MULTI-MODALITY FUSION: REGISTERING PHOTOGRAPHS, VIDEOS, AND LIDAR RANGE SCANS

A Dissertation presented to
the Faculty of the Graduate School
at the University of Missouri

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy of Computer Science

by
Brittany Morago
Dr. Ye Duan, Dissertation Supervisor

MAY 2016

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

MULTI-MODALITY FUSION:
REGISTERING PHOTOGRAPHS, VIDEOS, AND LIDAR RANGE SCANS

presented by Brittany Morago,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Ye Duan

Dr. Chi-Ren Shyu

Dr. Prasad Calyam

Dr. James Keller

ACKNOWLEDGMENTS

This work is supported in part by the NSF Graduate Research Fellowship award #0943941, the US Department of Education award #P200A100053, the Robert E. Waterson Fellowship, the NSF CC-NIE award #1245795, and the NSF CMMI award #1039433. We thank the researchers who provided the public datasets in [1, 2, 3, 4, 5, 6, 7] and code [8, 7] that we used for testing and evaluating the work in this thesis as well as the engineers and researchers who designed and implemented the OpenCV [9] and VLFeat [10] libraries and Jonathan Shewchuk who designed "Triangle", all of which were incorporated into this work.

I would also like to acknowledge and thank the faculty members and other students who contributioned to this work and advised me during graduate school including Ye Duan, Prasad Calyam, Chi-Ren Shyu, James Keller, Giang Bui, Truc Le, Ronny Bazan Antequera, and Norbert H. Maerz.

Lastly, I would like to thank all my friends and family for their support, enocouragement, and laughs as I completed my Ph.D. including Brandon Morago, Danny Gough, Chris Gough, Dan Gough, Rengarajan Pelapur, Cheryl Maslar, and my zoo.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

2D images and 3D LIDAR range scans provide very different but complementing information about a single subject and, when registered, can be used for a variety of exciting applications. Video sets can be fused with a 3D model and played in a single multi-dimensional environment. Imagery with temporal changes can be visualized simultaneously, unveiling changes in architecture, foliage, and human activity. Depth information for 2D photos and videos can be computed. Real-world measurements can be provided to users through simple interactions with traditional photographs. However, fusing multi-modality data is a very challenging task given the repetition and ambiguity that often occur in man-made scenes as well as the variety of properties different renderings of the same subject can possess. Image sets collected over a period of time during which the lighting conditions and scene content may have changed, different artistic renderings, varying sensor types, focal lengths, and exposure values can all contribute to visual variations in data sets. This dissertation addresses these obstacles using the common theme of incorporating contextual information to visualize regional properties that intuitively exist in each imagery source. We combine hard features that quantify the strong, stable edges that are often present in imagery along object boundaries and depth changes with soft features that capture distinctive texture information that can be unique to specific areas. We show that our detector and descriptor techniques can provide more accurate keypoint match sets between highly varying imagery than many traditional and state-of-the-art techniques, allowing us to fuse and align photographs, videos, and range scans containing both man-made and natural content.

Chapter 1

Introduction

Registering 2D and 3D imagery can create an enlightening venue for visualizing sets of photographs and videos and conveying spatial relationships among different cameras. 3D information in the form of LIDAR (Light Detection and Ranging) range scans shows us the full structure of a scene with depth information and allows users to navigate through a virtual environment to gain new perspectives of real-world data. 2D photographs and videos of the same location can reveal high resolution details on surfaces and capture dynamic objects and people at specific times. When these two modalities are combined, 3D models are created that present all of this information at once and the ability to explore a variety of exciting applications is gained. A series of photos taken over different seasons can be registered with a range scan and displayed together to show how the scene has changed over time. Combining data from different seasons can even be used to create 3D models of a scene under conditions when it would not be feasible to collect a LIDAR scan, such as during a blizzard. The time consuming scan be taken under more favorably conditions, photographs can quickly be captured in the snow, and they can be later registered to simulate a 3D representation of a winter scenario. Engineers can obtain needed real-world, 3D measurements from simple interactions with traditional photographs. Animations of people captured in videos can also be presented in a LIDAR-based virtual environment to aid in better scene understanding than that offered by a grid of 2D displays. 3D depth information can be mapped back down to the 2D image planes and used to create 3D stereo videos. The exact camera location of historical photographs can be identified and virtual 3D museums can be built [14, 15]. 3D models rendered with non-photorealistic techniques [16, 17, 18] can be created relatively quickly and easily [19, 20] to produce environments for artistic and entertainment purposes. Scenes can be constructed around 3D models creating an augmented reality scenario that can be used for visualizations of landscaping and urban planning. This dissertation describes techniques and methodologies designed for achieving many of these interesting goals.

No matter what the purpose of the registration process may be, determining the 2D-3D correspondence can be described as a pose-estimation problem [21]. Solving for a camera's pose in relation to a 3D model has been explored in a variety of contexts with some using 3D range scan models [22, 23, 24, 25] and others using structure from motion (SfM) point clouds [26, 27, 28, 29, 30]. Every one of these methods requires defining and matching some type of features either exclusively in 2D or from 2D to 3D. We have developed a pipeline that can take either route depending on what type of data is available. For the former, we match photos and videos to photographs captured by a LIDAR range scanner to create a 2D-3D link. In the latter case, we match 2D imagery to synthetic views of the 3D model.

Images can be matched for the purpose of 2D-2D alignment using information from locally defined feature descriptors [29], regional descriptors, or 2D-2D direct methods that rely on global pixel-pixel correspondences [3]. Locally defined keypoints use data such as gradient direction and magnitude from neighboring pixels within a relatively small window size whereas regional descriptors use information over a larger portion of the image to describe an area and potentially match it to corresponding regions in other images. While several methods already exist for detecting and matching sets of pixels using image intensity patterns, many will encounter difficulties when images are captured under highly varying conditions. Changes in lighting, camera exposure values, scale, perspective, exposure values, time, scene content, and artistic styles can greatly affect local gradient information surrounding keypoints. This presents the challenge of determining how to extract and describe features in a manner that is invariant to all of these factors [5, 7]. Popular keypoint detection and matching methods such as Harris Laplacian [31] and SIFT [32] that rely on such information may not perform well under these circumstances. In difficult situations, they may provide non-repeatable features, too few matches, or so many incorrect matches that a correct image alignment may not be identified after applying robust methods like Random Sample Consensus (RANSAC) [33]. On the opposite end of the spectrum, repetition in datasets, such as that found in architectural and other man-made scenes, can overwhelm 2D-2D matching algorithms with non-distinctive features, creating ambiguity in how to align images. In this dissertation, we propose a 2D-2D matching pipeline that builds upon traditional keypoint matching techniques and uses contextual information and mid-level information to handle these challenging scenarios. We hypothesize that using extra regional information can help clear up a great deal of the ambiguity presented by small scale gradient information to find a larger set of matches for registration purposes. We also present a method for estimating corner keypoint locations for situations where other repeatable features may be hard to detect and use this same contextual information to match them.

For the second 2D-3D registration scenario, where no pre-registered photographs are used or matches cannot be found with the pre-registered imagery, the difficulty lays in visualizing the 3D point cloud and the 2D images in a manner that they look alike so that defined features can be matched across the dimensions. We can create synthetic views of the 3D scan from a variety of perspectives [34] that visualize certain properties of the point cloud, such as the edges of major planes or the range of returned laser intensity values, that can also be identified and matched in the images and videos. Our pipeline for fusing 2D and 3D data and following one of these two paths is outlined in Figure 1.1.

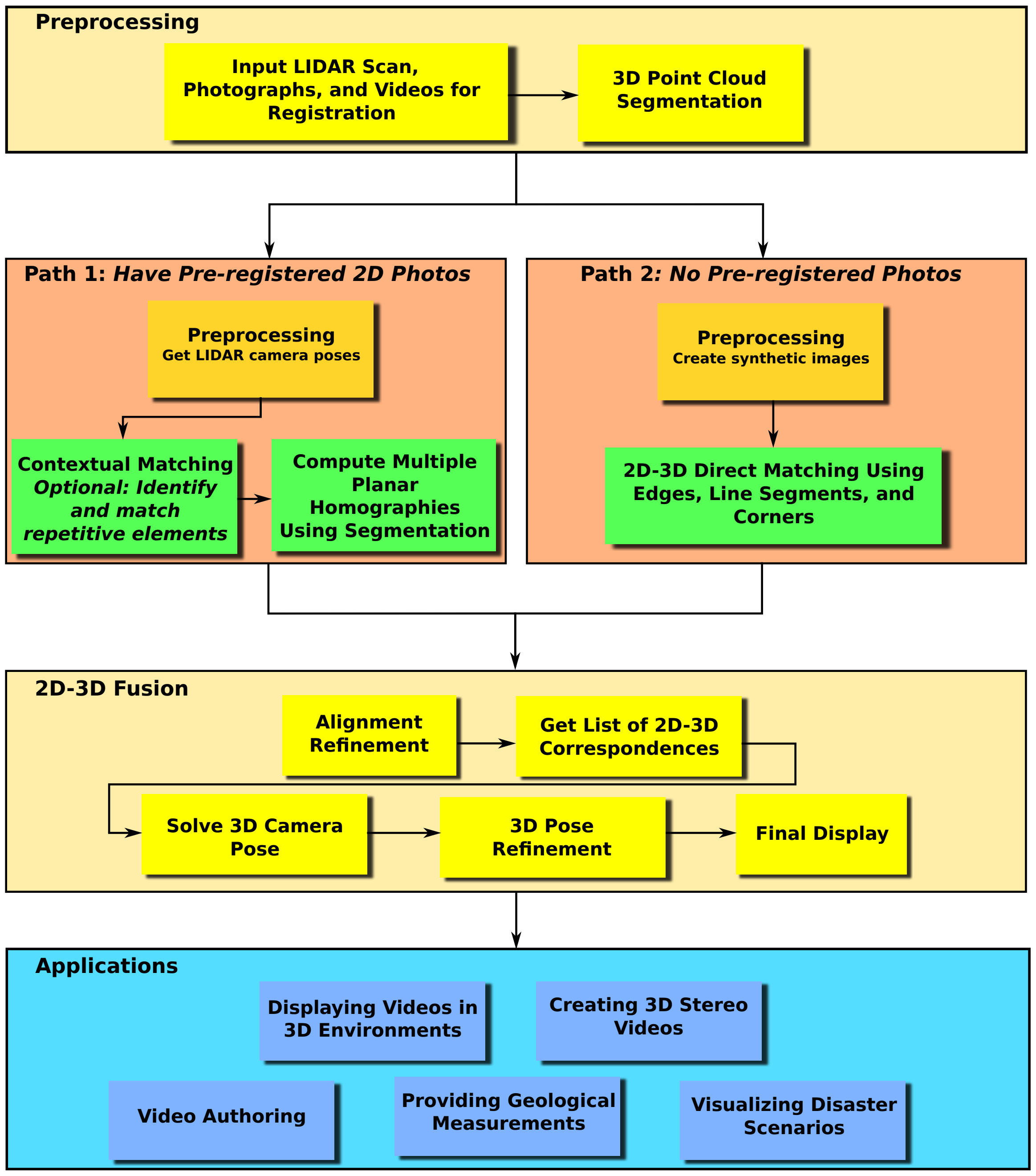

Preprocessing
Input LIDAR Scan, Photographs, and Videos for Registration
3D Point Cloud Segmentation
Path 1: Have Pre-registered 2D Photos
Preprocessing
Get LIDAR camera poses
Contextual Matching
Optional: Identify and match repetitive elements
Compute Multiple Planar Homographies Using Segmentation
Path 2: No Pre-registered Photos
Preprocessing
Create synthetic images
2D-3D Direct Matching Using Edges, Line Segments, and Corners
2D-3D Fusion
Alignment Refinement
Get List of 2D-3D Correspondences
Solve 3D Camera Pose
3D Pose Refinement
Final Display
Applications
Displaying Videos in 3D Environments
Creating 3D Stereo Videos
Video Authoring
Providing Geological Measurements
Visualizing Disaster Scenarios

Figure 1.1: Pipeline overview. 3D scan is read in along with photographs and videos that are to be registered with it. Depending on if pre-registered photographs are available or not, we match the new imagery with LIDAR photographs or synthetic views of the segmented scan. 2D-3D correspondences are collected and used to solve the camera's projection matrix. Once the 2D-3D registration is completed, we can use the data for a variety of applications.

We use our registration techniques to view multiple photographs and videos simultaneously in a 3D environment and to explore other exciting applications such as creating 3D stereo videos and authoring 3D videos. We are also able to use our work for very practical, safety related purposes such as assisting geological engineers and emergency response coordinators. Civil and mining engineers need to measure the orientations of faces of rocks and cracks along their surfaces to anticipate failures along walls of rocks. Knowing the orientations of individual or groupings of discontinuities can lead to successful stability predictions. However, traditional methods for obtaining these measurements pose many difficulties. Physically accessing sites can be dangerous, if not impossible, and sampling biases can affect drawn conclusions. Photographs and 3D LIDAR scans can be taken of a site with far greater ease than physical measurements can be obtained and can be used to collect comparable data. With our registration pipeline, we are able to fuse photographs that are simple for a user to interact with and LIDAR scans which contain accurate real-world measurements and can provide engineers with their needed data for analysis.

For emergency responders, we have combined the fields of computer vision, cloud computing, and high-speed networking to create 3D visualizations of disaster scenarios for scene understanding. In the later part of this dissertation, we present results for the three stages of a system that collects videos of a scene, performs the necessary computations to register them with a 3D LIDAR scan on a remote server, and transfers them to mobile devices for consumption and scene viewing. Our tests demonstrate how a high-speed network used in a disaster scenario can greatly increase the speed at which data can be shared amongst officials and emergency responders spread out over numerous locations making crucial decisions. We also show that multiple videos with recordings of many moving objects such as people and cars can be sent to a cloud server, registered, and viewed in 3D simultaneously using thin-clients by emergency response teams. We are able to avoid having to download all the data and necessary analysis software, increasing the usefulness of this system in critical moments when officials need to make quick, informed decisions to save lives.

1.1 Data Acquisition

We use a Leica C10 HDS LIDAR scanner which provides a high resolution point cloud of a scene and 2D images of the scanned subject using a built-in camera. The output data from the scanner also consists of files containing the internal and external camera parameters for each image. From this, we have the option of establishing a link between the 2D LIDAR images and the LIDAR range scan as shown in Figure 1.2. We can register photographs and videos with the LIDAR scan by taking advantage of these provided LIDAR photographs or we can ignore them altogether and perform direct 2D-3D matching. Our photographs and videos for registration were taken with a variety of cameras including a Nikon D80 and several smartphones. All of our data was collected on the University of Missouri-Columbia campus and in Rolla, Missouri. The LIDAR scans and LIDAR photographs from our university campus were collected during the summer months whereas the photographs and videos we register were taken during the summer, fall, and winter.

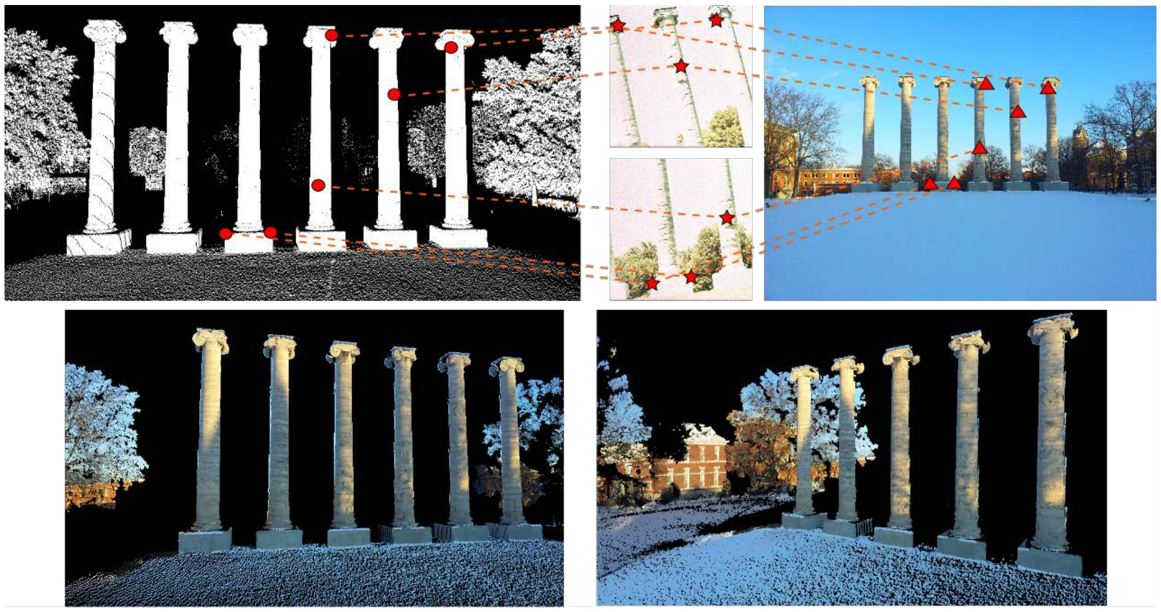


Figure 1.2: Matching 2D and 3D data in Path 1 of Figure 1.1. *Top:* Diagram showing 3D-2D link from LIDAR scan (*circles*) to LIDAR images (*stars*) to regular photograph (*triangles*). *Bottom:* 3D registration result shown from two perspectives.

1.2 Contributions

In this dissertation, we present the following contributions to the area of imagery registration and its applications:

- We present a unique contextual matching framework for finding keypoint correspondences that is designed to deal with the fact that images captured by different sensors and under different conditions may actually have very different image intensity and gradient patterns surrounding corresponding points. We combine our contextual information with local keypoint matching methods to create an ensemble feature. We propose using robust SIFT feature points or plentiful corner keypoints as anchor points for exploring potentially matching regions by comparing both HOG's and line segments. The MSER's that encompass each SIFT or corner keypoint, and that are often used to identify homogeneous regions, give us a general idea of the dimensions of local planar patches and what the neighborhood size should be for stable matching. We match both hard and soft edge features (lines and HOG's) in these regions to increase the matching feature set size between images with visual discrepancies. Line segments represent the salient features and structures found in an image and are relatively invariant to perspective and lighting changes. However, using line segments creates a very minimalist representation of an image. It does not provide as much distinct information as HOG, which picks up curved structures and textures. HOG, on the other hand, is sensitive to clutter in the image so we aim to strike a balance by using both measurements. For line segment extraction, we use the Line Segment Detector method [35]. This algorithm has been shown to accurately identify existing lines in an image without also including many false detections. This method achieves low false positive rates by incorporating line-support regions that require pixels on or near a line segment share very similar gradient orientations. Including our contextual matching method in our pipeline helps us to match stable regions across lighting and time changes in images. (Section 3)

- We describe a method for matching line segments. One of the main issues encountered in line-based matching is the lack of a robust distinctive line-based feature descriptor. Lines in general have no area and the sizes of their neighborhoods are not known or are hard to determine. The length of a line segment is fragile and therefore is not a stable attribute for comparison. However, the relative orientations and locations of line segments can be computed if a proper anchor point is available. In our work, we combine SIFT and corner points with line segments to conduct line matching. By exploring multiple scales of regions and using dominant gradient information in a local neighborhood, we are able to develop a coordinate system which can be used to encode the relative locations and orientations of the line segments in local neighborhoods. We avoid including any information dependent on the unstable end points of the line segments and focus on matching groupings of segments as opposed to individual lines. Our experimental results show that making use of line segments in this fashion can provide accurate information for matching on a large variety of imagery. (Section 3)

- In cases where the data is very repetitive and potential matches are ambiguous, such as in urban imagery, we apply a unique method for identifying both repetitive and salient regions and use these to guide the matching process. Our main contributions to the area of architectural imagery registration include methods for aligning rectified images one dimension at a time which reduces the search space and increases the image matching robustness, computing intra-image saliency maps in an unsupervised manner with no training set, and directly accounting for image region saliency when quantifying keypoint match quality. We perform our analysis directly on the images of interest without running machine learning techniques on a large dataset to identify unique regions. Our pipeline is designed to overcome several of the limitations and assumptions made in existing work. We use a technique for identifying repetitive patterns in images regardless of the spatial distribution of similar elements and we are able to match images of repetitive facades that only partially overlap [36, 37, 38]. We also remove the ambiguity in the feature space that is common in architectural imagery by collapsing repetitive elements into single representative patches. Finally, instead of only using either repetitive or salient features for matching [36, 37, 39], we take advantage of all the information available in images, and incorporate both types of features in our registration pipeline. (Section 4)

- For the direct 2D-3D matching stage, we present a unique method using a piecewise planar segmentation of the range scan to extract edge information from a 3D point cloud. The 3D segmentation intrinsically includes the geometry of the point cloud and provides edges that are lighting independent. These edges, which represent depth and normal discontinuities in the 3D space, usually appear as intensity differences in photographs and, thus, can be identified by extracting edges in 2D as well. We run Harris-Laplacian on the binary edge images and match corners by comparing HOG's of the surrounding areas. We use corners as keypoints as opposed to sampling the edges [40] to increase our confidence that our extracted features are repeatable across the images. Also, using HOG to study soft edges as opposed to other types of descriptor methods used in other 2D-3D direct matching algorithms [40, 25] like shape context or direct line matching between hard edges makes our work relatively robust to broken edges and takes into account all the available edge information regardless of curvature. (Section 5)

- We employ high-level semantic segmentation information for matching 2D and 3D data. We observe through our direct 2D-3D matching process how we can use both our piecewise planar segmentation and the Line Segment Detector method [41] to help us focus on keypoints on or near stable sections of the imagery that remain the same over time without requiring the use of machine learning techniques. Our segmentation allows us to conduct semantic labeling on the point cloud and divide the 3D data into three categories; buildings, trees, and ground. By only matching keypoints in the 3D data that lie on building segments and near line segments with keypoints in 2D photographs that are also near line segments, we are able to focus on matching regions of the images with similar attributes and reduce our keypoint matching search space. We can achieve similar goals as [39, 5, 42], but with this much simpler approach of only considering gradient, semantic, and geometric information in the scene. (Section 5)

- We have designed a program for geological engineers to collect measurements of rocky walls alongside roadways easily just by interacting with 2D photographs and 3D virtual models. (Section 7.4)
- We explore combining computer vision with high speed networking to create 3D virtualizations for scene understanding in disaster scenarios. (Section 7.5)

We show that the construction of our registration approach is general enough to work on imagery containing varying subjects including highly regularized architecture with repeating patterns and more natural, unorganized subjects such as buildings with curved features and non-standard designs, foliage, and images with rendering style differences. Examples of the imagery we work with are shown in Figure 1.3. We use a variety of methods to find image pair correspondences and make no assumptions about a scene's structure. Much of the current work in 2D-3D registration will face difficulties in complex scenes, including those heavy in trees or containing non-uniform architectural, as they rely on only extracting numerous straight lines or planes, identifying symmetrical features, or computing the normals of 3D points to match features [25, 22, 24, 43]. We do not determine correspondences between data by assuming a specific type of structural feature is abundant and identifiable. We use both local and regional feature descriptors to match data that will strongly respond to different image properties which we believe makes our pipeline more flexible across different data sets.

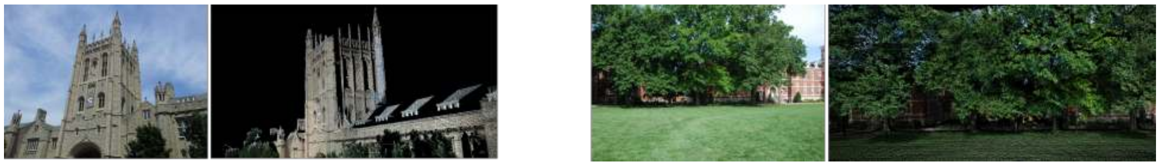


Figure 1.3: Examples of the variety of data we can register including images with heavy foliage and very structured architecture. *Left:* Original photographs. *Right:* 2D-3D registration results.

1.3 Organization

The remainder of this thesis is organized as follows: Chapter 4.1 provides a review of literature related to the work in this thesis. Chapter 3 details our method for registering 2D images with 2D images. Chapter 4 expands upon our contextual 2D-2D matching approach to handle scenarios with highly repetitive data. Chapter 5 discusses how we use and build on the 2D-2D work to align 2D images with 3D LIDAR scans. Chapter 6 covers details on displaying the 3D model after images have been fused with a LIDAR point cloud. Chapter 7 presents several applications for our registration work and Chapter 8 summarizes this thesis.

Chapter 2

Literation Review

Our work for registering multi-modality data touches on detecting significant features, defining unique descriptors, and visualizing specific properties of images that can be defined in both 2D and 3D data. Many of the research projects that explore these topics focus primarily on either general 2D-2D matching, finding matches in the presence of repetition, computing a 3D structure, or 2D-3D registration between images and existing models. Some of the 3D work in this field also includes steps for building a 3D geometric model to represent and replace original 3D point clouds and to fill in missing data.

2.1 2D-2D Image Matching

Various aspects of our pipeline require that different types of 2D images be aligned, which poses a number of difficulties even before attempting to fuse data from across different dimensions. Images can be aligned using information from locally defined feature descriptors, regional descriptors, direct methods that rely on global pixel-pixel correspondences, or structural features such as line segments. Locally defined key-points use data, such as gradient direction and magnitude, from neighboring pixels within a relatively small window size. Regional descriptors use information over a larger portion of the image to describe an area and potentially match it to corresponding regions in other images. Global matching techniques, such as optical flow and Fourier-based alignment, can provide a very accurate image alignment, but generally require a good initialization to be successful [2].

We try to take advantage of the strengths of several of these methods by combining them together into one ensemble feature matching and alignment refinement approach that uses local, regional, global, and structural information.

Local Matching Methods

The Harris detector and use of the Hessian matrix are two classical methods for finding local interest points in images [2] [44] [31]. Both detectors use the second moment matrix to study the gradients of pixels surrounding an interest point. The scales for points' feature descriptors are often chosen using the Laplacian [45]. These methods work to identify local areas with significant visual changes, such as corners, and are fairly invariant to lighting changes. However, these early methods were shown to be sensitive to large scale changes and had limited affine invariance, which led to the development of more sophisticated detectors and descriptors. For example, the Scale Invariant Feature Transforms method (SIFT) [32, 46] is a commonly used and robust local feature matching technique that assigns a scale and orientation to each interest point. A descriptor is constructed based on local image gradients that is generally unique to a single area in the image. The neighborhood size used to construct the descriptor is usually around 16x16 pixels at the appropriate image scale. Several groups have expanded on the SIFT descriptor to develop feature matching methods that are faster and more robust to various transformations [47] [48] [49].

The relatively small window sizes used by these methods to create keypoint descriptors can be beneficial in many cases. They will be able to identify locally distinct areas accurately. However, cases where image pairs have large scale, lighting, and/or content changes may require a larger window size to correctly pinpoint very distinct keypoint matches since the images will have so many visual differences.

Regional Matching Methods

Several methods for matching and aligning photographs have expanded on these locally defined feature descriptors to use visual information covering a larger region within images. Yang et al. [3] designed a pipeline using bootstrapping and region growing to search for a dense set of corner and edge features. They use these features to compute a global homography relating two images. Since this method uses a single homography to overlay an image pair, it is best-suited for planar-approximate scenes. Hacohen et at. [50] also identifies dense matches across image pairs. They align image patches by searching across a set of transformations such as rotations and translations that overlay pairs of patches as well as possible and use groupings of matches to calculate the coherence of a region.

Other groups have developed region detectors and descriptors that rely on less-structured features. MSER's are used to identify image neighborhoods that are visually similar and contrast greatly with adjacent regions. The SIFT descriptor can be used to describe and match corresponding MSER's in image pairs [51]. MSER is affine invariant but has been shown to be biased to "round" shapes [52]. If an image has repetitive patterns, such as a row of windows, it is possible that the MSER feature will be ambiguous if its size is not large enough. Shape contexts [53] describe feature points that are uniformly sampled along contours based on nearby keypoint's relative locations. These groupings of points are used to find optimal image correspondences and are useful for object recognition. Contours throughout the image must be correctly extracted, which is not always an easy task.

Measuring the similarity of all overlapping pixels is another approach for region matching. Normalized cross correlation, the sum of squared differences, and the sum of absolute differences all perform a pixel-by-pixel comparison and can be used to find the best alignment for an image pair [54]. Histograms of gradients (HOG) have also been used as unstructured region descriptors for image alignment verification [55] and object recognition [56]. While these methods can be very informative, scale and orientation information should be known in order to use them efficiently. To maximize their robustness, and to avoid exhaustively searching for overlapping regions, they can be combined with other information or descriptors to provide this information.

VLAD (vector of locally aggregated descriptors) [57] is a method for describing the feature descriptors of an image in a vector of limited size. This can be used to speed up image searching in a very large database. First features are clustered with a Bag of Features (BOF) type methodology. The image is represented in k x d dimensions where k is the number of clusters and d is the dimension of the feature descriptor being used. For each cluster, the sum of the distances of each of its assigned descriptors from the cluster center is calculated. Each cluster vector is L2 normalized by this distance. A graphical representation of VLAD vectors for sets of images show that images of the same scene taken from different perspectives have very similar appearing VLADs.

Linear Features

Extracting long linear features is another approach used for matching images. This can be a very difficult task as line features are relatively fragile and their lengths may vary with changing viewpoints and lighting conditions. Wang et al. [58] match clusters of line segments, or line signatures, in image pairs using the assumption that local neighborhoods of strong structural features will maintain the same relative arrangements throughout baseline and lighting changes. Fan et al. [59] also match lines by using spatially proximate features. They consider two lines to be a matching pair if they have similar distance ratios to neighboring keypoints that are known matches. Since each potential linear match requires nearby correctly matched keypoints, this method is dependent on having a relatively large number of correct feature point correspondences. Color histograms of the color profiles surrounding a line segment have also been used for line matching under the observation that the changes in color between two intersecting planes is relatively invariant to camera motion [60].

2.2 Matching Repetitive Data

Several groups have developed techniques to use different types of structural and high-level information to guide image matching. The goals of these methods are generally to overcome pitfalls of locally or even regionally defined descriptors [61, 32, 31, 2, 47, 12] that are ambiguous in the presence of repetitive features. While locally defined descriptors can provide very accurate information that is invariant to scale changes and scene clutter, in certain cases they do not take into account large enough image regions to be truly distinct [62].

