Interactive Image-Based Rendering Using Feature Globalization

Daniel G. Aliaga — Lucent Bell Labs
Dimah Yanovsky — Harvard University
Thomas Funkhouser — Princeton University
Ingrid Carlbom — Lucent Bell Labs

Abstract

Image-based rendering (IBR) systems enable virtual walkthroughs of photorealistic environments by warping and combining reference images to novel viewpoints under interactive user control. A significant challenge in such systems is to automatically compute image correspondences that enable accurate image warping.

In this paper, we describe a new algorithm for computing a globally consistent set of image feature correspondences across a wide range of viewpoints suitable for IBR walkthroughs. We first detect point features in a dense set of omnidirectional images captured on an eye-height plane. Then, we track these features from image to image, identifying potential correspondences when two features track to the same position in the same image. Among the potential correspondences, we select the maximal consistent set using a greedy graph labeling algorithm.

A key feature of our approach is that it exploits the multiple paths that can be followed between images in order to increase the number of feature correspondences between distant images. We demonstrate the benefits of this approach in a real-time IBR walkthrough system where novel images are reconstructed as the user moves interactively.

Keywords: Image-based rendering, reconstruction, interactive, correspondence, image features.

1. INTRODUCTION

One of the most challenging problems in computer graphics today is rendering visually compelling 3D environments. For real-world environments, a key problem is to recreate the complex interaction between geometric and photometric properties. Image-based rendering (IBR) addresses this problem by densely capturing images of the environment and then creates novel views by re-sampling the images that already contain the geometric and photometric properties [Max95, McMillan95].

The main tasks for an IBR system are: (1) to acquire a dense sampling of calibrated reference images, (2) to map image samples from reference images to the view plane of the virtual observer, and (3) to combine the mapped image samples from multiple reference images to form a novel image. The first and third of these tasks have been well studied in recent years. For instance, several papers have described how to acquire a dense set of reference images for real-world scenes, and related papers have described how to blend samples from the reference images during reconstruction of novel images. However, the second task (mapping image samples from one viewpoint to another) has still not been addressed adequately – it requires computing either pixel correspondences or accurate 3D scene geometry. As a result, most IBR systems have produced novel images with noticeable blurring and ghosting artifacts, or they have been demonstrated only with distant or synthetic scenes.

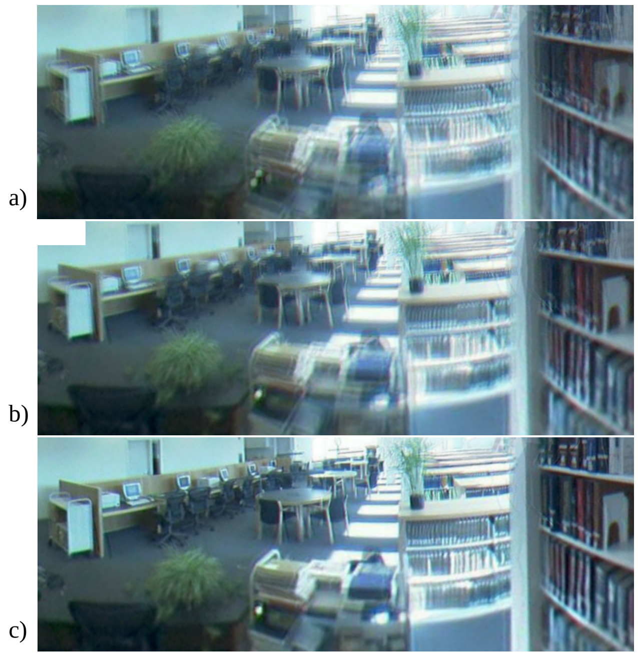


Figure 1. ***Feature Globalization.*** *We show cylindrical projections reconstructed for a novel view of a captured environment using one of three methods. (a) Simply blending together neighboring references images. (b) Using a proxy to warp and blend reference images. (c) Using our approach that combines feature tracking with the construction and labeling of a correspondence graph, enabling correspondences over a wide range of viewpoints and producing high quality reconstructions without requiring dense depth information or a full 3D reconstruction.*

The goal of our work is to develop a method for mapping image samples so as to enable high-quality image reconstruction in interactive IBR walkthrough systems. We would like the method to be: (1) able to produce novel images without noticeable ghosting or blurring artifacts, (2) robust to inaccuracies in camera pose estimates, (3) robust to inaccuracies in 3D proxy models, (4) fully automatic, and (5) able to warp any combination of images to form a novel image. This last goal is motivated by the real-time needs of an IBR system where a pre-fetching process may not be able to load all images into memory quickly enough to keep up with the virtual observer motion. In these cases, the reference images captured from locations closest to the observer viewpoint may not be available in memory, and the system must warp and combine more distant images from its memory-resident cache.

Our approach creates novel views by warping and combining a set of omnidirectional images densely captured throughout a plane. Features detected in each reference image are tracked to other images within a local region. Then, tracked features are matched in every image -- anytime two features track to the same location in the same image they are flagged as a potential correspondence.

A greedy algorithm is then used to select the best subset of potential correspondences while ensuring a consistent global feature labeling. This approach allows us to find feature correspondences across a wide range of viewpoints in difficult scenes (e.g., with occlusions) where many features would be lost by a typical feature tracker. Moreover, it allows us to produce high-quality novel images in a walkthrough system, even when reference images are separated by large distances and/or have poor camera pose estimates (Figure 1).

The main contributions of this paper are:

- **Algorithm for feature globalization:** we describe a new approach for finding correspondences between features in images spread over a wide range of viewpoints. Using a graph formulation over a dense "sea of images," our algorithm is able to find more feature correspondences than traditional algorithms based on tracking of features along a single sequence of images.
- **Evaluation of image warping methods:** we compare the results of rendering novel images by simply blending reference images, warping them according to a coarse 3D proxy, and warping them based on globalized feature correspondences. We find that warps based on globalized feature correspondences produce fewer ghosting and blurring artifacts than the other methods.
- **Interactive IBR walkthrough system:** we describe how to include globalized feature correspondences in an interactive IBR walkthrough system. We find our system can produce novel images at 15 to 20 frames per second as a user interactively controls the simulated observer viewpoint.

