Spatiotemporal Sampling and Interpolation for Dense Video Camera Arrays*

Bennett S. Wilburn† Stanford University
Neel S Joshi‡ Stanford University
Katherine Chou§ Stanford University
Marc Levoy¶ Stanford University
Mark Horowitz‖ Stanford University

Abstract

We explore the application of dense camera arrays to view interpolation across space and time for dynamic scenes. Large video camera arrays are typically synchronized, but we show that staggering camera triggers provides a much richer set of samples on which to base the interpolation. We do not increase the total number of samples—we merely distribute them more effectively in time. We use optical flow to interpolate new views. Within this framework, we find that the dense space-time sampling provided by staggered timing improves the robustness of the interpolation. We present an novel optical flow method that combines a plane plus parallax framework with knowledge of camera spatial and temporal offsets to generate flow fields for virtual images at new space-time locations. We present results interpolating video from a 96-camera light field using this method.

CR Categories: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Time-varying imagery; I.4.9 [Image Processing and Computer Vision]: Applications

Keywords: spatiotemporal sampling, image-based rendering, view interpolation, optical flow, multiple camera systems

1 Introduction

Video cameras are rapidly becoming commodity hardware, causing image sensors, lenses, and video compression electronics to drop rapidly in price. Today one can easily build a modest camera array for the price of a high-performance studio camera, and it is likely that arrays of 100s or even a 1000 cameras will soon reach price parity with these larger more expensive units. The increased flexibility of this type of camera array creates a number of new applications and interesting research challenges. For example, researchers have already shown how to use such an array for dense sampling in time, demonstrating a virtual 1560fps camera composed of 52 densely packed cameras [Wilburn et al.] while others have used dense video camera arrays for view interpolation [J.-C.Yang et al. 2002]. Rather than viewing these results as independent, this paper explores the use of dense camera arrays for view interpolation in both space and time. Thus we look at the more general problem of optimal sampling patterns and interpolation methods for the spatiotemporal volume of images that the camera array records.

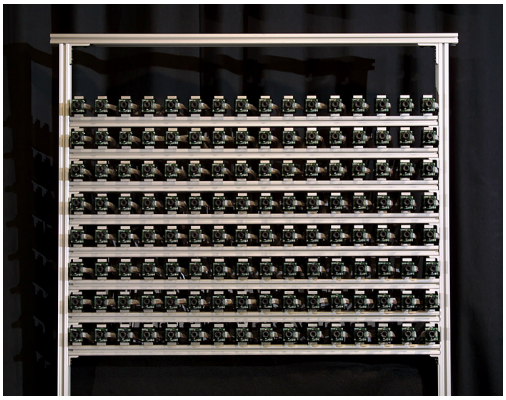


Figure 1: The video light fields in this paper were captured using a 96-camera subset of this array.

The simplest spatiotemporal interpolation method is extending light field rendering to video by linearly interpolating in time. For this reconstruction to work, the image volume must be bandlimited. Such prefiltering adds undesirable blur to the reconstructed images. Even with very large camera arrays, the sampling density is not sufficiently high to make the blur imperceptible. If the images are not bandlimited, the output exhibits ghosting artifacts. This occurs for large disparities or temporal motions.

To avoid the conflicting requirements for sharp images with no sampling artifacts, most image based rendering systems use more sophisticated interpolation schemes based on an underlying scene model. The simplest method is to estimate motion in an image based on local information from neighboring views. Other methods generate increasingly sophisticated three-dimensional models of the scene. Motion estimation grows less robust as the distance between cameras increases. More complicated models can handle more widely separated images, but their runtime increases as more global information is incorporated.

This paper explores how sampling affects reconstruction. Traditionally, designers of camera arrays have striven to synchronize their cameras. This often leads to much more temporal motion between camera frames than there is parallax between neighboring cameras. Instead, staggered triggers are a better sampling strategy. Improved temporal sampling decreases temporal image motion, allowing us to use simpler, more robust interpolation methods such as optical flow. As section 2 shows, planar arrays are well-suited to using a simple plane + parallax formulation that allows one to robustly calibrate images and also provides a very nice framework to set up flow. It provides a constrained space for computing optical flow between disparate cameras.