To make matching images with repetitive architectural patterns a more tangible task, researchers have explored using larger-scale properties such as feature symmetry [63, 24] and repetition to identify corresponding regions [11]. Hauagge et al. [7] search for horizontal, vertical, and rotational symmetries about various axises and scales across images. Self-similarities can be identified in patterns of colors, edges, and repeated visual elements by measuring how similar a small region is to parts of the larger surrounding region [64]. Wu et al. [36] match SIFT features within an image to identify repeating and symmetrical elements that occur at regular intervals and the boundaries between repetitions. Kushnir et al. [37] also match SIFT features across images to find repeating elements and use agglomerative clustering to group similar regions. This group presents two versions of their matching algorithm. Similar to [36], one version focuses on images where grids of periodic elements can be identified. The other handles cases where repetition exists in a less predictable fashion. Matches between non-repetitive features are identified and used to solve a transformation between an image pair. Chung et al. [38] use graph matching to globally align images of buildings that capture the same portion of a structure's facades. This work identifies repetitive MSER patches and constructs a semantic sketch of a building to match between highly varying viewpoints. Bansal et al.[65] also work to align images of buildings taken from highly different viewpoints such as aerial to ground level. Aligning images of high-rise buildings that have many repetitive elements and appearance differences due to view changes poses difficulties. They construct an embedding of a facade that captures how similar or dissimilar image regions are to their neighbors. This group uses the periodicity of a repeating element as an estimate of the scale of interest. This group's contribution of a "scale-selective self-similarity" can be extracted from any pixel in the image bypassing the need for a repeatable detector. PatchMatch [66, 67] provides a framework for matching patches across images by randomly assigning and searching for correspondences. This method uses a combination of searching through different scales, random sampling, and distance propagation to guide the matching process using a cooperative hill climbing strategy to find a patch's set of nearest neighbors. Ceylan et al. [11] construct grids of repetitive elements within images and limit an aligning transformation to be one of a discrete set of possible solutions that overlay different images' grids.

A different perspective for matching data that has repetitive elements is to find the salient regions in images and use these areas to guide registration. Often times, machine learning techniques are applied to identify these types of unique regions as is described later in this section.

2.3 Computing 3D Structure

Once keypoint correspondences are established amongst an image set, the 3D structure of a scene can be computed by estimating the relative camera position between images [68, 69] and triangulating matching keypoints into a 3D space [70, 29, 26].

Furukawa et. al. present details for accurately solving camera calibration parameters and the 3D structure of a scene in several works [71, 72, 73, 74]. In [71, 72], the model surface is represented with a dense set of oriented, rectangular patches that are visible from varying camera perspectives. Photometric consistency and visibility consistency of the patches are enforced during the refinement stage of this algorithm. The NCC of the reprojected patches on different image planes must be below a certain threshold, and no image patch can occlude/be occluded by other patches. [73][74] build image pyramids and run PMVS [75] on levels to obtain a set of oriented points with visibility information. The studied pyramid levels are chosen based on their size relationship to the expected reprojection error. For efficiency purposes, a certain number of feature points are randomly selected from each image patch (of a predetermined size). Refinement of feature points is performed by comparing local image textures. A 3D patch is constructed surrounding an image point and projected into other image planes. The feature point is corrected based on its relationships to correspondences on these planes. This is repeated on each lower level of the image pyramid. If the point moves too much at the bottom level, it is discarded. SBA is then called on to refine the 3D structure of the scene. Viewpoint invariant 2D features are created by extracting planes from the 3D structure and normalizing the 2D features in relation to these planes. The homography between the 3D plane and the image plane is determined and used to warp the image plane [76].

Wendel et al. [77] created a system for performing dense reconstruction using mobile devices such as tablets. A video can be taken from a moving car in normal traffic and the user is shown a reconstruction in real time with tracking information that he can correct. Information is transfered to a server over a 3G or Wifi connection that is used to create a 3D reconstruction. Global bundle adjustment is only performed on the reconstruction when the mapper is idle. The 3D structure is sent back to the mobile device and used for tracking the camera position. A mesh is created of the scene by fusing depth maps. The maps are represented implicitly and fused by minimizing a convex energy function. The mesh is visualized using a GPU-accelerated raycaster that can work directly on implicit geometry.

Durand et al. [14] present a method for pinpointing the exact camera location and orientation of a historical photograph. Two new photographs, with about a 20 degree baseline are captured and used to construct a 3D model. This model is referenced to calculate the camera parameters for the historical photograph. The user is then guided on how to move her camera to match the historical viewpoint as the program solves the relative position between the camera and the historical image in real-time.

2.4 2D-3D Registration

Many groups have worked on 2D-3D registration, following the general framework of taking existing 3D information, such as a range scan [25], and fusing images and videos with it [22, 23, 24, 25]. The techniques used to perform this type of alignment often build on the type of work described in the previous sections (Sections 2.1 and 2.3). Oftentimes various types of features (image-based, structural, etc.) [78], certain system constraints (limiting camera motion to rotation only) [79] and/or environmental information (GPS, GIS information, etc.) [23, 80] are assumed to be present and this prior knowledge guides the registration process. This overall body of work can be broken down into two main modes of performing 2D-3D registration, i.e. direct 2D-3D matching and using pre-registered images. Machine learning is sometimes incorporated in both of these categories to obtain helpful information for registration. Directly matching 2D photographs to 3D range scans is a common technique, generally matching large structural features such as lines or circles [25]. These methods assume that certain features are present in the scene which is reasonable since many groups focus on urban data that mainly contains regularized architecture such as multi-story buildings with rows of windows, doors, and balconies. They also often require some sort of user guidance and usually work exclusively with images and range scans that contain the same content. The second approach discussed here is to assume that there is a set of photographs that is pre-registered with the range scan because they were taken by either a camera built into the scanner or a camera mounted on the scanner with a known configuration. In this situation, potential obstacles in 2D-2D matching must be considered such as how different camera sensors, lighting conditions, time changes, and varying perspectives affect image intensity patterns.

Direct 2D-3D Registration

The photo-realistic urban modeling system produced by Stamos et al. [25, 78] identifies edges and vanishing points in the 2D imagery and matching 3D linear features to estimate the camera parameters used in the 2D-3D registration. Wan et al. [30] also take advantage of the linear features inherent in urban scenes to compute camera locations from vanishing lines. Schindler et al. [24] register 2D and 3D urban data by identifying patterns in the structural features in both the 2D and 3D data and then matching them. The work of Li et al. [22] identifies repetitive structures as well and utilizes vanishing lines to rectify photographs before registering them with the LIDAR range scans. [81] also relies on the abundant orthogonal lines found in aerial urban data, using vanishing lines for initial camera pose estimates. They obtain corners from intersecting lines and match corners across dimensions based on proximity using their initial pose estimates. Matei et al. [82] build closed 3D models of buildings present in aerial LIDAR scans. They consider the similarities of HOG's of edges surrounding corners in the 3D models where 3D planes intersect and potentially corresponding areas in ground-view photographs. Another approach to direct registration is to create a structure from motion point cloud from a set of photographs and match the 3D output with the range scan [25, 80].

Mutual information can also be used for direct 2D-3D registration. 2D images can be constructed from a LIDAR scan that visualize various properties of the scan such as the reflectivity of the laser [83], normals [84], or the relative height throughout a point cloud [85, 21]. The entropy between these types of images and regular photographs is minimized to uncover the relationship between the two. A scan's reflectivity image will not contain the same lighting inconsistencies (such as shadows) as a regular image, limiting the flexibility of this approach. The height image created in the work of Mastin et al. [21] is appropriate for the task of registering aerial urban imagery. The shading present in regular photographs in such scenarios will often have a relationship with the height of the buildings.

Using Pre-registered Images for Registration

Using pre-registered photographs to guide the 2D-3D registration may be a more tractable approach since data is available that is similar to the photographs being registered. However, there are still many difficulties to handle especially when the goal is to register 2D and 3D data that are obtained on different days as the scene may have changed. Challenges also arise if different modalities are used to depict a scene such as infrared images or paintings. Yang et al. [43] match difficult image pairs by identifying one stable matching feature between an image with unknown location and a pre-registered image and use a region growing method to find more correspondences. As the feature search space increases, they consider matches between corners, edges, and normals. They feel that their iterative approach to searching for and checking feature correspondences helps to overcome some of the inaccuracies encountered when matching keypoints between images that differ in qualities such as illumination and viewpoint. Russell et. al make alignment estimates between multiple modalities such as structurally-accurate paintings and photo-realistic structure from motion-based 3D meshes by using the gist scene descriptor [86] and edge information of different regions to match paintings and OpenGL renderings of synthetic views of the 3D model [40, 39]. Weinmann et al. [87] handle the challenge of aligning regular and infrared photographs by matching general shapes of the images' gradient fields across planar areas. [88] uses LIDAR with color intensity information, implying use of a pre-registered camera, and creates 2D images of synthetic views. They then match SIFT keypoints between the synthetic views and regular photographs. This process requires a high density point cloud and intensity information that matches relatively well to that of the photographs being registered.

Machine Learning for Registration

Machine learning techniques can be used to identify unique regions in photographs, paintings, and 3D models to match imagery with highly varying properties and to perform object recognition [5, 42, 39]. These mid-level discriminative features ideally occur often enough in the data to be learned, but are different enough from the rest of the imagery to be clearly located in photographs and models with visual dissimilarities. [58] matches the contours of "regions of interest" across dimensions for aerial views of urban scenes. They use learning methods to determine what contours and shapes constitute a building outline requiring clean, closed contours and many orthogonal edges. Sun et. al uses a data-driven approach to identify and match unique features using a combination of complimentary descriptors across domains [89]. Training a system in this manner to learn salient features tends to be time consuming and computationally intensive.

2.5 3D Model Augmentation

Once a 2D-3D fused model has been computed, or in some cases in order to aid the registration process, steps may be taken to add in or replace existing 3D information. This may be done by filling in missing data in the model, computing a basic-geometric model of a 3D scene [90] to replace a point cloud [91], or registering multiple 3D point clouds together that capture different information [92]. This augmented 3D information can be beneficial for visualization since models consisting of basic primitives such as planes and spheres tend to be more complete and denser than point clouds which may have missing information. It is also possible that the planes and edges extracted from the geometric process can provide useful information for aligning 2D and 3D data.

2.5.1 Filling in Holes

It is common for LIDAR scans to have missing information or holes in the structures scanned. There are three main causes of these holes [90]:

1. Occlusions: An object (Object A) in the path between the scanner and an object further away (Object B) will prevent the laser from reaching the entire "viewable" surface of Object B. The result is a hole of the general shape of Object A will be present in the scan of Object B.

2. Laser reflectance. If the laser is aimed at an object that will not reflect it back to the scanner, such as a window or the sky, no information will be returned to the scanner for that particular angle.

3. Data out of range. LIDAR scanners have a limited range of view that can be manipulated by the user by specifying that only a certain area be scanned. Each scanner also has physical limitations on the range of angles it can scan. For instance, the scanner may not be able to collect data on the ground directly below it.

One approach to filling in such holes in order to create a more complete 3D model is to use inpainting [90][93]. An image that is registered with the range scan is searched to find a source area with a similar texture to the target area corresponding to the missing 3D information. One way to use this information is to create a library of texture patches and their corresponding 3D gradient information as was performed in [90]. This library can be searched and matched to small sections of a hole's border and the selected patch can be registered in 3D to find its correct location in the scan. Another approach for finding the 3D depth information for missing data is to create a depth image of the range scan and find a source region that has similar gradients to the border of the target region [93]. The depth values at the boundaries of the 3D target area are used to guide the creation of the new 3D points using the gradient information from the source area. Fruh et al. [94] also developed a method for filling in holes and fixing inaccuracies in range scans due to windows and use the updated information to create an accurate 3D mesh. Goesele et al. [95] proposed using "ambient point clouds" in which views that have not been explicitly captured are interpolated using streaks between points.

2.5.2 Computing 3D Geometric Models

Gallup et al. [96] identify planes in a 3D point cloud to create a more complete model. This paper calls for performing multiple sweeps through a scene to identify planes with different normals. These planes are then texture mapped with images and video frames. A later work [97], builds on this to also identify non-planar regions. A texture mapped polygonal mesh is created from the planar and non-planar sections. [27] also computes planes in 3D models of urban settings. This method calculates camera positions without performing bundle adjustment, using panoramic street-view images. Alharthy et al. [98] create 3D polygonal models from aerial LIDAR point clouds by considering geometric properties and depth variations through the point cloud. Sinha et al. [99, 28] process SfM clouds and identify planes in the 3D structure. Similarly to many other works in this area, they texture map the 3D planes from an image to create a 3D photo-realistic model. The 3D planes are constructed either with user-guidance or by using information from the line-segments and 3D points. Vanishing points are calculated in 2D images and matched to 3D line segments. They utilize graph-cuts to segment the image prior to texture mapping the planes.

Grzeszczuk et al. [100] focus on creating 3D architectural models using different data sources than those commonly used in this area. Geographic Information System (GIS) databases are relied upon to obtain actual building dimension information and GPS information is used to get a general idea of the camera location. The optimal images for performing the final texture mapping are chosen by determining how well a building view is represented from a particular perspective. This measure looks at the amount of building visible based on viewpoint, how directly a camera is facing the building, and the number of occlusions.

2.5.3 3D-3D Point Cloud Registration

Yang et al. [92] register LIDAR range scans with wide baselines. They use a range scanner that provides both 2D photographs and a 3D point cloud along with 2D-3D correspondences. They match regular camera images to the range scanner images that are taken from similar vantage points using SIFT. This group then creates an SfM point cloud from the camera images and register it with range scan using the approach described in Liu and Stamos's work [25]. As more and more camera images are considered, the feature matching between images is improved using the 3D information. The improved 2D matches are then used to register multiple range scans. Yao et al. [101] perform planar segmentation to range scans and fuse neighboring point clouds by fusing segments.

Chapter 3

2D-2D Contextual Matching

Matching photographs and finding image correspondences is necessary for a variety of applications in computer vision from creating structure from motion point clouds to image classification. These applications have encouraged the proposal of many different feature detectors, descriptors, and matching techniques, all of which have their own strengths and weaknesses. This being the case, each technique may be more well-suited for specific tasks than others [31]. On one hand, defining features on a very local scale can create a relatively large set of keypoints that are highly informative for matching and invariant to certain amounts of perspective and content changes. However, limiting the amount of visual information accounted for in a descriptor may obscure the differences in these local keypoint descriptors [62]. This is especially true when highly different viewing conditions, clutter, and occlusions are present, causing local visual properties to change, making it difficult to match locally-defined keypoints correctly. Using a larger scale and considering a wider context can help define more discriminative features. By combining local and regional information into one ensemble feature, we feel we can achieve the best of both worlds by taking advantage of both precise local information and distinct regional data.

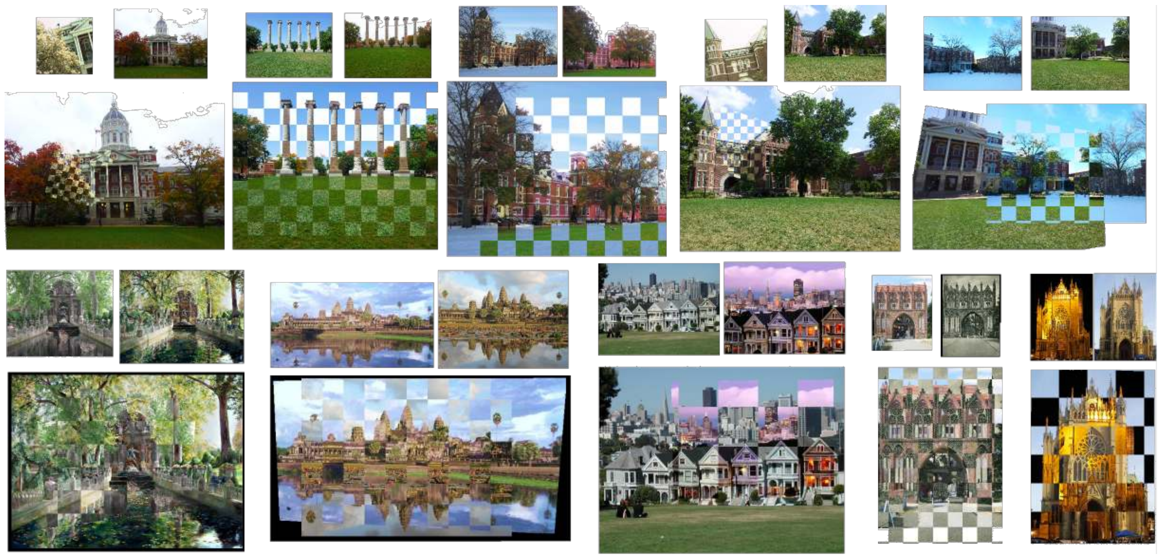


Figure 3.1: Registration results after applying our pipeline to several challenging image pairs from our dataset as well as from the datasets provided in [5] and [7]. Original images are shown above registered images.

In this chapter, we present a framework for using contextual information for image matching with the goal of increasing the number of keypoint matches found between challenging image pairs in order to find an aligning transformation relating the images. We mainly work with piece-wise planar man-made scenes that need not be time adjacent, examples of which are shown in Figure 3.1. For planar-approximate image patches, we believe that using relatively large regions for keypoint comparison helps features remain more distinct throughout scene and imaging condition changes. Our contextual framework for registering images can take three forms. The first is to build upon and complement existing sparse local feature matching techniques such as SIFT. We describe the surrounding regions of keypoints in terms of salient structural features via line segment extraction and their rich texture information using histograms of gradients (HOG). These neighborhood sizes are determined adaptively using maximally stable extremal regions (MSER) [61]. To eliminate the requirement that we incorporate an existing matching algorithm, we can also use our regional contextual information to describe and match Harris-Laplacian corners. The third option is to estimate corner locations using extracted line segments, which we call Pseudo Corners, freeing this algorithm from any dependencies on existing detectors or descriptors. After combining several local keypoint and regional template matching techniques to make an educated estimate about the image alignment, we follow up with iterative and global refinement stages using corner, edge, and gradient information across the entire image planes. Our pipeline is outlined in Figure 3.2.

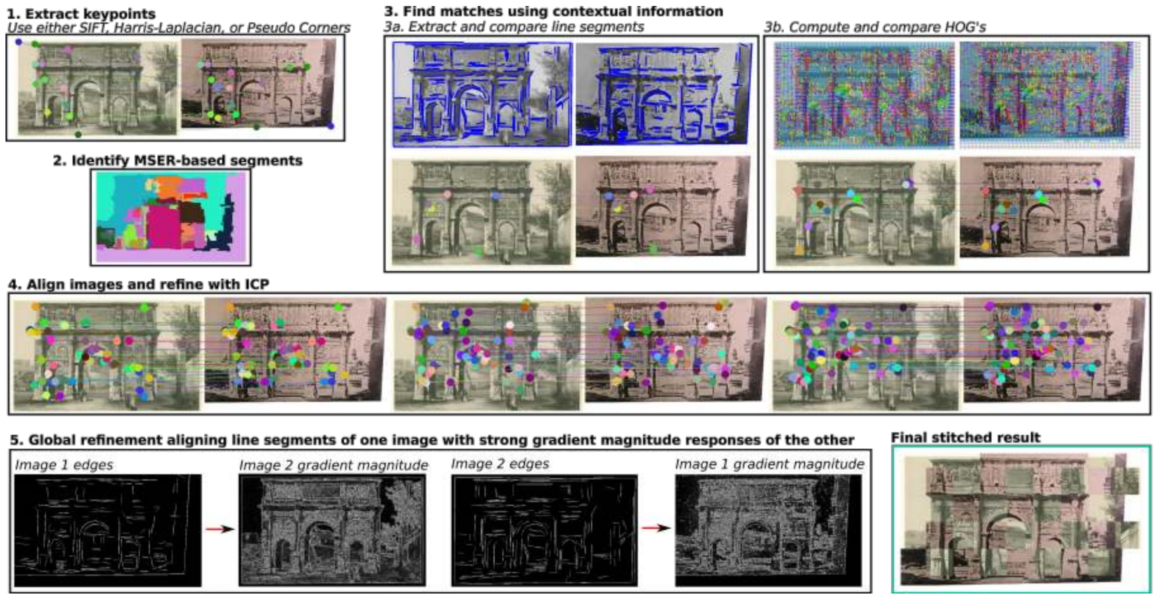

1. Extract keypoints
Use either SIFT, Harris-Laplacian, or Pseudo Corners
2. Identify MSER-based segments
3. Find matches using contextual information
3a. Extract and compare line segments
3b. Compute and compare HOG's
4. Align images and refine with ICP
5. Global refinement aligning line segments of one image with strong gradient magnitude responses of the other
Image 1 edges
Image 2 gradient magnitude
Image 2 edges
Image 1 gradient magnitude
Final stitched result

Figure 3.2: Outline of our registration pipeline. Keypoint features are first extracted. SIFT keypoints, Harris-Laplacian corners, or our proposed Pseudo-Corners can be used. If SIFT is used, keypoints are matched using the SIFT descriptor. We then search for matches using contextual features. I_1 is segmented into MSER-based neighborhoods which are used to determine neighborhood sizes for studying line segments and HOG's. (HOG information in top right image is colored according to the dominant gradient direction for each cell.) Linear and HOG information is used to verify new matches. An aligning homography is calculated and refined using ICP and gradient-based methods.

3.1 Notation

For simplicity during descriptions in this chapter, we denote an image pair as I_1 and I_2. A feature point in I_1 is represented as A and its two closest matches in I_2 are B and C. When I_2 is transformed to I_1's coordinate system using a homography, H, I represent it as I_2'.

Throughout our discussion, we use the term "contextual" to refer to the set of matches verified using both linear and HOG contextual information. "Ensemble" will refer to the set of features combining both SIFT matches and contextually-verified matches.

3.2 Contextual Feature Matching

The goal of our method is to extract highly distinct interest points and obtain a large number of correspondences that can be used to correctly register image pairs [62]. Our pipeline begins by using regional contextual information to match keypoints. We explore using three different types of keypoints as anchors for matching contextual descriptors. The first approach we study is matching existing locally defined keypoint descriptors, such as SIFT, and expanding upon them with our contextual information to create an ensemble feature. The second keypoint we test is Harris-Laplacian using only our contextual work for matching. The third, Pseudo Corners, is a new keypoint type we propose which is identified using line segments. We then identify inliers in the match set from any of these methods using geometric verification, globally estimate and refine the image alignment, and increase our set of matches.

3.2.1 Adaptive MSER Neighborhoods

We use MSER's to determine the neighborhood that is considered during contextual matching. MSER's are often used to find homogeneous areas in images that contain distinctive information that is useful for matching [102, 103]. An ideal neighborhood needs to contain enough regional information to clear up any ambiguity at the local keypoint level but not include so much information that no regions actually match. We also want every pixel in an image to belong to an MSER so that we have a "stable" neighborhood ready to select for every potential keypoint match. To find potential MSER's that satisfy these criteria, we create a Gaussian pyramid for an image and extract MSER's at each level, as shown in Figure 3.3. At the original image resolution, we get many MSER's that tend to be relatively small. Also, initially, not all pixels are assigned to an MSER. As we move down the pyramid and extract MSER's on increasingly blurred images, we end up with fewer, larger regions and more pixels assigned to a region. Just to ensure that every pixel is indeed assigned to a region, we have one post-processing step after MSER's are extracted at each level. Connected components of pixels that have no region label are identified. Each component is merged with the region that touches the highest number of pixels along the component's perimeter. In general, segmentation is a very difficult task, but this adaptive MSER labeling gives us a reasonable approximation of the regions we need.

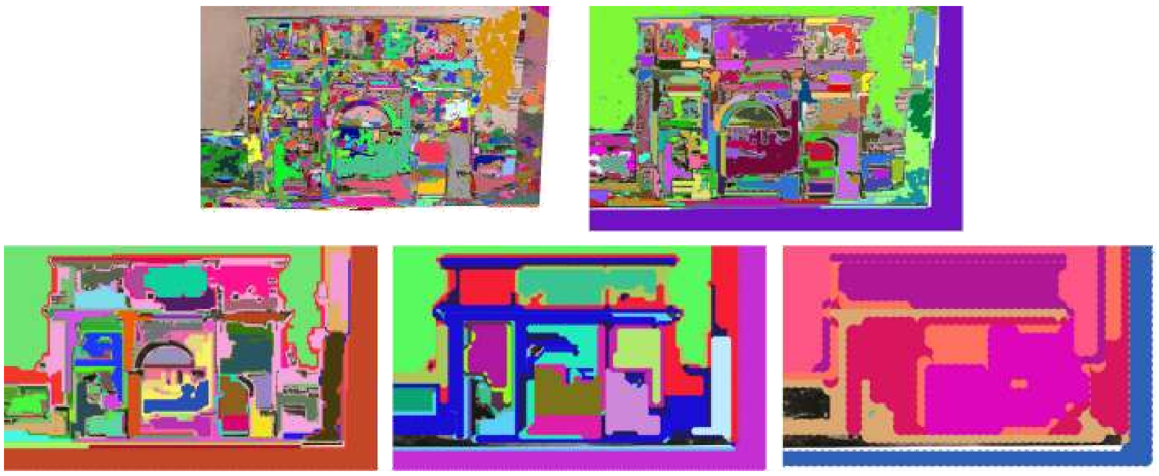


Figure 3.3: MSER's extracted at different levels of a Gaussian pyramid. Segments are combined from different levels to create adaptive MSER segments shown in Figure 3.2.

To help achieve our goal of assigning every pixel to a mid-sized MSER, we allow different pixels in the image to be assigned to MSER's from different Gaussian pyramid levels. Initially, every pixel is assigned its corresponding label at the bottom level (original image resolution) of the pyramid. Then we begin moving up the pyramid. If a pixel's corresponding region in the next level of the pyramid has an area that is no more than 10% of the image area (a threshold determined experimentally for our work), the pixel takes on the higher level region area.

The keypoint's neighborhood in I_1 used during contextual matching is an orthogonal bounding box that encompasses an entire chosen MSER. Using this bounding box essentially grows the region slightly and allows us to take into account boundary information surrounding the distinctive area. During the matching stage, the MSER bounding box in I_1 is mapped to I_2 at multiple relative scales to take into account scale changes between the images. The bounding boxes are oriented such that each one's x-axis is aligned along the dominant gradient direction within the image patch. An example of using this information to find a corresponding region between I_1 and I_2 is shown in Figure 3.4.

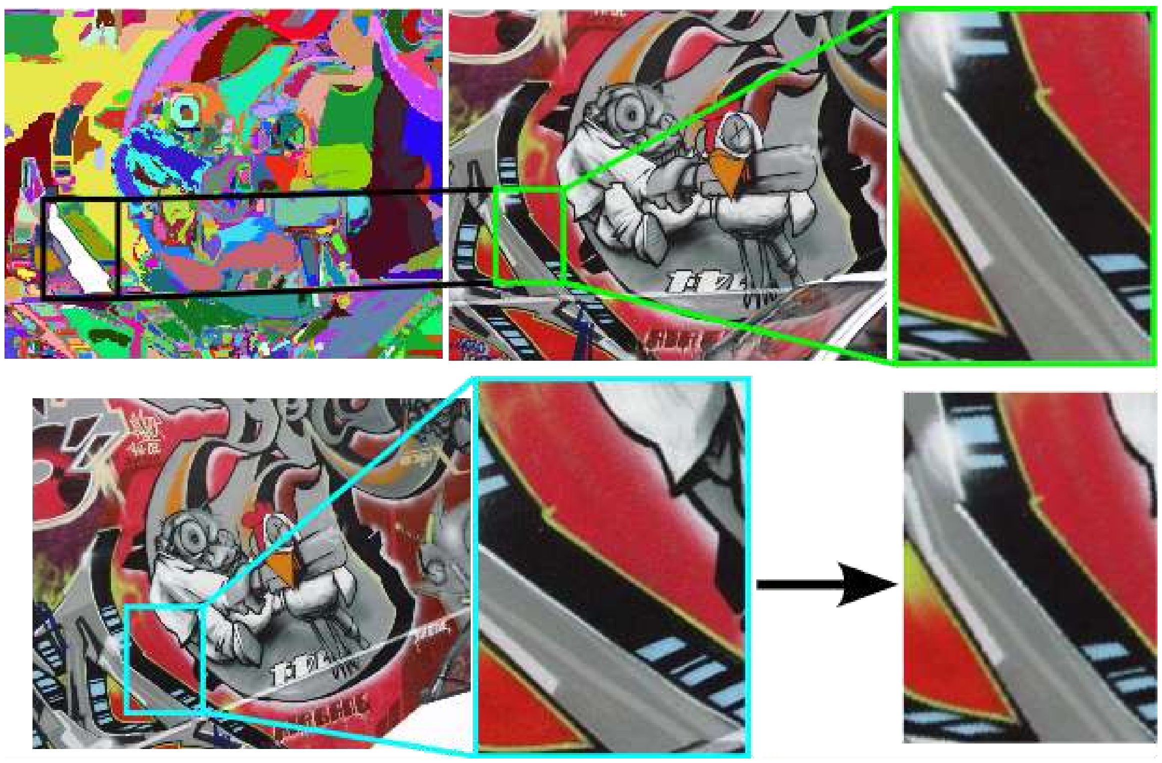


Figure 3.4: Mapping bounding box of MSER neighborhood in I_1 to I_2. *Top:* MSER segments and MSER defined neighborhood in I_1. MSER segment being used is shown in white. *Bottom:* Original MSER neighborhood in I_2 and I_2's patch after it has been transformed to the same coordinate system as I_1's patch via a scaling and rotation. The patch in I_2 is extracted at multiple scales and each patch is rotated such that its x-axis points along the dominant gradient direction in the patch.

3.2.2 Contextual Line Segments

We use linear contextual matching to describe potential matches by searching for and matching line segments near keypoints. Stable line segments in the MSER-based neighborhood of potentially matching keypoints are identified using the Line Segment Detector method [35]. Line segments are extracted from I_1 and I_2 at three different levels of a Gaussian pyramid. The lines found at each Gaussian level are all combined into one set of lines. Extracting lines from multiple image scales helps to address the fragile nature of line segments and gives us more information for matching. Each line segment must have a length of at least $0.03 * max(image_{width}, image_{height})$ and its midpoint must be located within the selected MSER bounding box to be taken into account. All the stable line segments in this neighborhood are transformed to a polar coordinate system determined by the neighborhood's dominant gradient orientation and normalized by the neighborhood's dimensions and are represented as (ρ, θ). We then match groupings of line segments in corresponding neighborhoods using Hungarian graph matching [104]. Figure 3.5 outlines this process. Two line segments (l_A, l_B) are considered to be similar if they have a low dissimilarity score according to Equation 3.1.

$$dissim(l_A, l_B) = (\rho_A - \rho_B)^2 + \omega(\theta_A - \theta_B)^2 \tag{3.1}$$

ω is a weight parameter that normalizes the range of θ ($\theta \in [0, 2pi]$) to that of ρ which is based on the contextual neighborhood dimensions. This dissimilarity measure is only calculated if at least three lines are identified in the contextual neighborhood of the keypoint. If a required percentage of the line segments have low dissimilarity scores, the keypoint match is saved. Table 3.1 shows how the precision of linear contextually verified matches changes as the required percentage of matching lines in a region and the dissimilarity score thresholds change. These values were obtained by running our method on the symmetry dataset in [7]. Figure 3.6 shows a set of matching lines.

