Texture mapping 3D planar models of indoor environments with noisy camera poses

Peter Cheng, Michael Anderson, Stewart He, and Avideh Zakhor

University of California, Berkeley
Berkeley CA 94720, USA

Abstract. Automated 3D modeling of building interiors is used in applications such as virtual reality and environment mapping. Texturing these models allows for photo-realistic visualizations of the data collected by such modeling systems. While data acquisition times for mobile mapping systems are considerably shorter than for static ones, their recovered camera poses often suffer from inaccuracies, resulting in visible discontinuities when successive images are projected onto a surface for texturing. We present a method for texture mapping that starts by selecting images whose camera poses are well-aligned in two dimensions. We then align images to geometry as well as to each other, producing visually consistent textures even in the presence of inaccurate surface geometry and noisy camera poses. Images are then composited into a final texture mosaic. The effectiveness of the proposed method is demonstrated on a number of different indoor environments.

Keywords: Texture Mapping, Reconstruction, Image Stitching, Mosaicing

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 digital models can be time consuming, and as such, automated 3D site modeling has garnered much interest in recent years.

The first step in automated 3D modeling is the physical scanning of the environment's geometry. An indoor modeling system must be able to recover its pose within an environment while simultaneously reconstructing the 3D structure of the environment itself [1–4]. This is known as the simultaneous localization and mapping (SLAM) problem, and is generally solved by taking readings from laser range scanners, cameras, and inertial measurement units (IMUs) at multiple locations within the environment. Mounting such devices on a platform carried by an ambulatory human provides unique advantages over vehicular-based systems on wheels in terms of agility and portability, but can also result in larger localization error [4]. As a result, common methods for texture mapping generally produce poor results.

In this paper, we present an approach to texture mapping 3D models of indoor environments made of planar surfaces in the presence of uncertainty and noise in camera poses. In particular, we consider data obtained from a human-operated backpack system with a number of laser range scanners as well as 2 cameras facing left and right, each equipped with fisheye lenses reaching an approximately 180° field of view and taking photos at a rate of 5 Hz [4]. Applying multiple localization and loop-closure algorithms on the raw data collected by the onboard sensors [1, 3, 4], the backpack is localized[1] over its data collection period. This involves recovering the 6 degrees of freedom for the backpack itself as well as the cameras rigidly mounted on it. Once this is complete, the data from the laser range scanners is used to generate a 3D point cloud of the surrounding environment, from which a 3D planar model is created [5]. This model, consisting of 2D polygonal planes in 3D space, along with the set of images captured by the backpack's cameras and their noisy 3D poses, can be considered the input to our texture mapping problem.

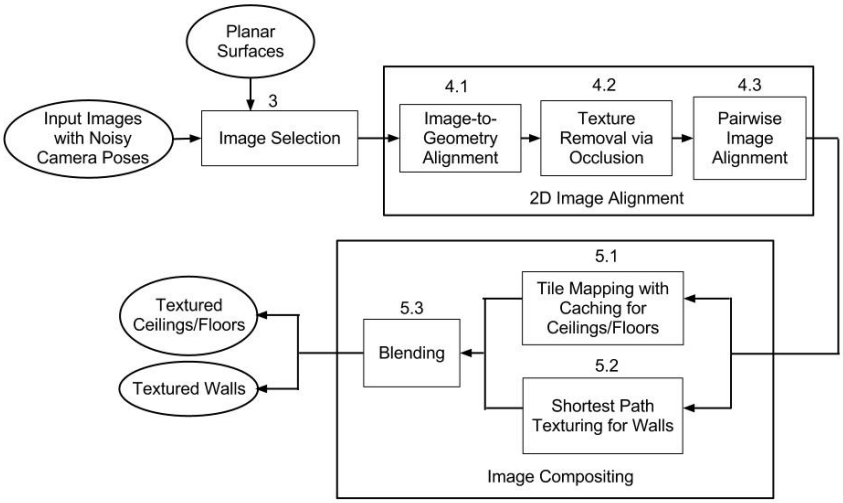


Fig. 1: The proposed texture mapping procedure

The overall block diagram for the proposed texture mapping procedure is shown in Figure 1, where the number attached attached to each box indicates the section in which the concept in the box is explained in this paper. We texture map each planar surface independently and in parallel. For each surface, we begin by selecting a set of images that spans the entire surface with high resolution

[1] In this paper, we use the terms localization and pose recovery interchangeably, in that they both refer to recovering position and orientation.

imagery. We then use our noisy camera poses to project these selected images onto the surface. These projections are then refined in 2D, in order to maximally align them with the surface's geometry, as well as to each other, allowing us to handle both errors in geometry as well as camera poses. For surfaces and camera poses at arbitrary locations or orientations, generally ceilings and floors, we propose a tile-based approach for sampling high-resolution portions of images and compositing them into a texture. In cases where cameras have consistently perpendicular viewing angles to the surfaces under consideration, generally walls, we demonstrate a superior approach that leads to more seamless textures.