The rest of the paper is organized as follows. In the next section we present a summary of background and related work. Section 3 describes our feature globalization algorithm, while Section 4 presents our image reconstruction method. Section 5 contains some of the implementation details. Experimental results and examples appear in Section 6. Finally, we conclude and present ideas for future work in Section 7.

2. BACKGROUND & RELATED WORK

There are several possible approaches to warping image samples in an IBR system. The simplest approach is just to interpolate images without any warping [Levoy96] -- this method is sufficient only if the plenoptic function is sampled extremely densely. A second approach is to re-project image samples according to depths derived from an approximate 3D model (or proxy) [Debevec96, Hanrahan96, Buehler01] -- this method works well only for very detailed proxy models, which are difficult to obtain. A third approach is to estimate the depth value [Nyland01, Levoy00] or pixel correspondence [Chen93, McMillan95, Kang96] for every pixel in every image -- these methods produce the most accurate warps, but current methods for automatic camera calibration and depth acquisition have difficulties in environments with specular and occluding surfaces.

Another approach is to find and use correspondences between a relatively small set of distinctive image features (e.g., corners) and to define a warp based on their correspondences (with linear interpolation between features) -- this approach works well only if a large number of features correspondences can be found, which has previously been possible only for closely located reference viewpoints. Finding feature correspondences over a wide region of viewpoints is difficult for several reasons:

- **Feature tracking limitations:** feature drift and visibility changes in the environment make robust tracking of long sequences difficult. This difficulty thwarts simple algorithms that try to detect features in one image and track them to other images over a wide region.
- **Feature detection limitations:** because of occlusions, lighting differences, and jitter due to the sampled-nature of images, a feature detector may select significantly different features in different images of the same scene, even when the viewpoints are very close. This difficulty hinders simple algorithms that try to match features detected in neighboring images.

Several people have addressed feature correspondence in 3D reconstruction. For example, structure-from-motion approaches [Morita94] use a linear sequence of images to estimate global camera pose and the global position of a set of sparse features. Perhaps the work most similar to ours is by Pollefeys et al [Pollefeys98]. Their algorithm detects features in images of a video sequence and tracks them to later images while relabeling the ones that correspond. Our approach differs from theirs in two ways. First, we start from calibrated omnidirectional images arranged on a 2D plane, which allows us to track features between images in multiple directions, not just from earlier images to later ones in a linear sequence. Second, we cast feature globalization as a graph-relabeling problem, which allows us to use algorithms to consider potential feature correspondences in best-first order, rather than in the order in which they are detected in the video. As a result, we expect to find more/better feature correspondences.

Our work is motivated by the "Sea of Images" IBR system described by Aliaga et al. [Aliaga02]. They capture omnidirectional images on an eye-height plane and store them in a hierarchical structure suitable for real-time memory management. As a virtual observer walks through the environment, they pre-fetch and warp reference images using a 3D proxy to form novel images. Unfortunately, since they rely upon a coarse 3D proxy model for warping image samples, and their camera pose estimates have errors, the system often reconstructs images with ghosting artifacts. One of the goals of our work is to develop better warping algorithms for IBR systems of this type.

3. FEATURE GLOBALIZATION

The focus of this paper is an algorithm for finding a set of feature correspondences in a large collection of images suitable for warping reference images in an interactive IBR system. The input to our algorithm is a set of omnidirectional images captured densely on an eye-height plane throughout a large environment and calibrated with position and orientation information [Aliaga02]. The output is a set of feature positions for each image along with a globally consistent labeling, where two features have the same label if they correspond to the same scene element. The features common to a group of images define a warp for the reconstruction of novel views in an interactive IBR walkthrough system.

The simplest approach would be to detect features in one "source" image (e.g., in the middle of the environment) and track them to all other "destination" images. Obviously, this approach would only work for a small range of images around the source, as features quickly become lost due to occlusions, lighting changes, and feature drift.

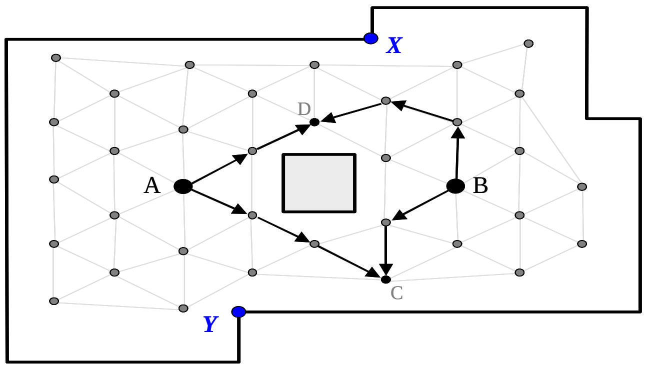


Figure 2. Multiple Tracking Paths*. This figure represents the floor plan of an example environment. Each dot corresponds to a reference image. There are multiple viewpoint paths to track the features X and Y from image A to image B (or vice versa). It is difficult to track both features in the same path (e.g., because of obstacles, changing visibility, and feature tracking limitations). Our algorithm finds more features common to two images by taking advantage of the redundancy of the multiple viewpoint paths.*

Our approach is to detect features in many source images and track features from each source image only to destination images within a local area (i.e., where tracking is relatively robust). Then, we identify potential correspondences whenever two features track to the same location in the same destination image. We use an iterative feature-relabeling algorithm to extract a globally consistent set of correspondences over the entire environment. This approach overcomes the limitations of feature tracking because it relies upon tracking only over short distances. Yet, it identifies feature correspondences over a large range of viewpoints because feature relabeling propagates correspondences across regions.

