Texturing Long Planar Surfaces with Imprecise Camera Poses for Indoor 3D Modeling

No Author Given

No Institute Given

Abstract. Automated 3D modeling of building interiors is useful in applications such as virtual reality and environment mapping. Texture mapping walls is an important step in visualizing the results of an indoor 3D modeling system. Methods to localize the camera in the 3D scene often are not pixel accurate, meaning that when multiple images are used for texture mapping there are seams and discontinuities between these images. Several approaches to this problem have been proposed but each suffer from a distinct problem of error accumulation for long chains of images, such as those from a long corridor. We propose a new approach to texture mapping planar surfaces that eliminates most discontinuities between images but does not suffer from error accumulation for long chains. We validate this approach using images from several long hallways with data generated by a human operated backpack 3D indoor modeling system.

1 Introduction

Three-dimensional modeling of indoor environments has a variety of applications such as training and simulation for disaster management, virtual heritage conservation, and mapping of hazardous sites. Manual construction of these models can be time consuming, and as such, automated 3D site modeling has garnered much interest in recent years.

An indoor modeling system must first be able to simultaneously estimate the camera's location within an environment and the 3D structure of the environment. This problem is studied by the robotics and computer vision communities as the simultaneous localization and mapping (SLAM) problem. It is usually solved with the aid of laser range scanners, cameras, and inertial measurement units (IMUs) that survey the environment in a vehicle or human-operated backpack [**?**,**?**]. The devices, along with various localization algorithms, can generate a point cloud which is then processed by a surface reconstruction algorithm to infer structure such as walls and ceilings. Finally, the reconstructed surfaces are textured using captured images and the estimated camera position for each photo.

Though the localization errors resulting from these laser based algorithms are quite low even in these complex environments, when the resulting recovered pose is used to texture map camera imagery onto the resulting 3D triangular mesh models, there is significant misalignment between successive images used to texture map neighboring triangles. This implies that the scan matching based localization algorithms are not pixel accurate. The misalignment problem is the focus of this paper.

Previous work on this problem proposed an image based approach in which the pose from laser scan matching based localization is refined using camera imagery [?]. Traditional image stitching approaches using image correspondences have also been tried [?]. However both these approaches suffer from error accumulation when used for long chains, for example 20 or 40 images. These long chains of images are common in indoor modeling when buildings have long corridors. Though image stitching can produce excellent results in some cases, it breaks down when there are very few features in the overlapping portions of the images, such as on empty walls. The probability of such a breakdown increases for longer chains of images, as does the error accumulation.

We separate the texture mapping problem into two independent subproblems: camera localization refinement and image selection. To refine the camera localization we use an image-based approach (SIFT matches [?]) to shift the cameras right, left, up, or down on the plane. The direction and the amount to shift each image is determined globally within each plane using a linear least squares approach. For image selection, we choose a subset of the available images that covers the plane while minimizing the appearence of seams or discontinuities in the final mosaic. We present two approaches to this. The first is a greedy heuristic that covers the plane by iteratively selecting the image that will cover the largest number of empty pixels on the plane. The second approach specifies a cost function, such as the total image energy in the vicinity of the seams, and chooses images that cover the plane while minimizing this cost. The second approach can be done efficiently using dynamic programming when the images overlap in a directed acyclic graph (DAG) structure from left to right, as they do in long hallways using our system.

The approach presented here gracefully handles long chains of images without error accumulation, which we demonstrate on data from our human operated backpack 3D indoor modeling system [?]. This paper is organized as follows. Section 2 outlines the problem of texture mapping for indoor 3D modeling and demonstrate existing approaches. Section 3 describes our approach. Section 5 shows results using data generated from our backpack system and presents conclusions.

2 Texture Mapping Planes

The geometry of the the texture mapping problem for indoor 3D modeling is shown in Figure 1. We are given a set of M images. Each image has a camera matrix P_i for $i = 1..M$, which translates a point in the world coordinate system to a point in image i's coordinates. A camera matrix P_i is composed of the camera's intrinsic parameters, such as focal length and image center, as well as the extrinsic parameters which specify the rotation and translation of the camera center's position with respect to the world coordinates at the time that

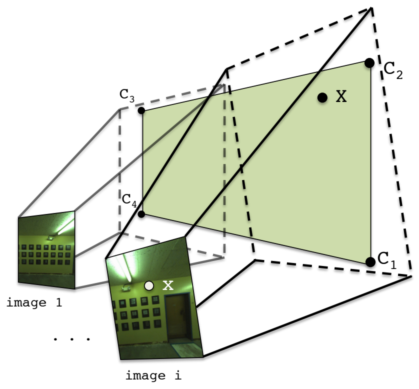

Fig. 1. The plane is specified in 3D space by the four corners C_1 to C_4. Images are related to the plane through the camera matrices $P_{1..M}$.

image i was taken. These extrinsic parameters are determined by the localization hardware and algorithms as part of the indoor modeling system. A point X on the plane can be related to its corresponding pixel x in image i through the following equation:

$$x = project(P_i X)$$

where

$$X = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \text{ and } project(X) = \begin{pmatrix} x/z \\ y/z \end{pmatrix}$$

We are also given a plane generated by the surface reconstruction system, which is to be texture mapped by these images. The plane is defined by four corner points C_1 to C_4 in world coordinates and a normal vector indicating the front facing side of the plane. The challenge is to texture the plane using these images, while eliminating any discontinuities or seams that would suggest that the plane was not composed of a single continuous image.