Table 3.1: Average Precision on Symmetry Dataset for Varying Graph Matching Parameters - dissimilarity score and threshold on required % of matching lines

-	$dissim = 1.0$	$dissim = 2.0$	$dissim = 3.0$
10%	0.552	0.617	0.412
30%	0.371	0.426	0.365
60%	0.000	0.081	0.081

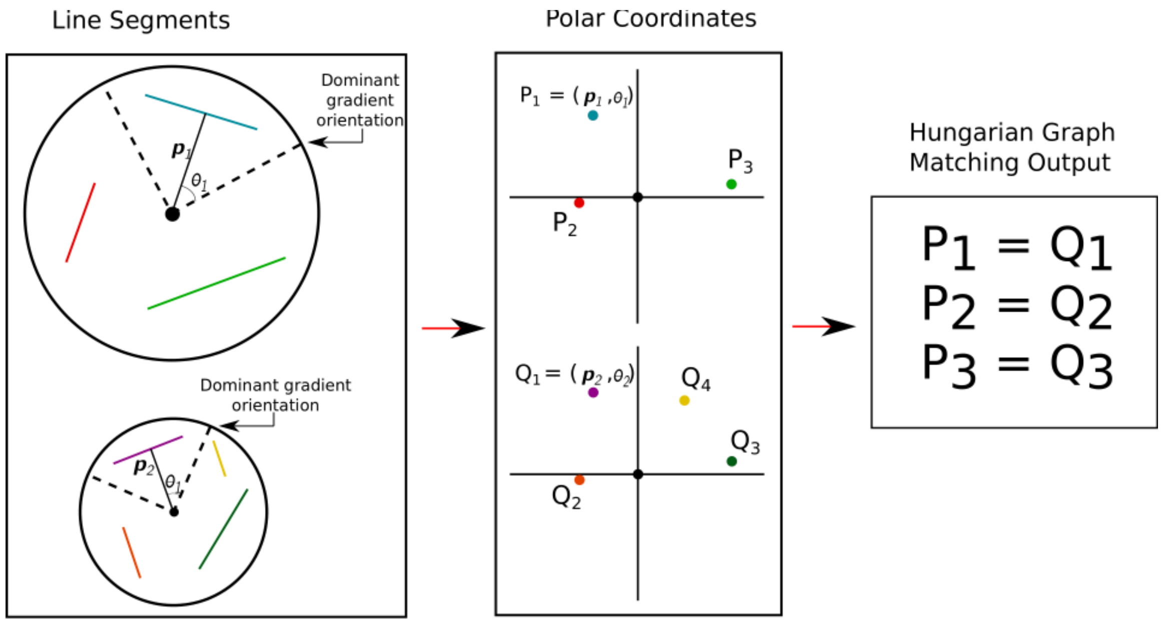


Figure 3.5: Matching lines in the neighborhood of a feature point. *Left:* Line segments are identified within neighborhoods centering a potential match. The keypoint is shown in black in both neighborhoods and the relative scales are represented by black circles. *Middle:* Line segments are converted to polar coordinates based on their distance from the keypoint and their orientation in relation to neighborhood's dominant gradient direction. *Right:* Hungarian graph matching is applied to the set of polar coordinates to figure out the optimal set of matching lines.

3.2.3 Contextual HOG

Our second approach for contextual matching takes advantage of every pixel in MSER-based neighborhoods surrounding potential keypoint matches. The potentially matching neighborhood in the second image is calculated at multiple scales and its orientation is determined using the region's dominant gradient direction, giving us a region that contains the same content and is based in the same coordinate system.

To measure how well the two neighborhoods match, we begin by computing histograms of gradients for each image patch. Just as was discussed in Section 3.2.2, a three-level Gaussian pyramid is built for the corresponding neighborhoods. A vector is constructed for each HOG cell by concatenating its nine bin values and normalizing using the vector's magnitude. We calculate the L2 distance between the vectors for corresponding cells in the matching neighborhoods across all levels of the pyramid. If the average of all these distances is under a defined threshold ($threshold = 0.5$ for our experiments) we say that the neighborhoods surrounding the match have a strong correspondence and save the keypoint match. By requiring a region to have matching HOG's along multiple scales, we are placing a stricter requirement on the matching criteria thereby removing noise that may be present when only one scale is used. Figure 3.6 shows matching HOG cells for two corresponding neighborhoods.

This method makes no assumption about the structure of a region. It can be used to find features in both natural, highly varied regions such as areas of images containing trees as well as in areas containing uniform architecture.

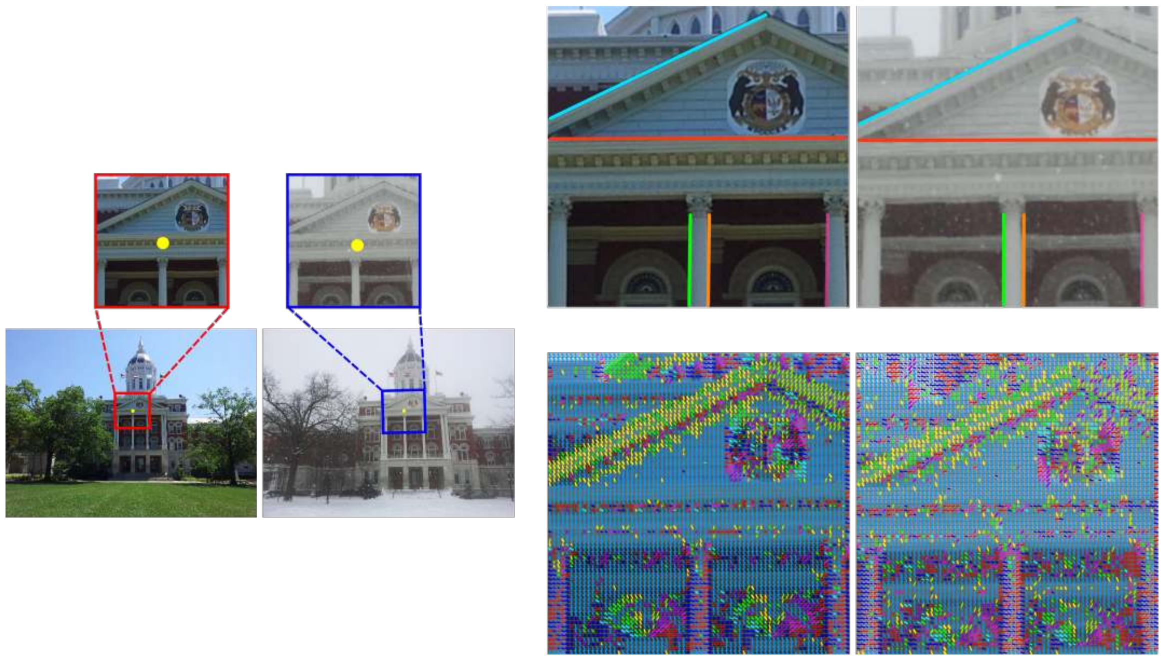


Figure 3.6: Identifying matching neighborhoods for contextual matching. This example uses a SIFT match as an anchor for exploring contextual neighborhoods. *Left:* Two images are matched and share a corresponding point, shown in yellow. This keypoint pair has a distinctiveness ratio between 0.7 and 0.9 and is not verified by SIFT. *Top Right:* Matching line segments within neighborhood. *Bottom Right:* HOG descriptors for cells in the neighborhood. Cells are colored according to dominant gradient direction. The line orientation in each cell shows the cell's dominant gradient direction.

3.2.4 Ensemble SIFT

The first way we incorporate contextual information in our pipeline begins with obtaining an initial set of matches for an image pair using SIFT, a very robust and scale invariant blob detector that describes the rich and varied gradient information surrounding an interest point. In the traditional SIFT pipeline, A and B are identified as a true match if the distance between their descriptors passes a distinctiveness ratio test (i.e. $\frac{||A_{des}-B_{des}||}{||A_{des}-C_{des}||} < 0.7$). When matching photographs with highly varying image properties, we may not obtain enough keypoint matches, or enough correct matches, that are very distinct to unearth a transformation mapping the images to each other. In order to identify correct point matches that may not pass the ratio test (and increase our pool of matches), we also take into account contextual information. This entails measuring the similarities of the larger neighborhoods surrounding a potential match. Sample matching neighborhoods are shown in Figure 3.6. We can use our two different types of contextual descriptors to describe these larger neighborhoods, taking advantage of hard linear features and soft histograms of gradients (HOG). Our SIFT + line segment method is robust to lighting and content changes and helps by identifying salient linear features. Our SIFT + HOG method is useful for matching regions that lack dominant lines but are rich in texture. These methods are both used to clear up some of the ambiguity between possible keypoint matches. If the neighborhoods of the matches with relatively higher distinctiveness ratios ($0.7 < ratio < 0.9$) match well using one of our measures, we include them in our list of putative correspondences that are used for future registration. Figure 3.7 outlines how we use the ratio test for collecting matches.

An example of how using our contextual features in addition to SIFT can help improve alignment is shown in Figure 3.8.

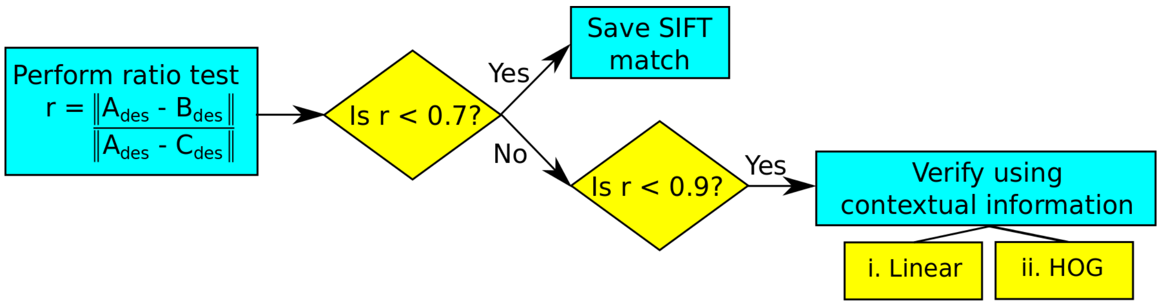

Perform ratio test
$r = \frac{\|A_{des} - B_{des}\|}{\|A_{des} - C_{des}\|}$
Is r < 0.7?
Yes
Save SIFT match
No
Is r < 0.9?
Yes
Verify using contextual information
i. Linear
ii. HOG

Figure 3.7: Outline of how we use the distinctiveness ratio test to collect matches and to determine when to use contextual matching in conjunction with the SIFT matching scheme.

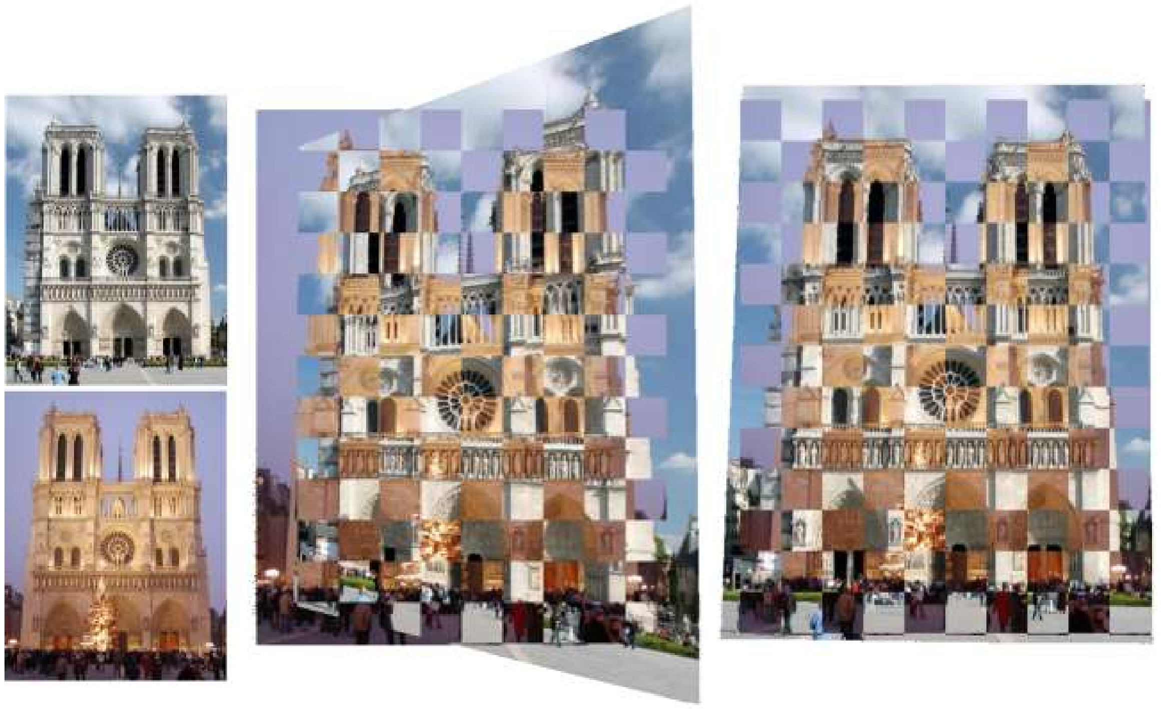


Figure 3.8: Image alignment improvement using our ensemble SIFT method. *Left:* Image pair to be aligned. *Center:* Image alignment using only SIFT features. *Right:* Image alignment using our ensemble features.

3.2.5 Contextual Harris-Laplacian

To eliminate our dependency on SIFT so that we can handle cases where an insufficient number of potential SIFT matches exists, we can also apply our contextual matching scheme to corner matching. In this scenario, we extract Harris-Laplacian corners from a pair of images and the Line Segment and HOG neighborhoods surrounding each detected keypoint. For every Harris-Laplacian corner in I_1, we compare its Line Segment neighborhood and its HOG neighborhood to the respective contextual descriptors of every corner in I_2. Neighborhoods are extracted at multiple scales as discussed in Section 3.2.2. Pairs of corners with a sufficiently low Euclidean distance between either their Line Segment or HOG-based descriptors are saved as matches.

3.2.6 Identifying Pseudo Corners

We may also encounter cases where the scale, lighting, content, or rendering style changes between an image pair are so great that it is difficult to extract repeatable SIFT keypoints or Harris-Laplacian corners in both images, leaving us with insufficient keypoint information to work with for matching and registration. For these scenarios, we propose using line segments extracted from the images to estimate corner locations, which we refer to as Pseudo Corner keypoints. Line segments are extracted at multiple scales and merged together. Two segments are connected into one larger line segment if they are near collinear and are in close proximity. Similarly, two line segments will be merged into a single line segment if they are near parallel and are in close proximity. We can also incorporate a saliency requirement on the line segments used by thresholding out lines with average gradient magnitude values below a certain value as done in [59]. Our new corner keypoints are the intersection points between nearby segments. We avoid using the inaccurate intersection points of near parallel segments by restricting that any Pseudo Corner must be within a certain distance of each of the line segments used to find it. This process is especially tuned to planar, architectural scenes where we can have a high degree of confidence that the intersection point of segments laying on the same plane will correspond to a real corner. We match our Pseudo Corners using the same approach described in Section 3.2.5 by comparing the contextual neighborhoods of keypoints in the image pair. An example of our results using this approach to align a difficult image pair entitled Vatican is shown in Figure 3.9 Bottom. We can see from Figure 3.17 that this image pair is not registered well using the SIFT-based ensemble method or the contextual Harris-Laplacian method that are discussed earlier in this paper. Figure 3.9 Top shows the matches identified using our ensemble SIFT method. The middle rows show the line segments extracted from the images and the Pseudo Corners identified. The bottom image shows the image pair aligned using matched Pseudo Corners to estimate the pair's homography.

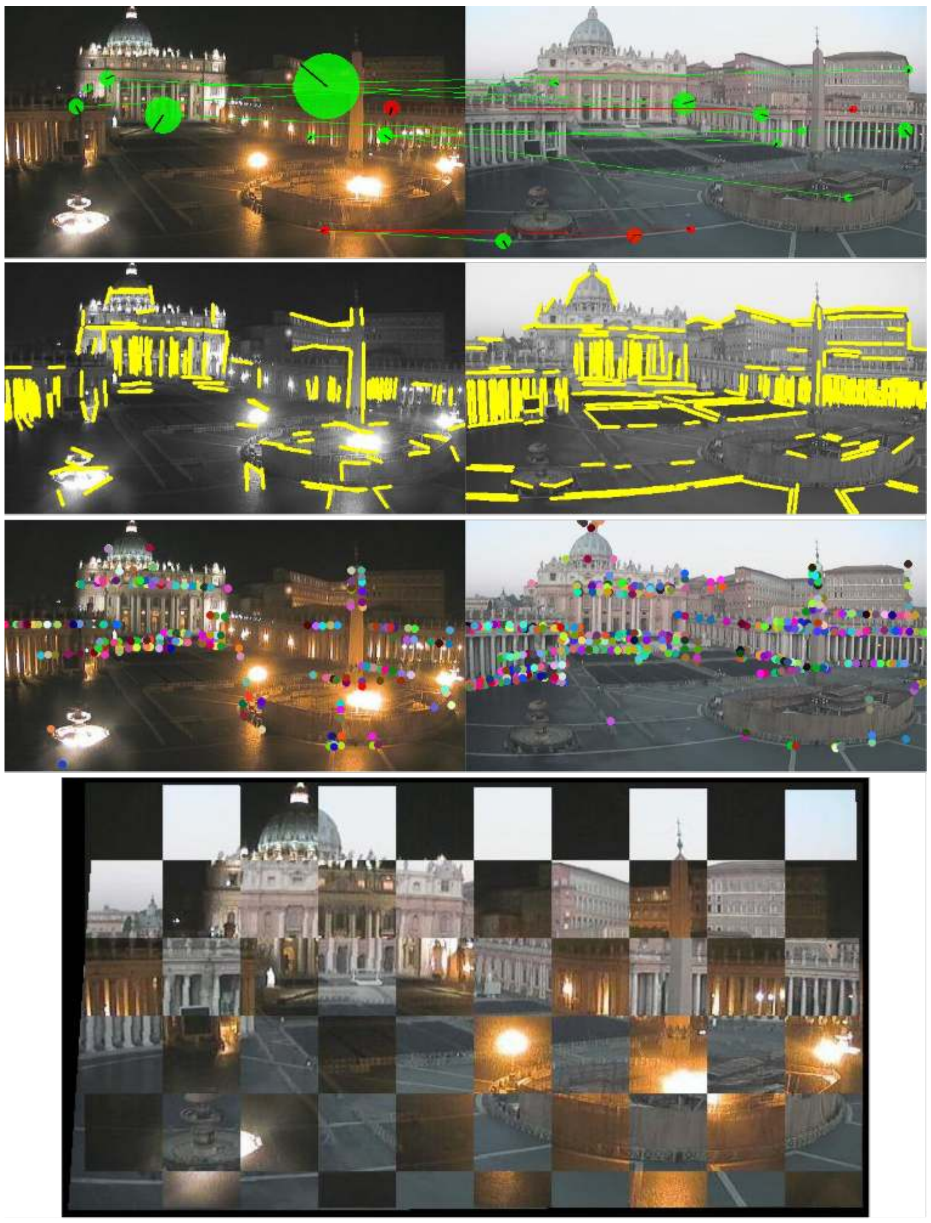


Figure 3.9: Using Pseudo Corners to align a difficult image pair, called Vatican. *Top:* The majority of the SIFT (green) and linearly-verified (red) keypoint matches for this pair are incorrect and cannot be used to align the images. No SIFT+HOG matches were identified in this example. The relative SIFT scales are represented by the size of the circles and the features' orientations are denoted by the black lines. *2nd Row:* Line segments extracted from images (shown in yellow) that are used to find Pseudo Corners. *3rd Row:* Pseudo Corners identified after intersecting line segments. *Bottom:* Image pair is aligned using the homography calculated from the matched Pseudo Corners.

3.3 Homography Estimation, Refinement, and Verification

All 2D matches identified using any form of our ensemble approach are used together to calculate a homography relating the image pair. The initial homography matrix is computed using RANSAC and Direct Linear Transform (DLT) [105]. See Appendices A.1, A.3, and A.5 for mathematical details on how we compute the homography matrix. All 2D matches found so far are geometrically verified by calculating the homography relating the image pair and identifying inlying matches. Multiple homographies are calculated for each image pair to take into account various 3D planes that may be represented in the images [106]. Each homography matrix H_i is computed using RANSAC and the chosen solution is the transformation matrix supported by the largest consensus set among the 2D matches. Matches that support a particular homography are removed and are not considered in future homography calculations.

Our iterative approach tries to find a balance between single, global homography calculations that are limited to planar-approximate scenes and multiple-homography methods that explicitly require a planar segmentation of photographs [107, 108] or more complicated image deformation models [109, 110, 111, 112, 113]. Generally, our iterative approach chooses matches belonging to different planes on different iterations as shown in Figure 3.10, but we do not complicate our method by enforcing such constraints. We conduct iterative homography fitting for the remaining features (i.e. features that are considered outliers based on the homographies estimated in the previous loops).

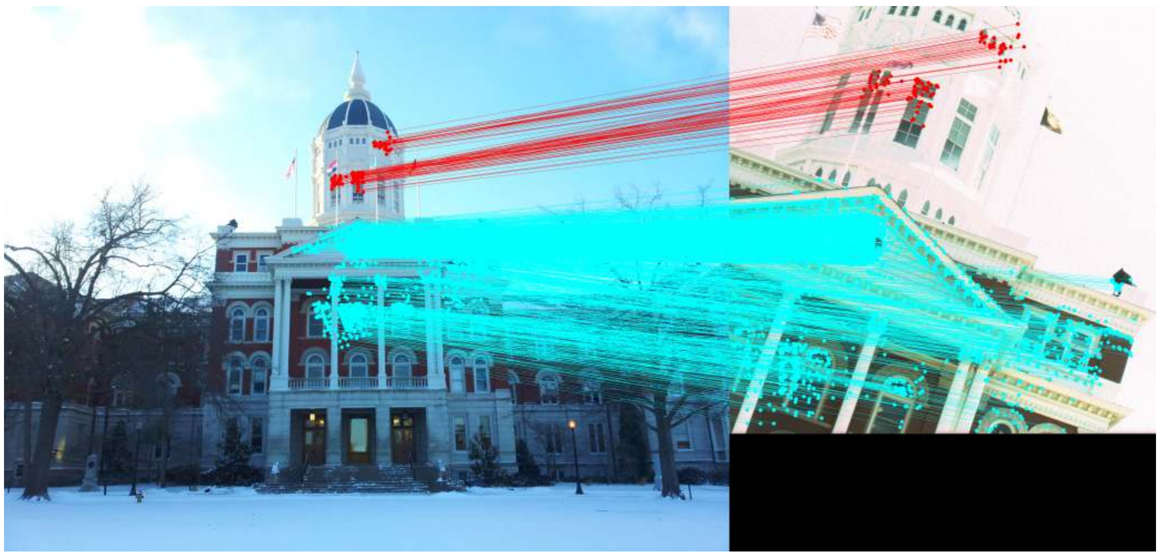


Figure 3.10: Matches identified using multiple homographies. Each color corresponds to a different homography. We can see how each homography tends to find matching points in different regions and planes of the image.

We take a two step approach to refine our image alignment that deviates from the popular approach of reestimating each H_i based on the reprojection error of the match set. We begin by searching for a dense set of matches on a relatively local scale. This first stage gives sets of keypoints the flexibility to search for correct matches along a variety of directions and distances. This is followed up by computing an optimal global alignment that is more rigid, but represents a general consensus for H_i throughout the image planes.

For refinement with local information, we apply the iterative closest point (ICP) algorithm [114] on the edge and corner pixels of I_1 and find their best matches in I_2. Edge points are identified by extracting line segments and uniformly sampling them. Ideally, by finding edge points using the Line Segment Detector algorithm, we are working with “strong” edges. Edges and corners in I_1 are transformed to the matching image plane using H_i and are compared to all of I_2's keypoints within a specified search radius. Pairs of keypoints with SIFT descriptor distances passing the traditional distinctiveness ratio test are saved. H_i is re-estimated using RANSAC and DLT. The search radius for matching keypoints plays an important role in our optimization algorithm. If the current homography estimation is far away from the optimum, we would like to have a large initial radius. On the other hand, if the current estimation is close to the optimum, a smaller radius is all that is necessary. However, this information is not known a priori, so we start a relatively large radius (10% of $max(image_{width}, image_{height})$) and divide it in half for each iteration of our refinement.

After reestimating H_i, we check the new alignment by comparing HOG's across the entire overlapping region of the two images within the bounds set by the matches used to compute H_i. If the relative HOG distance after performing ICP is higher than that of the alignment before running ICP, the new alignment is rejected. This is essentially a quality control step to ensure that the refinement stage does not lead to an alignment that diverges from the best solution we can find.

We next use gradient information from the image pair to perform one last global alignment. The gradient magnitude field for I_1 and the binary edge image for I_2 are computed. We wish to find the homography matrix that aligns the edge points of each image with the strongest gradient responses in the matching image as much as possible. The varying strengths of I_1's gradient field are used to guide I_2's edges to their ideal alignment and vice versa. To accomplish this, H_i is perturbed using Levenberg-Marquardt Optimization to maximize Equation 3.2. Only edge points that are within the bounded region of the image that H_i represents are considered in

Equation 3.2.

$$E = \arg\max_{H_i} \left(\sum_{j=0}^{\#\ I_2 \text{ edges}} \| \bigtriangledown I_1(H_i' pt_j) \| \quad + \sum_{k=0}^{\#\ I_1 \text{ edges}} \| \bigtriangledown I_2(H_i pt_k) \| \right) \tag{3.2}$$

Figure 3.11 displays gradient information that can guide our alignment and the result of using this approach. This second stage of refinement is applied at the end of our pipeline after we have already established a relatively strong estimate of the image pair alignment. Line segments in one image need to be transformed closely to a linearly shaped area of relatively strong gradient responses in the matching image for these pixels to contribute to the refining of the image alignment. Equation 3.2 tries to find a general agreement amongst the transformation of all the line segments in the overlapping regions of the images. The combination of using the Line Segment Detector method, which is robust to noise, and searching for a general consensus amongst all the line segments in the images helps our optimization avoid being overly influenced by noise and clutter in the images. Figures 3.2, 3.9, and 3.11 show examples of lines extracted using the Line Segment Detector method. We can see in these cases that the lines picked up mainly correspond to structures that exist in both images despite very different qualities in the photograph pairs. We pick up very few lines along the ground, trees, people, or other clutter or noise in the images.

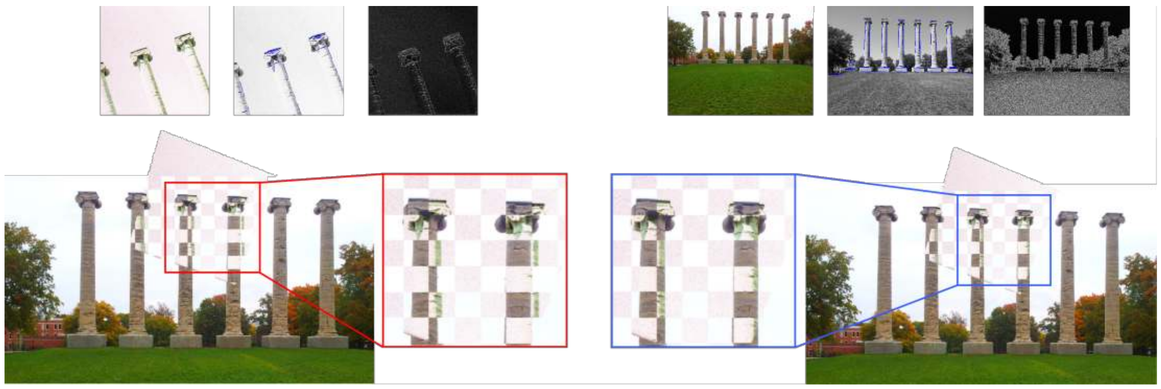


Figure 3.11: Using gradient information to refine homography. *Top:* Two original images and their edge images and gradient field images. We aim to align the edge and gradient field images as well as possible. *Bottom Left:* Stitching result before our refinement method is applied. *Bottom right:* Result after our refinement is applied. *Bottom middle:* Zoomed in views of the area where the alignment is corrected through refinement.

The benefit of using these types of refinement techniques instead of trying to simply minimize a match set's reprojection error is apparent when the initial point correspondences are very sparse or confined to one section of the image pair. In this situation, we are very limited in how much we can actually refine our transformation. On the other hand, our refinement approach that considers edges, corners, lines, and gradient information takes into account visual properties spanning the majority of the image planes regardless of the distribution of our point correspondences.

We have several different criteria for verifying each H_i. We first make sure that the points used to calculate the homography matrix are not collinear. Once the matrix is computed, we also take into account the layout of the transformed images and the numerical properties of the homography matrix. To check against our layout criteria, we project the four corners of I_2 to I_1's image plane using H_i. The top left and bottom left corners must remain to the left of the top and bottom right corners, and the top left and right corners must remain above the bottom left and right corners after the transformation. If this condition is not met, H_i is discarded. We also calculate the determinant of our normalized H_i. Experiments have shown that this value should not be close to zero [106].

3.4 Results and Discussion

We ran our 2D-2D matching pipeline on many of the image pairs provided in the architectural symmetry dataset [7] which contains images with a variety of visual discrepancies. This dataset includes paintings, drawings, and historical photographs matched to modern day images and several image pairs taken at very different times. Some image pairs are obtained under different lighting conditions, including changes in season and time of day, and have scale and perspective changes. The dataset includes ground truth homographies for aligning image pairs. We ran several tests on this dataset to determine the effectiveness of the first version of our pipeline which builds upon a local feature matching technique such as SIFT. The first of these tests is shown at the top of Figure 4.10. The goal of these tests is to show how we can increase the accuracy and robustness of a widely used feature matching technique with our contextual framework. This test consisted of stitching the images using the homography calculated from four different matching techniques. We show results for using traditional SIFT matches with a 0.7 distinctiveness ratio, SIFT matches with a 0.9 distinctiveness ratio, our ensemble features, and our ensemble features with our refinement pipeline. The contextual features identified in conjunction with our ensemble features are based on SIFT keypoints that have distinctiveness ratios between 0.7 and 0.9. We use the first homography calculated with our pipeline since only one ground truth transformation is provided for each image pair. We have included both standard SIFT results to show that just increasing our search range to potential matches with ratios up to 0.9 does not account for the increased match pool and accuracy produced by our pipeline. The overall accuracy of using all SIFT matches with a distinctiveness ratio below 0.9 is very low.