The next section describes previous work in capturing and interpolating between space-time image samples. Following that, we describe our method for calibrating the camera array and rendering

†e-mail: wilburn@stanford.edu
‡e-mail:nsj@cs.stanford.edu
§e-mail:seneca@stanford.edu
¶e-mail:levoy@cs.stanford.edu
‖e-mail:horowitz@stanford.edu

[0]*Submitted to Siggraph 2004*

new views. We review plane + parallax geometry, which plays a central role in our calibration and optical flow algorithms. We then describe a framework for determining how best to distribute our camera array samples in time. We show for linear interpolation between views that better sampling greatly improves reconstruction. Finally, we present an optical flow method for determining spatial and temporal motion between several views in space and time.

1.1 Previous Work

The increasing availability and lower cost of cam arrays has enabled a number of researchers to look at the potential of these systems. Virtualized Reality^{TM} [Rander et al. 1997] and its successor the 3D-Room[Kanade et al. 1998] are two pioneering large-array designs. The systems capture video from widely spaced, synchronized cameras. Because the system captures samples that are sparse in space and time, interpolation depends strongly on the quality of the inferred scene model. Yang et al. took a different approach. Rather than construct a scene model, they capture denser samples and use simple light field interpolation to run in real-time. This interpolation method produces ghosting artifacts in their output. They are not concerned with interpolation in time. The most visible multiple camera work is the linear array of over 100 cameras used to produce the "Bullet Time" effects for *The Matrix*. They simulate a virtual high-speed camera flying at impossibly high speeds around a dynamic scene. Their system captures one trajectory through time and space.

Our goal is to investigate how well one could do "Bullet Time" effects as a post-processing step for a captured set of images without specifying the view trajectory in advance. We explore this using a 100 camera array that combines selected features from these designs. It captures all of the data from a large number of precisely timed, inexpensive CMOS video image sensors. Although this array was built to enable a wide range of research, the technologies required for this work are commonly available. The key feature of the array for the results presented in this paper is a programmable trigger offset at each camera. Many CMOS sensors offer the digital synchronization inputs required to implement this sort of precise timing control.

Image-based rendering methods for view synthesis differ primarily in how they resample acquired images to generate new views. Light Field Rendering [Levoy and Hanrahan 1996] assumes a flat scene, which leads to aliasing artifacts. Lumigraphs [Gortler et al. 1996] assume a known geometry, either synthetic data or a manually specified geometric proxy [Buehler et al.]. [Lin and Shum] present a maximum camera spacing for light fields with a constant depth assumption, and [Chai et al. 2000] analyze the minimum spatial sampling rate for light fields including geometry information. We are the first to investigate the minimum temporal sampling rates for video light fields.

While hardware allows us to capture samples in the spatiotemporal volume, our next task is to interpolate between images to produce new views. View interp using image warps[Chen and Williams 1993; Seitz and Dyer 1995] allows one to produce sharp images even when large image motion is present but requires some means of determining how to warp pixels between views. The simplest of these schemes are image based methods using optical flow[Avidan and Shashua 1998; Irani et al. 1998], which is not robust over large displacements or lighting changes. For more widely separated cameras, 3D models have been computed using disparity maps[Rander et al. 1997], voxel coloring[Seitz and Dyer 1997], or visual hulls[Matusik et al. 2000]. Although models compensate for larger camera spacings, inferring the structure of complex, real scenes is a challenging task.

2 Calibration and Rendering

A challenge of working with many cameras is finding a framework for combining information from the different views. Even simple camera hopping will produce poor results if the cameras have different color responses. Because inexpensive image sensors rely on human sensitivity to relative, not absolute, color differences, the responses of image sensors varies greatly. We correct this using the automatic color calibration method of [Wilburn et al.], iteratively adjusting camera gains so each sensor approximates a desired linear response, then applying a post-processing step to correct for sensor nonlinearities and color differences.

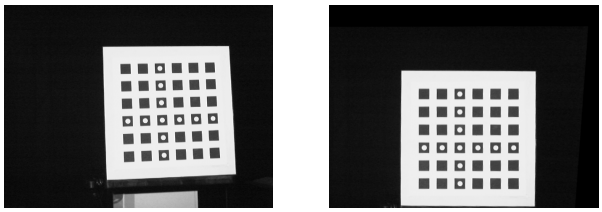
Once the camera color responses are matched, we need to understand the geometry of our cameras and images. Our goals here are twofold. First, we must correct for nonuniformities between cameras, such as varying focal lengths and orientations. As with color variations, view hopping across cameras will look poor if the virtual camera's perspective properties jump discontinuously from view to view. Second, we need a framework for combining data from different images that captures the geometric relationship between our cameras but does not require knowledge of the scene geometry. We use the method of [Vaish et al.] for calibrating a planar array of cameras using plane + parallax [Shashua and Navab 1994; Kumar et al. 1994; Irani et al. 1997]. This method aligns views in image space instead of relying on full geometric calibration, making it both simpler and more robust. As we will see, it also provides an effective framework for computing optical flow in space and time and for analyzing spatiotemporal sampling.