The remainder of the paper is organized as follows. Section **??** covers existing approaches to image stitching, and their performance on our datasets. Section 2 explains how to downsample the set of available images by selecting those with the best orientation and distance from surfaces under consideration. Section 3 contains our approach towards 2D image alignment, followed by Section 4, which describes two methods of selecting and compositing images to create the final texture. Sections 5 and 6 contain results and conclusions.

2 Image Selection

Since the backpack system takes pictures at a rate of 5 Hz, hundreds of images are available for texturing each surface in the 3D model. To obtain a selection of images to work with, we discretize each surface into small square tiles, and for each tile, gather the images with a clear line of sight to it. We then find the image that maximizes the scoring function $\frac{1}{d}(-1 \cdot \boldsymbol{c}) \cdot \boldsymbol{n}$ for each tile, as shown in Figure 3, where d is the distance between the centers of a camera and a tile, and $\boldsymbol{n}$ and $\boldsymbol{c}$ are the directions of the plane's normal and the camera axis respectively. In Figure 4(a), an image has been textured using the images selected for each tile. While images with good viewing angles and high resolution have been used, there are a significant amount of discontinuities, though many appear reconcilable with 2D transformations.

By aggregating the images selected for tiles on each surface, we obtain a collection of images that can be used as a starting point for the proposed texturing procedure, as explained in Section 3. For our datasets, this process generally selects around 10% of all the images that could be used for texturing the surface. This roughly 10:1 subsampling removes images with poor viewing angles, ensuring that the most promising images are considered in the following steps.

3 2D Image Alignment

Our proposed 2D alignment procedure consists of three parts, as shown in the diagram in Figure 1. First, images are projected onto the surface and lines within these projected images are detected. Images are then transformed such that these lines match geometric lines composing the boundaries of the surface being textured. Second, occlusion checks are performed to remove invalid parts of each image for the target surface. Third, SIFT feature matches are detected between

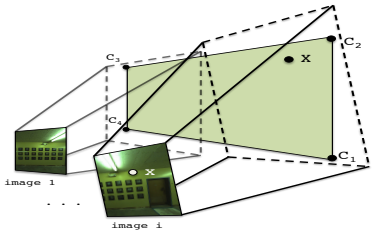


Fig. 2: Surfaces to be textured are specified in 3D space by corners C_i. Images are related to each surface through the camera matrices $P_{1..m}$.

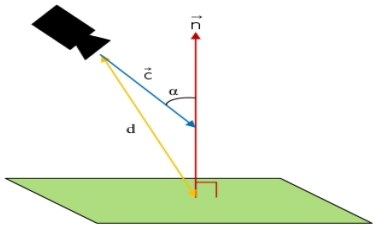


Fig. 3: We minimize camera angle α and distance d by maximizing the scoring function $\frac{1}{d}(-1 \cdot \boldsymbol{c}) \cdot \boldsymbol{n}$

pairs of images, and a weighted linear least squares problem is solved in order to maximize all image and geometry-based alignments. Each step will now be explained in detail.

3.1 Geometry-based Alignment

To align images to geometry, line segments are detected in images via Hough transforms. We also gather a set of geometry-based lines, which correspond to lines comprising the target surface's border, as well as lines formed where other surfaces intersect the target surface. An example of these lines is shown in red for a ceiling surface in Figure 5(a). Ideally, for perfect camera poses and surface geometry, the lines in images corresponding to corners between surfaces should match up exactly with corners in the 3D model. By inducing such lines to match, we fit camera poses more accurately to the surface, and therefore to each other as well.

First, pairs of image-based and geometry-based line segments within a distance and angular difference threshold of each other are collected. We have found a distance threshold of 250 mm and an orientation difference threshold of 10° to work well for our datasets. Each such pair signifies a rotation and translation with one degree of freedom that matches an image line to a geometry line. Pairs are then binned by rotation in 1° increments, which reveals the rotation that maximizes line matches. From the pairs with that corresponding rotation, the two longest noncollinear image-based lines are used to compute a fixed translation. An example of this is case (1) of Figure 5(c). If a rotation is calculated, but a fixed translation cannot be obtained, the pair with the longest image-based line is selected, and the rotation and the minimal translation to match its lines are applied. This corresponds to case (2) of Figure 5(c). Finally, if no line pairs were detected, images are rotated using a RANSAC [6] approach to maximize parallel image and geometry lines, without regard for translational distance. This corresponds to case (3) of Figure **??**(c). The results of this procedure are recorded, as the translational degrees of freedom for each image are used in Section 3.3.