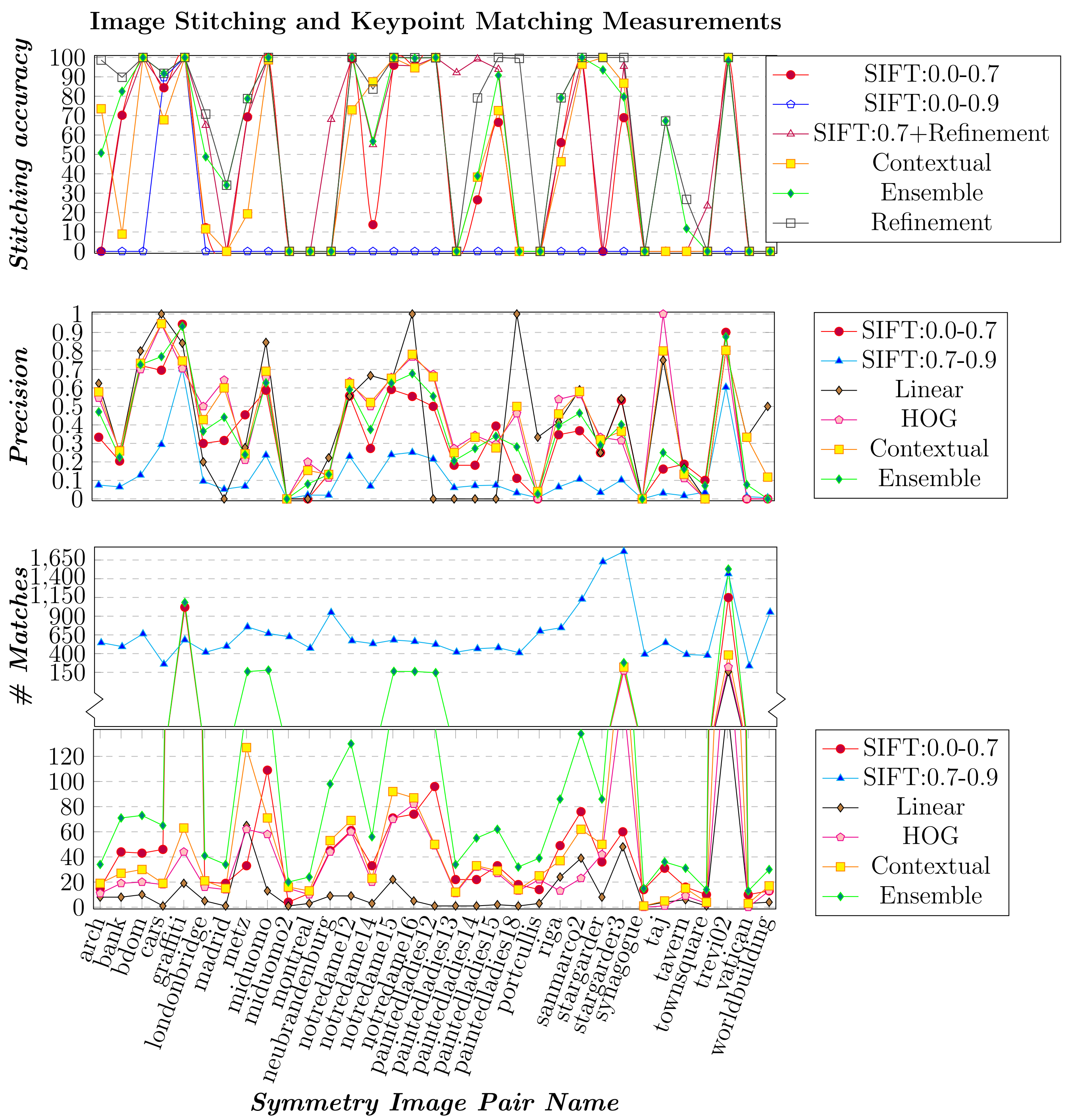


Figure 3.12: Quantitative results for stitching image pairs. *Top:* Percentage of pixels in image plane that are correctly aligned after using solved homography. *Center:* Percentage of matches that are identified using different methods that are correct. *Bottom:* Number of matches identified using several different methods.

For each of these four matching technique tests, we translate every pixel in I_1 to I_2 using both the ground truth homography and the first homography calculated using the method being tested. We compute the percentage of pixels that are transformed to the same coordinate. To allow for rounding errors and small inaccuracies in the ground truth data, we set a threshold on what the 2D distance between projected pixels can be to label a pixel as having been transformed correctly. This threshold is 0.6% of $max(image_{width}, image_{height})$ which is the same threshold used on 2D symmetrical transfer errors for identifying inliers in [29].

Our proposed ensemble feature compares favorably to SIFT across this dataset. From our tests, we also have determined that blindly raising the distinctiveness ratio allows too many inaccurate matches for RANSAC to find a correct homography between the image pairs and yields very unstable results. However, our approach for looking through these more ambiguous matches and considering contextual information appears to be a very valuable source of information. We increase the size of our match pool, making it easier to find a correct alignment, without letting in so much noise that RANSAC is overwhelmed by the outliers.

We also performed a test to observe what percentage of the keypoint matches identified in the different techniques are indeed correct. These results are shown in Figure 4.10 Center. From this chart we can see that, in general, adding in contextually verified matches increases the percentage of accurate correspondences in the pool of matches. In theory this should increase the odds of a correct aligning transformation being calculated using RANSAC. In conjunction with this, we also charted the number of each type of match that was used in Figure 4.10 Bottom.

In Figure 3.13, we show the precision-recall curves for four different image pairs in the symmetry dataset. These curves were obtained by increasing the distinctiveness ratio (r) for accepting matches. r increases from left to right. We show the curves for using just SIFT features, using our ensemble features, and using only our contextually-verified features. The ensemble feature contains SIFT features with ratios under r and contextually-verified features found within a distinctiveness ratio range of $[r, r + 0.2)$. The contextual curve shows only SIFT features that passed either the linear or HOG requirements for all matches with ratios under r. The contextual and ensemble features perform quite well under this metric. Combining the precision-recall measurements with those shown in Figure 4.10 demonstrates that the contextual information we propose using can complement local keypoint matching techniques very well.

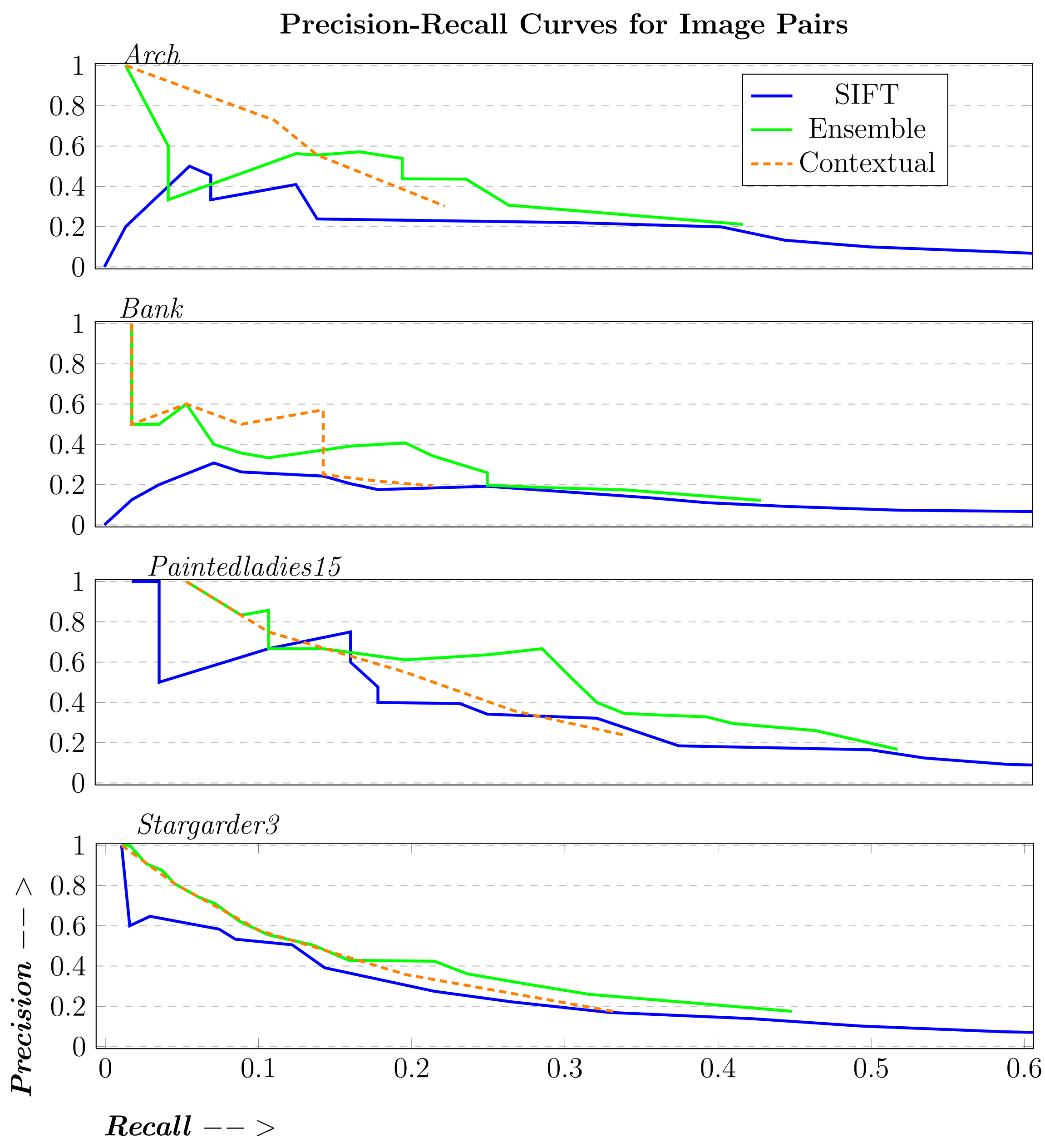


Figure 3.13: Precision-recall curves for four different image pairs from [7]. Curves are provided for using SIFT matches, our ensemble feature, and just the contextually verified SIFT features identified using our method. Data points correspond to increasing distinctiveness ratios moving from left to right.

In Figure 3.17, we also provide the image alignment accuracy results for each of the three versions of our pipeline. These consist of building on SIFT and creating an ensemble feature of SIFT and contextual descriptors, matching Harris-Laplacian corners with contextual descriptors, and matching our proposed Pseudo Corners with contextual descriptors. This chart shows the alignment accuracy using the initial set of matches from each of these methods and the accuracy after refinement is applied. We can see here that each of the images tested can be aligned very well using one of our techniques. Our contextual descriptor framework can be used successfully in combination with a variety of keypoint detectors.

Further tests of our contextual corner methods are shown in Appendix B in which we used both our Harris-Laplacian-based method and Pseudo Corner method to align images in the png-ZuBuD dataset [1], the Stanford Mobile Visual Search dataset, and the dataset provided in [5]. Visual results for several of these image pairs are shown in Figure 3.1 and in the Appendix B, Figures B.1 and B.2. For these tests, we compute a homography using the matches calculated from both tested techniques and report the number of matches found and the average symmetrical transfer error [105] for all of the matches using the estimated homography. These tests show that we can obtain sufficient numbers of matches between image pairs with a wide variety of visual properties to find an aligning transformation. They also show, via their low average symmetrical transfer error, that the majority of the matches found agree with each other about the value of the solved homography matrix. This indicates that the matches found are structurally consistent.

To test our proposed feature detector, Pseudo Corners, we computed its repeatability scores on the Oxford Affine Covariant Region Detectors Dataset [2] which provides image sets demonstrating changes in perspective, lighting, scale, and JPEG compression. We compared the repeatability of our Pseudo Corners to that of SIFT and Harris-Laplacian. These values are shown in Figure 4.11. Our Pseudo Corners method tends to out perform SIFT and Harris-Laplacian on all of the image sets except “bark” and “wall”. Given the nature of Pseudo Corners, which looks for the intersections of line segments, it is reasonable that it does not perform as well on these natural images.

We have also included several visual examples of image registration using our pipeline (Figure 3.1) for photographs from our own dataset, the symmetry dataset (Figure 3.15), and several other public datasets (Figure 3.14 and Appendix B, Sections B.1 and B.6). These image pairs present challenges including changes in season and content, lighting, scale, camera orientation, blur, and rendering styles. The runtimes to register many of these image pairs using our pipeline are also provided in Appendix B, Section B.5. Both quantitative and qualitative comparisons of our method to other feature extraction and matching techniques including MSER, SIFT Flow [8], and local symmetry features [7] are presented in Appendix B, Section B.2 as well.

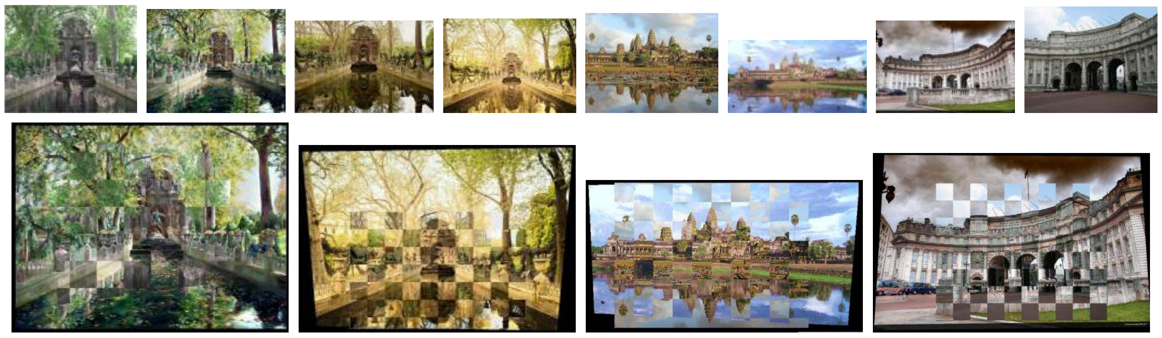


Figure 3.14: Registration results after applying our pipeline to images from dataset in [5].

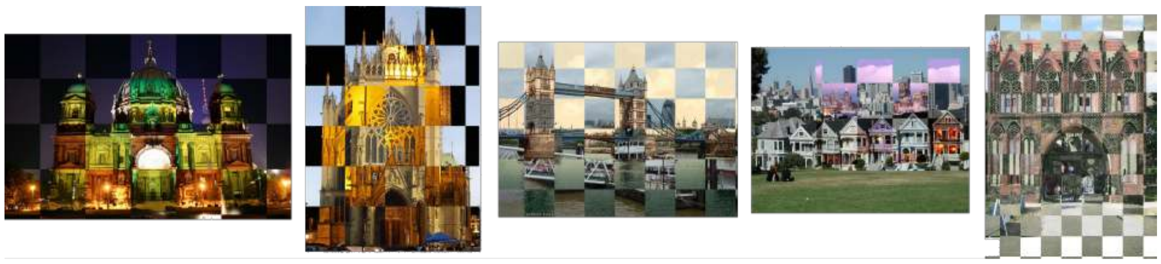


Figure 3.15: Registration results after applying our pipeline to several symmetry dataset image pairs. Images provided courtesy of [7]. From left to right, pair names are bdom, metz, londonbridge, paintedladies15, and stargarder.

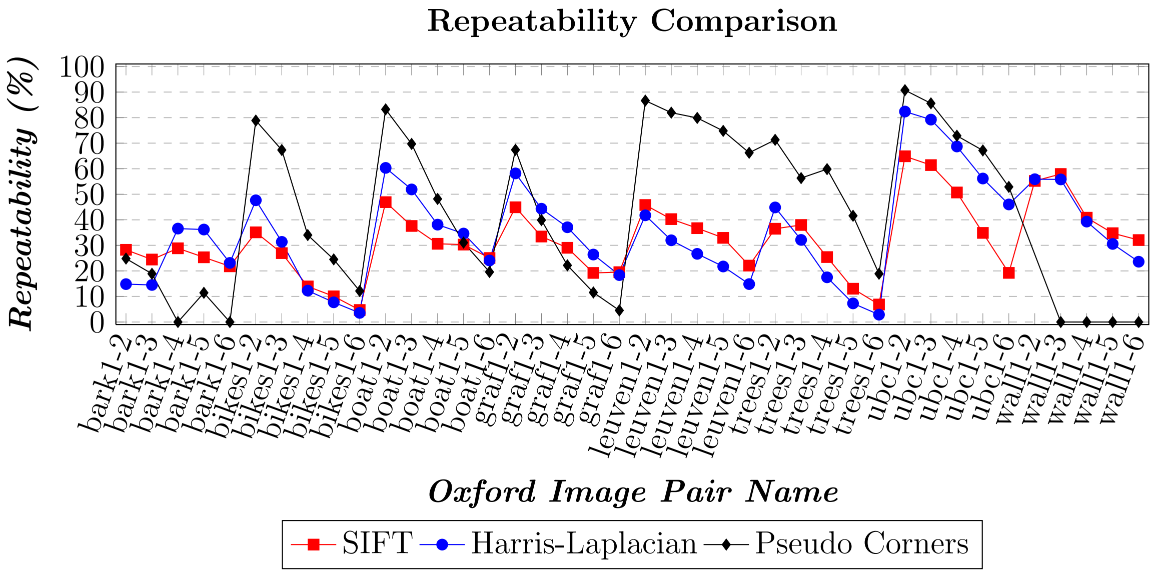


Figure 3.16: Repeatability measurements for SIFT, Harris-Laplacian, and Pseudo Corners on Oxford Affine Covariant Region Detectors Dataset [2]. The sets of images test the following changes in image properties: bark - scale and image rotation changes on textured scene, bikes - increased blur on a structured scene, boat - scale changes, graf - viewpoint changes, leuven - illumination changes, trees - increased blur on a structured scene, ubc - changes in JPEG compression, wall - viewpoint angle changes on textured scene.

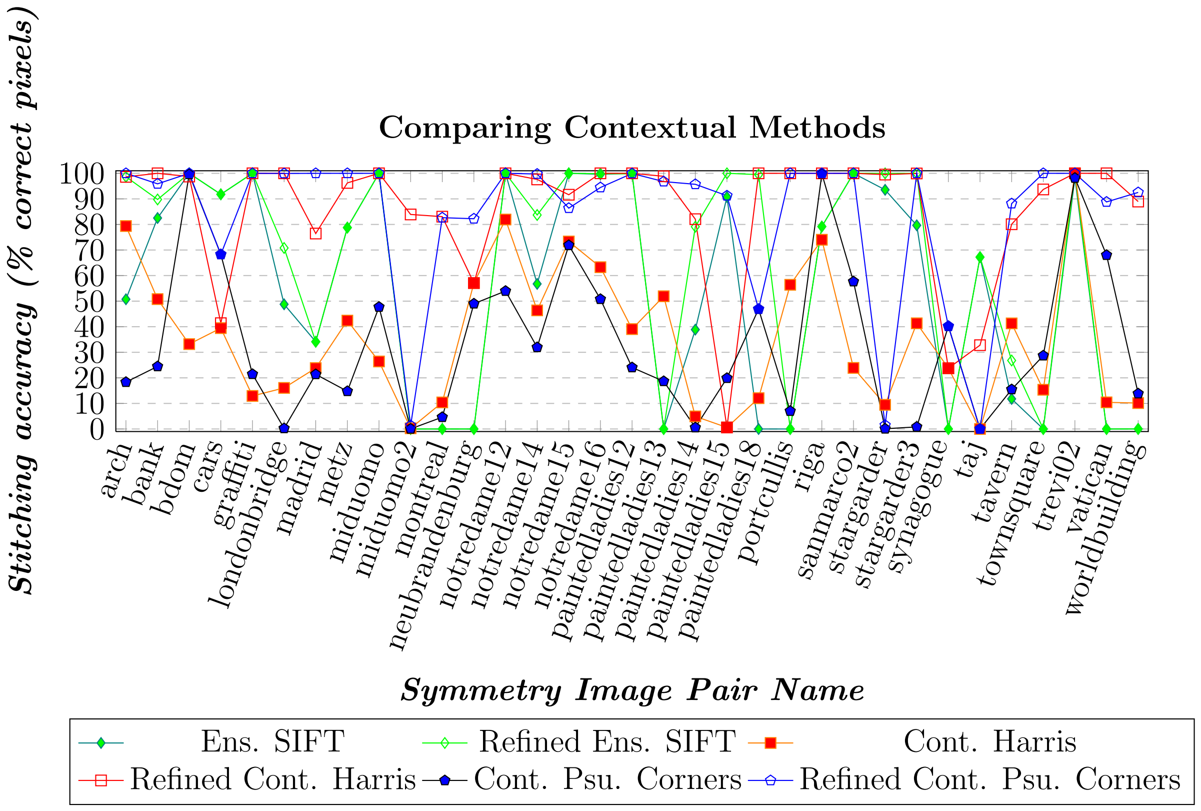


Figure 3.17: Comparing our contextual methods. Using the symmetry dataset [7], we show the alignment accuracy using our ensemble SIFT features before and after refinement, contextual Harris-Laplacian before and after refinement, and contextual pseudo corners before and after refinement. As mentioned in Section 3.1, ensemble features refer to the version of our method that uses an existing keypoint descriptor, such as SIFT, in addition to our contextual linear and contextual HOG information for matching. Contextual features use only our contextual descriptor for matching. The accuracy is measured in terms of the percentage of pixels in I_1 that are transformed to the correct location in I_2 using the homography estimated from the match set.

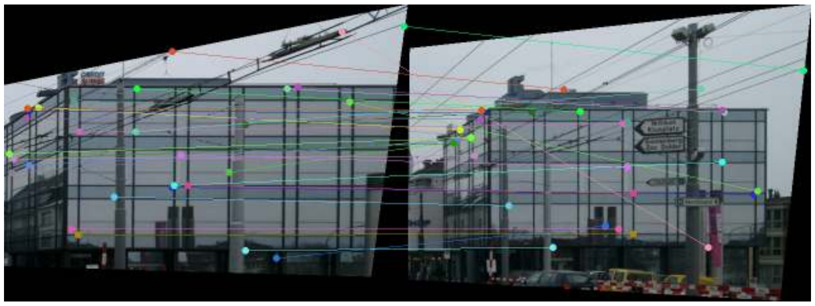
Chapter 4

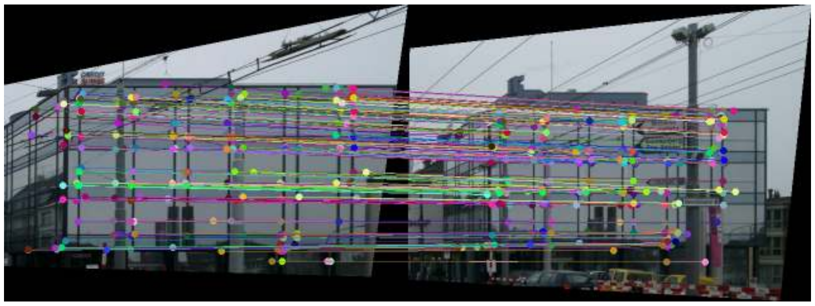
Matching Repetitive Data

Registering architectural imagery poses several interesting challenges. Many architectural designs include repetitive elements such as rows of windows, doors, or balconies that all have similar appearances. If either local or regional descriptors are extracted for different elements of a repeating patten, they may take on near-equivalent values. This creates ambiguity when the descriptors from a series of images are compared. A matching algorithm being used to find the correspondence for a window in one image may not be able to distinguish between the descriptors of several repetitive windows in a second image. Figure 4.1 shows an example of how a common local keypoint descriptor and matching technique, SIFT [32], cannot distinguish between different areas of a building that look nearly identical.

We hypothesize that by using specific urban imagery-based criteria to reduce the search space during matching, we can bypass the difficulties encountered when matching ambiguous data. We have observed that regardless of the height of a building, architectural repetition often occurs at least in the horizontal direction, if not also in

a.


b.


c.

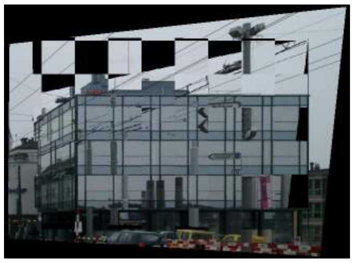

d.

Figure 4.1: Challenge of matching images containing repetitive features. (a.) Original image pair. (b.) SIFT keypoint matches on rectified images. Close inspection shows that the majority of the matches are incorrect and keypoints in the left image are matched to similar looking repetitive areas in the right image. We do not have enough correct SIFT matches for RANSAC to find a correct homography. (c.) Keypoint matches after applying our proposed pipeline for handling repetition. (d.) Image pair aligned and overlaid using our pipeline.

the vertical direction [36]. Since we are focused on images of man-made structures that frequently have easy to identify vanishing lines, we rectify all images to be front facing [105, 115] so that horizontal repetitive elements have the same y-coordinate. Once images are in this front-facing format, we propose aligning images one dimension at a time. We first collapse each image along the vertical axis and use the y-coordinates of matching features to vertically align the image pairs. Once we know the relative scale and row alignment across images, we expand the data back out along the x-axis and compute the horizontal alignment.

To perform dimension reduction to compute a vertical alignment, we create an abstraction of each image by identifying the repetitive elements along the horizontal axis and choosing just one element to represent each group of repeated features. Figure 4.3 provides a high-level example of how we perform dimension reduction to abstract a building into its distinctive elements. By choosing one representative patch for each row of repeated horizontal elements as well as identifying salient patches that do not repeat across the image, we can abstract an image into a vertical column of distinct regions and remove potential repetition in the feature space. We match these representative regions across images to compute a pair's vertical alignment.

This simplifies the second alignment stage since our only remaining unknown is the displacement along the horizontal axis. At this stage, we have a reduced search space to find distinctive matches that can be used to align repetitive facades. For a match to be distinctive, the matching regions must be unique within their own images. Using our assumption that the data can be rectified to be front-facing and vertically aligned, we propose a simple, but very-effective technique for computing the intra-image saliency along the rows of urban images. We use this information to

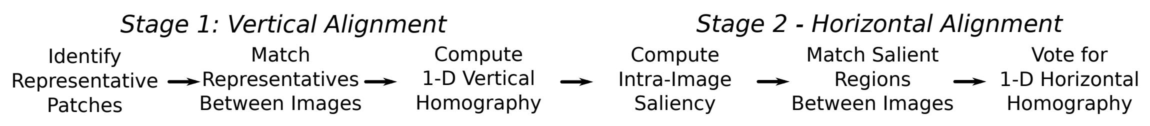

Stage 1: Vertical Alignment
Identify Representative Patches
Match Representatives Between Images
Compute 1-D Vertical Homography
Stage 2 - Horizontal Alignment
Compute Intra-Image Saliency
Match Salient Regions Between Images
Vote for 1-D Horizontal Homography

Figure 4.2: Pipeline of our proposed registration method. Our method has two main stages: vertical alignment and horizontal alignment. For vertical alignment, we perform dimension reduction, identify representative patches for each image, and use the matching representatives to compute a 1-D homography that transforms the images to the same scale and aligns the pair along the vertical axis. For horizontal alignment, we compute the intra-image saliency and search for strong, salient matches along corresponding rows between images to vote for the correct displacement between the pair.

gauge the quality of keypoint matches. We are able perform this computation directly on the images being matched without relying on the existence of a large database of similar images to determine the uniqueness of keypoints being matched.

4.1 Overview

To summarize, the main steps of our registration methodology are identifying both repetitive and salient elements within images, using this information to first align image pairs vertically, and finally computing a horizontal alignment using a dramatically reduced and unambiguous search space. We use image regions that represent both repeated areas as well as salient patches that do not resemble any other area of the image to guide our matching process. To aid in horizontally registering an image pair, we compute both the intra-image saliency and the saliency of matches between images. We perform this analysis directly on the images of interest without running machine learning techniques on a large dataset to identify unique regions. Our pipeline is outlined in Figure 4.2.

Our pipeline is designed to overcome several of the limitations and assumptions made in the work described in Section . We use a technique for identifying repetitive patterns in images regardless of the spatial distribution of similar elements and we are able to match images of repetitive facades that only partially overlap. We also remove the ambiguity in the feature space that is common in architectural imagery by collapsing repetitive elements into single representative patches. Finally, instead of only using either repetitive or salient features for matching, we take advantage of all the information available in images, and incorporate both types of features in our registration pipeline.

4.2 Vertical Alignment

To register an image pair, we first align the photographs vertically using a 1-D homography. This transformation will encode the relative scale change between the image and align matching rows with each other as shown in Figure 4.12c. We remove some of the ambiguity of matching horizontally repetitive facades by initially only considering the y-coordinates of potential matches. We further remove confusion in the feature space by only choosing one representative patch from each repetitive region to perform the 1-D alignment.

4.2.1 Identifying Representative Patches and Matching Image Pairs

We begin by performing dimension reduction and selecting one representative patch of each type of region in an image so that all the chosen representative patches are

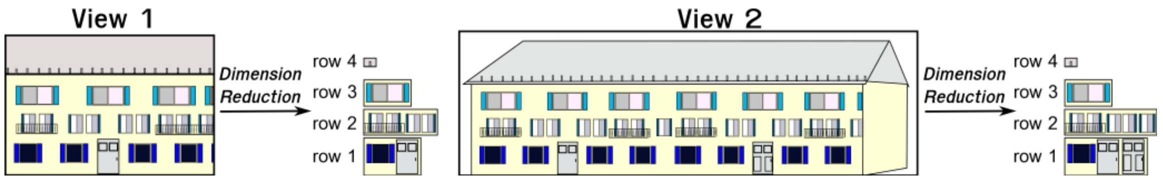
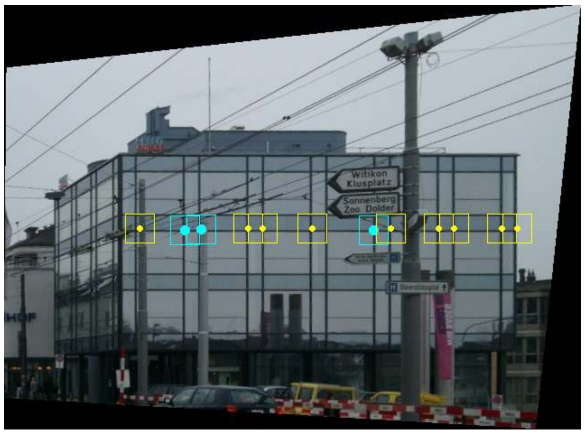


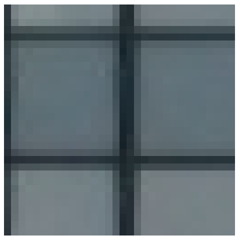
Figure 4.3: Overview of how we use dimension reduction to represent an image. Repetitive and salient regions are searched for along the horizontal axis and a single representative patch for each group is chosen. By first focusing on just aligning the representative patches extracted from different images in just one dimension, we remove the ambiguity caused by attemping to match repeated elements across images.

distinctive. This entails identifying portions of an image that are duplicates of each other and represent a repeating pattern as well as image portions that are truly unique. When determining if a patch is part of a repetitive group or is unique within the image, we use a combination of local and regional descriptors. This helps overcome some of the inherit ambiguity of very-locally defined descriptors [62] and increases our confidence as we identify regions as either salient or repetitive. Incorporating regional descriptors at varying scales also allows us to explore how different image regions may switch between repetitive and salient as the considered area changes.

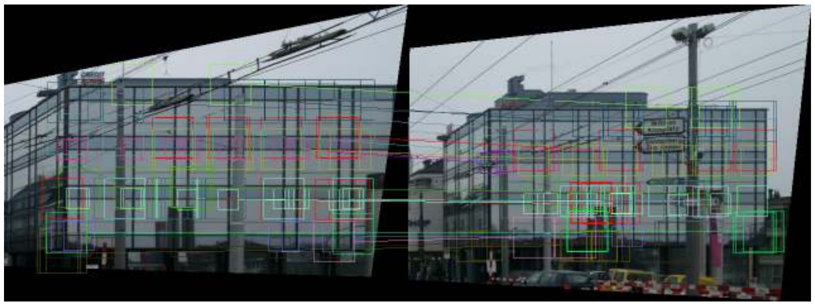
To initially identify repetitive elements, we extract local SIFT descriptors and regional HOG descriptors surrounding some sort of sparse, repeatable local keypoints in an image, such as SIFT or corner features. Throughout our discussion, we will refer to these local keypoints as anchor keypoints. The choice of type of anchor keypoint used here is very flexible and is discussed further in Section 4.4.2. The main point is that we have some sort of feature to use as an anchor for studying and comparing regions that reduces our search space from needing to compare every pixel to every other pixel in an image. We extract multi-scale patches centered at each anchor keypoint and use SIFT and HOG descriptors [12] to group repetitive elements that are centered on the same image row. For each group, we use the element that is most similar to group's mean patch as the representative patch. Figure 4.4 provides a visual example of how we identify repetitive elements in an image and choose a group's representative patch.



a.



b.



c.