A key feature of our approach is that two features are said to correspond if they track to the same location in *any* destination image via any of the multiple viewpoint paths. For example, consider trying to find the correspondences for two scene features, X and Y, between two images, A and B, among a sea of images (black dots) captured throughout the environment shown in Figure 2. Due to occlusion of the obstacle in the middle of the room, it is difficult to track both features along any single path of viewpoints between images A and B. Feature X may track along a sequence of images captured on one side of the obstacle, and feature Y may track along a different sequence of viewpoints on the other side of the obstacle. However, no single path can track both features all the way from one image to the other. Moreover, features are lost along any path due to changes in lighting or sampling. Our algorithm is able to find the correspondence for both X and Y because it matches features if they track to the same location in any image (e.g., C or D). This redundancy allows our algorithm to find potential feature correspondences more robustly and across a wider range of viewpoints than traditional feature tracking methods.

In the following three sections, we explain the main components of our algorithm. First, we describe the feature propagation method that starts with features detected in a set of source images and tracks them radially outwards to destination images. Second, we elaborate on our feature-relabeling algorithm. Third, we describe several practical optimizations that enhance the algorithm performance. The entire algorithm is summarized in Figure 3.

```
// Initialize
Build Delaunay triangulation
Initialize priority queue

// Detect features
for each image A
 Detect features F
 if (quality(F) > threshold)
  Insert F into A

REPEAT
 // Propagating features
 for each image A
  for each [neighbor] image B
    for each untracked feature F in A {
      Track F to B
      if (track quality > threshold) {
        for each feature G in B
          if (match (F,G) > threshold)
              Insert (F,G) into priority queue
        if (F matched no features of B)
          Insert F into B
      }
    }

 // Relabeling features
 While priority queue is not empty {
  Pop “best” potential correspondence (F,G)
  If (consistent(F,G)) Relabel G->F
 }

UNTIL no untracked features or max iterations
```

Figure 3. Feature Globalization Summary*. Pseudocode that summarizes both feature propagation and feature relabeling.*

3.1 Feature Propagation

There are multiple image paths along which detected features can track and propagate outwards from each source image. Our algorithm attempts to minimize feature-tracking error by using a shortest-path traversal of the 2D Delaunay triangulation of the image viewpoints. Starting with a source image, feature tracking extends outwards from each image along a set of disjoint *tracking paths* (Figure 4a) until either no more features can be reliably tracked or a maximum number of images have been traversed.

Our algorithm initializes feature propagation by detecting features in every source image using the algorithm of Shi and Tomasi [Shi94]. We assign each feature a unique *label*, mark it as *untracked* in its image, and insert it into the *feature list* associated with its image. Figure 4b shows an example image A with a few labeled features. (The actual reference images are omnidirectional images – we show planar re-projections for clarity).

To propagate the features, our algorithm iteratively tracks features to the next neighboring image along every tracking path. In figure 4c, we show the next neighboring image B along one tracking path from image A. To obtain candidate correspondences for this neighboring image B, we determine the subset of untracked features in A with labels different than those already appearing in B. For each feature in that subset (e.g., f_1, f_2, f_3), we track it to image B. If the track quality is above a user specified threshold, we consider merging the tracked features into the set of features associated with image B. If a feature f_i tracks to a position within a user specified distance threshold ε of another feature g_j in image B, then we consider the pair (f_i, g_j) as a potential correspondence. Otherwise, we simply add the tracked features to the feature list of image B and mark them untracked in image B.

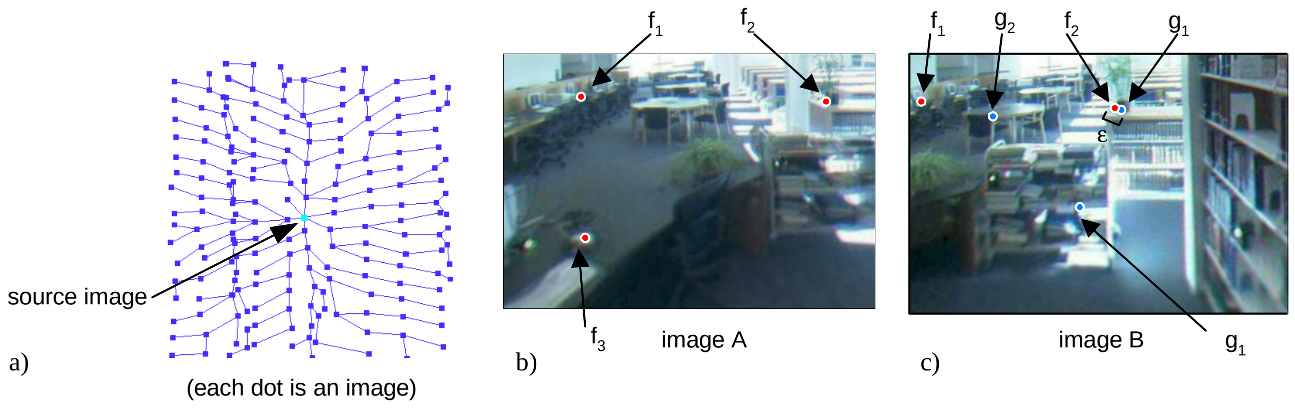


Figure 4. Feature Propagation. *(a) Using the edges of a 2D Delaunay triangulation of the image viewpoints, we track outwards from each source image along disjoint paths. (b) For a source image A, we detect features, such as f_1,f_2, and f_3 and add them to the untracked list of features for that image. (c) Then, we iteratively track all untracked features to the next neighboring image B along each disjoint path. If a successfully tracked feature (such as f_1 or f_2) is within ε of an existing feature in an image B, the pair (f_1,g_1) is potentially corresponded.*

3.2 Feature Relabeling

After every image has tracked features to its neighbors, we are left with a list of potential feature correspondences (i.e., pairs of features that track to the same location in some image). Since there is no guarantee that a feature detected in a source image will be tracked to the same position through different paths to a destination image, inconsistencies may appear in which two distinct features in one image are thought to correspond to the same feature in another image.