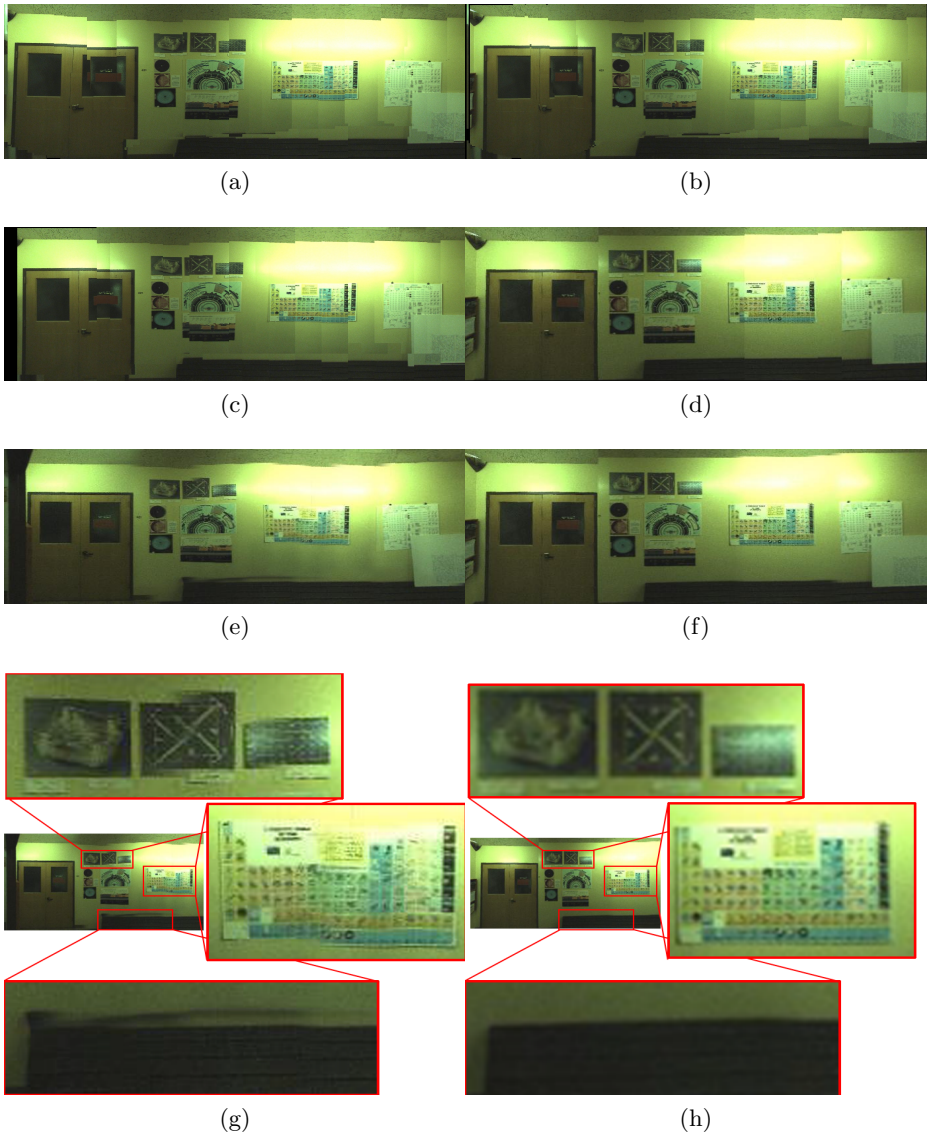


Fig. 4: (a) Tile-based texturing; (b) Tile-based texturing after image alignment; (c) Tile-based texturing after image alignment with caching; (d) Shortest path texturing after image alignment; (e,f) Blending applied to (c) and (d); (g,h) Zoomed in views of discontinuities in (e) vs. in (f).

