# Capturing and View-Dependent Rendering of Billboard Models

Oliver Le, Anusheel Bhushan, Pablo Diaz-Gutierrez and M. Gopi

Computer Graphics Lab
University of California, Irvine

**Abstract.** In this paper, we propose a method for obtaining a textured billboards representation of a static scene, given a sequence of calibrated video images. Each billboard is a textured and partially transparent plane into which the input images are mapped using perspective projection. Binning using Hough transform is used to find the position of the billboards, and optic flow measures are used to determine their textures. Since these billboards are correct only from specific view-points, view-dependent rendering is used to choose and display appropriate billboards to reproduce the input.

## 1 Introduction

A fundamental problem in Computer Vision is the retrieval of a geometric model from a set of images. Even when the input images are accurately calibrated, it is easy to find two different objects that yield identical images after being projected. However, some applications do not require exact reconstruction, but only a compelling reproduction, and a simple model representation. Billboards are a good approach, as they provide an unstructured, light weight representation of an object that can be combined with other rendering systems. The automatic billboard generation from calibrated video sequences is our goal in the present paper.

The problem we try to solve can be shortly stated as finding a minimal number of textured planes that reproduce the input images when seen from the appropriate position. *We define an* **equivalent geometry** *to a textured geometry $G$ for a set of viewpoints $V$ as a textured geometry $E$ that looks identical to $G$ when viewed from $V$.* The set of geometries equivalent to $(G, V)$ forms an equivalence class. Our target is to find an equivalent geometry for the static scene in the input. Finding the accurate geometry might be neither feasible nor desirable, as explained. Since our input is a calibrated video sequence of *unknown scene geometry*, our computation of billboards is purely image-based and shall not assume anything about the shape of the objects in the scene. In essence, all planes in the object geometry are perfect choices for billboards, but not all billboards belong to the object geometry.

Since the goal of stereo-based techniques is to extract exact geometry, such a reconstruction is overly restrictive for our purposes of obtaining an approximate

reproduction of the input. On the other hand, our approach tolerates certain geometric distortion between the original scene and the obtained representation, so long as the result is visually equivalent to the input.

Specifically, the following are the main contributions of this paper.

- Given a calibrated video sequence of an unknown geometry, we calculate an equivalent geometry using optical flow properties.
- We reduce the complexity of the set of billboards using Hough transform and also based on the constraint of maintaining the geometric equivalence.
- We show the use of these billboards in the reconstruction of video images from viewpoints not completely aligned with any input viewpoint.

## 2 Related Work

The trade-off between representing a scene with geometry and images is a classical conflict in image based rendering (IBR) in computer graphics. Image-based methods take a collection of images as input, construct a representation of the color or radiance of the scene, and use it to synthesize new images from arbitrary viewpoints.

Levoy and Hanrahan [1] acquire many hundreds of images, which are resampled to lie on a regular grid in a two-plane parameterization. They apply vector quantization to obtain compressed representations of light fields. Gortler et al. [2] present a similar two-plane parameterization that they call a lumigraph. They use approximate surface geometry derived from photograph silhouettes to perform a depth correction that substantially reduces ghosting and blurring artifacts.

Schirmacher et al. [3] use light field and per pixel quantized dense depth for rendering. Heigl et al. [4] use unstructured camera images to perform image-based rendering from such an unstructured data. A similar algorithm is presented by [5] where an IBR method known as unstructured light fields is designed to satisfy a set of goals that includes making use of information about surface geometry, if available. This method also generalizes many IBR algorithms, including lightfield rendering and view-dependent texture mapping. In all these methods, input images are directly used for rendering, without any resampling or reformatting to any other representation.

Another class of algorithms require geometric proxies to approximate the scene geometry and are closely related to this paper. View-dependent texture mapping [6–8] uses geometric information to re-project each input image from the desired camera viewpoint. The re-projected input images are then blended together using weights based on primarily the view direction, and possibly other factors such as sampling rate. In these works, the geometric information about the scene is provided by the user and the input set of images are retained and used during rendering from novel viewpoints. Similarly, Decoret et al. [9] automatically refactor the triangulated model into a representation purely based on billboards. Our system extracts the required equivalent geometry, and the input video sequence, once reindexed into this geometry, is not used any longer.

These are the primary differences between our work and view-dependent texture mapping techniques. A formal analysis of the trade off between the number of images from different camera positions and the fidelity of geometry in order to produce satisfactory IBR from novel viewpoints is presented in [10].

For compression of light field data, Magnor and Girod [11, 12]develop an MPEG-like scheme to compress two-plane lightfields that achieves better compression ratios than those obtained by Levoy and Hanrahan. Further in [13], they use the accurate scene geometry to develop two methods for multi-view coding: one is the texture-based coding where the scene geometry is used to convert images to view-dependent texture maps and the other is the model-aided predictive coding where the geometry is used for disparity compensation and occlusion detection between images. In either case, the geometry of the scene has to be provided as input. Further, the representation is used for multi-view coding and rendering of scenes and not directly applicable for video compression.