We resolve these inconsistencies by casting the problem of finding the best globally consistent set of feature correspondences as a graph-labeling problem. Using a greedy graph-labeling algorithm, we "accept" potential correspondences in best-first order. The best correspondence is that of the feature pair closest to each other and with the highest track quality. Initially, we create a graph in which each vertex represents a unique feature detected in a reference image and each edge represents a potential correspondence between two features. Using a priority queue, we iteratively pop the best potential feature correspondence off the queue and check to see whether "accepting" it produces a consistent labeling (i.e., no two features in the same image have the same label). If so, the potential correspondence is accepted and the features are "merged" into one. We use the lower-value label as the new label.

Figure 5 shows an example of our graph algorithm on a simple example with three images (dotted ovals), seven detected features (black dots), and many potential correspondences (dashed lines). The highest priority potential correspondence associates the two top-most features, and accepting this correspondence does not produce an inconsistent labeling. So, we relabel these two features (1) and remove other potential correspondences that would obviously produce inconsistencies (Figure 5b). The process continues accepting potential correspondences in priority order (Figure 5c-d) until no more can be found. Sometimes, the highest priority potential correspondence remaining would produce an inconsistent labeling (Figure 5e), in which case it is rejected. Finally, at the end of the algorithm, every feature has been given a unique global label and feature correspondences are indicated by the remaining (dark solid) edges in the graph. Of course, this example is simple for exposition purposes only. The graphs computed for our datasets typically have hundreds of clusters and thousands of vertices per cluster.

3.3 Practical Optimizations

In theory, this basic algorithm can find the largest, globally consistent set of labeled features (if we detect features in every image and track them to every other image before any features are relabeled). However, feature tracking is very compute intensive, and a naive implementation of this algorithm would take months of compute time for our data sets (~10,000 images with ~1,000 features per image). So, instead, we employ several optimizations that result in a more practical algorithm.

First, in order to reduce the effects of feature jitter and to reduce the number of features to track, we detect features only in a subset of the images. Specifically, we partition the images into *regions* and detect features only at the central image within each region (Figure 6). Of course, the size of the regions has great impact on the performance and success of the algorithm. On one extreme, if every image is in its own region, then we detect features at every image, and the system must track a large number of features. On the other extreme, if all images are in a single region, then features are detected and tracked from only one image. In our system, we choose region sizes by estimating the distance along which features can be tracked reliably. This approach initializes all images within each region with a large set of corresponded features and allows the algorithm to begin by considering potential correspondences at the boundaries between regions.

Second, in order to avoid unnecessary feature tracking, we alternate between feature tracking and feature relabeling (Figure 3). Rather than tracking features as far as possible before resolving potential correspondences, we interleave tracking features to immediate neighbor images and resolving potential correspondences with the greedy relabeling algorithm. This approach avoids further tracking of features that can be merged with other features at the same location in the same image.

These two practical optimizations not only reduce the compute time required by our algorithm, but they also help its stability. Detecting features in only one image of each region avoids much of the noise typically found in feature detectors, and merging/relabeling nearby features reduces the chances of erroneously finding correspondences between tracked features.

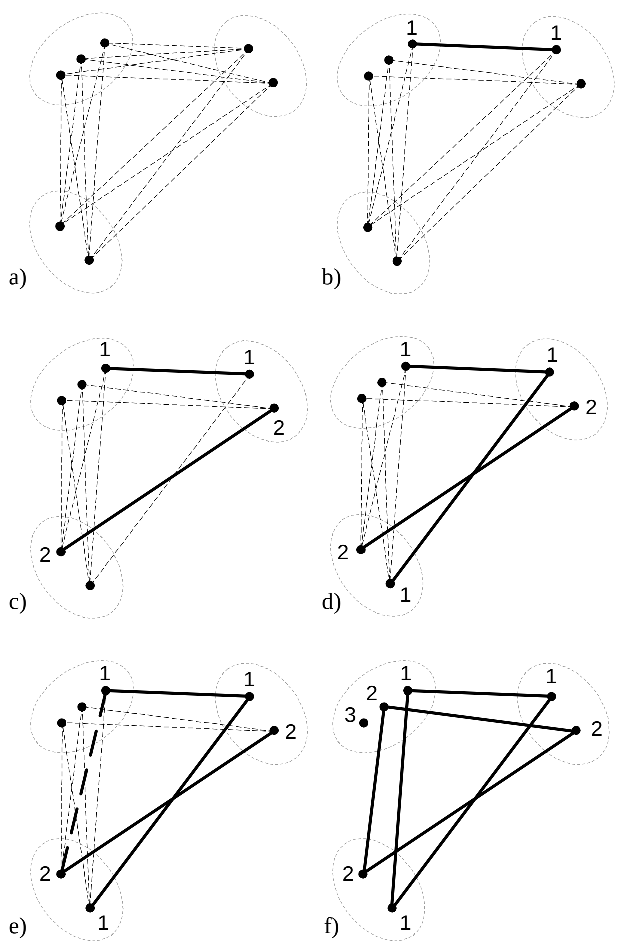


Figure 5. Graph Construction*. A graph labels features and encodes correspondences. This figure shows a sequence of steps to construct the correspondence graph for 3 images. (a) The initial graph contains all proposed correspondences edges between adjacent images. (b) The algorithm selects the best correspondence and labels the vertices "1". The other proposed correspondence edges for the same vertices are removed. (c) The algorithm selects the next best correspondence edge and labels the vertices "2". (d) The algorithm selects a third edge but connects the vertex to the previous connected component of label "1". (e) The algorithm attempts to accept a correspondence edge that yields an inconsistent assignment of vertices: features "1" and "2" are distinct features in the upper right image but are being considered equivalent features in the upper left image. (f) After several more steps, the final correspondence graph is obtained. Notice that a feature is not necessarily present in all images (e.g., feature "3").*

The net result is a practical and automatic algorithm for finding a globally consistent set of image features.

4. IMAGE RECONSTRUCTION

Once we have a globalized set of features, a novel view is generated by warping and combining surrounding reference images. Building on the approach of Aliaga et al [Aliaga02], we triangulate the viewpoints of the reference images and find the triangle that contains the novel viewpoint. Then, we warp the three reference images to the novel viewpoint. Since we have a globalized set of features, we are not limited to combining only adjacent images. If only a sparse subset of the captured images is available, the images will still have common features.