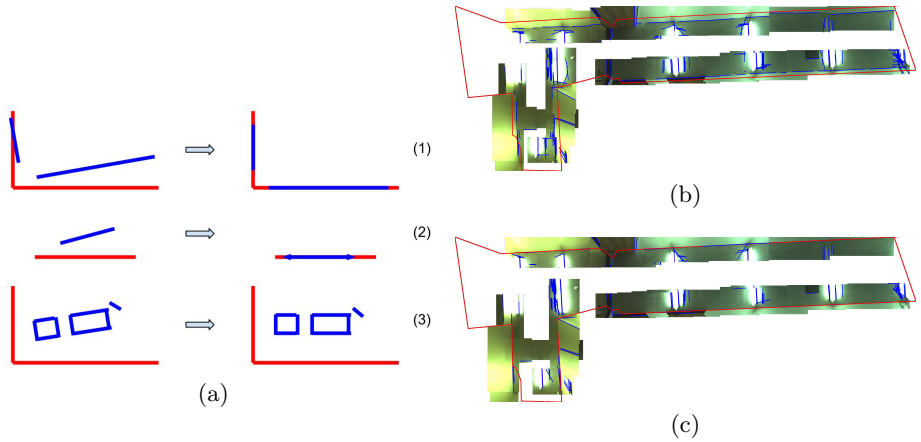
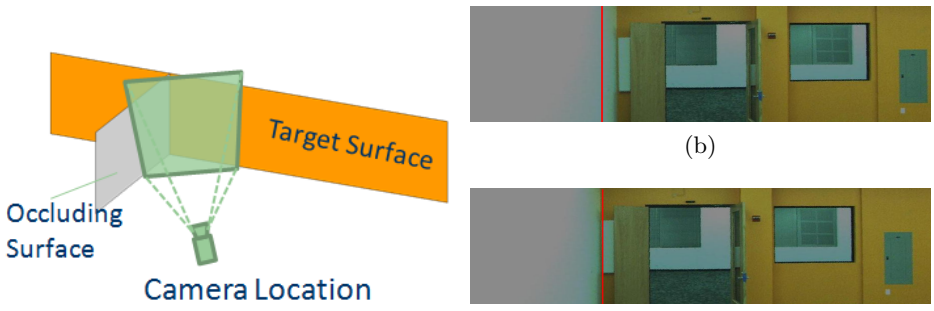


Fig. 5: Images projected onto a ceiling surface, where geometry-based lines corresponding to the ceiling's boundary are shown in red. Image-based lines detected by Hough transform in the image projections are shown in blue; (a) examples of matching lines in cases with ≥ 2 line pairs, 1 line pair, and zero line pairs, from top to bottom; (b) images projected with their original noisy camera poses; (c) after image alignment to maximize line matches between images and geometry;

After this step, image projections line up well with target surfaces, as shown in Figure 5(b), which is considerably more aligned than Figure 5(a). This procedure reconciles both errors in camera poses as well as in geometry, and results in sharp, continuous borders across images, which is crucial when checking for occlusion.

3.2 Image Occlusion

In order to correctly texture surfaces, it is important to detect and remove parts of image projections containing texture for occluding surfaces. For instance, in Figure 6(a), an image used to texture the orange target surface also contains part of a gray occluding surface. We remove this incorrect texture by recursively performing ray-polygon intersection tests between the ray from the camera location to locations on the target surface [7]. These intersection tests are performed at the corners of a grid overlaid upon the target surface. Where all four corners of a grid section are occluded, texture is removed. Where one or more corners are occluded, the grid is subdivided into four, and the process repeats. Occlusion checking works entirely with geometry, so by ensuring that images match geometry using 3.1's alignment procedure, texture belonging to other surfaces is accurately removed, as seen in Figure 6(b).



(a) (b) (c)

Fig. 6: (a) The image from the camera in this diagram contains texture that belongs to the gray occluding surface, which should not be projected onto the orange target surface; (b) without geometry alignment, texture to the left of the red line would be removed, which would leave some erroneous texture projected onto our target surface; (c) after geometry alignment, the image is shifted, resulting in the correct amount of texture being removed.

3.3 2D Feature Alignment

The next step is to align images to each other by detecting feature point matches between all pairs of overlapping images. We use feature alignment rather than pixel or intensity-based alignment due to the differences in lighting as well as possible occlusion among images, both of which feature alignment is less sensitive to [8–10]. We use SiftGPU [11] for its high performance on both feature detection as well as pairwise matching. Detected matches determine dx and dy distances between each pair of features for two image projections, though distances often are not the same for different features. Since indoor environments often contain repetitive features such as floor tiles or doors, we need to ensure that SIFT-based distances are reliable. First, we only detect feature matches in the parts of images that overlap given the original noisy poses. Second, feature matches that correspond to an image distance greater than 200 mm from what the noisy poses estimate are discarded. In order to utilize the remaining feature matches robustly, RANSAC [6] is used to estimate the optimal $dx_{i,j}$ and $dy_{i,j}$ distances between two images i and j.

We now use the feature-based distances between pairs of images as well as geometry alignment results from Section 3.1 to refine all image positions using a weighted linear least squares approach. An example setup for a weighted linear least squares problem $\min_{\boldsymbol{\beta}} ||W^{\frac{1}{2}}(A\boldsymbol{\beta} - \boldsymbol{\gamma})||_2^2$ with 3 images is as follows.