Figure 4.4: Identifying groups of repetitive elements and matching groups. (a.) Similar patches are identified along a row of an image. Teal boxes surrounding a larger dot show the matching anchor keypoints. Yellow boxes surrounding a smaller dot show the dense features that were matched to the anchor keypoint for the row. For this example, we use Pseudo Corners as anchor keypoints as is discussed in Section 4.4.2. (b.) The chosen representative patch of the group. This patch is most similar to the average of all the elements contained in the group. Note that sometimes in architectural designs, the repetitive elements may not occur at regular intervals as shown in this example. Our method is designed to search for repetition regardless of the regularity or irregularity of the interval. (c.) Subset of the corresponding groups after computing a 1-D homography (Section 4.2.2) are highlighted in matching colors. Every element of the groups are displayed.

Our next task is to match patches between photographs to determine how to align the images. We begin by collapsing the groups and only matching the representative patches picked from each group. When we match just the representative patches of groups (that may have either one or multiple elements), we can focus on matching unique areas without worrying about the ambiguity that arises with repetition. This gives us enough information to begin determining the structure of the correspondences between images and their arrangement along the y-axis. Once we know this, we can expand the groups and align the repetitive elements along the x-axis using salient regions as a guide. Since the images have already been rectified and are front facing, the only parameters we need to solve to align them are the scale and displacements in the x and y directions.

To actually match representative patches, we extract the SIFT descriptor at the center point of each patch and the HOG descriptor of the entire representative region. For each representative patch in I_1, we compare its SIFT descriptor to all of the representative patches in I_2. If the patch in I_2 with the most similar SIFT descriptor to the patch in I_1 also has sufficiently low Euclidean distances between the center points' SIFT descriptors and the patches' HOG descriptors, we label this as a group match. We consider all of the representative patches in I_1 and I_2 that were constructed at different scales in one round of matching. We use the set of group matches to begin computing the aligning image transformation. Figures 4.4 and 4.12b show examples of matching groups between an image pair.

4.2.2 Computing a 1-D Vertical Homography

Using the set of matching representative patches, we can compute a 1-D vertical homography. The only two parameters that we need to construct this transformation (H_v) are the scale (s) and displacement along the y-axis (d_y) as shown in Equation 4.1. H_v only transforms the y-coordinate of a pixel in I_2 (y_2) to it's correct row or y-coordinate in I_1 (y_1) as demonstrated in Equation 4.2. When we preform a vertical alignment, the x-coordinates of the transformed pixels remain unchanged.

$$H_v = \begin{bmatrix} s & d_y \\ 0 & 1 \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} y_1 \\ 1 \end{bmatrix} = H_v \begin{bmatrix} y_2 \\ 1 \end{bmatrix} \tag{4.2}$$

Since H_v has two unknown parameters, we need two sets of matches along the y-axis to compute these values. The representative patches we choose in Section 4.2.1 have the same y-coordinate as each of their groups' members since they represent repetitive elements along the x-axis. Therefore, pairs of matching representative patches provide all the information we need to solve for a vertical alignment. We use RANSAC [33] to randomly choose sets of two matching representative patches between the image pair to compute a series of potential H_v's. For each H_v, we check how many other representative patch matches have a symmetric transfer error [105] lower than $0.5 * winSize_1$ and choose the H_v with the largest support. We then re-estimate H_v using this set of inliers.

4.3 Horizontal Alignment

Once we have the vertical alignment and the relative scale change between the image pair, all we need to calculate is the displacement along the x-direction d_x. Again, we can construct a 1-D homography to transform points between I_1 and I_2, this time focusing on shifting points horizontally. Equation 4.3 shows the form of our horizontal 1-D homography (H_h) and Equation 4.4 shows how we can use this transformation to map the x-coordinates of pixels in I_2 (x_2) to their corresponding column in I_1 (x_1). The scale s used here is the same scale computed in Section 4.2.2 in Equation 4.1.

$$H_h = \begin{bmatrix} s & d_x \\ 0 & 1 \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} x_1 \\ 1 \end{bmatrix} = H_h \begin{bmatrix} x_2 \\ 1 \end{bmatrix} \tag{4.4}$$

Since we are working with data containing repetition, especially images with patterns along the horizontal direction, we need to identify matching areas where we can be sure the correspondences are non-ambiguous to compute d_x. This is especially important because we do not require that our image pairs contain completely overlapping facades, so we can not rely on the relative arrangement of repetitive elements to determine the horizontal alignment. Figure 4.5 shows an example of the problem of relying on the relative arrangements of repetitive elements to align images. Our goal is to find matching pixels (p_1 and p_2) that satisfy the following criteria:

1. p_1 and p_2 have very similar HOG descriptors

2. p_1's closest match within I_1 (q_1) is very dissimilar to p_1

3. p_2's closest match within I_2 (q_2) is very dissimilar to p_2

These conditions help us ensure that we find matching pairs of pixels that are both very similar to each other, but are distinct within their respective images, making the match distinct. In this paper, we propose a novel pairwise match saliency field that can accurately quantify these two conditions. At this point we have a dramatically reduced search space for finding these salient correspondences than at the beginning of our pipeline because we now know the relative scale between images and what rows correspond to each other using H_v. We have two steps for identifying salient matches. The first is to compute an intra-image saliency map to identify distinct areas in a single image. The second step uses this information to compute the pairwise saliency of each match. Matches that are determined to be salient are used to compute d_x.

4.3.1 Intra-Image Saliency

The goal of our intra-image saliency maps is to display how salient each pixel and its surrounding region in an image are compared to other regions in the image. Since at this point in our pipeline, for a given pixel in I_1, we are only looking for its match on a single row in I_2, we are only concerned with how similar each region is to all other regions centered on the same row. Following this criteria, we are able to reduce our intra-saliency computation by comparing each region only to the other non-adjacent regions on the same image row. We consider regions whose HOG windows do not overlap to be non-adjacent. Equation 4.5 shows how we compute each pixel's saliency. We use s_i to denote the saliency of a pixel, p_i, and its surrounding patch. q_i is the pixel on the same row as p_i whose surrounding HOG patch has the lowest L^2 distance to the HOG patch surrounding p_i and is not adjacent to p_i.

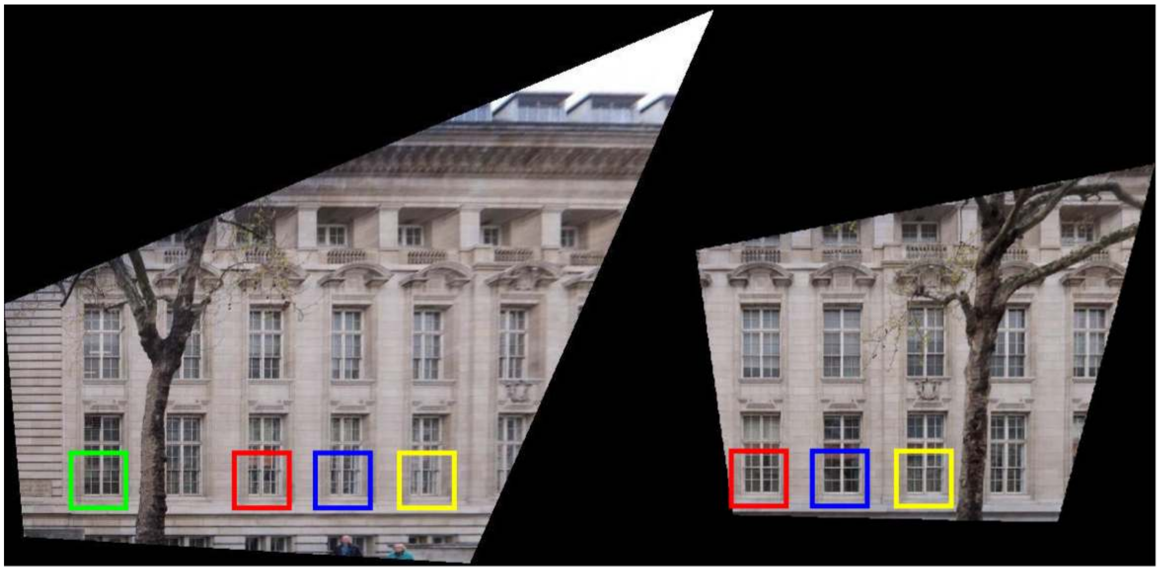


Figure 4.5: Motivation for searching for salient matches for horizontally aligning images. The highlighted image patches in the above image pair shows a set of matching repetitive elements. If we choose the horizontal alignment that maximizes the number of overlapping elements, we will have the wrong image pair transformation in situations like this where the images do not fully overlap. The highlighted patches are colored according to which patches will match if we try to maximize the number of overlapping patches. This is why we use our reduced search space (after vertical alignment) to identify distinct areas and salient matches to choose potential horizontal alignments.

$$s_i = \|HOG(p_i) - HOG(q_i)\| \tag{4.5}$$

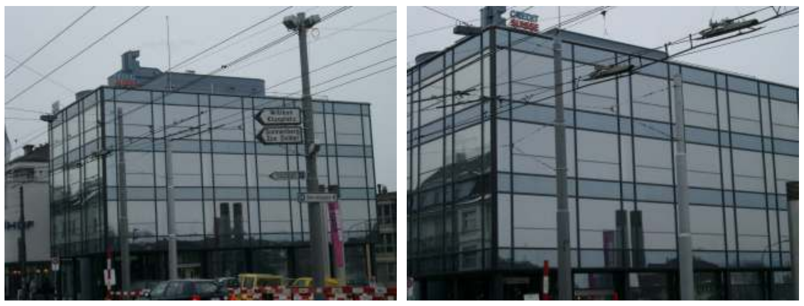
Using this equation, we gauge the saliency of each pixel and its surrounding region by the distance between its HOG descriptor and its closest match on the same row. We can visualize the range of these values as shown in Figures 4.6b and 4.12d. We compute the saliency at multiple scales or HOG window sizes as shown in Figure 4.12d to consider the saliency of different image region sizes. We can see in these examples how the more distinctive features and objects that are only seen once in an image stand out from the repetitive elements.

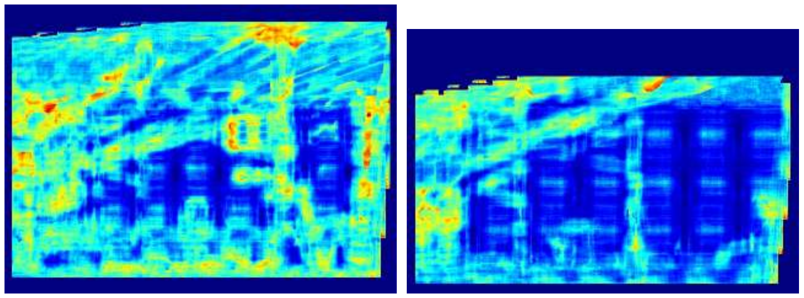
4.3.2 Pairwise Match Saliency

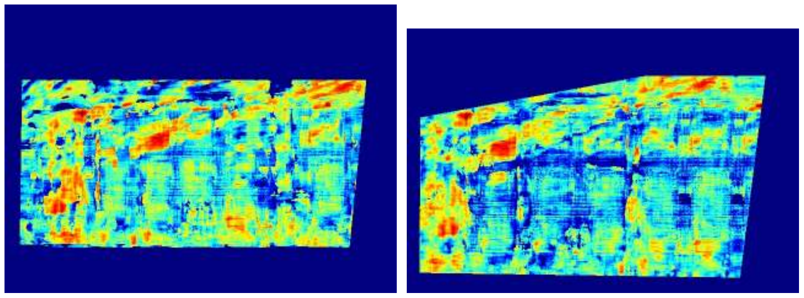
Once we know how salient each pixel is within its own image, we can use this information to gauge how confident we are in the saliency of keypoint matches found between the image pair. We define this value, m_s, in Equation 4.6. $HOG(p_1)$ is the HOG descriptor of a pixel, p_1, in I_1. We compare p_1 to the HOG descriptors of all pixels lying on its corresponding horizontal line in I_2. $HOG(p_2)$ is the HOG descriptor surrounding a pixel on this line that is most similar to $HOG(p_1)$. We enforce a bi-directional constraint on the matches here, meaning that p_2 is the best match for p_1 in I_2 and p_1 is the best match for p_2 in I_1. ϵ is a very small value to ensure both that m_s does not go to infinity and that a perfect match that is also repetitive does not have a high saliency score.

$$m_s = 1 - \frac{\|HOG(p_1) - HOG(p_2)\| + \epsilon}{\|s_1 + s_2\| + \epsilon} \tag{4.6}$$

If either p_1 or p_2 are elements of a repetitive pattern within I_1 or I_2, we cannot be confident that p_1 and p_2 are a unique and correct match. For this reason, we use the intra-image saliency scores s_1 and s_2 of both p_1 and p_2 respectively. We are confident in keypoint matches with high m_s values since this indicates that the matching pixels have similar descriptors and that they are both distinct within their respective images, leaving little room for ambiguity in the pair match. To distinguish the pixel matches that we are confident in from the ones that are either poor matches or ambiguous, we create an image visualizing the range of m_s values in the same manner we used to visualize the intra-image saliency values. The color map of m_s values for different image pairs are shown in as shown in Figures 4.6c and 4.12e. We apply Otsu thresholding [116] to identify the "foreground" pixels in this image, or those whose saliency values stand out from the rest of the image. Figures 4.6d and 4.12f show the saliency match fields we are left with after applying this thresholding. We choose to use Otsu thresholding since it is an automatic method that is adaptable to different types of images and does not require that we set one threshold value for all datasets. Other methods could also be applied at this stage to identify "foreground" pixels such as graph-cut, K-means, or ranking the matches based on their m_s values and taking the top n matches as the program designer sees fit.


a.


b.


c.

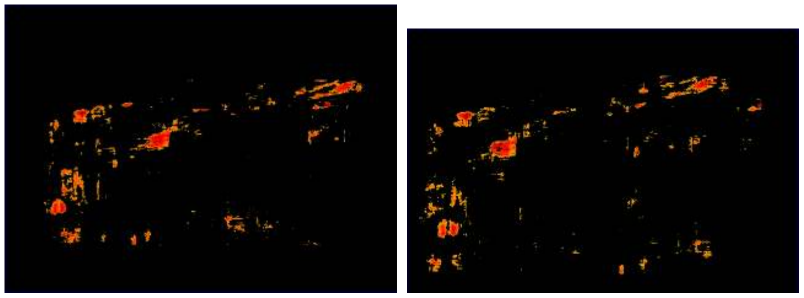

d.

Figure 4.6: Visualization of match saliency equation. (a.) Original images being matched. (b.) Intra-image saliency map of each image. (c.) Color fields of match saliency scores shown from the perspective of both images. (d.) Result after applying Otsu thresholding. All keypoint matches that are contained within the remaining colored regions are used to vote for a hortizontal alignment. The final aligned result using these matches is shown in Figure 4.1.

4.3.3 Voting for a 1-D Horizontal Homography

We use the pixel matches remaining in the pairwise saliency maps after thresholding to vote for H_h, with each match voting for a potential d_x value. Each match's vote is weighted by it's m_s score so that the most salient matches have a stronger voice in choosing the horizontal alignment.

We can encounter a potential obstacle during the horizontal shift voting stage if the scene in the image actually contains multiple planes that cannot be represented by a single homography. For instance, as shown in Figure 4.12h., if an object, such as a tree, is in front of the main building being photographed, one homography might align the two views of the tree captured between images, while a second homography will align the building of interest. Since, in general, foreground items like this also tend to be salient, we may get a large number of pixel matches voting for the shift to align the salient foreground as well as a strong support for aligning the salient features on the more repetitive building facades.

To handle this scenario, we consider several different potential d_x values. We have observed that the horizontal shift that aligns the salient foreground object does not also align the repetitive elements on the facade of interest, except by coincidence. Using this observation, if multiple d_x values have strong support, we choose the one that best aligns the repetitive facades in the images. We have already identified the image regions showing building facades when we extracted and matched groups of repetitive elements in Section 4.2.1.

To decide how many d_x values have strong support, we consider all shifts that have at least 50% of the number of votes as the shift with the most votes. We use each of these shifts to project all the members from the repetitive groups extracted at the beginning of our pipeline in I_1 to I_2 and vice versa. For all of the members that are projected within the matching image's plane, we maintain a counter. The counter is incremented every time a repetitive element is projected onto the location of a member of its matching group. If there is no matching group member at an element's projected location, the counter is decremented. After projecting all the group members between images, we divide the counter value by the number of group members that were projected within their matching image's boundaries. We choose the horizontal shift which has the largest counter score. An example of using this criteria to choose the horizontal shift that best aligns a building facade with salient objects in front of it is shown in Figure 4.12h.

4.3.4 Full 8 Degree of Freedom Homography and Refinement

At this point we have two 1-D homographies that can be merged into one 3x3 homography that encodes the scale change between the image pair and the translations along the x and y axises. If both images have been perfectly rectified, this is all the information we need to align them. However, since we are working with real-world data and applying an automatic rectification algorithm to large sets of images, there is naturally still some warping and distortion in the rectified results. To handle these effects, we can switch to a traditional 8 degree of freedom (DOF) homography at the end of our pipeline, using the original two 1-D transformations as an initial alignment estimate. One way to do this is to refine our transformation [12] using the iterative closest point algorithm [114], switching to a full homography in the process.

4.4 Evaluation and Discussion

4.4.1 Experimental Setup

We have evaluated our pipeline on the ZuBuD building dataset [1] and two datasets containing symmetrical elements in urban scenes [7, 11] which we will refer to as the SymFeat dataset and the SymUrban dataset respectively. All of these datasets consist of a number of photographs of buildings under changing photographic conditions. In the ZuBuD dataset, there are five images of each building with each image varying in some aspect such as perspective, scale, or rotation. We used the software provided by [115] to rectify the images in the ZuBuD and SymUrban datasets. The majority of the images in the SymFeat dataset are already rectified and front facing. For the ZuBuD dataset, using the images that were automatically rectified correctly, we have 716 pairs of images for testing. We ran SIFT and the contextual regional matching method described in Chapter 3 and [12] on these 716 image pairs. We chose these methods for comparison because both have been shown to be robust under changes in photographic conditions, but due to their relatively local nature (compared to using some sort of grouping and image abstraction), can be confused by repetition ambiguity in images. We manually identified about 50 image pairs in the original 716 that either had no or very few correct keypoint matches or a number of keypoint matches that incorrectly matched repetitive structures using both of these comparison methods. We used a similar selection criteria on the second two datasets. The SymUrban dataset consists of 9 image sets, each focused on a single building. There are around 30 photos of each view. From the subset of images that were correctly rectified, we again chose about 50 image pairs that represented all of the view changes presented by the dataset. We use these selected images with the image pairs in the SymFeat dataset that had reliable ground truth information to test our method to see if we can overcome some of the common challenges of aligning urban images. A list of the names and thumbnails of these challenging images are provided in Appendix C. To obtain ground truth for the ZuBuD and SymUrban images, we manually selected a set of four matching points between image pairs to compute an aligning homography. We use these ground truth homographies to quantitatively evaluate our results in Figures 4.10 and 4.11.

4.4.2 Results

We ran a variety of tests to evaluate both different design choices in our pipeline and how they build on each other as well as the overall effectiveness of our proposed methodology. Our first set of tests focuses on the contribution of breaking down the transformation estimation into two steps, focusing on aligning one dimension at a time. We next look at how matching accuracy increases as we progress through the steps of our pipeline, adding in the intra-image saliency and pairwise match saliency computations to the two-step framework. Included in our results is also an investigation of how the type of anchor point used for vertical matching impacts our results.

For all of our alignment accuracy tests, we compute a transformation matrix for each tested method. We translate every pixel in I_1 to I_2 using both the ground truth homography and the test homography. We compute the percentage of pixels that are transformed to the same coordinate. To allow for rounding errors and small inaccuracies in the ground truth data, we set a threshold on what the 2D distance between projected pixels can be to label a pixel as having been transformed correctly.

Two-Step and Saliency Computation Evaluation

We begin by performing tests to study the usefulness of our proposed two-step method for image alignment. We show that by aligning a single dimension first, instead of registering the full coordinate systems of an image pair, we can achieve a higher accuracy along that dimension. First, we aligned the test data using traditional SIFT matches with the 8-point algorithm [105] and RANSAC [33] to compute a full 8 degree of freedom homography. We then check the 1-D error of the matches along the y-axis. Then we use the y-coordinates of the SIFT matches to compute a 1-D homography (as shown in Equation 4.1) and again check the error along the y-axis. Our third test is to check the accuracy along the y-axis of the 1-D homography computed using the grouping method described in this paper. Figure 4.7 shows the results of these tests on the ZuBuD dataset and the results for the SymFeat and SymUrban datasets are provided in Appendix C. Both of the 1-D alignments yielded better results than the traditional 8 DOF computation across the test data. The benefit of our grouping method over both of the SIFT-based tests is seen especially on image pairs that have large scale, rendering, or lighting changes such as those encountered in the ZuBuD and SymFeat datasets.

Next, we look at what happens when we get to the second stage of the two-step approach. Figure 4.8 shows results after using the SIFT-based and grouping-based techniques described above in combination with different methods for determining the horizontal alignment. We first align the images vertically (using SIFT features or grouping features) and then match SIFT features only along corresponding rows.

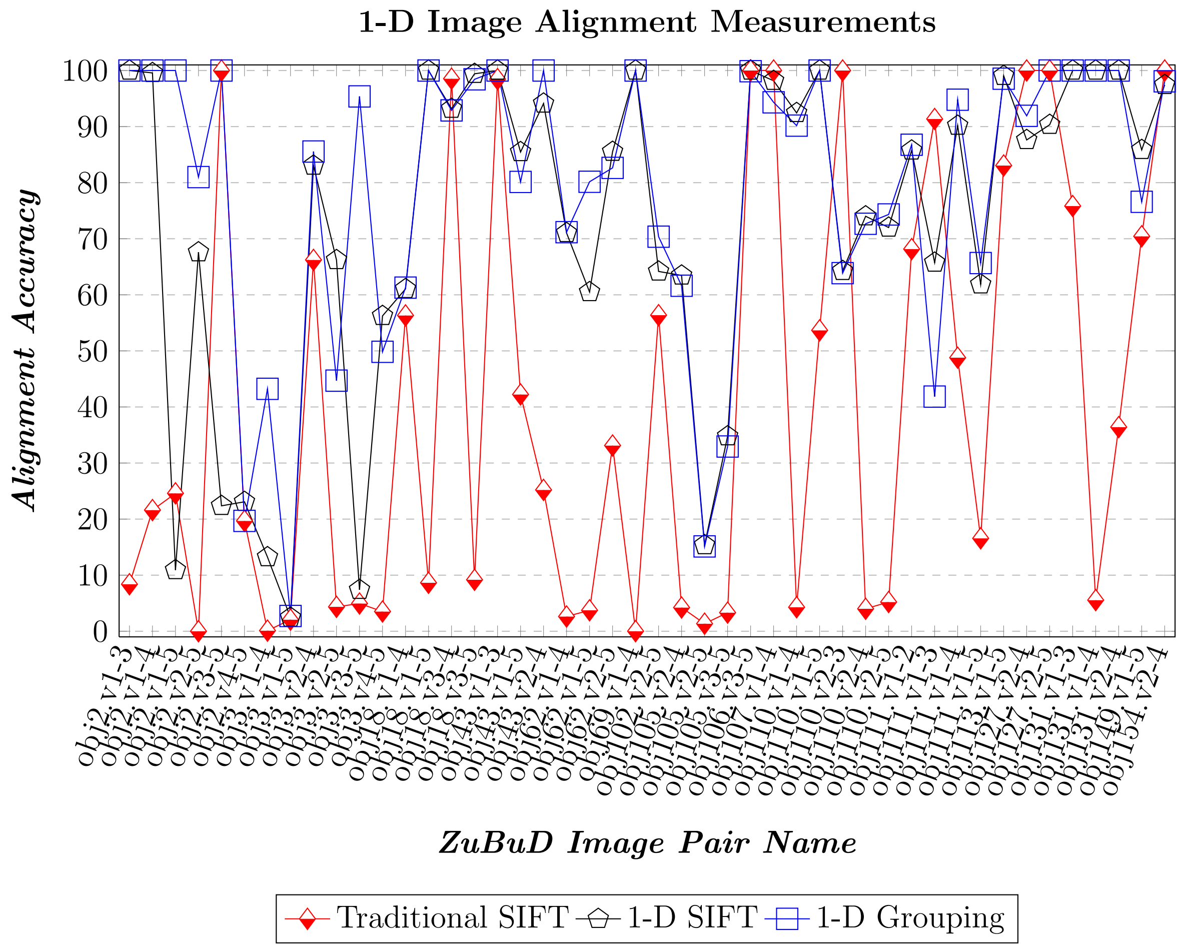


Figure 4.7: Test of our proposed dimension reduction on the ZuBuD dataset. Here we compare the 1-dimensional accuracy (along the y-axis) of different methods. The first is to using traditional SIFT matches with the 8-point algorithm to compute a full 8 degree of freedom homography. The second approach is to use traditional SIFT matches, but compute a 1-D, 2 degree of freedom homography (as described in Section 4.2.2) using just the y-coordinates of the SIFT matches. The third approach is the grouping method described in this paper to compute a 1-D vertical homography.

In both tests, distinctive SIFT matches are used to vote for a horizontal alignment. We can see that this minimal version of the two-step approach tends to outperform the traditional technique of solving a full 8 DOF matrix on challenging, repetitive images. The final test in Figure 4.8 shows the results of using our full pipeline, consisting of the two-step approach in combination with the intra-image saliency and pairwise match saliency computations. The general trend is that the basic two-step approach outperforms traditional SIFT matching and our full pipeline outperforms the basic two-step approach, showing the strength of our full pipeline.

Overall Pipeline Evaluation

To further test the overall effectiveness of our full pipeline, we compute homographies for aligning the test image pairs using our pipeline, SIFT keypoint matches, and contextual keypoint matches using the method discussed in [12]. The alignment accuracies for these methods are shown in Figure 4.10 Top. We also look at the accuracy of the keypoint matches obtained with each of these three test methods. Figure 4.10 Bottom shows the number of matches we obtain for each method and Figure 4.10 Center shows the percentage of these keypoint matches that are correct. Figure 4.10 shows the results of each of these tests for the ZuBuD dataset. The corresponding results for the SymUrban and SymFeat datasets are provided in Appendix C. We can see from all of these charts that our proposed pipeline tends to outperform the comparison methods. We do have several cases where the refined result has a significantly higher accuracy than the pre-refined result. Often times in these cases, the rectified images still have a small amount of warping which is not captured in our 1-D homographies. It takes the full 8 DOF transformation to model these distortions.

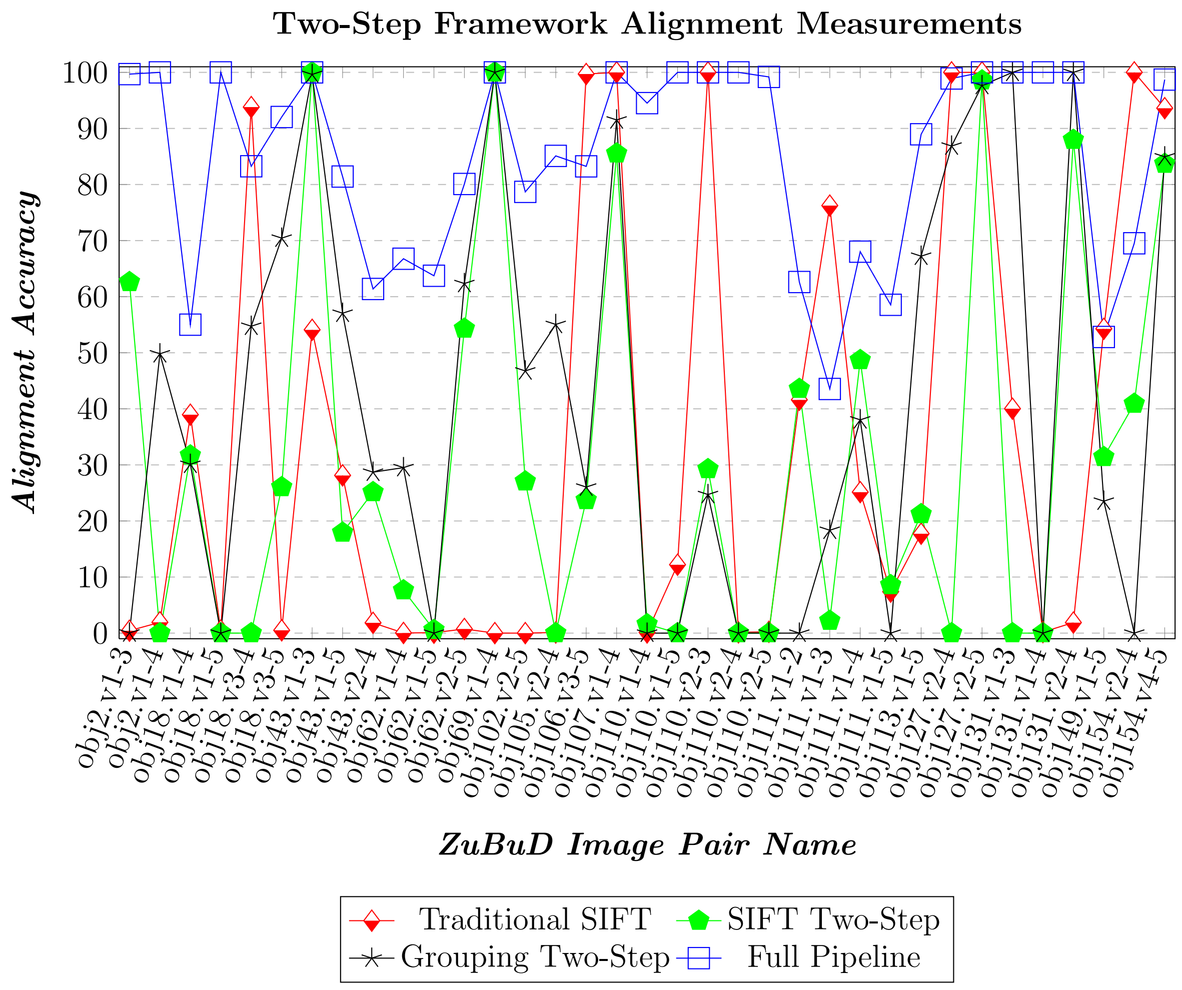


Figure 4.8: Test of our proposed two-step on the ZuBuD dataset. As a baseline, we show the accuracy of using traditional SIFT matching to compute a standard 8 DOF homography (*Traditional SIFT*). We then show tests for using 1) SIFT features (*SIFT Two-Step*) and 2) grouping features (*Grouping Two-Step*) to compute a 1-D vertical homography. For both of these tests, we using matched SIFT features along corresponding rows to vote for a horizontal 1-D homography. To show the strength of our full pipeline, we have also included in this chart the results of using the two-step approach with the intra-image saliency and pairwise match saliency computations (*Full Pipeline*).