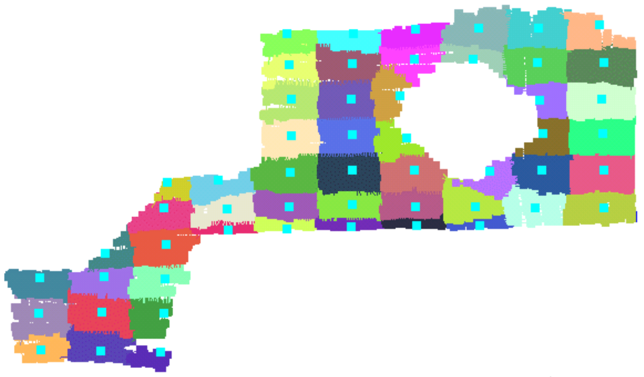


Figure 6. Image Regions*. The algorithm divides the plane of images into regions. For each region, feature tracking is initialized. Later, features from adjacent regions are iteratively corresponded and globalized.*

The algorithm finds the common features by iterating through all the features of one reference image and determining the subset present in the other reference images. In addition, we store with each feature, the original feature "quality" [Shi94], the feature tracking correlation (i.e., tracking quality), and the iteration at which the feature was globalized. If so desired, only features that exceed set quality and maximum iteration thresholds are used for reconstruction.

Since features are only present at or near image details, some large parts of an image might have few or no features. This creates large triangles that would not warp and render the image as desired. We dynamically subdivide these large triangles and insert feature points. To calculate the correspondences for these feature points, we fall back to using a geometric proxy.

5. IMPLEMENTATION DETAILS

Our algorithm is implemented in C/C++ using, OpenGL, GLUT, and GLUI on both a Pentium IV 1.7 MHz computer with an NVidia GeForce3 Ti 500 graphics card and on a SGI Onyx2 R10000 250Mhz computer with InfiniteReality2 graphics.

Images and features are sorted and stored in a database. We store with each image the list of features it contains and with each feature the list of images that contain it. Both of these access mechanisms are used during feature globalization.

We used our approach with two feature tracking algorithms. Initially, we used the publicly available KLT tracking library [Tomasi91]. This library was too costly for tracking thousands of features in 1024x1024 resolution images, such as ours. We then developed a custom feature tracking method. Our method estimates the position of the features to track in the destination image. This can be accomplished either by using the camera pose to predict the displacement of the feature along its epipolar curve or by using a geometric proxy to predict the feature displacement. Then, the method searches in the neighborhood of the predicted position for the tracked feature that best correlates with the original feature.

The reconstruction method is implemented by blending triangular meshes using either multi-texturing on the NVidia board or the accumulation buffer on the SGI. After extracting the common set of globalized features for a given group of reference images, the reconstruction method triangulates the features in one image and generates a mesh. Using each reference image's feature positions and the distance from the novel viewpoint to each reference