2.1 Plane + Parallax Calibration

We will briefly summarize the implementation and implications of the plane + parallax calibration described in [Vaish et al.]. We start by aligning images from all of our cameras to a reference plane that is roughly parallel to the camera plane. To do this, we take a picture of a planar calibration target roughly in the middle of our scene, frontoparallel to the camera plane. We use an automatic feature detector to locate and label points on the target. We designate a central camera to be the reference view and compute an alignment for it that makes the target appear frontoparallel while perturbing the imaged target feature locations as little as possible. We then compute planar homographies that align the rest of the views to the aligned reference view [Hartley and Zisserman 2000]. Figure 2 shows the original and aligned images of our calibration target from a non-reference view.

In the aligned images, there is a simple relation between a point's distance from the reference plane and its parallax between two views. Figure 3 shows a scene point P and its locations $p_0 = (s_0,t_0)^T, p_1 = (s_1,t_1)^T$ in the aligned images from two cameras C_0 and C_1. Let Δz_p be the signed distance from P to the reference plane (negative for this example), Z_0 be the distance from the camera plane to the reference plane, and Δx be the vector from C_0 to C_1 in the camera plane. Define the *relative depth* of P to be $d = \frac{\Delta z_p}{\Delta z_p + Z_0}$. Given this arrangement, the parallax $\Delta p = p_1 - p_0$ is simply $\Delta p = \Delta x \cdot d$.

This has two important consequences for our work:



(a) (b)

Figure 2: Alignment using planar homographies. Using images of a planar calibration target, we compute a planar homography that aligns each image to a reference plane. (a) shows an image of the target from a corner camera of the array. (b) show the same image warped to the reference view. The planar target appears frontoparallel in all of the aligned images.

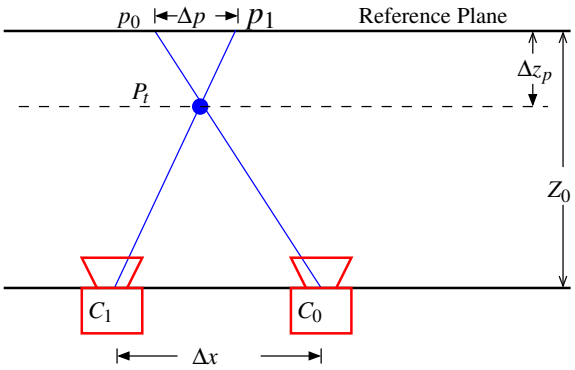


Figure 3: Planar parallax for planar camera arrays. A point P not on the reference plane has distinct images p_0, p_1 in cameras C_0, C_1. The parallax between these two is the product of the relative camera displacement Δx and the relative depth Δz_p.

- The parallax between aligned images of a single point off the reference plane is enough to determine the relative locations in the camera plane of all of the cameras. Typically, one measures the parallax of many points to make the process more robust.
- Once we know the relative camera locations, determining the relative depth of a point in one view is enough to determine its location in all other views.

This gives us the framework we need for interpolation. The aligned images provide a common space in which to analyze and combine views. In fact, they correspond to the (u,v) parameterized images for light field rendering (assuming constant depth at the reference plane), so measuring motion in reference plane indicates how much aliasing we will see in reconstructed light field images. As we will see later, parallax being a function of relative depth permits a simple optical flow method for determining image flow between neighboring spacetime views.

2.2 Rendering

Aligning our images to a reference plane automatically corrects for geometric variations in our cameras (excluding translations out of the camera plane and radial distortion, which we have found to be negligible for our application). The aligned images are generally off-axis projections, which are visually disturbing. This is clear from the aligned views of the calibration target, in which the reference plane target always appears frontoparallel regardless of the camera position.