Given the fact that our horizontal alignment stage takes into account dense matches in salient regions of the image pairs, as we saw in Figures 4.6 and 4.12, our pipeline also tends to have a dramatically higher number of keypoint matches in the end than our comparison methods. Despite the high number of matches we consider, they are comparably more precise to matches from other methods.

We show several visual examples of the result of our pipeline for qualitative evaluation in Figure 4.9. The images shown here are a sampling of the different types of data we handle. These image pairs have varying ratios of salient and repetitive regions. Several of the pairs also have changes in rendering style, time captured, scale, and lighting, all image characteristics that are known to cause challenges for many traditional keypoint matching methods [31]. Our method is able to handle many of these problems on top of the issues presented by repetitive data.

Anchor Keypoint Evaluation

To identify repetitive and salient regions for initial matching, as was discussed in Section 4.2.1, we need anchor keypoints to use for searching for and comparing regions throughout an image. There are a number of feature extraction methods that can be used for this purpose such as SIFT or Harris-Laplacian corners. Since most architectural images have abundant linear features that capture a great deal of a building's structure, we take advantage of this information to find anchor keypoints. We intersect line segments extracted from these images to estimate corner locations, which we refer to as Pseudo Corner keypoints [12]. A more detailed description of how we identify Pseudo Corners is provided in Section 3.2.6.

To test the usefulness of the feature detector we have included in our pipeline, Pseudo Corners, we computed its repeatability scores on the images we used for evaluation from the ZuBuD dataset. We compared the repeatability of our Pseudo Corners to that of SIFT and Harris-Laplacian. These values are provided in Figure 4.11 which shows that our Pseudo Corner detector tends to out perform SIFT and Harris-Laplacian on our test images. Given that Pseudo Corners are based on line segments, it is very well-suited for architectural imagery. Since it tends to be more repeatable on this type of data, we chose to incorporate it in our work to help increase the chances of extracting groups for matching that will exist in both images. However, depending on the type of data being used and the designers' preferences, other feature detector may be used in its place with the rest of our pipeline being run exactly the same. To demonstrate this, we also tested out our pipeline on the ZuBuD dataset using SIFT features as the anchor keypoints for grouping. The result of doing so is shown in Figure 4.10 Top with the labels “Repetition SIFT” and “Refined SIFT”.

Runtime

Table B.4 shows the average runtime of our pipeline on the tested ZuBuD images. These images were matched on an Acer Laptop with 6 GB of memory and a 2.2 GHz Intel Core i7 Processor running Ubuntu 12.04.

Table 4.1: ***Average Runtimes in Seconds*** and *Number of Features* for Stages of Our Pipeline on ZuBuD Dataset

Pipeline Steps	-
Average Image Dimensions	*628 x 457*
Line Segment Extraction (Section 4.4.2)	***0.5115***
# Line Segments	*1313*
Pseudo Corner Extraction (Section 4.4.2)	***1.2838***
# Pseudo Corners	*566*
SIFT Descriptor Extraction (Section 4.2.1)	***8.2625***
HOG Extraction (Section 4.2.1)	***3.9168***
# HOG Patches	*141493*
Repetition Identification (Section 4.2.1)	***11.9155***
# of Groups	*259*
Group Matching (Section 4.2.1)	***8.1449***
Vertical Alignment (Section 4.2.2)	***0.5165***
Intra-Image Saliency Computation (Section 4.3)	***17.5684***
Saliency Match Computation (Section 4.3)	***11.9831***
Horizontal Alignment/Voting (Section 4.3)	***0.0165***
Refinement (Section 4.3.4)	***5.9628***

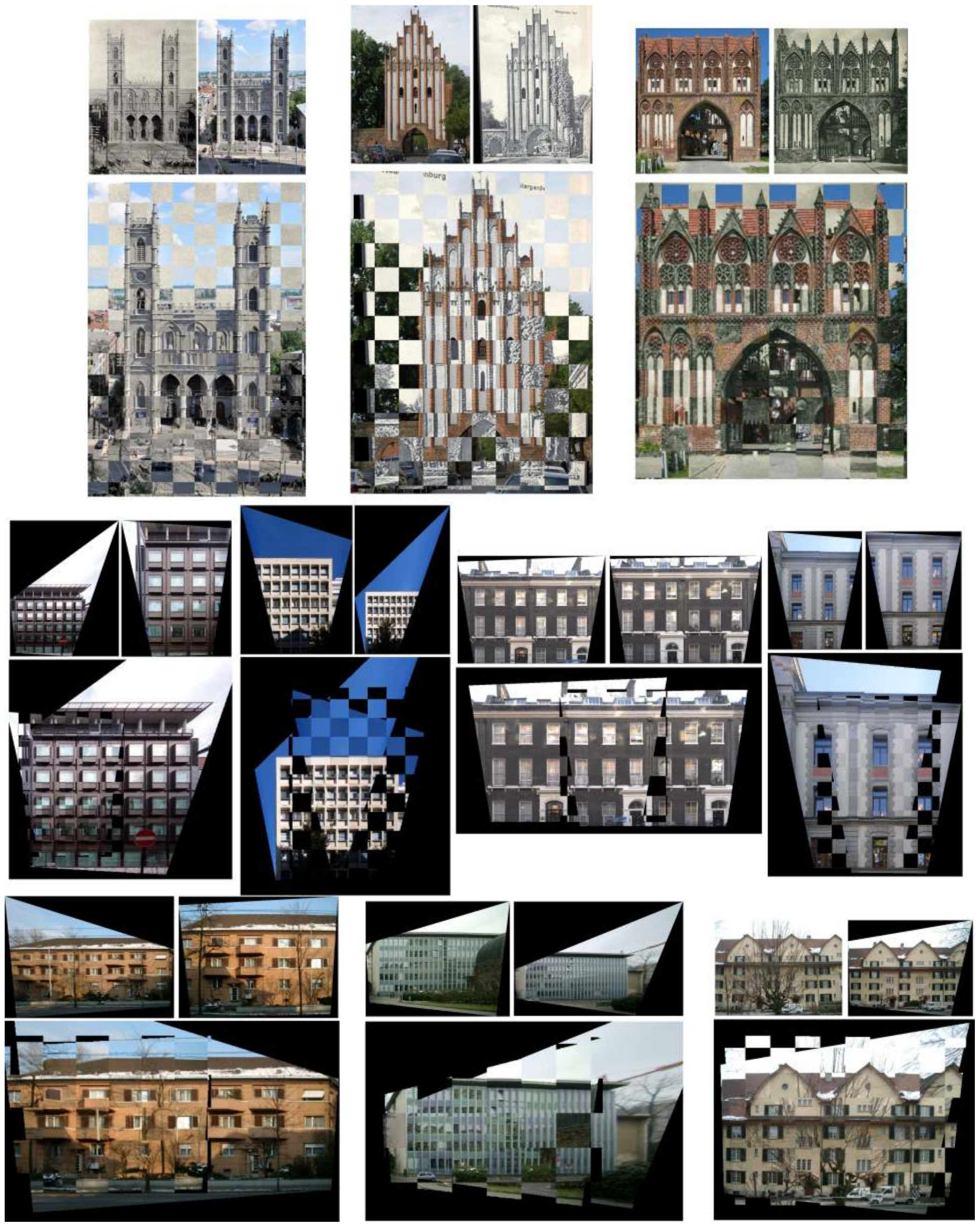


Figure 4.9: Visual alignment results. The original image pair is shown above each of the checkerboard views of the aligned result. These image pairs poses a variety of challenges to matching in addition to being repetitive such as different rendering styles, time changes, partial facade overlap, scale changes, and lighting changes. Image pairs shown are from datasets provided by [7], [11], and [1].

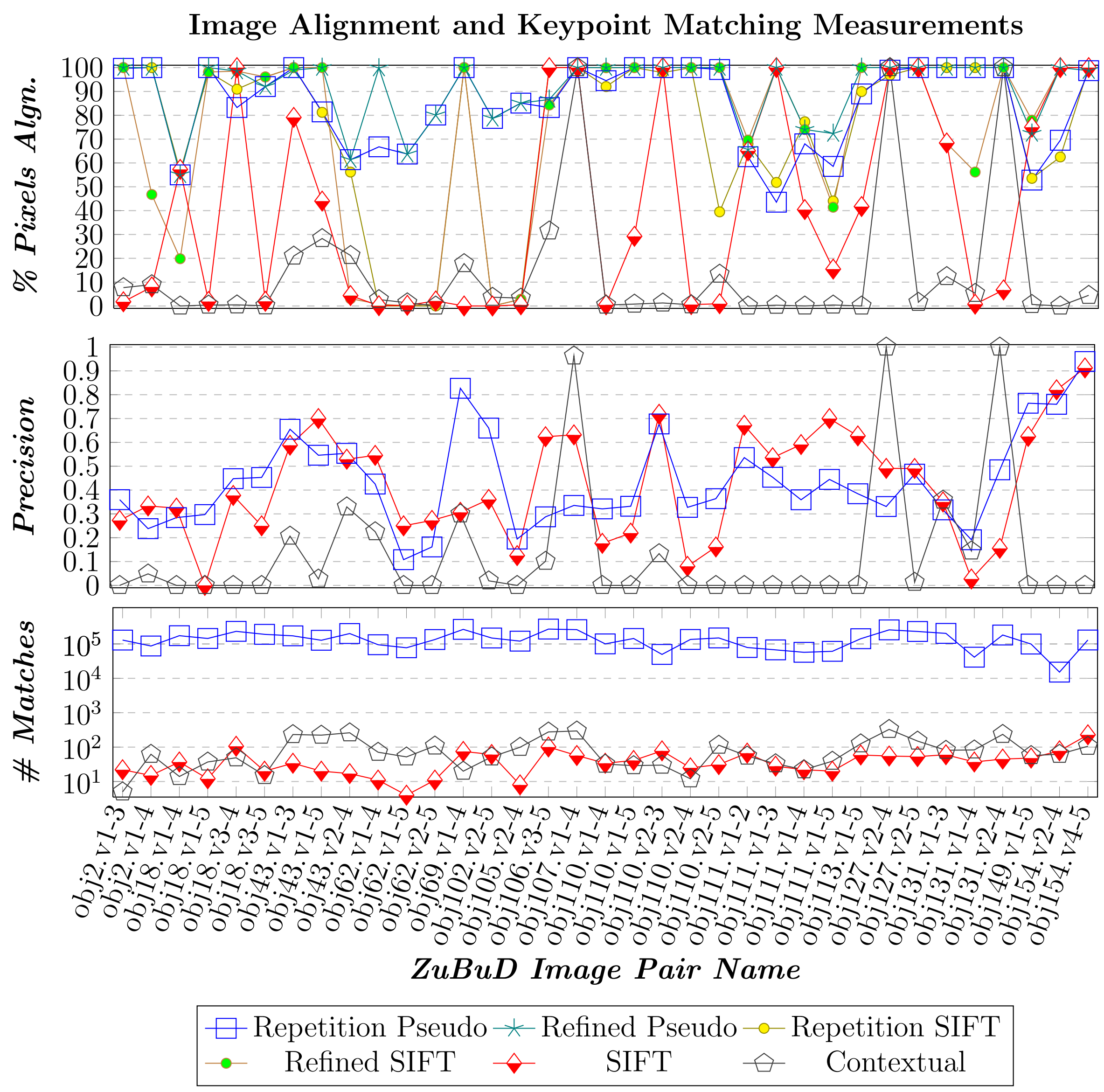


Figure 4.10: Quantitative results for stitching image pairs. *Top:* Percentage of pixels in image plane that are correctly aligned after using solved homography. *Center:* Percentage of matches that are identified using different methods that are correct. *Bottom:* Number of matches identified using several different methods. Here we show the result of using 1) our proposed pipeline with Pseudo Corners as the original anchor keypoints for grouping (*Repetition Pseudo*), 2) our proposed pipeline after refinement is applied (*Refined Pseudo*), 3) our proposed pipeline using SIFT features as anchor keypoints for grouping (*Repetition SIFT*), 4) our pipeline with SIFT anchor keypoints after refinement is applied (*Refined SIFT*), 5) using traditional SIFT extraction and matching with the 8 point algorithm + RANSAC (*SIFT*), and 6) the contextual method described in [12] (*Contextual*).

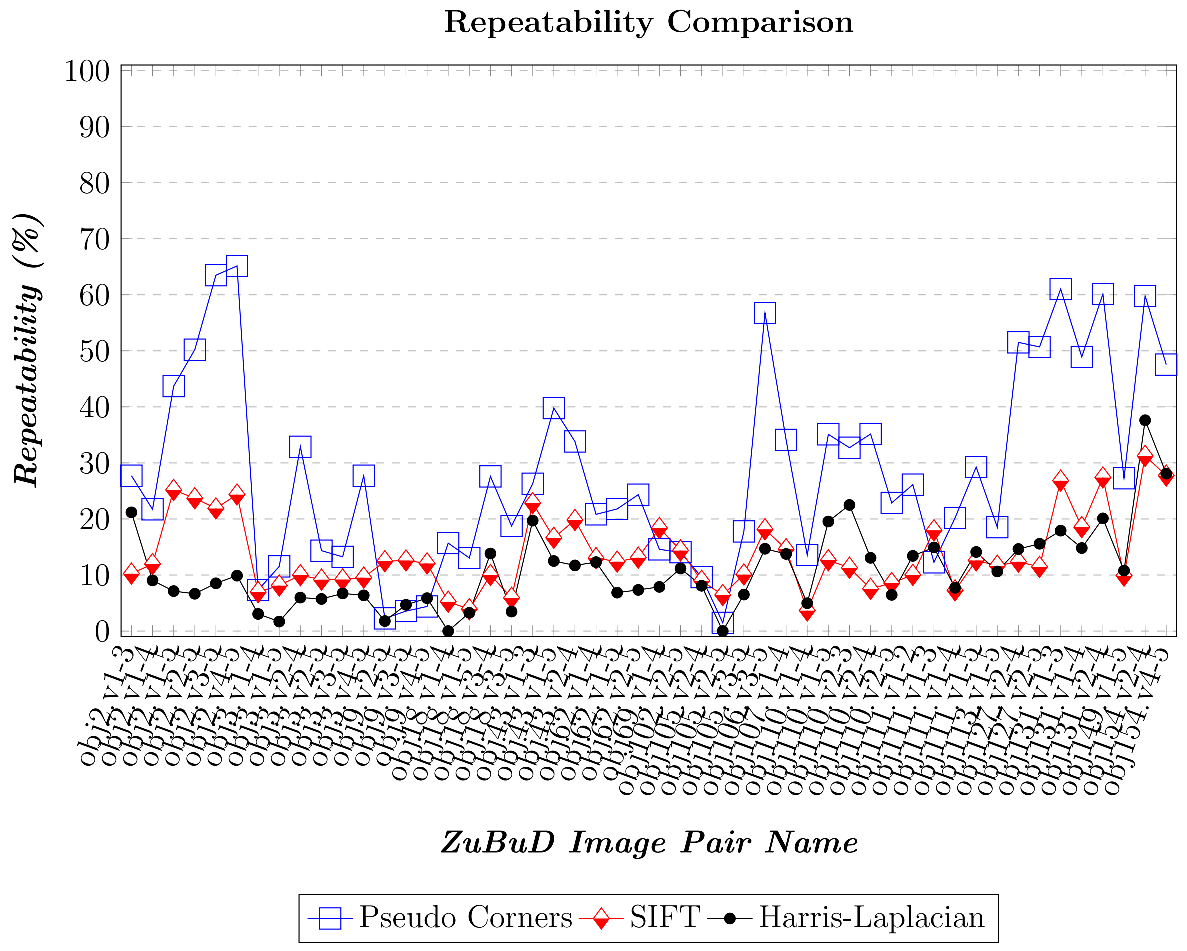
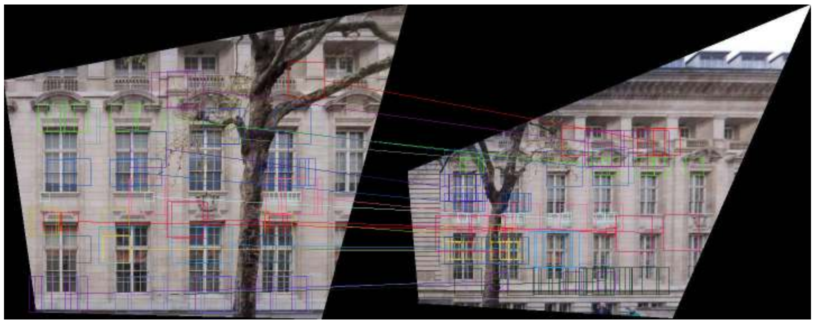
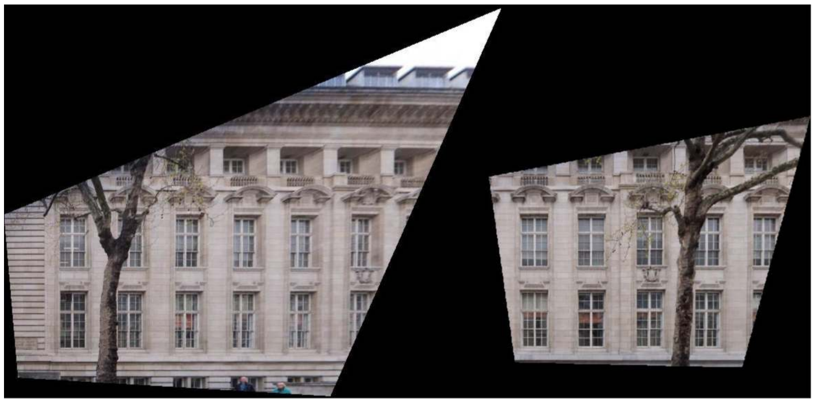
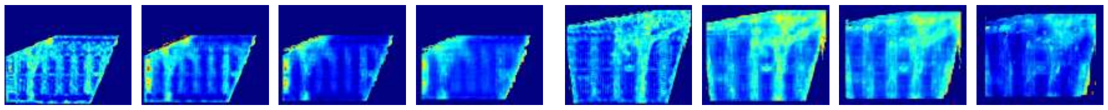


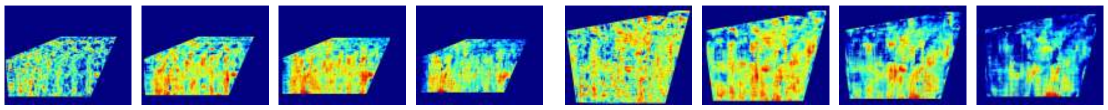
Figure 4.11: Repeatability measurements for SIFT, Harris-Laplacian, and Pseudo Corners on ZuBuD Dataset [1]. Our Pseudo Corners tend to be more repeatable than other common feature extractors on architectural data that can be characterized by linear features.

a.


b.


c.


d.


e.

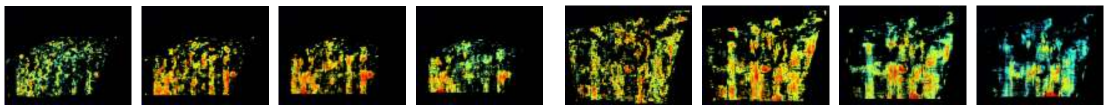
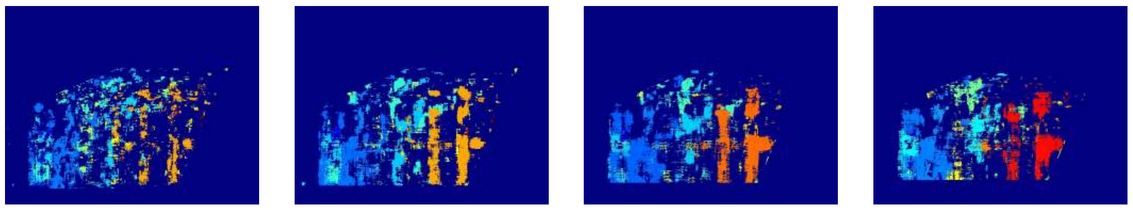
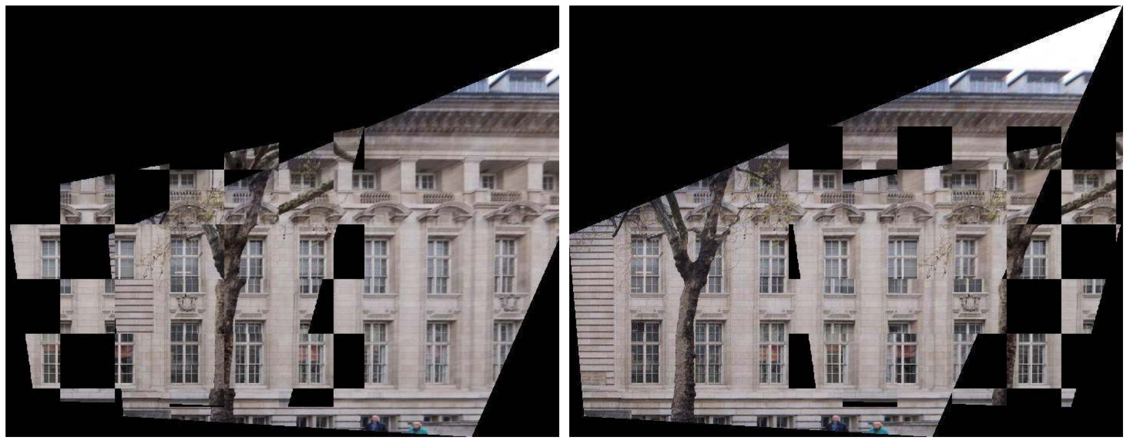

f.

Figure continued on next page.



g.



h.

Figure 4.12: Alignment of images containing multiple planes. (a.) Original images, before and after rectification, that are being matched. (b.) Subset of the corresponding repetitive groups after computing a 1-D homography (Section 4.2.2) are highlighted in matching colors. Every element of the groups is displayed. (c.) Images shown after they have been vertically aligned. Note that they now have the same scale and matching rows are aligned with each other. (d.) Intra-image saliency map of each image. The four color fields for each image show the inner-image saliency computed at different scales. The scale increases from left to right for each image. (e.) Color fields of match saliency scores shown from the perspective of both images. Again, the color fields are shown at four different scales for each image, increasing from left to right. (f.) Result after applying Otsu thresholding to match saliency score images. All keypoint matches that are contained within the remaining colored regions are used to vote for a hortizontal alignment. (g.) Displacement maps. Each match visualized in (f.) votes for an alignment. The range of these shifts is scaled from [0-1] and displayed as a color map. By doing this, we can see that at each of the four levels, two different displacements have strong support. (h.) Result of aligning the image pair using the two different horizontal displacements chosen in (g.) The left result aligns the salient tree that is in front of (and on a different plane than) the building of interest. The right result uses the salient matches on the building facade to correctly align the region of interest. Our selection criteria chooses the alignment on the right side as the one that best aligns the repetitive elements, and the thus, the building facade.

Chapter 5

2D-3D Registration

Our next task in our registration pipeline (Figure 1.1) is to align 2D images with 3D LIDAR scans. This entails calculating a camera's pose in relation to the 3D point cloud which requires that we define correspondences between the 2D and 3D imagery. We have a two-path pipeline for finding this set of correspondences depending on what type of data is available. Both methods use contextual information to perform matching between imagery captured by varying sensors with 3D peicewise planar segmentation and semantic labeling to guide the process. The first takes advantage of the set of pre-registered LIDAR photographs taken by the scanner at the time we obtain the 3D data. We know from the pre-processing stage what 3D point matches to what 2D LIDAR image pixel. By matching our 2D image to the LIDAR images, using the techniques discussed in Chapters 3 and 4, we can determine what 3D points match to the feature points of our photograph as is shown in Figure 1.2. Our second method assumes that no pre-registered photographs are available and we match 2D photographs directly with synthetic views of the 3D data. Regardless of which method we use, we obtain a single list of 2D-3D correspondences. We consider the 3D points to be very reliable and treat them as the absolute locations for the feature points.

5.1 Creating Segmentation Images

For the first path in our pipeline, in which we use the pre-registered images, we encounter many difficulties caused by the fact that these images tend to have small fields of view and exposure inconsistencies making them difficult to match to typical images people take with regular cameras. Path 1 (covered in Chapter 3) of our pipeline is designed to be robust to these types of visual discrepancies between images so that we can use this data for registration.

For Path 2 of our pipeline, we bypass our dependence on having pre-registered photographs to estimate camera poses for new photos and videos. Instead we create synthetic cameras throughout the scene and use their pose information to visualize a variety of perspectives of the LIDAR scan.

Both paths in our pipeline require that the main planes in the 3D scan be identified, so we use a RANSAC-based segmentation algorithm to gather this information similar to the one described in [117]. Since our LIDAR point cloud contains no information about which 3D points neighbor each other, we build a kd-tree to find each point's n nearest neighbors. Using this proximity data, we compute the 3D normals for each point using Principle Component Analysis (PCA) [118]. We then choose random sets of points and fit 3D planes to these groups. Given a plane, we try to expand it to see if more points in the cloud also lie on or near the plane. The plane is iteratively re-estimated as new points are added. If the plane has a large enough support, it is saved.

This segmentation, an example of which is shown in Figure 5.1 Top Right, tends to separate the 3D data into more divisions than are generally visible in our photographs, so we merge these smaller segments into larger ones. The result of this merging is shown in Figure 5.1 Bottom Left.

For Path 2 of our pipeline, we mainly want to work with the stable portions of the data, such as buildings, that do not change drastically with time. Our 2D-3D matching stage is already tackling finding correspondences between images from different sensors that have many visual discrepancies. By only looking at the stable portions of the images, we remove confusion in the matching caused by actual physical changes in the real-world that occur between the times that 2D and 3D data are captured. This also helps our registration process, because the boundaries and relative depths of 3D planes often appear in the original 2D imagery which generally contains edges at depth changes. However, planar information of foliage tends to mainly have edge clusters of physically proximate leaves that do not generally correspond to edges in 2D images. From our segmentation algorithm we are able to divide the segments into the three semantic categories of buildings, trees, and ground using a labeling method similar to [119]. All segments that cannot be described as a plane are labeled as trees. We are also able to identify the ground using the observation that in our 3D scans, the largest plane is the ground plane. All remaining planes are labeled as buildings. Figure 5.1 Top Right shows which of the original segments remain after the ground and tree segments have been removed. We use this information to remove merged segments that are likely to be ground or trees as well. Each merged segment votes if it is ground or trees or not by computing what percentage of pixels belong to a ground or tree segment in the original segmentation. If at least 50% of the pixels in a segment identify themselves as ground or trees, we remove all the pixels in the segment that are labeled as ground or trees. The result of this is shown in Figure 5.1 Bottom Right.

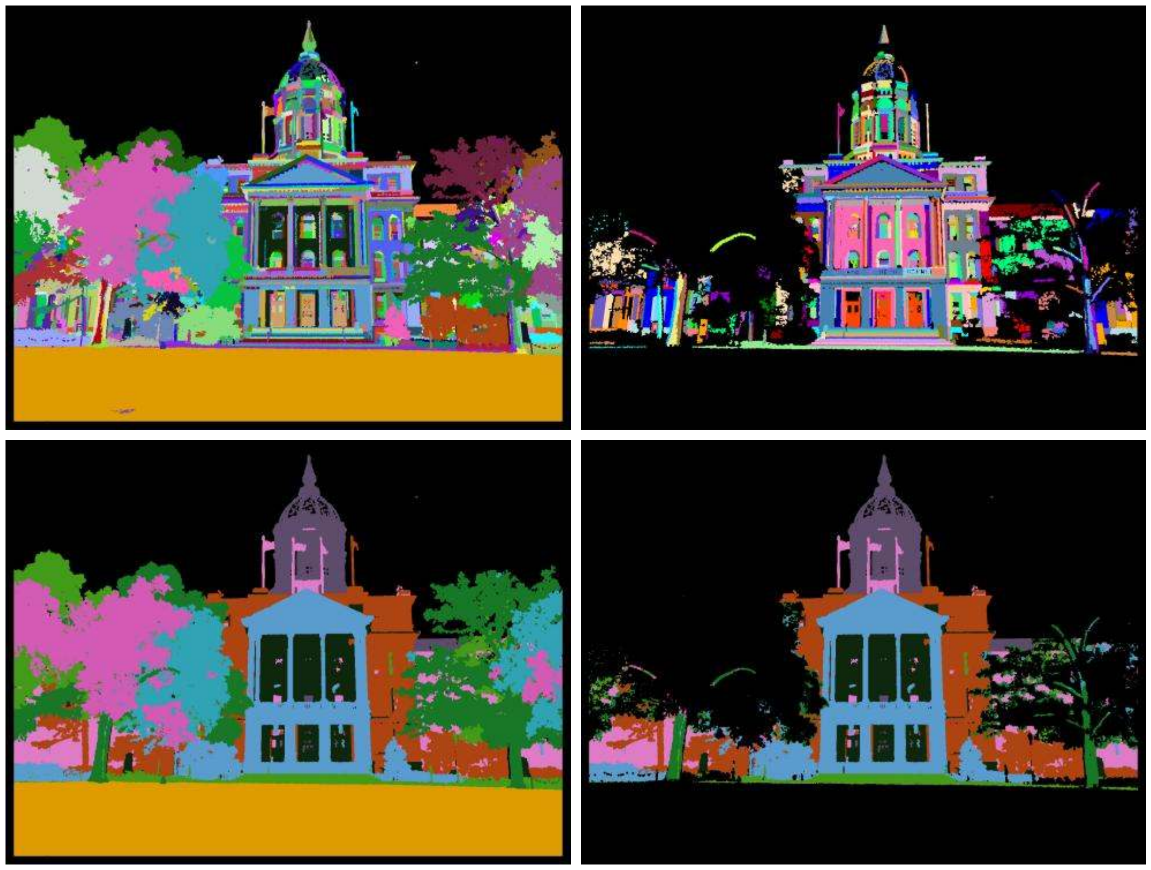


Figure 5.1: 3D segmentation images. *Top left:* Initial 3D segmentation results. *Top right:* Initial segmentation with segments marked as ground or trees removed. *Bottom left:* Initial 3D segments merged into ten main clusters. Segments are merged based on 3D depth. *Bottom right:* Main clusters after those that have voted themselves to be ground or trees are removed.