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & -m_2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{\beta} = \begin{pmatrix} x_1, \\ x_2, \\ x_3, \\ y_1, \\ y_2, \\ y_3 \end{pmatrix} \boldsymbol{\gamma} = \begin{pmatrix} dx_{1,2}, \\ dy_{1,2}, \\ dx_{2,3}, \\ dy_{2,3}, \\ -m_2 gx_2 + gy_2, \\ gx_1, \\ gy_1, \\ tx_1, \\ ty_1, \\ tx_2, \\ ty_2, \\ tx_3, \\ ty_3 \end{pmatrix} \boldsymbol{W} = \begin{pmatrix} 1, \\ 1, \\ 1, \\ 1, \\ 1, \\ 1, \\ 1, \\ 0.01, \\ 0.01, \\ 0.01, \\ 0.01, \\ 0.01, \\ 0.01 \end{pmatrix}$$

The variables to solve for are the x_i and y_i positions of images, while equations are the feature-based distances between pairs of images, images fixed to geometry with 0 or 1 degrees of freedom, and the original noisy camera poses. In this scenario, a feature-based distance of $dx_{1,2}$, $dy_{1,2}$ was calculated between images 1 and 2. This corresponds to the first and second row of A, while the third and fourth row of A represent the same for images 2 and 3. Rows 5 through 7 correspond to results of the geometry alignment procedure in Section 3.1. Specifically, row 5 corresponds to a geometry-based constraint of image 2's location to a line of slope m_2, passing through point gx_2, gy_2, while rows 6 and 7 correspond to a fixed location for image 1 without any degrees of freedom. Rows 8 through 13 correspond to the original camera locations for each image (tx_i, ty_i).

The original camera poses are needed due to lack of feature matches in all images, or lack of enough geometry alignment results to generate a single solution. Since it is desirable to minimally use the original noisy poses, we assign to them a weighting factor of 0.01, while all other equations are weighted at 1.

This problem is linear, and it is solved quickly. After applying the resulting shifts, images overlap and match each other with far greater accuracy. Using the simple tile texturing scheme from Section 2 on these adjusted images results in Figure 4(b), which has far fewer discontinuities than in 4(a), though some discontinuities remain visible.

4 Image Compositing

In Section 2, a simple tile-based image selection and texturing method was described. After performing image alignment, the image in Figure 4(b) was obtained. It is clear that discontinuities in the image occur at tile boundaries, and Section 4.1 adds a caching mechanism to reduce such boundaries. On the other hand, for special cases where images have consistently perpendicular viewing angles, Section 4.2 develops another image compositing method that tends to produce cleaner textures. Both methods are followed by a blending step, as shown in Figure 1.

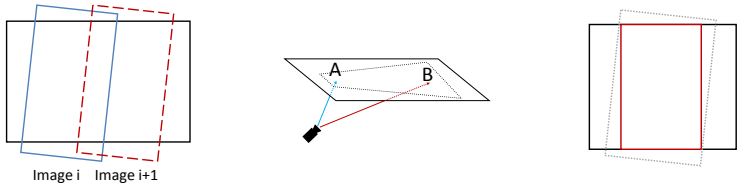
4.1 Tile-Mapping with Caching

To reduce discontinuities between tiles, this method reuses images for adjacent tiles where possible, and otherwise selects images with similar poses. This is accomplished via the use of a spatiotemporal cache for selected images.

The best image for a tile t is selected by searching through two subsets of images for a viable candidate, before searching through the entire set of valid images obtained in Section 2. The first subset of images is those selected by adjacent tiles that have already been textured, and that also contain texture for t. Of these images, one is selected according to the scoring function in Figure 3. Before reusing this image, we check the criteria $\alpha < 45°$, in order to ensure a high resolution projection, with α as the camera angle as shown in Figure 3.

If no satisfactory image is found in the first subset, a second subset of images is used, consisting of those taken near the ones in the first subset, both spatially and temporally. These images have quite similar localization and projection, and are likely to match well. If no viable image is found according to the $\alpha < 45°$ criteria, we use the image selected from Section 2.

The result of applying this caching approach to the images for the surface in Figure 4(a) is shown in Figure 4(c), where seams are considerably reduced as compared to Figure 4(b). However, some discontinuities are still present, as visible in the posters on the wall with breaks in their borders.



(a) (b) (c)

Fig. 7: (a) Images for vertical planes are tilted, but their camera axes are more or less normal to their respective planes. (b) Camera axes for ceiling images are at large angles with respect to plane normals. (c) Wall images are cropped to be rectangular.