The final category of IBR algorithms require accurate geometry of the scene. Surface light field Nishino et al. [14, 15] uses the user specified surface of the object to index and compress the view dependent radiance information. The accurate geometry of the object is used for two purposes. The colors indexed on the surface of the object would exhibit high degree of coherence, and the rendering of the object can use the traditional geometry rendering pipeline.

In this paper, we use ideas similar to those used in image based modelling to compress video of static scenes; but with additional complexity of extracting the *equivalent geometry* automatically from the video sequence.

## 3 Optic Flow as a measure of geometric proximity

Our goal is to find the *equivalent geometry* of a model as a set of billboards. Instead of operating at a per-pixel level, we obtain geometric proxies for contiguous pixels regions. A region on a plane that is proximate to geometry has the property that texture projected from multiple input viewpoints will match in the region. So, we apply projective texturing to map the input images onto the billboard and then use texture similarity as a measure of geometric proximity.

We can use different measures of texture matching: Color distance at each pixel is noisy and unreliable, while using intensity correlation can return good results, but is expensive. We use optical flow between the projected images as a measure of texture matching. Regions with low optic flow project similar textures from the input viewpoints and thus are candidates for being part of the *equivalent geometry*. We use an optic flow algorithm for texture matching. Given the optic flow value, we perform an adaptive thresholding to get regions of low flow, based on the location of the cameras and the billboard.

## 4 Algorithm description

Our algorithm takes a set of calibrated images of a video sequence and produces an *equivalent geometry* for the represented scene. That is, a set of textured planes, or billboards, that look like the input when seen from the input viewpoints. The

process comprises two fundamental steps, repeated until the input data has been exhausted. These steps are the spatial sampling of the input scene and the search for fitting planes. In the first one, the geometry of the objects in the input scene is sampled using measures of optic flow. In the second step, we find a set of planes that fit the approximate sampled geometry obtained in the first step.

### 4.1 Spatial sampling of the input scene

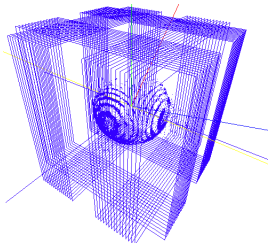

**Fig. 1.** Grid of planes for dense spatial sampling of the input geometry.

The first step in the process of finding an equivalent geometry sampling the input scene to calculate an estimate of the geometry contained in the input. Such sampling can be as well split in two stages. First we must roughly identify the area where the input geometry is to be located. In the second one, we actually sample that geometry. The first sampling stage can be accomplished in a number of ways. Under constrained conditions, such as controlled video capture or when using synthetic models like in this paper, the approximate location of the object might be already known, and this step can be skipped.

The second stage in the sampling procedure is a dense sampling of the volume resulting in the first stage. In order to extract a set of points near the input geometry, we place a regular grid of parallel planes(see Figure 1) along the three dimensions, spanning all the sampling volume from previous stage.

For each of those planes, we project pairs of images from the input viewpoints and use an optic flow measure, as explained in Section 3, to determine which points on the plane lie nearby the input geometry. Points on the plane with an optic flow value under a threshold are treated as being near the input geometry. After this operation has been performed on all sampling planes, we triangulate the obtained 3D points (see Figure 1) to produce an approximation to the surface in the input scene. This set of triangles will be the input to the next part of the algorithm, described in the following subsection.

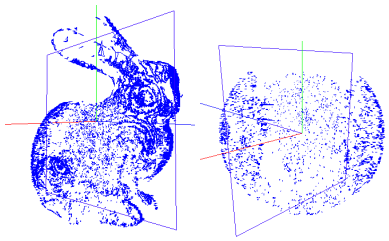

**Fig. 2.** Triangulated approximations to the input geometry.

### 4.2 Finding fitting billboards

This step in the algorithm takes a set of triangles near the surface, and returns the equivalent geometry for the input scene. We say a plane fits a triangle if all its three vertices lie within a user defined distance from the plane. The equivalent geometry is made of planes fitting the triangles that approximate the geometry. To find a set of planes that fit a large number of triangles, we use a variant of the stochastic method RANSAC: Iteratively, we choose three points from different

triangles in the surface approximation. These three planes define a plane $P$ that fits $n$ triangles from the surface. $n$ is a measure of the *goodness* of $P$ fitting the data. RANSAC methods guarantee that a good solution will be reached with high probability after a certain number of samplings are done. After we have a set of good fitting planes, we do a local optimization of the position and orientation of the candidate planes to maximize surface coverage.