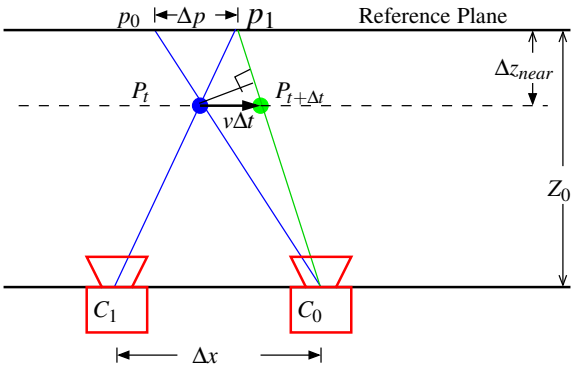
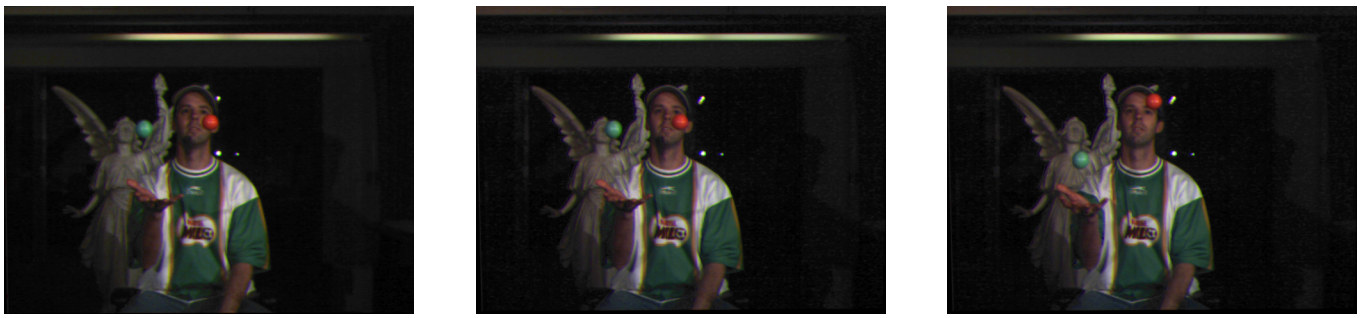


Figure 5: The temporal and spatial view axes are related by image motion. For a given scene configuration, we can determine a timestep Δt for which the maximum image motion between temporal samples is equal to the maximum parallax between spatially neighboring views. If we measure time in increments of Δt and space in increments of the camera spacing, then distance between view coordinates corresponds to the maximum possible image motion between views.

The transformation that corrects the off-axis projection is equivalent to taking a picture of the aligned plane from the virtual camera position. The plane + parallax calibration does not provide enough information to do this. If we fix the relative camera locations produced by our calibration, the missing information corresponds to the field of view of our reference camera and the distance from the camera plane to the reference plane. These quantities can be determined either by calibrating the reference camera relative to the reference plane or simple manual measurement. In practice, we have found that small errors in these quantities produce very subtle perspective errors and are visually negligible.

3 Spatiotemporal Sampling

We now turn our attention to the temporal distribution of our samples. We assume that our cameras all run at a single standard video rate (30fps for our array), that they are placed on a planar grid, and that the desired camera spacing has already been determined. Figure 4 shows aligned synchronized images from our array of 30fps video cameras. Differences between images are due to two components: parallax between views and temporal motion between frames. From the images, it is clear that the temporal image motion is much greater than the parallax for neighboring views in space and time. This suggests that we should sample more finely temporally to minimize the maximum image motion between neighboring views in space and time. In the next section, we show how temporal and spatial view sampling are related by image motion.



(a) (b) (c)

Figure 4: For synchronized cameras, the motion due to parallax between neighboring cameras is often much less than the temporal motion between frames for the same camera. (a) and (b) are images from adjacent cameras at the same point in time. Disparities between images are small. (c) shows a picture from the same camera as (b), one frame later. The motion is obvious and much larger.

3.1 Normalizing the Spatial and Temporal Sampling Axes

For a given location of the reference plane at a distance Z_0 from the camera plane, if we bound the maximum parallax in our aligned images, we can establish near and far depth limits for our scene, z_{near} and z_{far}. Alternatively, we could determine the minimum and maximum depth limits of our scene and place the reference plane accordingly [Chai et al. 2000]. The near and far bounds and camera spacing Δx determine the maximum parallax for any point between neighboring cameras. Given this near depth limit z_{near} and a maximum velocity of v for any object in the scene, we can determine the time for which the maximum possible temporal image motion equals the maximum parallax between neighboring views. This is shown in figure 5. The temporal motion for P in camera C_0 is greatest if it is at the near depth limit and moves such that the vector P_tP_{t+1} is orthogonal to the projection ray from C_0 at time $t+1$. If we assume a narrow field of view for our lenses, we can approximate this with a vector perpendicular to the reference plane, shown as $v\Delta t$. If P has velocity v, the maximum temporal motion of its image in C_0 is $\frac{v\Delta t Z_0}{Z0+\Delta Z_{near}}$. Equating this motion to the maximum parallax for P in a neighboring camera yields

$$\Delta t = \frac{\Delta X Z_{near}}{v \Delta Z_0} \tag{1}$$

This is the timestep for which maximum image motion equals maximum parallax between neighboring views.