2.1 Best-Candidate Mapping

Ignoring the fact that the camera matrices $P_{1..M}$ are inaccurate, one can texture map the plane simply by discretizing the plane into tiles or triangles and texturing each tile or triangle seperately. It makes sense to texture each tile or triangle by selecting a best candidate image. This is usually the nearest and/or most direct facing image to the tile or triangle.

As Figure 2 demonstrates, the best candidate mapping leads to significant misalignment between successive tiles. This suggests that while the errors in the localization system are quite low, they are not pixel accurate. For photorealistic texture mapping, either the camera matrices need to be refined such that the

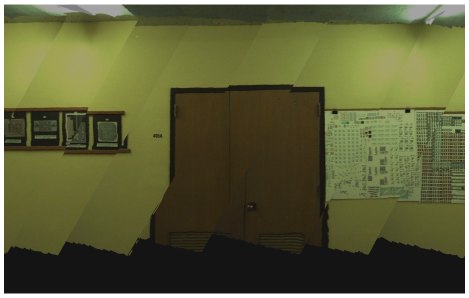

Fig. 2. The result of naive texture mapping based on the imprecise camera matrices estimated by the localization system.

localization is pixel accurate, or image stitching techniques need to be applied to provide this illusion. The following are two existing techniques to solve this problem.

2.2 Image Mosaicing

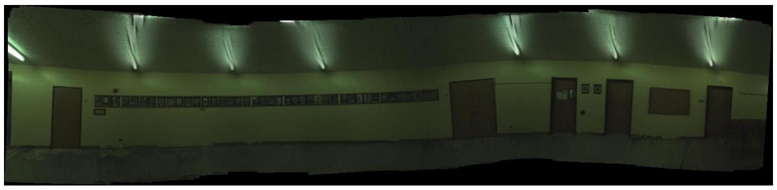

Fig. 3. Image mosaicing.

When images are taken of a plane from arbitrary overlapping positions, they are related by homography [**?**]. Thus, existing homography-based image mosaicing algorithms are applicable [**?**]. However, errors can compound when long chains of images are mosaiced together using these approaches. For example, a pixel in the nth image in the chain must be translated into the first image's coordinates by multiplying by the 3×3 matrix $H_1 H_2 H_3 ... H_n$. Any error in one of these homography matrices is propagated to all further images until the chain is broken. For some chains of images this can happen almost immediately due to erroneous correspondence matches and the resulting image mosaic is grossly misshapen.

Figure 3 shows the output of the AutoStitch software package which does homography-based image mosaicing. This plane is nearly a best-case scenerio with many features spread uniformly across it. Even so, the mosaicing produces errors that causes straight lines to appear as waves on the plane. This image was generated after careful hand tuning. Many planes that had fewer features simply failed. This led us to conclude that image mosaicing alone is not enough to reliably texture map our indoor 3D modeling dataset.

2.3 Image-Based 3D Localization Refinement

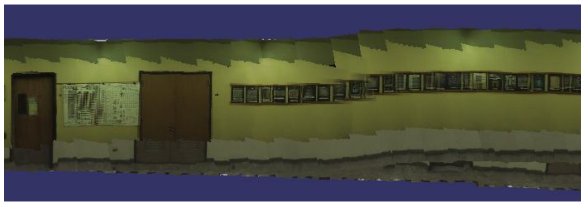

Fig. 4. Using the graph-based localization refinement algorithm from [11] suffers from the problem of compounding errors.

Another approach is to refine the camera matrices using image correspondences to guide the process. Each camera matrix has 6 degrees of freedom. Previous work attempted to refine the camera matrix by solving a non-linear optimization problem [**?**]. This process is carried out at the same time as the laser based backpack localization and is therefore specific to the system in [**?**,**?**]. Unfortunately, this approach suffers from a similar error propagation problem shown in Figure 4. In this work we also refine the placement of images using image correspondences. However, we do so in two dimensions on the plane where as this previous work did so over all 6 degrees of freedom. Refining in two dimensions on the plane is less flexible in that it cannot fix certain projection errors. Emperically we find that it avoids the error propagation problem.