4.2 Shortest Path Texturing

As mentioned earlier, our data comes from a mobile backpack system carried by am ambulatory human operator, typically bent forwards at 15 to 20 degrees with respect to the vertical direction. As a result, cameras facing sideways are head on with respect to vertical walls, as shown in Figure 7(a), while cameras oriented towards the top or bottom of the backpack are at an angle with respect to horizontal floors and ceilings, as shown in Figure 7(b). These oblique camera

angles for horizontal surfaces translate into textures that span large areas once projected, as shown in Figure 7(b). Using the tile-based texture mapping criteria from Figure 3, such projections have highly varying scores depending on the location of a tile on the plane. Thus, the tiling approach in Section 4.1 is an appropriate choice for texturing floors and ceilings, as it uses the parts of image projections that maximize resolution for their respective plane locations, e.g. areas near point A and not near point B, in Figure 7(b).

For vertical walls however, most images are taken from close distances and head-on angles, resulting in high resolution fronto-parallel projections. As a result, for each tile on a wall plane, the scoring function of Figure 3 is relatively flat with respect to candidate images, as they are all more or less head on. Since the scoring function is less discriminative for walls, it is conceivable to devise a different texturing strategy to directly minimize visible seams when texturing them. This is done by choosing the smallest possible subset of images from the set selected in Section 2 and aligned in Section 3 such that it (a) covers the entire plane and (b) minimizes the visibility of borders between the images. A straightforward cost function that accomplishes the latter is the sum of squared differences (SSD) of pixels in overlapping regions between all pairs of images. Minimizing this cost function encourages image boundaries to occur either in featureless areas, such as bare walls, or in areas where images match extremely well.

In its most general form, our problem can be defined as minimally covering a polygon i.e. the planar surface, using other polygons of arbitrary geometry i.e. image projections, with the added constraint of minimizing the cost function between chosen images. Given that wall-texture candidate images are taken from more or less head-on angles, and knowing that generally minor rotations are made in Section 3, we can crop our image projections to be rectangular with minimal texture loss, as shown in Figure 7(c). Furthermore, because the fisheye camera lenses have full floor-to-ceiling coverage of nearly all walls, and the backpack operator nearly always only moves horizontally, we only need to ensure lateral coverage of our wall planes. We can thus construct a Directed Acyclic Graph (DAG) from the images, with edge costs defined by the SSD cost function, and solve a simple shortest path problem to find an optimal subset of images with regard to the SSD cost function [12].

Figure 8 demonstrates the construction of a DAG from overlapping images of a hallway wall. Images are nodes in a graph, and directed edges are placed in the graph from left to right between overlapping images. The weights of these edges are determined by the SSD cost function. Next, two artificial nodes representing the ends of the plane are added, with equal-cost C_0 edges to the images crossing them. We now solve the shortest path problem from one end to the other. This results in a set of images completely covering the plane horizontally, while minimizing the cost of seams between images.

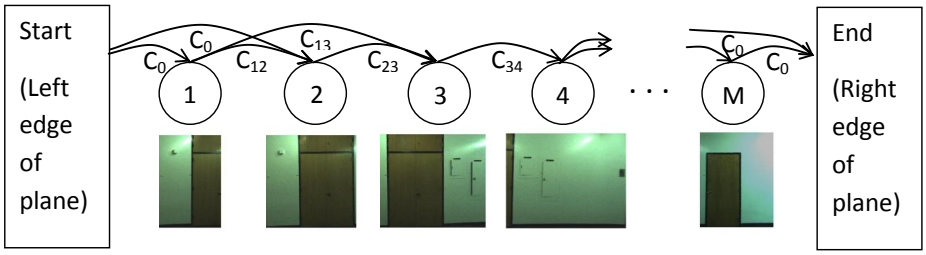


Fig. 8: DAG construction for the image selection process.

As seen in Figure 4(d), this shortest path method has fewer visible discontinuities than Figure 4(c) corresponding to the tile caching approach[2]. This is especially evident when comparing the posters in the images. This shortest path approach directly reduces the cost of each image boundary, while the tile caching method uses a scoring function that only approximates this effect. Furthermore, this approach guarantees the best selection of images to minimize seams, while the sequential tile caching method may select images early on that turn out to be poor choices once subsequent tiles have been processed.

When texturing an entire 3D planar model, we apply the shortest path method on walls, due to its superior results when provided with head-on images with lateral coverage. Floors and ceilings however, given their many images taken at oblique angles, are textured using the tile caching method of Section 4.1.

4.3 Blending

We now apply a blending procedure to the texturing methods in Sections 4.1 and 4.2. Although the image alignment steps and image selection in both methods attempt to minimize all mismatches between images, there are occasional unavoidable discontinuities in the final texture due to different lighting conditions or inaccuracies in model geometry. These can however be treated and smoothed over by blending over image seams. Whether the units to be blended are rectangularly-cropped images or rectangular tiles, we can apply the same blending procedure, as long as there is a guaranteed overlap between images.