Measuring time in increments of the timestep Δt and space in units of camera spacings provides a normalized set of axes to relate space-time views. A view is represented by coordinates (x,y,t) in this system. For nearest-neighbor or weighted interpolation between views, measuring view distance in these coordinates will minimize jitter or ghosting during reconstruction. Choosing a temporal sampling period equal to Δt will also ensure that maximum temporal motion between frames will not exceed the maximum parallax between neighboring views.

Determining maximum scene velocities ahead of time (for example, from the biomechanics of human motion, or physical constraints such as acceleration due to gravity) can be difficult. An alternative to computing the motion is filming a representative scene with synchronized cameras and setting the timestep equal to the ratio between the maximum temporal and parallax motions for neighboring views. One could even design a camera array that adaptively determined the timestep based on tracked feature points between views.

6	1	4
3	0	7
8	5	2

Figure 6: An example trigger pattern for a 3x3 array of cameras with nine evenly staggered triggers. The numbers represent the order in which cameras fire. The order was selected to have even sampling in the (x,y,t) space across the pattern. We tessellate larger arrays with patterns such as this one to ensure even spatiotemporal sampling.

3.2 Spatiotemporal Sampling Using Staggered Triggers

The timestep Δt tells us the maximum temporal sampling period that will ensure temporal resolution at least as good as the spatial resolution across views. One could increase the temporal sampling rate by using an array of high-speed cameras, but this could be prohibitively expensive and would increase demands on data bandwidth, processing, and storage. By staggering the cameras' trigger times, we can increase the temporal sampling rate without adding new samples.

Our goal is to ensure even sampling in space and time using our normalized axes. A convenient way to do this is with a tiled pattern, using the minimum number of evenly staggered trigger times that gives an offset less than Δt. To approximate uniform sampling, the offsets are distributed evenly within the tile, and the pattern is then replicated across the camera array. Figure 6 shows an example trigger pattern for a 3x3 array of cameras. For larger arrays, this pattern is replicated vertically and horizontally. The pattern can be truncated at the edges of arrays with dimensions that are not multiples of three.

3.3 Interpolating New Views

We can now create our distance measure for interpolation. The plane + parallax calibration gives up camera positions in the camera plane up to some scale factor. We normalize these positions by dividing by the average space between adjacent cameras, so the distance from a camera to its horizontal and vertical neighbors is approximately one. Let (x,y) be the position of each camera in these normalized coordinates, and let t be the time at which a given image is acquired, measured in timestep of Δt. Because we have chosen a timestep that sets the maximum parallax between views equal to the maximum temporal motion between timesteps, the euclidean distance between the (x,y,t) coordinates representing two views is a valid measure of the maximum possible motion between the two images.

The simplest way we could interpolate new views would be to use nearest neighbors. This is the method used by [Wilburn et al.] to create a virtual camera using a dense camera array. This method produces acceptable results, but as points move off the reference plane, their images jitter due to parallax between views. The perceived jitter can be reduced using interpolation between several nearby views. To determine which images to blend and how to weight them, we compute a Delauney tessellation of our captured image coordinates. For a new view (x,y,t), we find the tetrahedron of images in the tessellation containing the view and blend the images at its vertices using their barycentric coordinates as weights. Using this tessellation and barycentric weighting ensures that our blending varies smoothly as we move the virtual viewpoint. As we leave one tetrahedron, the weights of dropped vertices go to zero. Our temporal sampling pattern is periodic in time, so we only need to compute the tessellation for three 30Hz sampling periods to compute the weights for an arbitrarily long sequence.

Figure 7 shows the benefits of improved temporal sampling. In this experiment, we used a 12x8 array of 30fps video cameras to film a soccer player. The cameras were triggered according to the pattern in figure 6, tiled across the array. We then generated 270fps interpolated video using several methods. First, we used a cross-dissolve between sequential frames at one camera to simulate linear interpolation for a synchronized array. The motion of the soccer ball between captured frames is completely absent. Next, we used nearest-neighbor interpolation, which assembles a video sequence using video captured at the proper time from neighboring cameras. This produces sharp images and captures the path of the ball, but the motion is jittered due to parallax between views. Finally, we used the barycentric weighted averaging described previously. This reduces the ball's motion jitter but introduces ghosting.