To construct the actual segmentation images in 2D we need to project various perspectives of the 3D scan which has been colored according to its segmentation

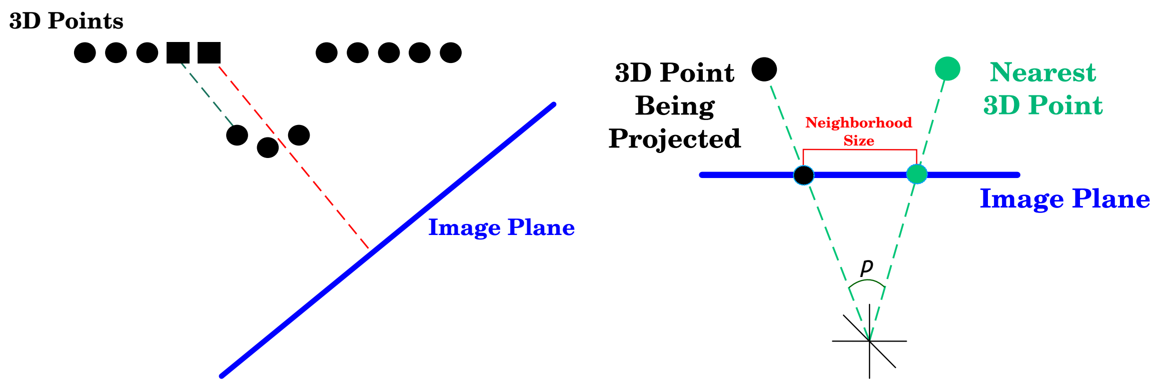


Figure 5.2: Projecting 3D scan to 2D image plane. *Left:* Circular points belong to areas visible from image plane. Square points lay on occluded areas in point cloud. Some occluded points are blocked by points with lower depths (green dotted line). Others, however, are not (red dotted line) and will erroneously be assigned to pixels without using our method. *Right:* The angular resolution ρ of the scan is used to determine the spacing between 3D points at particular depths. Two 3D points this distance apart are projected onto the image plane to determine the neighborhood size that adopt point A's depth. (*Notation corresponds to that used in Equation 5.1.*)

labels down onto 2D images planes. The 2D image plane used for construction can correspond to an actual camera's projection matrix, or a synthetic camera's projection matrix that is assembled by sampling various translations and rotations across the 3D scan. When a camera location is different than the LIDAR scanner, there may be areas of the scan that are occluded and are not visible from the camera's position. However, since the scan itself consists of points and not solid structures, a 3D point belonging to an occluded structure may actually be projected onto the image plane because no other point in the cloud blocks its path. This situation is demonstrated in Figure 5.2 Left.

We perform a depth testing step to overcome this scenario where 3D points are attached to a neighborhood of pixels when they are projected down onto the image plane. This neighborhood size is determined by the scan's angular resolution and the 3D point's distance from the camera center as is visualized in Figure 5.2 Right. Each pixel's final color is determined by the 3D point, with the smallest depth, whose neighborhood was projected onto it. The scan's resolution ρ and the depth d are used to determine how far the 3D point A is from the potential nearest point B using Equation 5.1.

$$|A - B| = dsin(\rho) \tag{5.1}$$

The imaginary point B is placed the calculated distance from A along the direction of the image's principal plane, which is parallel to the image plane and goes through the camera center. The 2D distance between A and B when they are projected on the image plane becomes the neighborhood size. All pixels within this distance of A are assigned A's depth. Points in the neighborhood are excluded from acquiring A's depth if they meet one of two conditions; the point has already been assigned a lower depth or it has a depth within a certain threshold of A's depth. The latter condition ensures that a point's depth is not attached to neighboring points on the same structure that may just be slightly further away from the image plane. The orientation of the principal plane is determined from the projection matrix. Any 3D point lying on this plane will have a depth of 0, as is shown in Equation 5.2.

$$P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{5.2}$$

For this to hold true, 3D points on the principal plane multiplied by the third row of the projection matrix will always yield a result of 0. We can describe the principal plane using the normalized vector representing the projection matrix's third row, n. The angles (θ_x, θ_y) between this vector and the x and y axes are given in Equation 5.3.

$$\begin{aligned} \theta_x &= arccos(n_x) \\ \theta_y &= arccos(n_y) \end{aligned} \tag{5.3}$$

This process is also used to color the 3D point cloud after image registration as well as to create depth images.

5.2 Finding 2D-3D Correspondences

5.2.1 Using Existing Images for 2D-3D Matching

If following Path 1 of our pipeline, we match new images to photographs obtained by the LIDAR scanner. We are able to compute the camera poses of the LIDAR photographs from data provided by the scanner (Section 1.1). We determine this 2D-3D mapping using each camera's focal length in pixels and extrinsic parameters (rotation and translation) that are read from a LIDAR file. Using this information, each 3D range scan point is projected onto each image plane to find its corresponding 2D point as is shown in Equations 5.4-5.5.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K(Ra + t), \quad K = \begin{bmatrix} f & 0 & -w/2 \\ 0 & f & -h/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

$$\hat{a} = Ra + t, \quad x = f\frac{\hat{a}_x}{\hat{a}_z} - \frac{w}{2}, \quad y = f\frac{\hat{a}_y}{\hat{a}_z} - \frac{h}{2} \tag{5.5}$$

where a is a 3D point, f is the focal length of the LIDAR camera in pixels, R is the rotation, t is the translation, w is the image width, and h is the image height. If the solved for x and y fall within the boundaries of the image dimensions, the 3D point has a corresponding 2D point in that image. All coordinates are in normalized homogeneous form. There is no guarantee that Equation 5.5 will produce homogeneous pixel coordinates that have a depth of 1 so each calculated point must be divided by its depth. The entire point cloud is projected onto each LIDAR image once and the 2D-3D correspondences are saved for future use.

We can then proceed by matching new images to the LIDAR photographs using the contextual method discussed in Chapter 3. The only change we make now that we have corresponding 3D information for the pre-registered images, is to incorporate it in the geometric verification stage that was covered in Section 3.3.

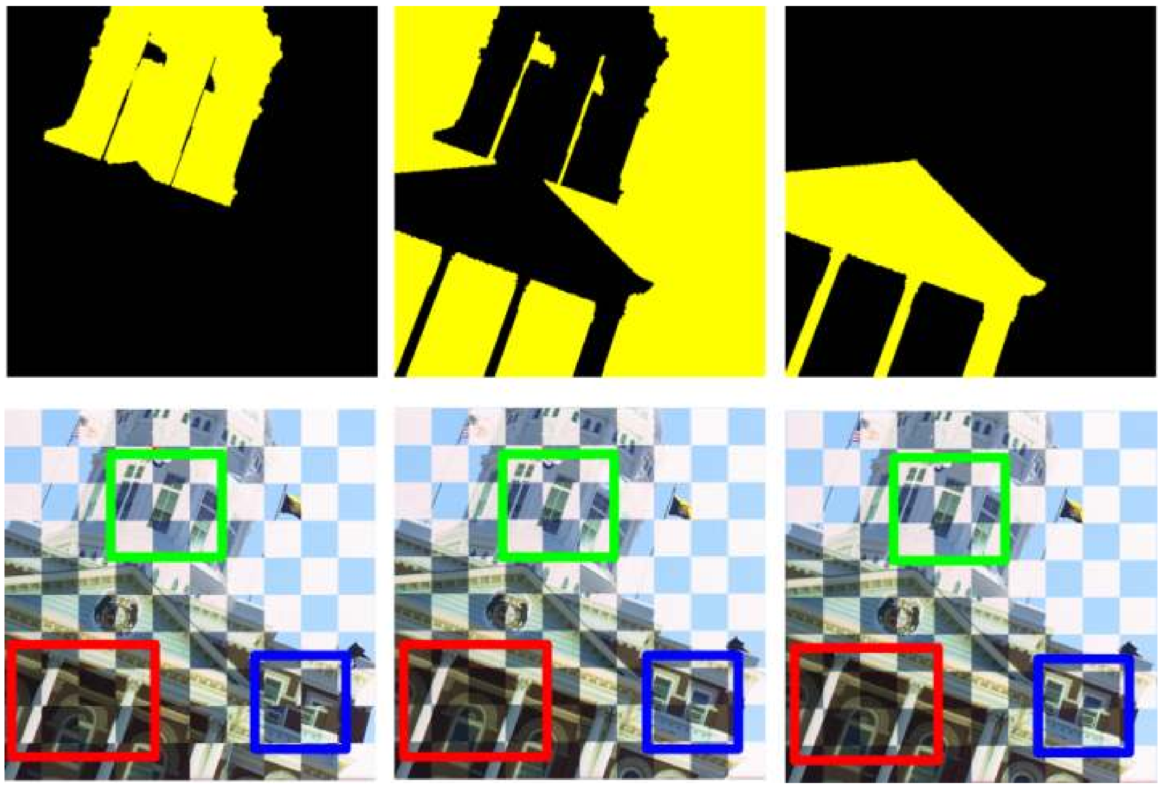


Figure 5.3: Computing multiple homographies using 3D segmentation as a guide. *Top:* Segmentation of 3D points viewable from a LIDAR photograph. *Bottom:* Images stitched for three different homographies using only keypoints in the corresponding LIDAR segment. Areas that have alignment changes between homographies are highlighted and are best aligned when using the plane they lay on is used to guide the homography calculation. For instance, the section highlighted in blue is best aligned in the center example where matches on is plane (shown in yellow on the top row) are used to calculate the homography.

Multiple Homography Estimation with Segmentation Information

We geometrically verify all our 2D matches by calculating the homography relating I_1 and I_2 and removing outlying correspondences. We also use the aligning information to search for additional matches. To account for various 3D planes that may be represented in the images, we compute multiple homographies for each image pair [106, 107, 108]. We know from our pre-processing stage what 3D planes are visible from the LIDAR camera, and we merge the planar segments until only three remain. Each homography matrix H_i is calculated using the keypoint matches laying on one of the three segments as shown in Figure 5.3.

Every H_i is refined by using gradient descent to align the edges in the aligned image pair as well as possible. This process is described in greater detail in Section 3.3. Once the image alignment has been optimized, we extract new matches along the edges of the images to increase our set of matches. We accept new matches at edge points whose local HOG's agree in both images.

5.2.2 Using Synthetic Views for Direct 2D-3D Matching

It is possible for us to visualize both the photographs and various perspectives of the 3D scan in such a way that these two modalities actually look very much alike. Doing so allows us to match the 2D and 3D imagery in a similar manner to that described in Section 3.2. We have explored visualizing our data in two different manners. The first focuses on finding hard edges that potentially represent depth changes in the imagery. Our second approach allows use to focus on texture variations along and between different surfaces captured in the data.

Direct Matching with Planar Segmentation Images

We have observed that oftentimes the edges between different planes in the 3D scan show up as intensity differences in photographs. These intensity differences can be picked up by extracting edges in the photographs. Potentially corresponding edges in the 3D data can be identified in the segmentation images also by extracting edges on the 2D image plane. Using this information, we have explored matching edge images across modalities to find 2D-3D correspondences. Examples of these edge images are shown in Figure 5.5 Second Row.

We use the segmentation image to extract geometrically intrinsic, lighting independent edges as opposed to using a camera's grayscale depth image because depth images do not always provide us with large enough intensity differences between planes to find important edges for matching. An example depth image and its edges are shown in Figure 5.4 Top. If the depths of 3D points visible from a camera have a very large range, when they are mapped to the grayscale values of $0-255$ points with relatively close depths will have very similar colors. Trying to increase the contrast by only coloring points within a certain range of depths or removing certain percentages of points that are the closest to or furthest from the camera is not feasible across an entire LIDAR scan where important objects and structures have all different arrangements and locations. In the segmentation images, on the other hand, individual segments are randomly colored according to their labels and provide much more contrast along boundaries of interest as can be seen in Figure 5.4 Middle. We can see in this example that the edge information in the segmentation image shows boundaries around windows and between different sections of the roof that are not included in the depth image edges.

The segmentation image also provides us with semantic information about the content of the scan that is not available in a low-information level depth image. As discussed in Section 5.1, each segment is labeled as building, tree, or ground. An example of this labeling is shown in Figure 5.4 Bottom Left. We are primarily interested in matching planar regions in this Path that will have depth and normal discontinuities that are visible as intensity differences in the 2D photographs, so we will focus on the sections of the segmentation that have been identified as buildings and only extract edges along these segments as shown in Figure 5.4 Bottom Right.

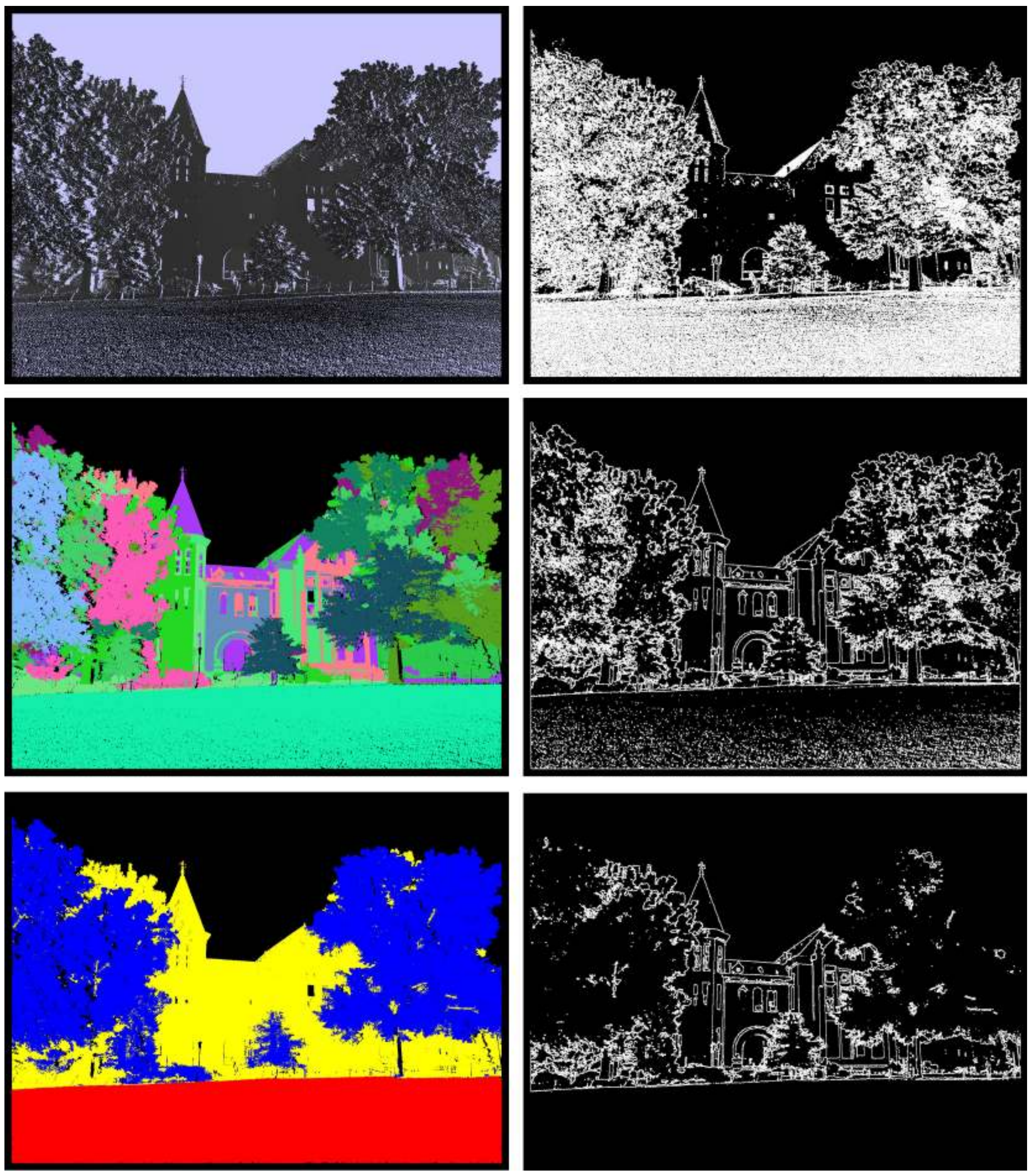


Figure 5.4: Using 3D segmentation image for 3D edge extraction. *Top left:* Grayscale depth image. *Top right:* Edges extracted from depth image. Many windows and details in the roof are not present in this edge image. *Middle left:* 3D segmentation image. *Middle right:* Edges extracted from segmentation image. *Bottom left:* 3D segmentation image after each segment has been labeled as building (yellow), tree (blue), or ground (red). *Bottom right:* Edges of segments that are labeled as building. Segmentation edges give us more information for matching than the depth image does.

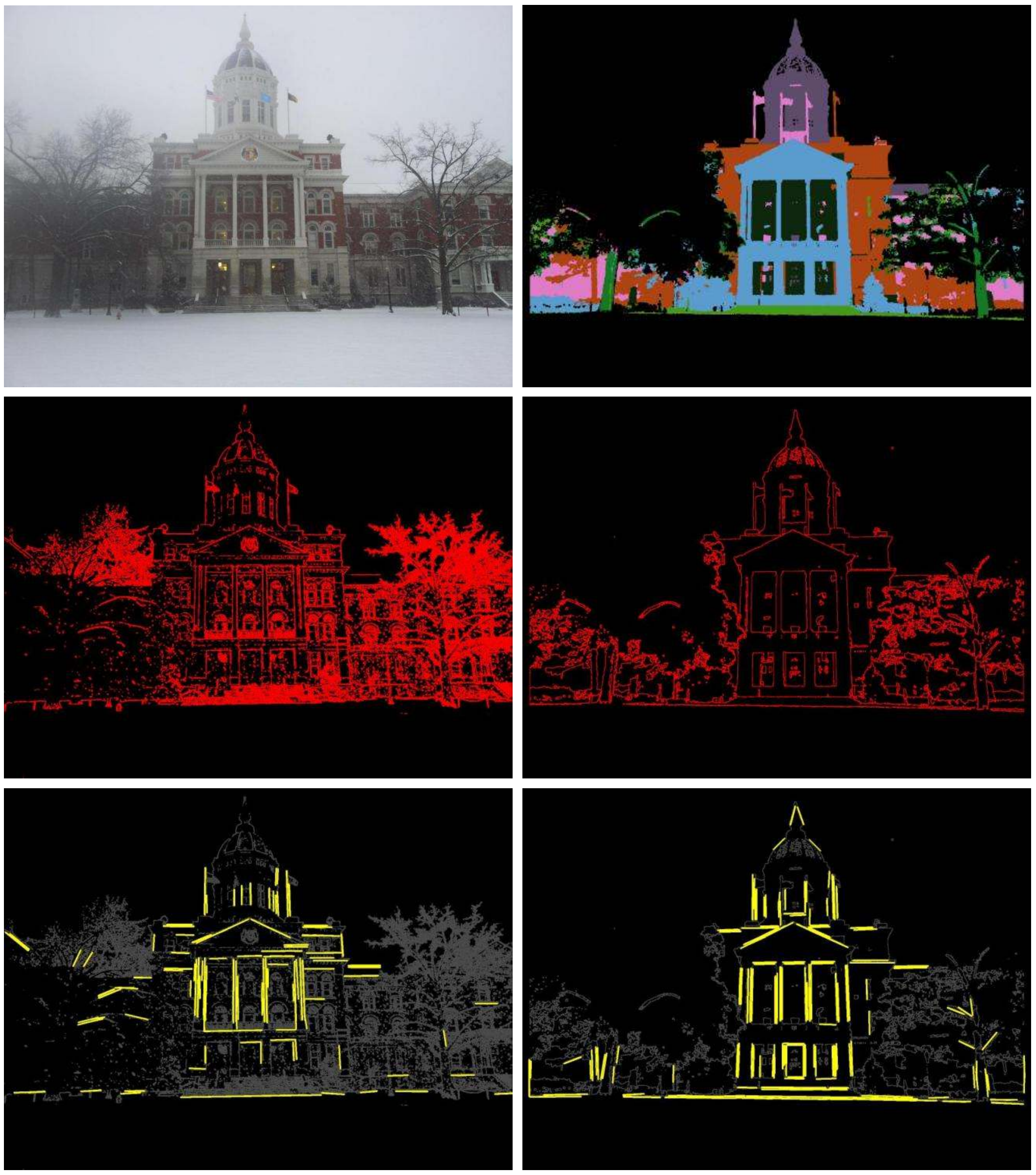


Figure continued on next page.

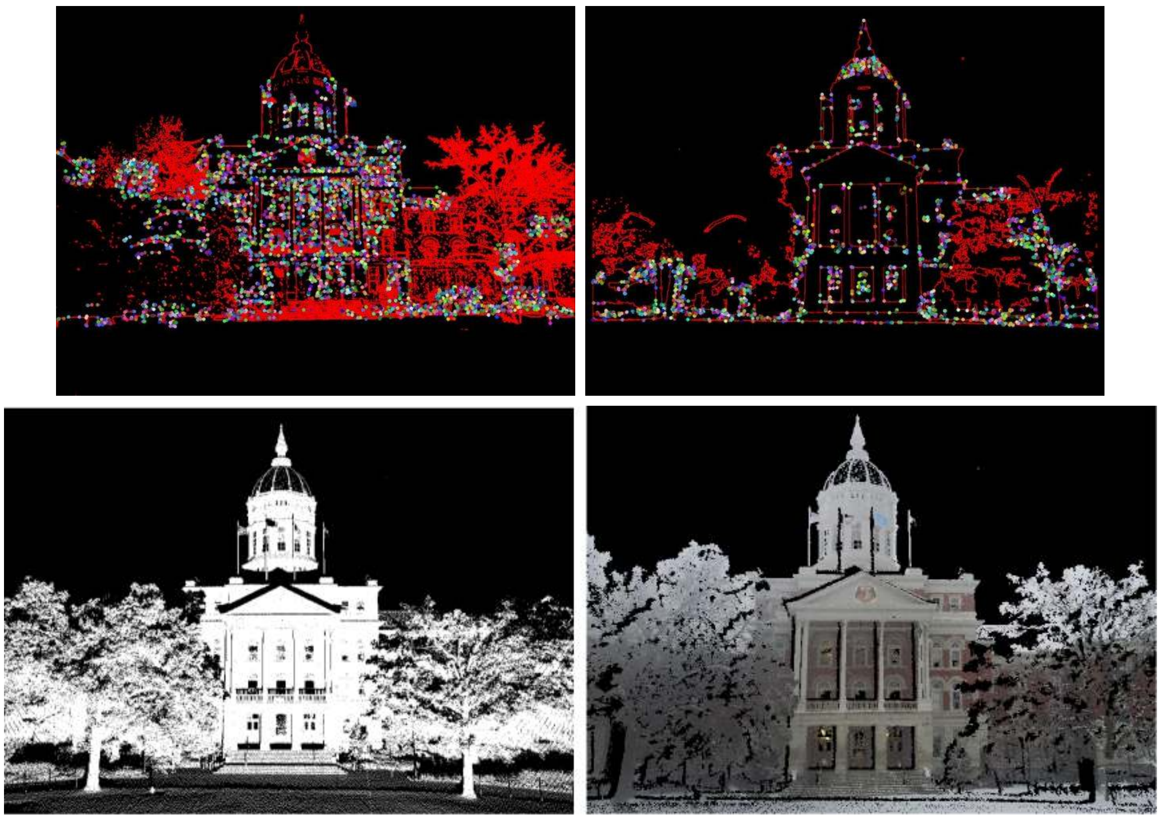


Figure 5.5: Extracting corners to match directly from 2D to 3D. *Top row, Left:* 2D photograph to be matched to 3D scan. *Top row, right:* 3D segmentation image with ground and tree segments removed. *Second row:* Edges of original images. *Third row:* Line segments overlaid on edge images. *Fourth row:* Corners extracted from edge images that are near line segments. *Bottom:* Final registration result shown in 3D environment.

We collect Harris-Laplacian corners throughout the edge images as our potential feature points. We are mainly interested in working with corners along the stable structures (buildings) in the images. To minimize the corners we pick up in other parts of the images (that are more likely to have content changes over long periods of time), we first identify line segments in the edge images using the Line Segment Detector method. Only corners that are close to these line segments are used in the matching stage. Using corners near line segments in both dimensions in addition to limiting corner extraction to building segments in 3D allows us to match regions of the images that have similar attributes and reduces our search space for feature matching.

The line segments for the edge images are shown in Figure 5.5 Third Row and the corners deemed to be on stable structures are displayed in Figure 5.5 Fourth Row. We extract line segments on the edge image instead of the original images so that the line structures will be picked up throughout the image regardless of the original color intensity patterns in the regions of interest. We use Canny Edge detection to extract edges with relatively low thresholds so that it picks up more potentially useful information in low contrast areas. Other edge extraction methods can be used here such as [120], but Canny works well for our purposes.

To match these corners, we extract HOG patches around corners in the 3D segmentation's edge image and match them to all the HOG patches surrounding corners in the regular photograph's edge image. If the closest HOG match in the regular photograph's edge image has a distance below a certain threshold to the segmentation's HOG patch, the match is saved. More details on our HOG contextual matching are provided in Section 3.2.3.

To increase our confidence in these corner matches, we estimate the homography aligning the photograph and 3D segmentation images and run the iterative closest point (ICP) algorithm [114] to find a better alignment and more accurate matches. A visual example of this process of shown in Figure 5.6.

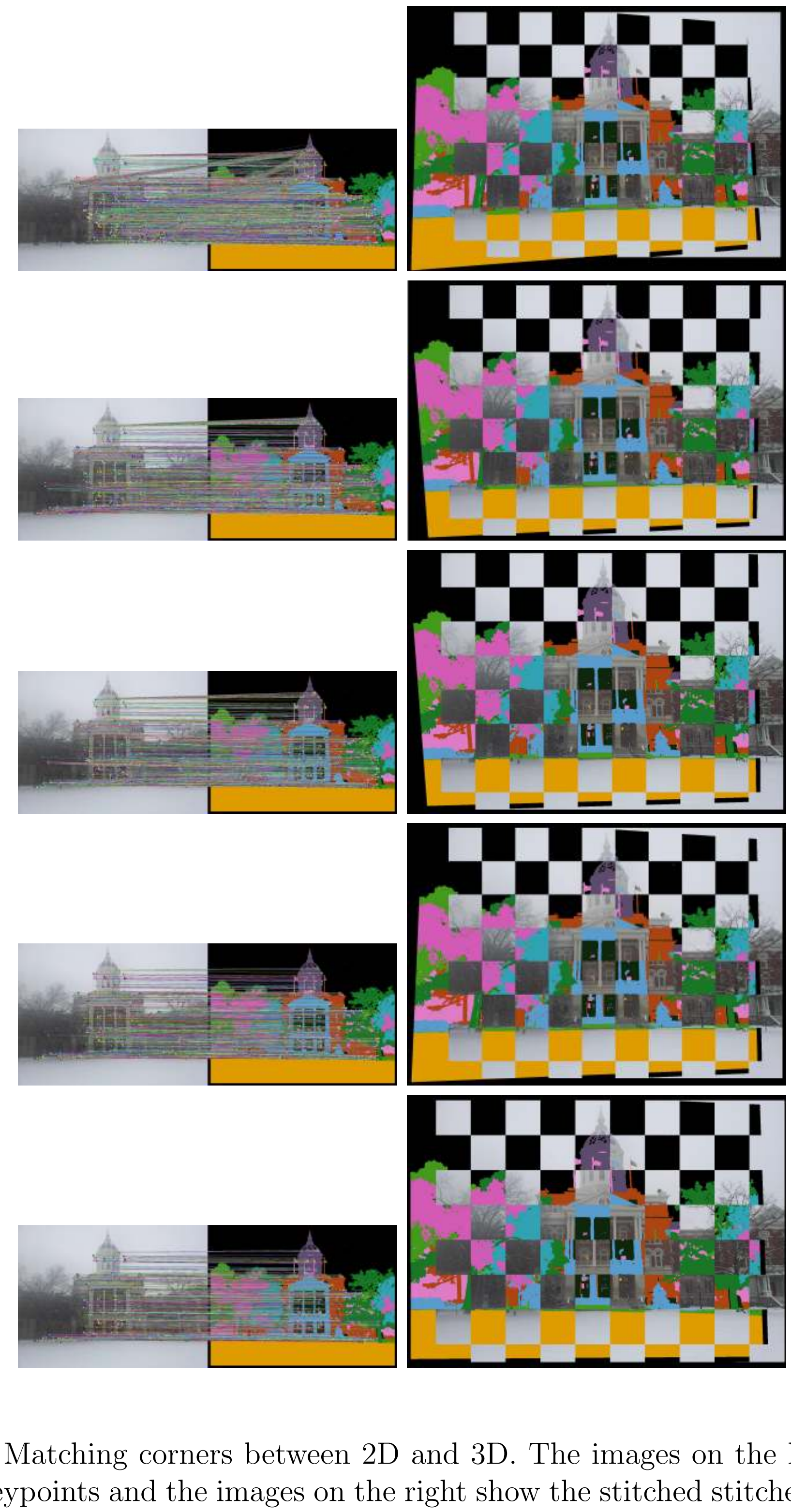


Figure 5.6: Matching corners between 2D and 3D. The images on the left show the matching keypoints and the images on the right show the stitched stitched images using the homography calcualted from the keypoint matches. The rows show progressive iterations of ICP.

The matches between the photograph and the 3D segmentation give us a set of 2D-3D matches that we used to solve the projection matrix defining the relationship between the photograph and the 3D LIDAR scan as shown in Figure 5.7. Examples of several photographs, the 3D segmentation image they are matched to, and their alignment with the LIDAR scan are shown in Figure 5.14.

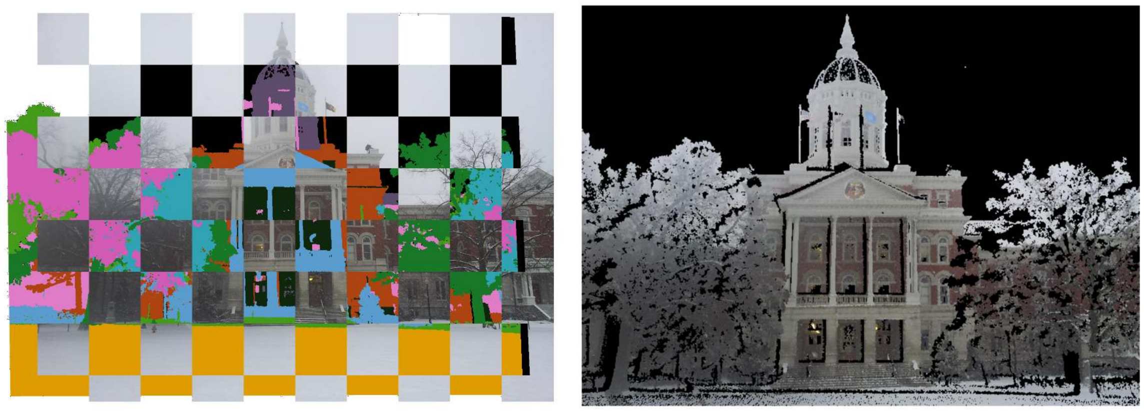


Figure 5.7: 2D-3D matching result. *Left:* Alignment between 3D segmentation image and photograph for registration. *Right:* Photograph projected onto 3D scan.

Direct Matching with Laser Intensity Images

The second type of synthetic image we create takes advantage of a particular property captured in LIDAR scans. LIDAR scanners use a laser beam to scan an area and measure the time it takes for the laser to leave the scanner, hit a surface, and reflect back to the scanner in order to determine the depth of the point hit. In addition to keeping track of the time of travel of the laser beam, it is also relatively standard for scanners to store the power of the returned laser beam. This reflectivity value provides some information about the type of surface struck. If a surface absorbs a large portion of the laser, only a small percentage of the original laser beam will be returned and a lower reflectivity value will be stored. However, if the texture of the surface hit is highly reflective, the power of the returned laser beam will be higher. The returned value is also influenced by a number of factors including angle of incidence, distance to target, and atmospheric conditions [121], but the overall trend of which surfaces have higher and lower laser reflectivity values tells us something about the changing surface texture found in a scene. By visualizing the range of these values, we can create images that catch very similar texture variations to those seen in normal photographs as demonstrated in Figure 5.11.