3 Our Approach

Our approach can be summarized as follows. First we place all the images on the plane using the imprecise camera matrices given to us by the localization algorithm. This is done using the naive approach outlined in Section 2.1. This naive placement looks bad because the camera matrices are not pixel accurate. Our first challenge is to improve the placement of these images either by fixing the camera's localization in 3 dimeensions, or through 2 dimensional image processing techniques. We choose to use image processing techniques. SIFT matches are found between overlapping images. The imprecise localization means that matching SIFT features coming from two different images may not map to the same location on the plane. To fix this, we shift the images left, right, up, or down in such a way to maximize the agreement among images for each SIFT match.

The next problem is image selection. The cameras on the backpack system produce far more images than are necessary to completely cover the plane. For example, a long hallway may have several hundereds of images when just 10 or 20 would suffice. We would prefer to use fewer images because this leads to fewer seams between images. However, we'd also like to make sure we are using good images that were taken perpendicular to the plane, for example, and whose seams fall in unimportant areas such as blank walls. In this section we outline two approaches to solving the image selection problem: a greedy heuristic solution that can work on any plane, and a dynamic programming formulation requires images overlap in a DAG structure.

3.1 Preprocessing

As a starting point, we naively texture map the plane using the given imprecise camera matrices for each image in the set. This is done by discretizing the plane into an arbitrary density of pixels and projecting each pixel into the image plane using the estimated camera matrix P_i. We later intend to shift some of these projected images around on the plane and remove some others. Therefore, each image is stored intact in its own data structure so that it can later be merged with the other images on the plane. We use a Hough transform to rotate the projected images such that vertical lines are pointing directly upwards. This can be effective for indoor modeling, since many indoor scenes contain strong vertical lines from features such as doors, wall panels, or rectangular frames.

3.2 Localization Refinement

The camera matrices generated by the localization algorithm are imprecise so the images are misaligned. To fix this, we choose to fix the locations in 2D. First we find corresponding points between all pairs of overlapping images using SIFT matches [**?**]. An illustration of this is given in Figure **??**. The SIFT matches allow us to determine the correct x and y distances between two images on the plane. We already have an estimate of the x and y distances between the two images which was given by the localization algorithms. Soon we will compare the original estimates to the SIFT-based distances and use this to improve the quality of the image locations.

Robust SIFT Distances using RANSAC First we need to ensure that the SIFT-based distances are reliable. Since SIFT matches frequently include outliers, the RANSAC framework [**?**] is used for a robust estimate of the x and y distances between the two images. The RANSAC framework will attempt to build a consensus among the SIFT matches about what the true x and y distance is between the two images while ignoring the influence of outliers that may skew the results. The framework handles the consensus-building machinery, and only requires that two functions be specified: the fitting function and the distance function. These functions are called for random subsets of the SIFT matches until the best set of inliers is found. For this application, the fitting function simply finds the average distance between matches. If the matches are exactly correct and the image is frontal and planar then the distances for various SIFT feature matches should be the same. Our distance function for a pair of points is the difference between those points' SIFT match distance and the average distance computed by the fitting function. We specified a 10 pixel outlier threshold to the framework. This means that a SIFT match is labeled as an outlier if its x and y distance is not within 10 pixels of the average distance computed by the fitting function.

Refining Image Positions using Least Squares There are a total of M^2 possible pairs of images, however we can only measure distances between images that overlap at SIFT feature points. Given these distances and the original image location estimates, we solve a least squares problem ($\min_\beta ||\beta X - y||_2^2$) to estimate the correct location of the images on the plane. The vector β of unknowns are the correct x and y locations of each image on the plane from $1 \ldots M$. Finding the correct x and y locations are independent from one another so we will only consider the x locations:

$$\beta = \big(x_1, x_2, x_3, \cdots x_{M-1}, x_M\big)$$

The matrix X is constructed with one row for each pair of images with measured distances produced by the SIFT matching stage. A row in the matrix has a -1 and 1 in the columns corresponding to the two images in the pair. For example, the matrix below indicates that we generated a SIFT-based distance between images 1 and 2, images 1 and 3, images 2 and 3, etc.

$$X = \begin{pmatrix} -1 & 1 & 0 \cdots & 0 & 0 \\ -1 & 0 & 1 \cdots & 0 & 0 \\ 0 & -1 & 1 \cdots & 0 & 0 \\ \vdots & \vdots & \vdots \ddots & \vdots & \vdots \\ 0 & 0 & 0 \cdots & 1 & 0 \\ 0 & 0 & 0 \cdots & -1 & 1 \\ 1 & 0 & 0 \cdots & 0 & 0 \end{pmatrix}$$

If only relative distances between images are included then there is no way to determine the absolute location of any of the images and the matrix becomes rank deficient. To fix this we choose the first image to serve as the anchor for the rest, meaning all the absolute distances are based on its original location. This is done by adding a row with a 1 in the first column and the rest zeros.