Staggering the cameras clearly improves our temporal resolution and results in much better results even for simple nearest-neighbor and weighted interpolation. Because our input images are not bandlimited in space and time, new views interpolated with either of these methods will always suffer from artifacts if the motion between views in time or space is too great. One could imagine prefiltering spatially as described in [Levoy and Hanrahan 1996], or temporally by using overlapped exposure windows, but prefiltering adds undesirable blur to our images. In the next section, we improve our spacetime view interpolation by analyzing the motion between captured images.

4 Optical Flow for Spatiotemporal View Interpolation

We have seen that distributing samples from a dense camera array more evenly in time improves spatiotemporal view interpolation using nearest-neighbor or weighted interpolation. Reducing the image motion between captured spatiotemporal views can also decrease the complexity or increase the robustness of other interpolation methods. We have found that the combination of dense cameras, improved temporal sampling, and plane + parallax calibration allows us to compute new views robustly using optical flow.

We extended the optical flow method of [Black and Anandan 1993] using code available on the author's web site. Their algorithm is known to handle violations of the intensity constancy and smoothness assumptions well using robust estimation. It uses a standard hierarchical framework to capture large image motions, but can fail due to masking when small regions of the scene move very differently from a dominant background[Bergen et al. 1992]. For our 30fps synchronized juggling sequence, the algorithm succeeded between cameras at the same time but failed between frames for the same camera. The motion of the small juggled balls was masked by the stationary background. Once we retimed the cameras, the motion of the balls was greatly reduced, and the algorithm computed flow accurately between pairs of images captured at neighboring locations and timesteps.

Our modified spatiotemporal optical flow algorithm has two novel features. First, we solve for a flow field at the (x,y,t) location of our desired virtual view. Typically, optical flow methods will compute flow between two images by iteratively warping one towards the other. This was inspired by the bidirectional flow of [Kang et al. 2003], who observe that for view interpolation, computing the flow at the new view position instead of either source image handles degenerate flow cases better and avoids the hole-filling problems of forward-warping when creating new views. They use this to compute flow at a frame halfway between two images in a video sequence. We extend the method to compute flow at a desired view in our normalized (x,y,t) view space. We iteratively warp the nearest four captured images toward the virtual view and minimize the weighted sum of the robust pairwise data errors and a robust smoothness error.

Motion cannot be modelled consistently for four images at different spacetime locations using just horizontal and vertical image flow. The second component of our algorithm is simultaneously accounting for parallax and temporal motion. We decompose optical flow into the traditional two-dimensional temporal flow plus a third flow term for relative depth that accounts for parallax between views. Plane + parallax calibration produces the relative displacements of all of our cameras, and we know that parallax between two views is the product of their displacement and the point's relative depth. The standard intensity constancy equation for optical flow is

$$I(i,j,t) = I(i+u\delta t, j+v\delta t, t+\delta t) \tag{2}$$

Here, (i,j,t) represent the pixel image coordinates and time, and u and v are the horizontal and vertical motion at an image point. We use i and j in place of the usual x and y to avoid confusion with our view coordinates (x,y,t).

Our modified intensity constancy equation includes new terms to handle parallax. It represents constancy between a desired virtual view and a nearby captured image at some offset $(d\delta x, d\delta y, d\delta t)$ in the space of source images. It accounts for the relative depth, d, at each pixel as well as the temporal flow (u,v):

(a) (b) (c)

Figure 7: Better temporal sampling improves interpolation. (a) Linear interpolation between frames in time for a synchronized camera array is just a cross-dissolve. (b) Nearest-neighbor interpolation using staggered cameras produces sharp images, but this composite of multiple images shows that the path of the ball is jittered due to parallax between different cameras. (c) Weighted interpolation using the nearest views in time and space reduces the perceived jitter but causes ghost images.

$$I_{virtual}(i,j,x,y,t) = I_{source}(i+u\delta t+d\delta x, j+v\delta t+d\delta y, t+\delta t) \quad (3)$$

We compute flow using four images from the tetrahedron which encloses the desired view in the same Delauney triangulation as before. The images are progressively warped toward the common virtual view at each iteration of the algorithm. We cannot test the intensity constancy equation for each warped image against a virtual view, so we instead minimize the error between the four warped images themselves, using the sum of the pairwise robust intensity constancy error estimators. This produces a single flow map, which can be used to warp the four source images to the virtual view. We currently do not reason about occlusions and simply blend the flowed images using their barycentric weights in the tetrahedron.