Finally, we need to texture the obtained planes, for which we calculate again the optic flow of the projected images on the planes, as described in Section 3. Each plane is then textured with the projection of one input image. In order to reduce texel distortion, the input image with the most normal orientation with respect to the plane is chosen.

The two described steps of the algorithm are repeated several times, with a number of new billboards added to the result in each iteration. Every time a billboard is added to the result, part of the input is *covered*. Covering of an image is defined as follows. Given an input image $I$, a pixel $I(x, y)$ with color $c$ in $I$ is said to be covered when visualizing the billboards onto $I$ produces an image with color $c$ at position $(x, y)$. Covered pixel regions are removed from the input images, and when all the input pixels have been covered, the algorithm terminates and the set of billboards is returned.

## 5 Results and conclusion

Although the algorithm described in Section 4 finishes after a bounded number of iterations, convergence is slow. To speed up the process, we associate a quadtree that keeps track of the covered regions in each input image. This way, the termination condition is checked more effifiently. Furthermore, increasing the granularity of billboard texturing from one pixel at a time to block sizes reduces the run time at the cost of a small decrease in the quality of results, like those in Figure 4.

We have introduced a novel method to construct a billboard-based representation of an unknown-geometry object from a sequence of calibrated images. We demonstrate it with a number of pairs of images from the input and the results. Planned future work includes an accurate sampling volume estimation, a more structured billboard identification and the necessary modifications to handle more general models, such as a room or an open space.

## References

1. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings SIGGRAPH 96, ACM SIGGRAPH (1996) 31–42
2. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings SIGGRAPH 96, ACM SIGGRAPH (1996) 43–54
3. Schirmacher, H., Heidrich, W., Seidel, H.P.: High-quality interactive lumigraph rendering through warping. In: Graphics Interface. (2000)
4. Heigl, B., R.Koch, Pollefeys, M., Denzler, J., Gool, L.: Plenoptic modeling and rendering from image sequences taken by hand-held camera. In: Proceedings of DAGM. (1999) 94–101

5. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings SIGGRAPH 2001, ACM SIGGRAPH (2001) 425–432
6. Debevec, P., Taylor, C., Malik, J.: Modelling and rendering architecture from photographs: A hybrid geometry and image based approach. In: Proceedings SIGGRAPH 96, ACM SIGGRAPH (1996) 11–20
7. Debevec, P., Yu, Y., Borshukov, G.: Efficient view-dependent image-based rendering with projective texture-mapping. In: Eurographics Rendering Workshop, Eurographics (June, 1998) 105–116
8. Pulli, K., Cohen, M.F., Duchamp, T., Hoppe, H., Shapiro, L., Stuetzle, W.: View-based rendering: Visualizing real objects from scanned range and color data. In: Proceedings Eurographics Rendering Workshop 97, Eurographics (1997) 23–34
9. Décoret, X., Durand, F., Sillion, F.X., Dorsey, J.: Billboard clouds for extreme model simplification. ACM Trans. Graph. **22** (2003) 689–696
10. Chai, J.X., Tong, X., Chan, S.C., Shum, H.Y.: Plenoptic sampling. In: Proceedings SIGGRAPH 2000, ACM SIGGRAPH (2000) 307–318
11. Magnor, M., Girod, B.: Adaptive block-based light field coding. In: Proc. 3rd International Workshop on Synthetic and Natural Hybrid Coding and Three-Dimensional Imaging. (September,1999) 140–143
12. Magnor, M., Girod, B.: Hierarchical coding of light fields with disparity maps. In: Proc. IEEE International Conference on Image Processing. (October, 1999) 334–338
13. Magnor, M., Ramanathan, P., Girod, B.: Multi-view coding for image-based rendering using 3-d scene geometry. IEEE Transactions on circuits and systems for video technology **13** (November, 2003) 1092–1106
14. Nishino, K., Sato, Y., Ikeuchi, K.: Appearance compression and synthesis based on 3d model for mixed reality. In: Proceedings of IEEE ICCV, IEEE (September, 1999) 38–45
15. Nishino, K., Sato, Y., Ikeuchi, K.: Eigen-texture method: Appearance compression based on 3d model. In: Proceedings of Computer Vision and Pattern Recognition. (June, 1999) 1:618–624

**Fitting plane selection** (set of triangles $\mathcal{T}$)
*current_best_plane* = random_plane()
**do** *no_of_ransac_iterations* times:
select 3 random triangles from $\mathcal{T}$.
let $\pi$ be the plane through the center of these random triangles.
**if** goodness($\pi$) > goodness(*current_best_plane*),
*current_best_plane* = $\pi$.
Return *current_best_plane*.

**Fig. 3.** RANSAC algorithm for finding a plane fitting a set of triangles.

**Fig. 4. Left:** Input images for the Blob and Armadillo scenes. **Right:** Resulting billboard sets for the same scenes.