For the tile caching method of Section 4.1, overlap can be ensured by texturing a larger tile than needed for display. For example, for a rendered tile $l_1 \times l_1$, we can associate it with a texture $(l_1 + l_2) \times (l_1 + l_2)$ in size. For the shortest path method, overlap between images is inherently ensured. To enforce consistent blending however, a minimum required overlap between images of 200 mm for the graph is added.

[2] In Figure 4(d), we arbitrarily chose one image for texturing where images overlap, as blending will be discussed in section 4.3.

After linear alpha blending across overlapping regions, the texture mapping process is complete. Figures 4(e) and 4(f) show the blended versions of Figures 4(c) and 4(d) respectively. The remaining images in Figure 4 highlight differences between the two methods, showing that Figure 4(f) has the cleanest texture and contains the best visual quality among the textures in Figure 4.

5 Results

Examples of ceilings and floors textured with the tile caching approach, and walls textured with the shortest path approach, are displayed in Figure 9. High resolution colored texture comparisons, as well as video and interactive walkthroughs of full models are available at [3].

As mentioned earlier, our approach is quite efficient. The top wall in Figure 9(a) was generated with 7543×776 pixels, and spans a 40-meter long wall. Given 41000 input images of the entire dataset, a 2.8GHz dual-core consumer-grade laptop takes approximately a minute to choose 36 candidate images, followed by under a minute to perform both image alignment and the shortest path texturing method, though over 75% of that time is spent calculating SIFT matches within the SiftGPU framework, which could feasibly be split into a separate preprocessing step. While not real-time, the process is capable of generating fast updates after changes in various parameters or modifications to input data, and if integrated directly into a 3D modeling system, could provide live visualization as data is collected. Our full models consist of an input model file, textures, and a mapping of image points to 3D model vertices. The models shown in Figure 9 are roughly 20 MB in size, and are visualized using the OpenSceneGraph toolkit [13], which allows for export to many common model formats, as well as efficient visualization, even in web browsers or mobile devices.

6 Conclusion

In this paper, we have developed an approach to texture mapping models with noisy camera localization data. We are able to refine image locations based on geometry references and feature matching, and robustly handle outliers. Using the tile-based mapping approach, we can texture both simple rectangular walls as well as complex floor and ceiling geometry. We also implemented a shortest path texturing method that produces seamless textures on planes where multiple head-on images are available. Both of these approaches are highly modular, and easily tunable for similar systems across multiple environments.

Our method is likely to fail in scenarios where 3D error is large. A logical progression of our approach to resolve camera error in 3D is to perform matching between image lines and geometry in 3D, which can be done reasonably efficiently [14, 15]. Using linear features in addition to SIFT features is also likely to result in improved matches, as indoor scenes often have long, unbroken lines

[3] `http://www-video.eecs.berkeley.edu/research/indoor/`

spanning multiple images [16]. Finally, the blending procedure is quite basic, and applying more sophisticated methods of blending as well as normalization would benefit the final visual quality, and more robustly handle motion-based or parallax errors.

References

1. Chen, G., Kua, J., Shum, S., Naikal, N., Carlberg, M., and Zakhor, A., "Indoor localization algorithms for a human-operated backpack system," in [*Int. Symp. on 3D Data, Processing, Visualization and Transmission (3DPVT)*], (2010).
2. Hartley, R. and Zisserman, A., [*Multiple view geometry in computer vision*], vol. 2, Cambridge Univ Press (2000).
3. Kua, J., Corso, N., and Zakhor, A., "Automatic loop closure detection using multiple cameras for 3d indoor localization," in [*IS&T/SPIE Electronic Imaging*], (2012).
4. Liu, T., Carlberg, M., Chen, G., Chen, J., Kua, J., and Zakhor, A., "Indoor localization and visualization using a human-operated backpack system," in [*Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*], 1–10, IEEE (2010).
5. Sanchez, V. and Zakhor, A., "Planar 3d modeling of building interiors from point cloud data," in [*Internation Conference on Image Processing*], (2012).
6. Fischler, M. and Bolles, R., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24**(6), 381–395 (1981).
7. Glassner, A. S., [*An introduction to ray tracing*], Academic Press (1989).
8. Lowe, D., "Object recognition from local scale-invariant features," in [*Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*], **2**, 1150–1157, Ieee (1999).
9. Mikolajczyk, K. and Schmid, C., "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **27**(10), 1615–1630 (2005).
10. Szeliski, R., "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision* **2**(1), 1–104 (2006).
11. Wu, C., "Siftgpu." `http://www.cs.unc.edu/~ccwu/siftgpu/`.
12. Dijkstra, E., "A note on two problems in connexion graphs," *Numerische Mathematik* **1**, 269–271 (1959).
13. "Openscenegraph." `http://www.openscenegraph.org/projects/osg`.
14. Elqursh, A. and Elgammal, A., "Line-based relative pose estimation," in [*Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*], 3049 –3056 (june 2011).
15. Koeck, J. and Zhang, W., "Extraction, matching and pose recovery based on dominant rectangular structures," (2005).
16. Ansar, A. and Daniilidis, K., "Linear pose estimation from points or lines," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **25**, 578 – 589 (may 2003).