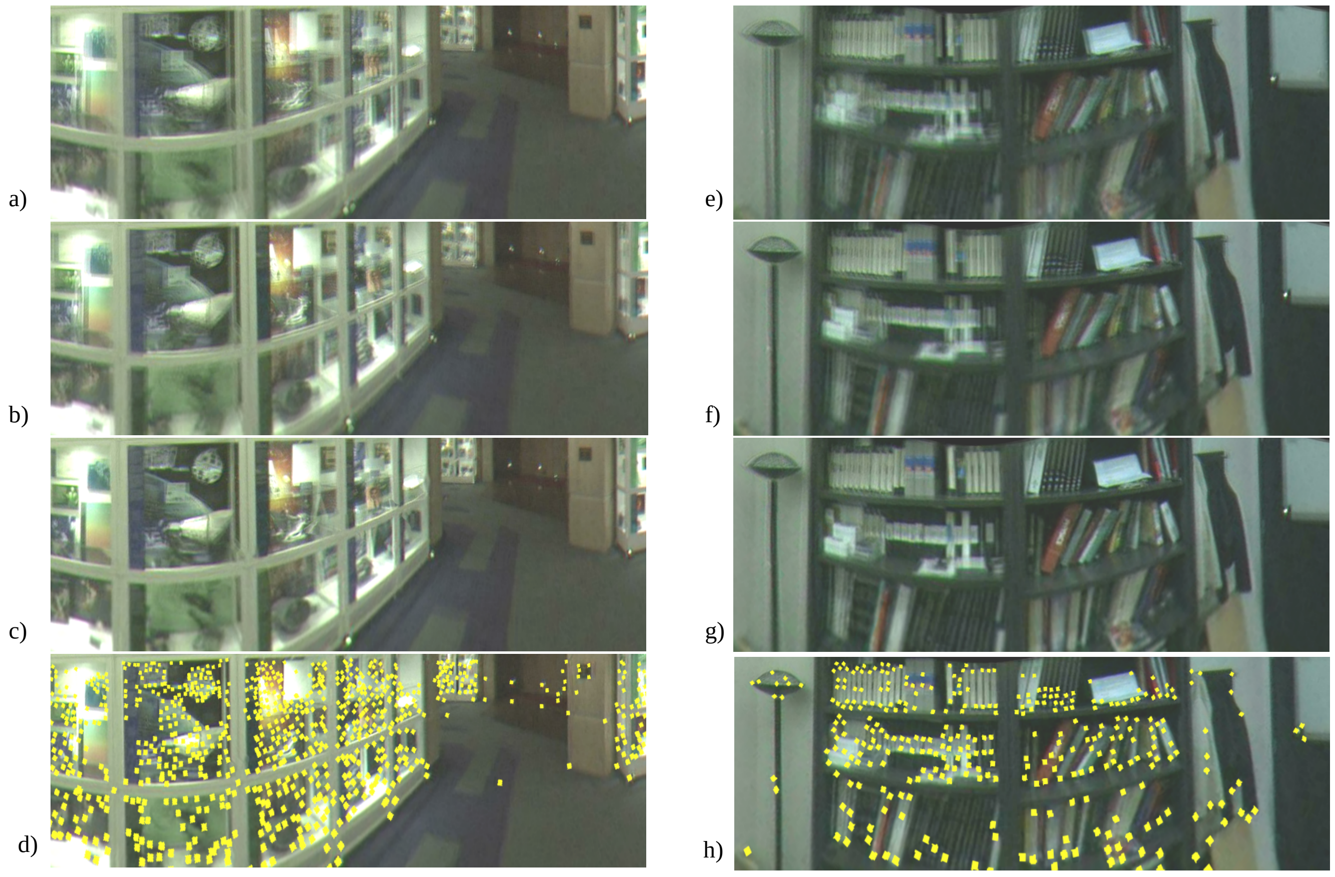


***Figure 7. Reconstruction Comparison**. Reconstructed cylindrical projections for novel viewpoints use one of three reconstruction methods. Novel viewpoints are approximately equidistant from the surrounding three reference images. Images a-d are from the Museum environment; images e-h are from the Office environment. (a, e) Neighboring references images are blended, yielding the most amount of ghosting. (b,f) A proxy is used to warp and blend reference images. Quality depends on proxy accuracy and camera pose estimation. (c, g) Feature globalization helps to warp and combine images. This approach is better able to correspond image details and less dependent on camera pose accuracy. (d,h) The globalized features are highlighted with yellow dots.*

image's viewpoint, the reconstruction method creates interpolated feature positions. Each reference image is then downloaded to the graphics engine and rendered as a textured mesh. A reference image is warped by rendering the mesh using the interpolated feature positions as vertices but using the original feature positions as the texture coordinates.

6. RESULTS

In this section, we present several results and observations about feature globalization. For our experiments, we use images captured in three real-world environments: a single-person office, the lobby of a library, and a museum covering almost 1000 square feet. The three datasets contain a total of 15,000 omnidirectional images captured in a plane (each image of 1024x1024 pixels) and the average distance between images is 1.5 inches [Aliaga02]. An omnidirectional camera mounted on a motorized cart captures the images. The camera pose for each image is derived using fiducials placed in the environment. Table 1 summarizes the image datasets.

6.1 Example Reconstructions

Figure 7 shows images reconstructed using simple blending, a proxy, and globalized features. All images are from viewpoints located approximately equidistant from the surrounding reference images. Simple blending produces the most amount of ghosting (Figures 7a, 7e). To prevent ghosting with this approach, images must be captured more densely, which for large environments can be exceedingly difficult. A proxy improves the correspondence between reference images, but the accuracy of the proxy and of the camera pose dictates the final quality (Figures 7b, 7f). Feature globalization is able to match details a proxy leaves out and is able to compensate for inaccuracies in camera pose estimation (Figures 7c, 7g). Figures 7d and 7h highlight the features.

In addition to improving the reconstruction quality when using the full set of captured images, feature globalization also allows us to establish correspondences between more distant images. As a practical matter, this allows us to produce novel images without noticeable artifacts using a sparser set of reference images. Figure 8 shows several images generated from sparser sets of reference images. Each reconstructed image uses a subset of the reference images approximately evenly distributed throughout the

Dataset	**No. Images**	**Size**	**Avg. Image Spacing**
Museum	9832	900 sq. ft	2.2 inches
Office	3475	30 sq. ft	0.7 inches
Library	1947	120 sq. ft	1.6 inches

***Table 1. Dataset Summary**. This table summarizes the datasets used: number of images, environment size, and average image spacing.*

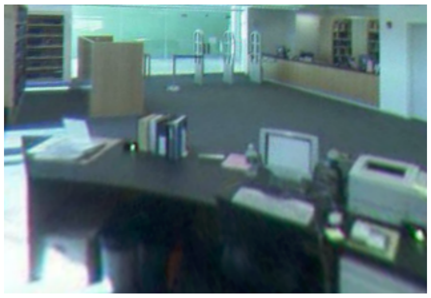
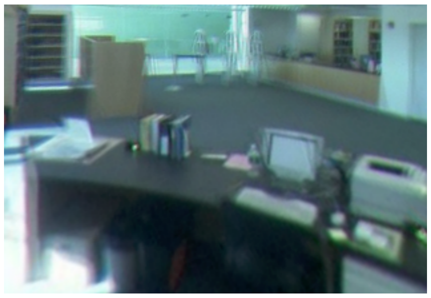
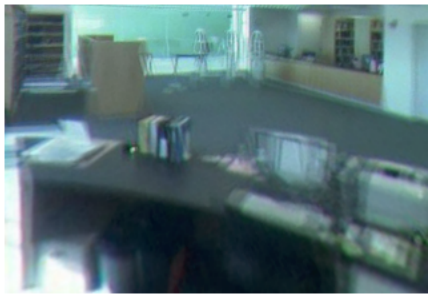
a)  b)  c) 

Figure 8. Reconstructions Using Sparse Reference Images*. These images of the Library environment are reconstructed using features globalized with small initial regions and at most 5 globalization iterations. Each example reconstruction uses a regularly-spaced subset of the original reference images: (a) the full set of 1947 reference images, (b) a subset of 153 images (13:1 reduction in images), (c) a subset of 89 images (22:1 reduction in images). For this environment, we can typically reduce the number of reference images to a few hundred with little impact on quality. There is some visible ghosting and blurriness in (b), in particular around the computer monitor and counter area in the distance. Further reference image reduction (c) often produces noticeable artifacts.*

environment. The reconstructed image is from a viewpoint near the middle of the space between the reference images. In this example, feature globalization allows us to increase the spacing between reference images by a factor of approximately 13:1 without significantly affecting quality.

6.2 Tracking

As with other tracking methods, feature drift and visibility changes in the environment make tracking long sequences difficult. The tracking requirements for feature globalization are stricter than those for typical feature tracking. We would like to detect features in a source image, track them to a destination image, and have all the tracked source features match the detected features of the destination image. Once we know through how many images we can satisfactorily track, we can gauge maximum region sizes and maximum tracking sequence lengths.

