332, Gifu, Japan, November 1998.

[Vedula *et al.*1999] S. Vedula, S. Baker, P. Rander, R Collins, and T. Kanade. Three dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Kerkyra, Greece, September 1999. (To appear).

[Waxman and Duncan1986] Allen M. Waxman and James H. Duncan. Binocular image flows: Steps toward stereo-motion fusion. *IEEE PAMI*, 8(6):715–729, 1986.