Figure 8 compares view interpolation results using our spatiotemporal optical flow vs. a weighted average. Because the computed flow is consistent for the four views, when the source images are warped and blended, the ball appears sharp.

5 Discussion

We have shown that using a well-constructed camera array allows one to control the image samples that are recorded from the spatiotemporal volume a scene generates, and that the sampling pattern chosen greatly affects the complexity of the view interpolation task. While in theory it is possible to simply resample a linear filtered version of the samples to generate new views, even with large numbers of inexpensive cameras, it seems unlikely one could obtain high enough sampling density to prevent either blurred images or ghosting artifacts. Instead, the correct placement of samples allows the use of simpler modeling approaches rather than none at all. The key question is, how sophisticated a model is needed and what sampling basis allows the most robust modelling methods to be used to construct a desired view? [Chai et al. 2000] et al. address this for spatial view interpolation.

For many interpolation methods, minimizing image motion leads to better quality view synthesis, so we use minimizing image motion to guide our sample placement. Given our relatively planar camera array, we use a very simple plane + parallax calibration for interpolation in space. For images aligned to a reference plane, spatial view motion results in parallax for points not on the reference plane. This motion must be balanced against temporal image motion. In our camera array this disparity motion is modest between adjacent cameras, and is much smaller than the true motion from frame to frame.

Staggering camera trigger in time distributes samples to reduce temporal image motion between neighboring views without adding new samples. In a way staggered time sampling is never a bad sampling strategy. Clearly the denser time samples help for scenes with high image motion. For scenes with small motion, the denser time samples do no harm. Since the true image motion is small, it is easy to estimate the image at any intermediate time, undoing the time skew adds little error. Since the spatial sampling density remains unchanged, it does not change the view interpolation problem at all. Better temporal sampling lets us apply relatively simple, fairly robust models like optical flow to view interpolation in time and space. We solve for temporal image motion and image motion due to parallax which improves our interpolation.

Because our flow based view interpolation methods are local, the only constraints on the camera timings are also local. They need to sample evenly in every local neighborhood. We use a simple tessellated pattern with locally uniform sampling at the interior and across boundaries. Algorithms that aggregate data from an entire array of cameras will benefit from different time stagger patterns and raises the interesting question of finding an optimal sampling pattern for a few of the more sophisticated model-based methods.

While it is tempting to construct ordered dither patterns to generate unique trigger times for all cameras there is a tension between staggered shutters to increase temporal resolution and models that exploit the rigid-body nature of a single time slice. This seems to be an exciting area for futher research.

Staggered trigger times for camera arrays increase temporal resolution with no extra cost in hardware or bandwidth, but have other limits. One fundamental limit is the number of photons imaged by the cameras if the exposure windows are nonoverlapping. The aperture time for each camera is set to be equal to the smallest time difference between the cameras. While this minimizes unintended motion blur, allowing sharp images in "Bullet time" camera motion, at some point the number of photons in the scene will be too small, and the resulting image signal to noise ratio will begin to increase. This gives rise to another dimension that needs to be explored—optimizing the relation between the minimum spacing between time

(a) (b)

Figure 8: View interpolation using spacetime optical flow. (a) Interpolated 270fps video using weighted average of four source images. (b) Interpolated 270fps video using optical flow. The four source images were warped according to the computed flow and then averaged using the same weights as in image a. No double images are present because parallax and motion for the ball were correctly recovered.

samples and the aperture of the cameras. Shechtman et. al. [2002] have done some promising work in this area, using multiple unsynchronized cameras with overlapping exposures to eliminate motion blur and motion aliasing in a video sequence. They perform a regularized deconvolution to synthesize the high-speed frames.

For our image-based methods, uniform spatiotemporal sampling limits image motion and enhances the performance of our interpolation methods. We analyzed spatiotemporal sampling from the perspective of interpolation with a constant depth assumption and related the temporal and spatial axes with maximum image motions due to parallax and time. That constant-depth assumption is one of the limitations of this work. In the future, we would like to enable more general scene geometries. Our spatiotemporal optical flow method generates significantly better results than weighted averaging, but still suffers from the standard vulnerabilities of optical flow, especially occlusions and masking.