Figure 9 shows a graph with two curves that measure the performance of tracking for globalization. Each data point is the averaged result of traversing outwards from images throughout the environment and comparing the tracked features to detected features, for all three environments. The horizontal axis represents distance (in images) from a source image. The vertical axis represents number of features. Starting with the detected features of a source image, the top curve indicates the number of successfully tracked features. The bottom curve shows how many of the tracked features match newly detected features for that image. The two curves are not equal for two reasons: (1) feature detection jitter causes an apparent "randomness" in feature detection, and (2) feature tracking errors (e.g., feature drift, visibility changes, etc).

As observed, successful matching between tracked features and detected features decreases rapidly. For our datasets, this graph indicates that tracking through more than 4 or 5 images should be avoided.

6.2 Globalization

Our feature globalization algorithm allows us to perform tradeoffs for finding feature correspondences. To vary the amount of globalization, we either change the region size or change the maximum number of iterations of the globalization algorithm. Recall that one iteration consists of a pass of feature propagation and feature relabeling (Figure 3). Using larger regions effectively "pre-globalizes" the features by tracking longer initial sequences. For each region, the features detected at the center image must be tracked outwards through a longer sequence of images before reaching the region border for subsequent globalization. This also reduces the amount of globalization work but depends on the reliability of long tracking sequences. The other option, increasing the maximum number of iterations, places the burden on globalization and not on long tracking sequences. To estimate the current amount of globalization for a set of parameters, we measure the number of globalized features for a set of novel viewpoints throughout the environment.

Figure 10 shows three graphs representing the amount of globalization for several region sizes, maximum iterations, and reductions of the number of reference images using the Library environment. Small regions consist of 5 or fewer images. Medium regions have about 15 images and large regions contain about 45 images. The iteration limit is one of 1, 3, or 5. For each sample point, we calculate the average number of globalized features at 128 viewpoints evenly distributed throughout the environment. The vertical axis represents the average number of globalized features. The horizontal axis indicates the total number of reference images available for reconstruction. Each subset of reference images is evenly distributed throughout the environment.

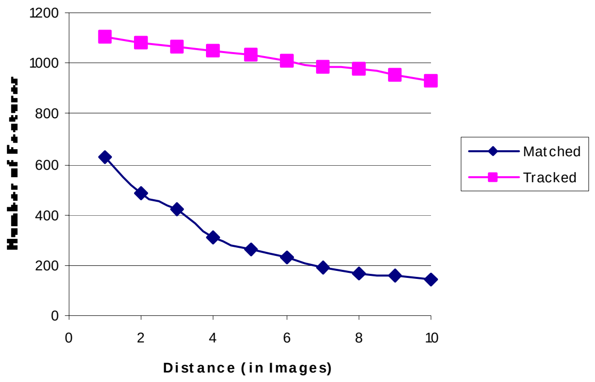
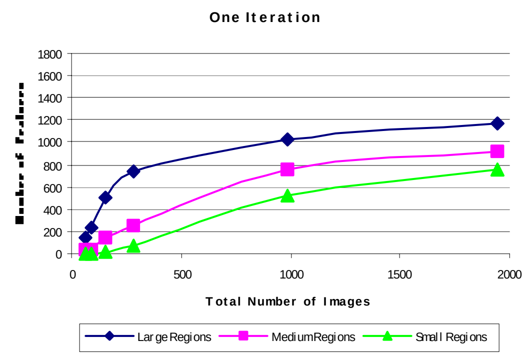
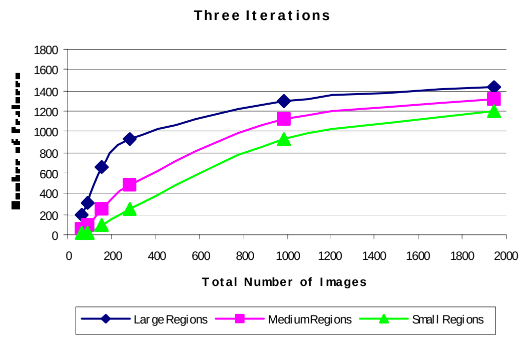


Figure 9. Tracking Performance. *This graphs plots the number of features tracked and matched when traversing outwards from an image. Each data point is the averaged result of a sampling of image paths through all three environments. The "tracked" curve indicates how many features were successfully tracked to the next image. The "match" curve indicates how many of the tracked features match newly detected features for that image. This graph allows us to estimate the number of features tracked and the number of matches.*





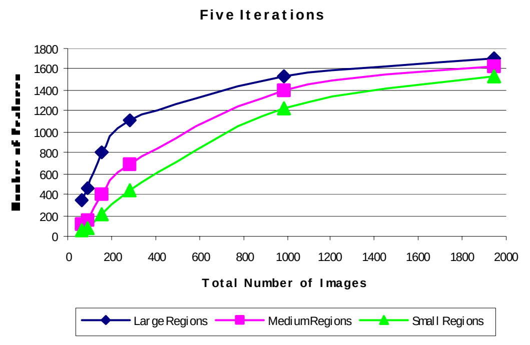


Figure 10. Feature Globalization*. These graphs show how varying the initial region size and varying the maximum number of iterations (and thus globalization) affects the number of features available for warping reference images. Each data point is computed as the average number of features used for reconstructing a novel view at 128 evenly distributed points through the library environment. See text for further interpretations of these graphs.*

These graphs show that feature globalization using more iterations generally outperforms using long tracking sequences in three ways. First, more iterations tends to have a larger benefit than using larger initial region sizes. For instance, using small regions and five iterations yields about twice as many features as using small regions and one iteration. Second, using small regions and more iterations reduces the average tracking sequence length. For large regions and one iteration, features are often tracked through 8 or more images (7 images from the region center to the border plus one image in order to match features with the adjacent regions). For small regions and a maximum of five iterations, features are tracked through only 4.5 images on average (the region typically has a radius of one image and the average iteration count is 3.5 images). As evidenced by Figure 9, tracking shorter sequences yields more matches and thus explains the overall increase in number of features. Finally, using more iterations makes feature globalization less sensitive to the size of the initial regions (compare the right side of both Figure 10a and Figure 10c). This is because as the number of iterations increases, feature matching and labeling approaches a full and complete globalization.