Finally, the observation vector y is constructed using the SIFT-based distances generated earlier in the matching stage. The distances are denoted as $d_1 \ldots d_N$ for N SIFT-based distances. The last element in the observation vector is the original location of the first image determined by the localization algorithm.

$$y^T = \big(d_{1,2}, d_{1,3}, d_{2,3}, \ldots d_{N-2,N-1}, d_{N-1,N}, x_1\big)$$

The β that minimizes $||\beta X - y||_2^2$ tells us a set of image locations on the plane that best honors all the SIFT-based distance measurements between images. A similar problem is solved for the y dimension. In practice there are often cases where there is a break in the chain of images, meaning that no SIFT matches were found between one segment of the plane and another. In this case we add rows to the X matrix and observations to the y vector that contain the original x and y distance estimates generated by the localization algorithm. Another way to do this is to add rows for all neighboring pairs of images and solve a weighted least squares problem where the SIFT distances are given a higher weight i.e. 1, and the distances generated by the localization algorithm are given a smaller weight i.e. 0.01.

3.3 Image Selection

The backpack system produces images at a much faster rate than is necessary to fully texture map all surfaces. Many of these images can be discarded because the range they cover is also covered by other images in the set. In fact, our system discards images as a preprocessing step that are either not directly facing the plane or are beyond a certain threshold distance from each plane. Beyond this, it can be advantageous to throw away redundant images at this stage because fewer images generate fewer seams and therefore fewer potential visual artifacts from misalignment. We describe two methods for image selection. The first is a greedy heuristic that makes no assumption about the plane or the images. The second method assumes that the images overlap in a DAG structure and employs dynamic programming to minimize a specific cost associated with a selection of images.

Greedy Heuristic We begin with an empty plane and a set of M images. The image that covers the largest number of pixels on the plane is chosen and copied to the plane. The pixels covered by that image are then considered to be covered. Next, the image that covers the largest number of uncovered pixels is chosen. It is copied to the plane as well. If it overlaps with any covered pixels then the image is blended according to the process described in Section 4.1.2. This process continues until the plane is completely covered.

Cost Functions The greedy algorithm will cover the plane, if possible, and will tend toward using fewer images. However, the solution is not optimal with respect to any image-based cost. We may be interested in covering the plane while minimizing the sum of squared pixel differences of the blended seam regions. This would encourage seams to be placed either in featureless areas, such as blank walls, or in areas in which the images are very well aligned. Another possible cost function is edge energy, i.e. the sum of the smoothed gradient of the blended seam regions. This would encourage seams to be placed in featureless areas even more than the sum of squared pixel difference cost function.

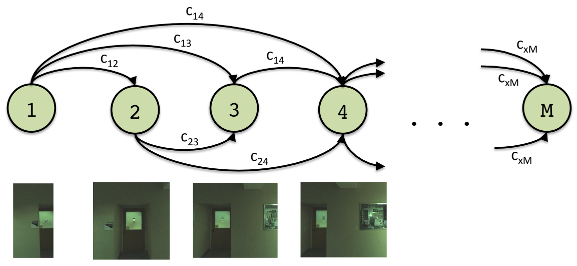

Fig. 5. Image Selection is done by constructing a graph of sorted images. Then we solve a shortest path problem where the edge weights represent the cost of a seam between two overlapping images.

Dynamic Programming In the general case, finding the optimal subset of images with regard to the cost functions appears to have exponential complexity in the number of images. For this reason, we did not attempt this approach to image selection. However, if the images overlap only in one direction, as they do in long hallways, we can construct a DAG from the overlapping images and solve a simple shortest path problem to find an optimal subset of images with regard to the cost functions.

Figure 5 demonstrates the construction of a DAG from overlapping images of a long hallway. Images are sorted by location (e.g. left to right) and become nodes in a graph. Edges are placed in the graph between images that overlap. The weights of these edges are determined by a cost function, such as image energy, or the sum of squared pixel differences between the two images in the seam region. Next, we solve a shortest path problem from the first node in the graph to the last node. This provides a set of images that completely covers the plane, but minimizes the cost of the seams produced. Note that this only works when all the images completely cover one dimension. In this case the vertical dimension of the plane is covered by all images.

3.4 Blending

The steps above eliminate visual artifacts from misalignment, but there still are obvious discontinuities in brightness between images resulting from differences in lighting. This problem has been successfully addressed in previous work using an alpha blending technique in which the pixel intensities in overlapping images are blended proportional to the pixel's location in the image [**?**]. We use a similar technique to improve the visual quality of our results. In the small area where images overlap, there is a linear weighting function in the horizontal direction that interpolates between the left image and the right, providing a gradual transition between images. The final location corrected, subsampled, and blended image is shown in Figure **??**.

4 Results

5 Conclusions and Future Work