6 Conclusion

In this paper, we show that for dense video camera arrays, sampling more efficiently temporally leads to much better spatiotemporal view interpolation results. For a given array of cameras, staggered triggers can provide increased temporal resolution without increasing the total number of samples or the frame rate of the cameras. We describe a method for computing the minimum temporal sampling rate by equating the maximum possible image motions due to parallax and temporal motion between neighboring views. Constraining image motion between neighboring views greatly aids optical flow algorithms. We present a spatiotemporal optical flow algorithm for view interpolation that uses plane + parallax calibration and knowledge of the trigger times to solve for both temporal flow and relative depth.

References

AVIDAN, S., AND SHASHUA, A. 1998. Novel view synthesis by cascading trilinear tensors. *IEEE Transactions on Visualization and Computer Graphics 4*, 4.

BERGEN, J., BURT, P., HINGORANI, R., AND PELEG, S. 1992. A three frame algorithm for estimating two-component image motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence 14*, 9, 886–895.

BLACK, M., AND ANANDAN, P. 1993. A framework for the robust estimation of optical flow. In *ICCV93*, 231–236.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. Unstructured lumigraph rendering. In *Proceedings of SIGGRAPH 2001*, 425–432.

CHAI, J.-X., TONG, X., CHAN, S.-C., AND SHUM, H.-Y. 2000. Plenoptic sampling. *Proc. ACM Conference on Computer Graphics (SIGGRAPH'00),* New Orleans, USA (Aug.), 307–318.

CHEN, S., AND WILLIAMS, L. 1993. View interpolation for image synthesis. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH'93)*, 279–288.

GORTLER, S., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. 1996. The lumigraph. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96)*, 43–54.

HARTLEY, R., AND ZISSERMAN, A. 2000. *Multiple view geometry in computer vision*. Cambridge University Press.

IRANI, M., ROUSSO, B., AND PELEG, P. 1997. Recovery of ego-motion using region alignment. *IEEE Trans. on Pattern Analysis and Machine Intelligence 19*, 3 (March), 268–272.

IRANI, M., ANANDAN, P., AND WEINSHALL, D. 1998. From reference frames to reference planes: Multi-view parallax geometry and applications. In *European Conference on Computer Vision*, 829–845.

J.-C.YANG, EVERETT, M., BUEHLER, C., AND MCMILLAN, L. 2002. A real-time distributed light field camera. In *Eurographics Workshop on Rendering*, 1–10.

KANADE, T., SAITO, H., AND VEDULA, S. 1998. The 3d-room: Digitizing time-varying 3d events by synchronized multiple video streams. Tech. Rep. CMU-RI-TR-98-34, Carnegie Mellon University.

KANG, S., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2003. High dynamic range video. In *ACM SIGGRAPH and ACM Trans. on Graphics*.

KUMAR, R., ANANDAN, P., AND HANNA, K. 1994. Direct recovery of shape from multiple views: A parallax based approach. *International Conference on Pattern Recognition*, 685–688.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proc. ACM Conference on Computer Graphics (SIGGRAPH'96)*, 31–42.

LIN, Z., AND SHUM, H. On the number of samples needed in light field rendering with constant-depth assumption. In *Proc. Computer Vision and Pattern Recognition 2000*, vol. 1, IEEE, 588–595.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S., AND MCMILLAN, L. 2000. Image-based visual hulls. In *Proceedings of ACM Conference on Computer Graphics (SIGGRAPH-2000)*, 369–374.

RANDER, P., NARAYANAN, P., AND KANADE, T. 1997. Virtualized reality: Constructing time-varying virtual worlds from real events. In *Proceedings of IEEE Visualization*, 277–283.

SEITZ, S. M., AND DYER, C. M. 1995. Physically-valid view synthesis by image interpolation. In *Proc. Workshop on Representation of Visual Scenes*.

SEITZ, S. M., AND DYER, C. R. 1997. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 1067–1073.

SHASHUA, A., AND NAVAB, N. 1994. Relative affine structure: theory and application to 3d reconstruction from perspective views. In *IEEE Conference on Computer Vision and Pattern Recognition*, 483–489.

SHECHTMAN, E., CASPI, Y., AND IRANI, M. 2002. Increasing space-time resolution in video sequences. In *European Conference on Computer Vision (ECCV)*.

VAISH, V., WILBURN, B., AND LEVOY, M. Using plane + parallax for calibrating dense camera arrays. In *Submittedto CVPR 2004*.

WILBURN, B., JOSHI, N., VAISH, V., LEVOY, M., AND HOROWITZ, M. High speed video using a dense array of cameras. In *Submittedto CVPR 2004*.