6.3 Performance

Feature globalization is divided into a runtime phase and a preprocessing phase. Our runtime reconstruction operates at about 15 to 20 frames per second on both the PC and the SGI. This includes extracting a common set of features for the current reference images, loading the images to texture memory, creating a mesh, and rendering warped textured meshes.

During preprocessing, we perform three main tasks: feature tracking, which takes up most of the time (68% on average), followed by rotating the images so as to align the images being tracked (22% of the time), and constructing the globalization graph (10% of the time).

For square regions and relatively small values for A, the number of tracking operations is approximately O($N + 4NA/S$), where N is the number of images, A is the maximum number of iterations, and S is length of a side of a region (in images). During initialization, we track N images. Then, each iteration tracks all the images on the perimeter of the regions. The perimeter of a region has $4S$ images and there are $N/(S*S)$ regions. Thus, in a single iteration, we track $4S*N/(S*S) = 4N/S$ images.

For our datasets, we typically track up to 1500 features per image. On the SGI, our method is able to track features from image to image in 3 seconds on average. On the PC platform, we obtain a faster tracking performance of 2 seconds on average.

The globalized feature sets take between 4 to 30 hours of computation, depending on the environment and the globalization parameters. In our current implementation, each feature occupies 24 bytes and the total number of features stored per environment ranges from a few million to approximately 20 million features (Museum environment).

7. CONCLUSIONS AND FUTURE WORK

In this paper, we describe an algorithm for detecting feature correspondences across a wide range of viewpoints, and we have used it to produce high-quality warping of reference images to novel viewpoints in an IBR walkthrough system. From our experience with this system, we conclude that warping with globalized features performs better than pure blending or geometric proxies not only when novel viewpoints are relatively close to scene elements, but also when available reference images are relatively far apart and camera poses of reference images are not perfectly calibrated. Our IBR walkthrough system is able to render images of complex scenes at 15 to 20 frames per second with little or no ghosting or blurring artifacts.

In the future we are considering iteratively adjusting the globalization parameters, including region size and maximum iterations, so as to arrive at a maximum amount and extent of globalized features. Furthermore, we will adjust globalization parameters to different parts of the environment. In addition, we are seeking to reduce feature-tracking time. With real-time tracking, our total pre-computation times would be reduced to a few hours.

The globalized feature set can also be used to compress the image dataset. We are currently investigating into compression methods that, in addition to intra-image redundancy, use feature globalization to extract inter-image redundancy.

Acknowledgments

We are grateful to Sid Ahuja, Multimedia Communications Research VP at Bell Labs, for supporting this research. In addition, we thank Bob Holt and Steve Fortune for their mathematical help.

References

[Aliaga02] Aliaga D., Funkhouser T., Yanovsky D., Carlbom I. "Sea of Images", *IEEE Visualization, October 2002*.

[Buehler01] Buehler C., Boose M., McMillan L., Gortler S., Cohen M., "Unstructured Lumigraph Rendering", *Proc. of ACM SIGGRAPH 2001*, pp. 425-432, 2001.

[Chen93] Chen E., Williams L., "View Interpolation for Image Synthesis", *Proc. of ACM SIGGRAPH 1993*, pp. 279-288, 1993.

[Gortler96] Gortler S., Grzeszczuk R., Szeliski R., and Cohen M., "The Lumigraph", *Computer Graphics (SIGGRAPH 96)*, 43-54, 1996.

[Lee98] Lee S., Wolberg G., Shin S. Y., "Polymorph: Morphing Among Multiple Images", IEEE Computer Graphics and Applications, 18(1):58-71, January/February, 1998.

[Levoy96] Levoy M. and Hanrahan P., "Light Field Rendering", *Computer Graphics (SIGGRAPH 96)*, 31-42, 1996.

[Levoy00] Levoy M. et al, "The Digital Michelangelo Project: 3D Scanning of Large Statues", *Proc. of ACM SIGGRAPH 2000*, pp. 131-144, 2000.

[Kang96] Kang S.B., Szeliski R., "3D Scene Data Recovery using Omnidirectional Multibaseline Stereo", *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 364-370, 1996.

[Max95] Max N. and Ohsaki K., "Rendering Trees from Precomputed Z-Buffer Views", *Rendering Techniques '95*: *Proceedings of the 6th Eurographics Workshop on Rendering*, 45-54, 1995.

[McMillan95] McMillan L. and Bishop G., "Plenoptic Modeling: An Image-Based Rendering System", *Computer Graphics (SIGGRAPH 95)*, 39-46, 1995.

[Morita94] Morita T., Kanade T., "A Sequential Factorization Method for Recovering Shape and Motion from Image Streams", *Proc. ARPA Image Understanding Workshop*, Vol. 2, pp. 1177–1188, 1994.

[Nyland01] Nyland L., Lastra A., McAllister D., Popescu V., McCue C., "Capturing, Processing, and Rendering Real-World Scenes", *Videometrics and Optical Methods for 3D Shape Measurement Techniques, Electronic Imaging Photonics West*, Volume 2309, 2001.

[Pollefeys98] Pollefeys M., Koch R., and van Gool L., "Self-Calibration and Metric Reconstruction in Spite of Varying and Unknown Internal Camera Parameters", *Proceedings Int. Conf. on Computer Vision (ICCV)*, pp. 90-95, 1998.

[Shi94] Shi J., Tomasi C., "Good Features to Track", *IEEE Computer Vision and Pattern Recognition (CVPR 94)*, pp. 593-600, 1994.

[Tomasi91] Tomasi C. and Kanade T., "Detection and Tracking of Point Features**",** Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.