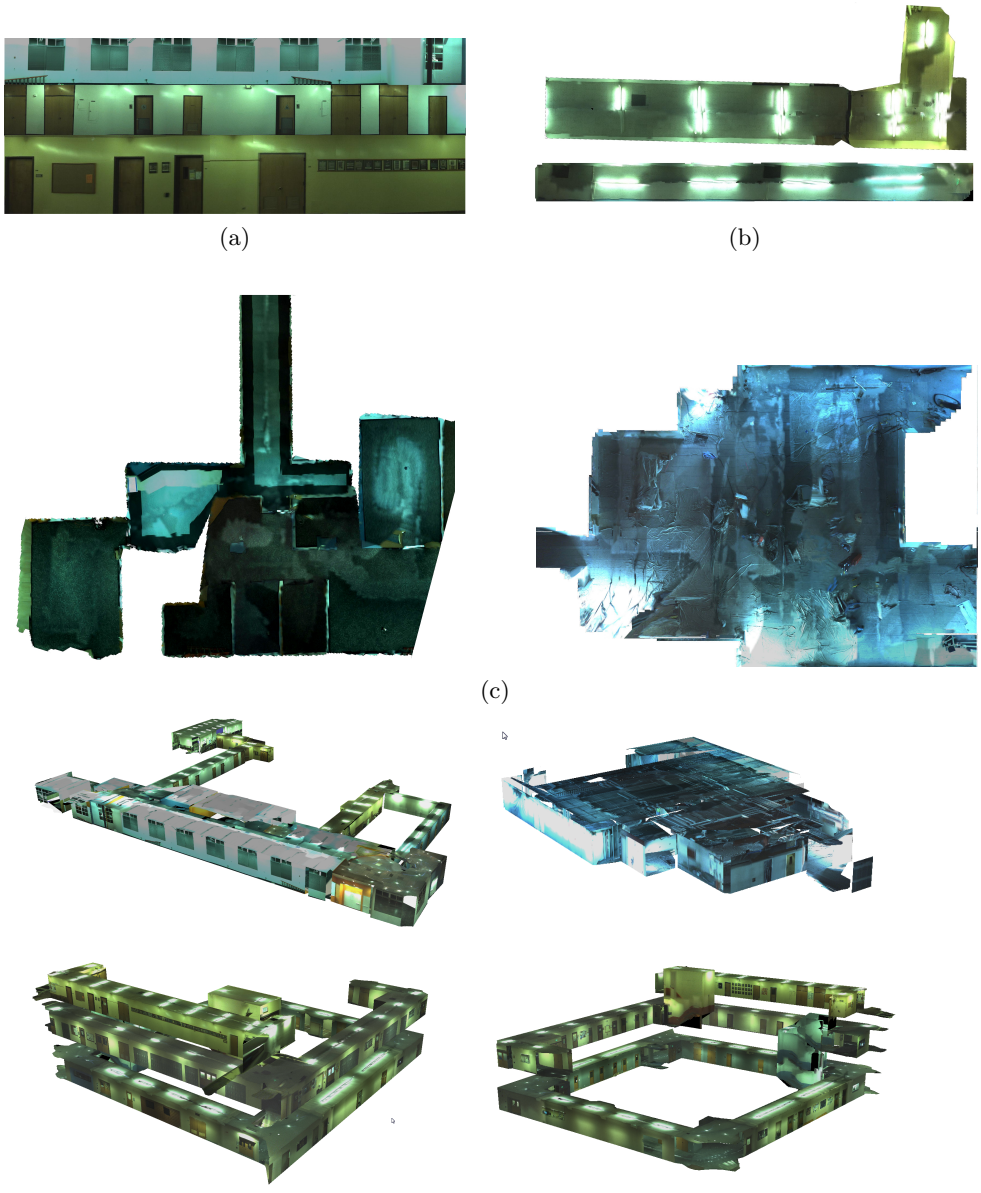


Fig. 9: Examples of our final texture mapping output for (a) walls, (b) ceilings, (c) floors, (d) full models.