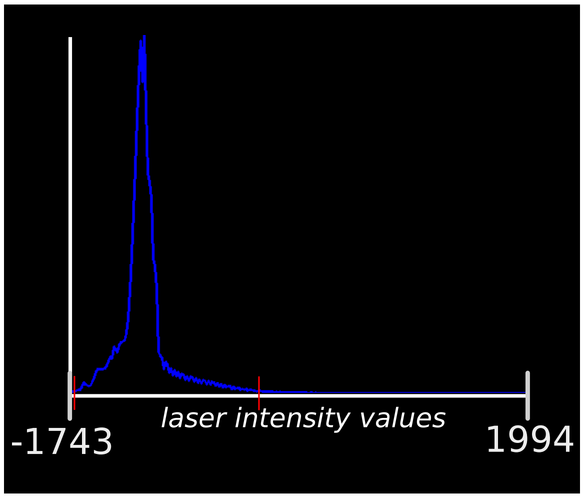


Figure 5.8: Histogram of laser intensity values for scan taken of Carnahan Quad on 6/19/13. This scan's values range from from $[-1743, 1994]$. The red markers show cut offs for the region in which most of the reflectivity values lie.

The full range of laser reflectivity values provided by the scanner is very large and, on the scans we have worked with, usually seems to be in the neighborhood of $[-2000, 2000]$. Figure 5.8 shows a histogram of the laser intensity values for our scan of *get histogram for rolla*. This distribution where the bulk of the values are in a subset of the full range appears to be the norm. We visualize each scan based just on the values within this concentrated area to increase the visual contrast of different laser reflectivity values.

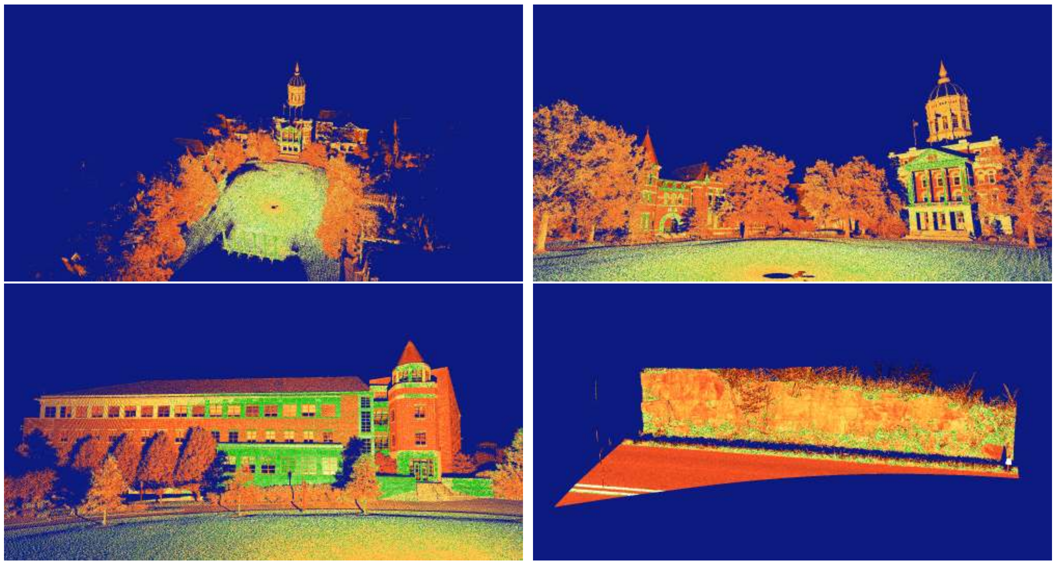


Figure 5.9: 3D LIDAR scans colored according to laser reflectivity values.

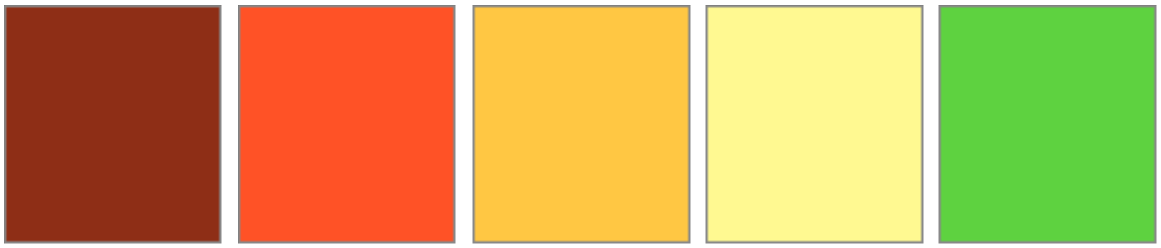


Figure 5.10: Anchor colors used to create color map for visualizing laser reflectivity values.

Using the boundaries obtained from the histogram, we assign a range of colors to the points of the scan as shown in Figure 5.9. We split the color range into n regions. In the examples below, we use $n = 4$, transitioning from a reddish/orange color scheme through yellow to green. Each of the four sub-ranges is assigned two colors; one for the beginning of the sub-range and one for the end. Within each sub-range, we can linearly interpolate between the anchor colors to determine colors for varying laser values. Figure 5.10 shows the anchor colors used in the examples in Figure 5.9. The value of n and the colors assigned to each sub-range are arbitrary.

Using these images, we can perform matching across dimensions as described above. Since the laser reflectivity images provide information about the texture changes across surfaces, we match the gradient magnitude of these images with images of original photograph's gradient fields. Again, we can focus on matching planar surfaces. We can use the edges found from the 3D planar segmentation to focus on areas in the reflectivity images that lay on planes. As we did with matching to the segmentation images, we use the locations of line segments in the 2D photograph to determine which feature points to use for matching. Figure 5.11 shows an example of the matches found using the gradient image of a reflectivity image and the gradient image of a standard photograph.

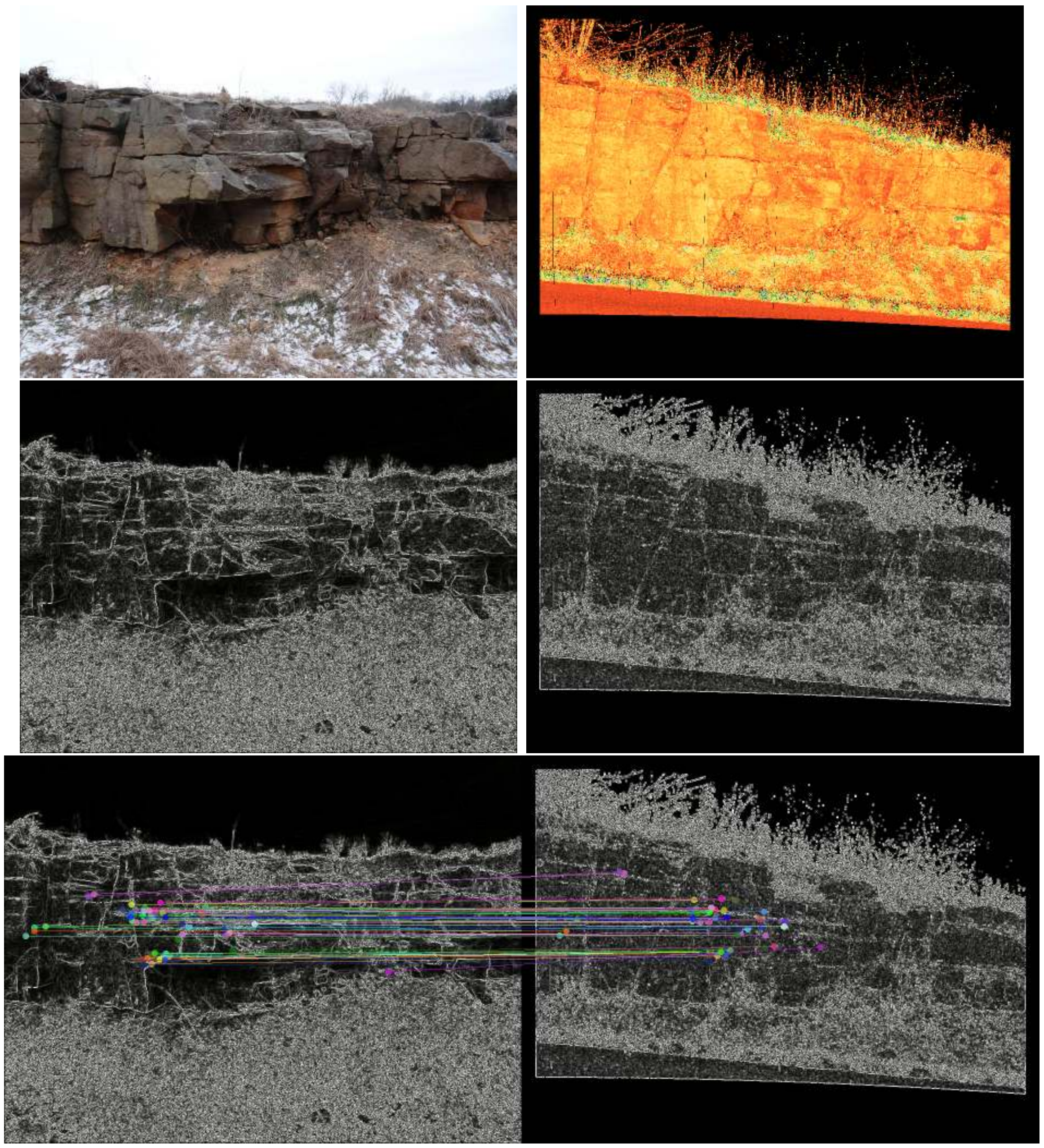


Figure 5.11: Matching photograph to laser reflectivity image. *Top:* Original photograph and laser reflectivity image being matched. *Middle:* Gradient magnitude images of original images. Notice that the images look much more similar in this view. *Bottom:* Keypoint matches between gradient magnitude images.

5.3 2D-3D Registration and Optimization

Using the set of 2D-3D matches we have obtained from either Paths 1 or 2, we proceed to our 2D-3D registration stage. Our set of 2D-3D correspondences is used to calculate the projection matrix of the camera. We carry out the six-point algorithm with DLT (Direct Linear Transform) [105] and RANSAC (Random Sample Consensus) [33] to find the set of matches that calculate the most accurate projection matrix P. See Appendices A.2, A.3, and A.5 for more details on computing the projection matrix. This process uses a relatively sparse set of matches to estimate the projection matrix. These matches may only have been found in and represent certain portions of the image and not the entire image plane I. To refine our camera pose estimate we find a denser set of 2D-3D matches across the image plane and then refine P using this set of dense matches with Levenberg-Marquardt optimization [105].

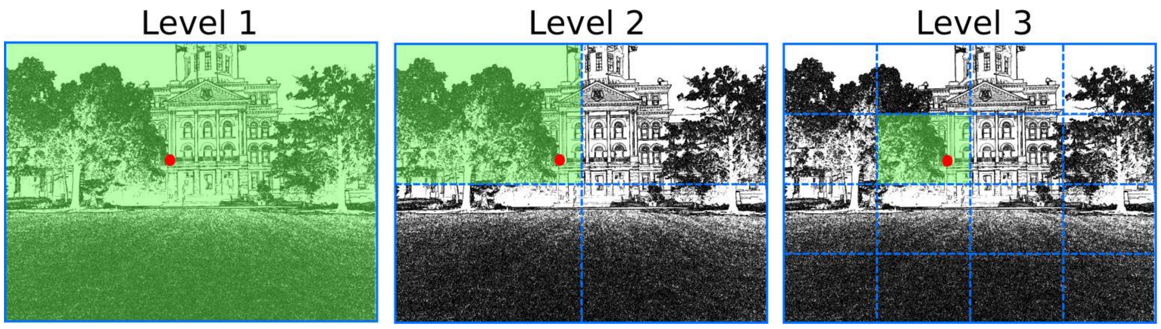


Figure 5.12: Hierarchical gradient descent for guided matching. One image from an aligned pair is shown. Gradient descent is applied on multiple window sizes (*outlined in blue*) and the best alignment for each patch is saved. When we match an edge pixel (*shown in red*) to its corresponding pixel in the second image, we choose the patch alignment that yielded the lowest energy at the pixel. In this example the red edge pixel has three estimates for its optimal alignment. The windows used to find these alignments are highlighted in green

.

To get this new set of matches, we create the 3D segmentation image S by projecting the segmented 3D point cloud onto I's 2D plane using P. This segmentation image is a valuable source of information because it contains the dominant edges of the 3D point cloud as they are viewed from our estimated camera pose and are in the same 2D coordinate system as I. Using the alignment of the segmentation image and the regular photograph, we perform guided matching along the segmentation image's edges to directly obtain a relatively denser set of 2D-3D matches that cover the image plane. These matches are used to refine the projection matrix estimate. S's edges ideally should align with the strong gradient responses in I. To ensure that this is the case we incorporate gradient descent in a hierarchical fashion. We first look for this alignment globally, finding one transformation to project S to I. We then divide S and I into smaller and smaller patches, calculating the best alignment for each patch pair. For every edge pixel in S, we save the alignment that gave us the smallest energy in an effort to minimize Equation 5.6, where (e_x, e_y, e_z) is the 3D point that corresponds to an edge pixel in S after it has been projected to the image plane using the projection matrix P. This process is visualized in Figure 5.12.

$$E = \sum_{j=1}^{\#\text{ of edges in } S} [1 - \bigtriangledown I(P * (e_{j_x}, e_{j_y}, e_{j_z}, 1)^T)]^2 \tag{5.6}$$

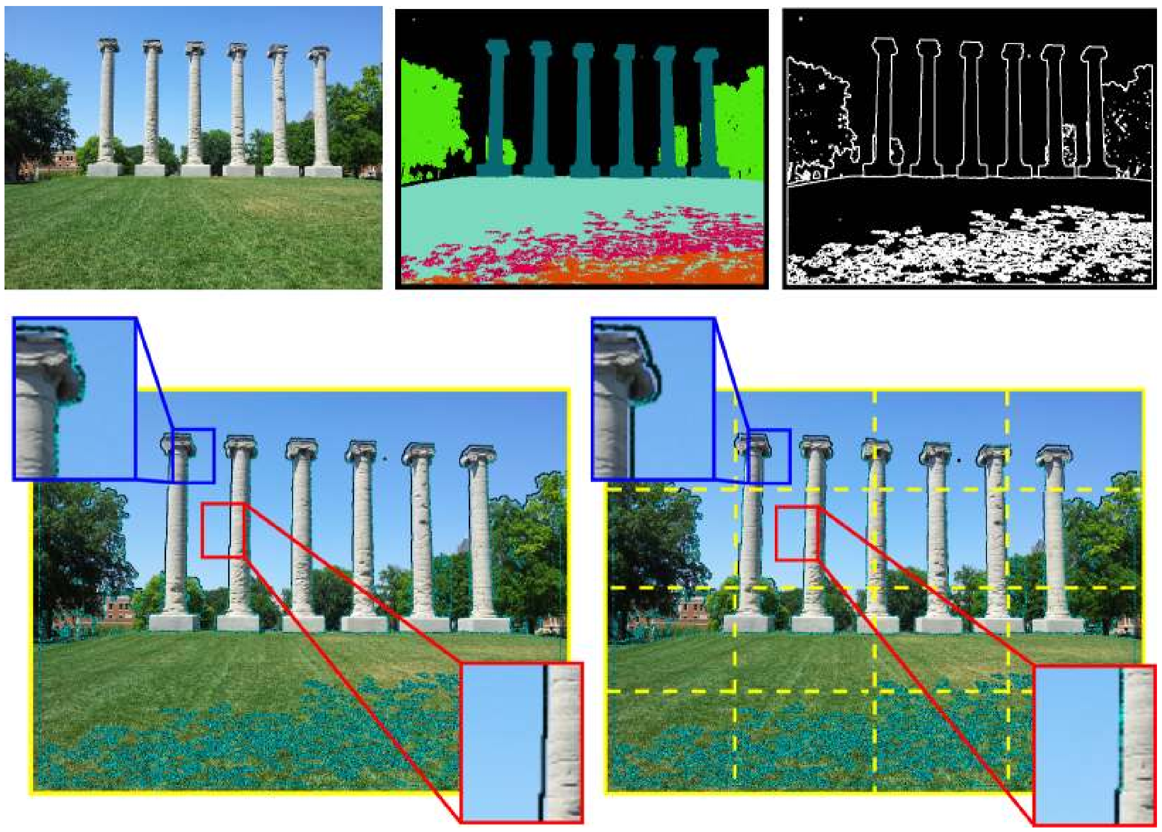


Figure 5.13: Hierarchical guided matching using segmentation image. *Top:* Original image, 3D segmentation image, and 3D segmentation edge image. *Bottom:* Edges of segmentation image are overlaid on original image and colored according to energy at each pixel. Brighter, turquoise pixels have lower energies and are better aligned. Different patch sizes are used to perform gradient descent and find new 2D-3D matches. Patches are outlined in yellow. The blue boxes highlight an area that is better aligned using global information for gradient descent. The red boxes highlight an area more strongly aligned using more local information.

We take this multiple patch approach because varying amounts of image information will help us find the best alignment for different areas. During the gradient descent process, very accurately matched sections of the segmentation image along the 2D plane may be shifted away from their optimal alignment due to noise and inaccuracies in surrounding regions. In these cases, it is expected that a smaller window size, looking at a subset of the image, will help maintain the best alignment for the edges in these sections. Less-well matched sections, however, may need to rely on the stable regions elsewhere in the image to help guide them to their best alignment, thus requiring a larger window. Figure 5.13 demonstrates how this hierarchical approach is helpful.

The new set of 2D-3D matches are searched for and extracted along the shifted edges of S. If S and I both have a strong edge at the same pixel location and the neighboring pixels have similar gradient orientations in both images we save the corresponding 2D and 3D points as a new match. We refine P using Levenberg-Marquardt optimization to minimize the reprojection error of this new set of matches. We are more confident in our final calculation of the projection matrix since it is based on a large set of matches located throughout a large percentage of the image plane that have been chosen using contextual information after multiple rounds of registration estimation and refinement.

The image can finally be registered with the scan by mapping every 3D point onto the image plane using the refined projection matrix. The color assigned to each 3D point is the color of the pixel it is closest to when projected onto the image plane. During this stage, the image's 2D-3D correspondences are saved so that it can be used to guide the registration of new images.

5.4 Results and Discussion

We judge the accuracy of our results both qualitatively and quantitatively. When the 2D images are viewed next to the registration of the images and the range scan (Figures 5.14 and 5.15) or next to the depth image created using their registered camera pose (Figure 5.16), one can easily visually judge the accuracy of the projected data. We also consider the reprojection error of 3D keypoints when they are mapped to the image plane. We gathered ground truth by manually choosing 2D-3D correspondences between photographs and segmentation images with mapped 3D coordinates. Table 5.1 shows the reprojection error of the ground truth matches using the projection matrices computed from the ground truth matches, the matches found in Path 1, and the matches found in Path 2. We also present the number of matches found and their reprojection errors when using Paths 1 and 2.

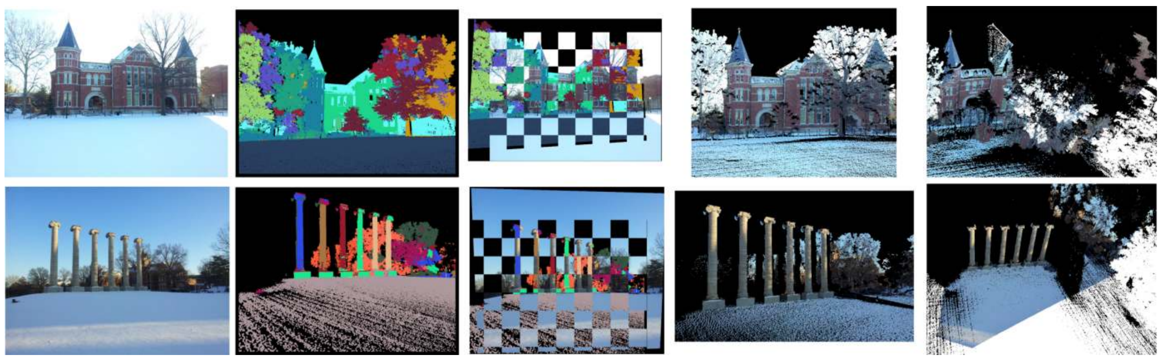


Figure 5.14: Result after matching photographs directly to 3D scan using Path 2. *From left to right:* Original photographs, 3D segmentation images that were matched to photographs, the original photos and the 3D segmentation images overlaid after they have been aligned, and two different views of the 3D scan after registration is complete.

Image	GT Avg. RE	Avg. RE GT (Path 1)	Avg. RE GT (Path 2)	# Matches (Path 1)	Avg. RE (Path 1)	# Matches (Path 2)	Avg. RE (Path 2)
Fig. 5.5	3.69	10.80	10.05	85	2.18	2394	5.16
Fig. 5.14 Top	7.75	N/A	12.62	N/A	N/A	1917	6.00
Fig. 5.14 Bottom	6.24	N/A	6.32	N/A	N/A	2329	5.56
Fig. 5.15 a. i.	6.37	6.64	12.37	609	2.53	1225	5.61
Fig. 5.15 a. ii.	7.75	13.71	N/A	280	3.45	N/A	N/A
Fig. 5.15 b. i.	4.51	11.60	6.42	2167	3.64	9497	5.64
Fig. 5.15 b. ii.	6.61	10.94	8.23	2169	3.63	7745	5.26
Fig. 5.15 b. iii.	4.04	15.48	15.45	1004	2.86	1337	5.91
Fig. 5.16 a	4.15	10.75	11.09	41	1.88	377	5.28
Fig. 5.16 b	9.08	10.95	N/A	105	1.55	N/A	N/A
Fig. 5.16 c	5.05	8.57	N/A	121	1.75	N/A	N/A
Fig. 5.16 d	5.87	7.33	9.09	33	1.15	4528	5.79
Fig. 5.16 e	11.68	14.13	15.44	99	2.27	4142	5.70

Table 5.1: Quantitative results for our 2D-3D registration. The resolution of the LIDAR images is 1920x1920 and that of the regular photographs is 1632x1224. (GT stands for ground truth. RE stands for reprojection error in pixels. N/A shows where the Path did not find a correct alignment.)

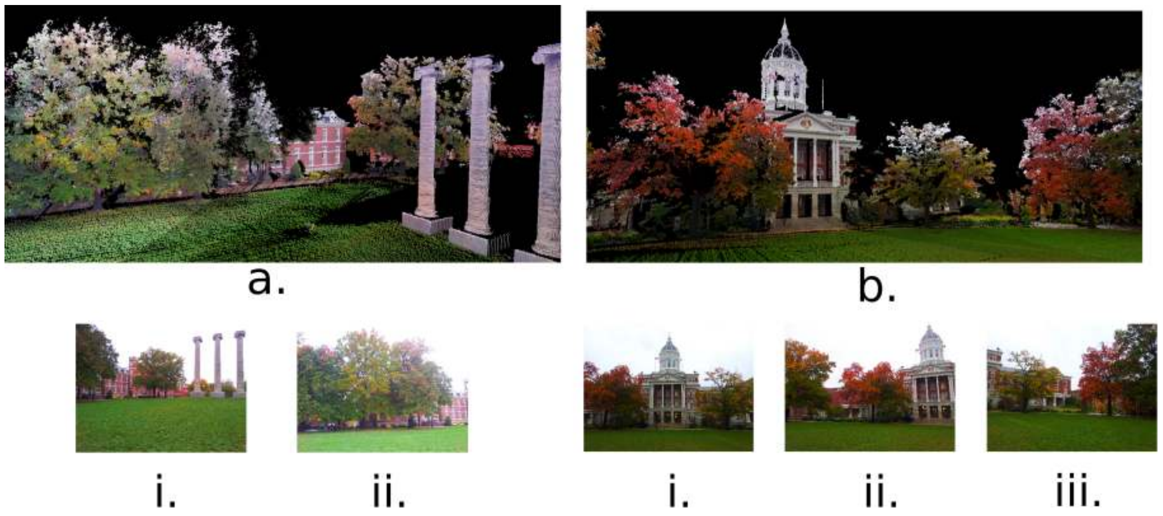


Figure 5.15: 2D-3D registration results using Path 1. Autumn photographs registered with summer scan. *Top:* 3D environment after registration. *Bottom:* 2D images being registered.

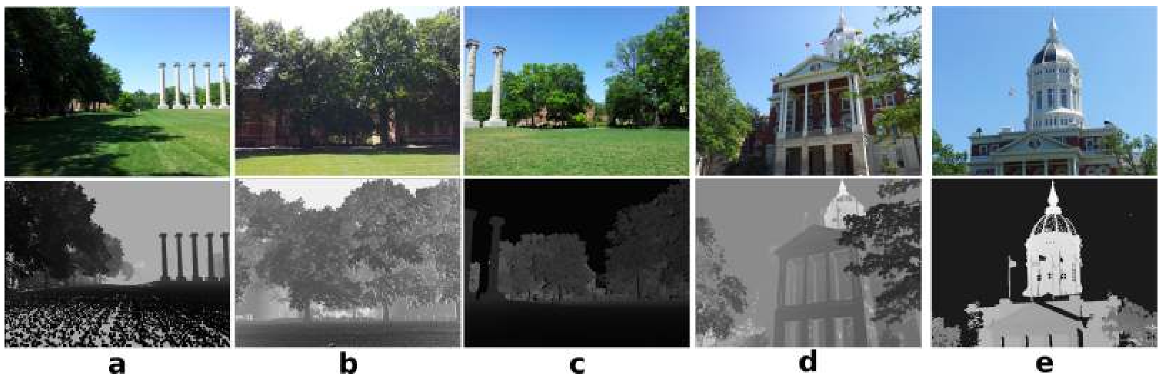


Figure 5.16: Depth images for photographs after they have been registered with the LIDAR scan.

These results show us that while both Paths 1 and 2 are robust to a number of challenges such as certain amounts of lighting, content, and perspective changes, each Path has its own strengths and weaknesses. Path 1 is able to register imagery very well, including photographs with heavy foliage and unstructured data, even under season changes where the content of the photographs and the LIDAR scan and photos vary very much visually. However, since Path 1 does not include any fully affine invariant feature descriptors, it does not perform well on photographs with very large perspective changes. Path 2, on the other hand, can handle these perspective changes better since synthetic views of the scan can be created from any location. However, since this Path relies on the peicewise planar segmentation of the scan, it will not work well for cameras that do not capture many regions labeled as buildings. In spite of the challenges encountered by these methods individually, our overall technique of contextual information to match the underlying structure across different images and modalities in both Paths of the pipeline allows us to perform accurate registration on a large variety of data.

The 2D imagery registered with the scan can be taken on the same day with the same camera or at different times, with different cameras, and/or with varying calibration parameters. Figure 5.17 demonstrates photographs taken months apart registered with the same scan. Two of the photos were taken during the summer and one was taken in the winter. The change in foliage between seasons is easy to view in this context. The underlying structure of the trees and their branches as well as the overall shapes of the trees when in full bloom are actually both visiable at the same time.

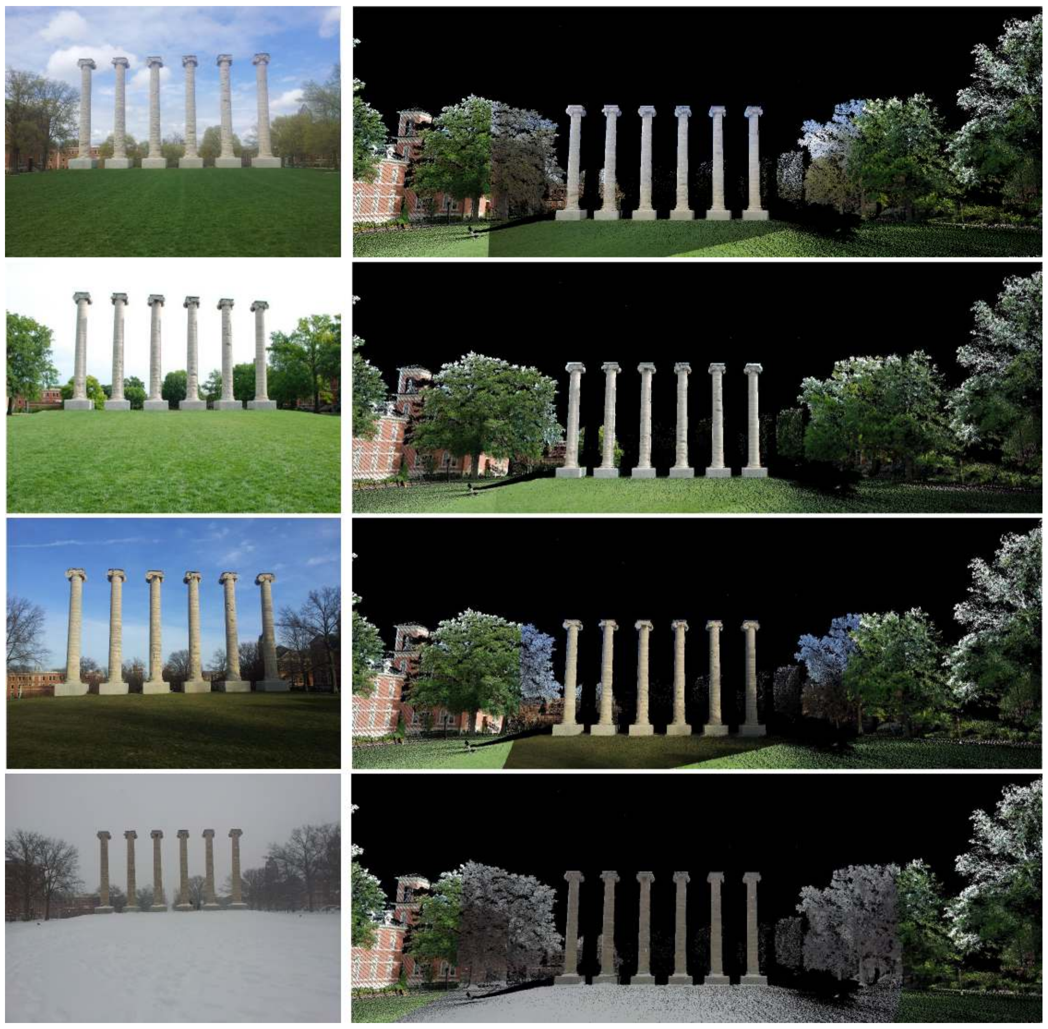


Figure 5.17: 2D-3D registration over four seasons. *Left Side*: Photographs from four different time periods of the Columns, a University of Missouri landmark. *Right Side*: Photographs projected onto range scan. Top to Bottom: Photos taken during 1) spring, 2) summer, 3) late fall, and 4) winter. Photos on left and right of the Columns in each row were taken during the summer.

Also, the photographs projected onto the scan need not be in their original state. The user can employ various sorts of image filters and non-photorealistic rendering techniques on the 2D data and project the modified images onto the scan instead. The matching step is still performed on the original data. The modified data is only used to determine the coloring for the final point cloud as is detailed in Algorithm 2. Employing this option can be useful for artists and simulation and game designers who may want to create non-realistic 3D models relatively quickly. Figure 5.18 shows various non-photorealistic renderings of photos and the result of projecting modified images onto range scans.

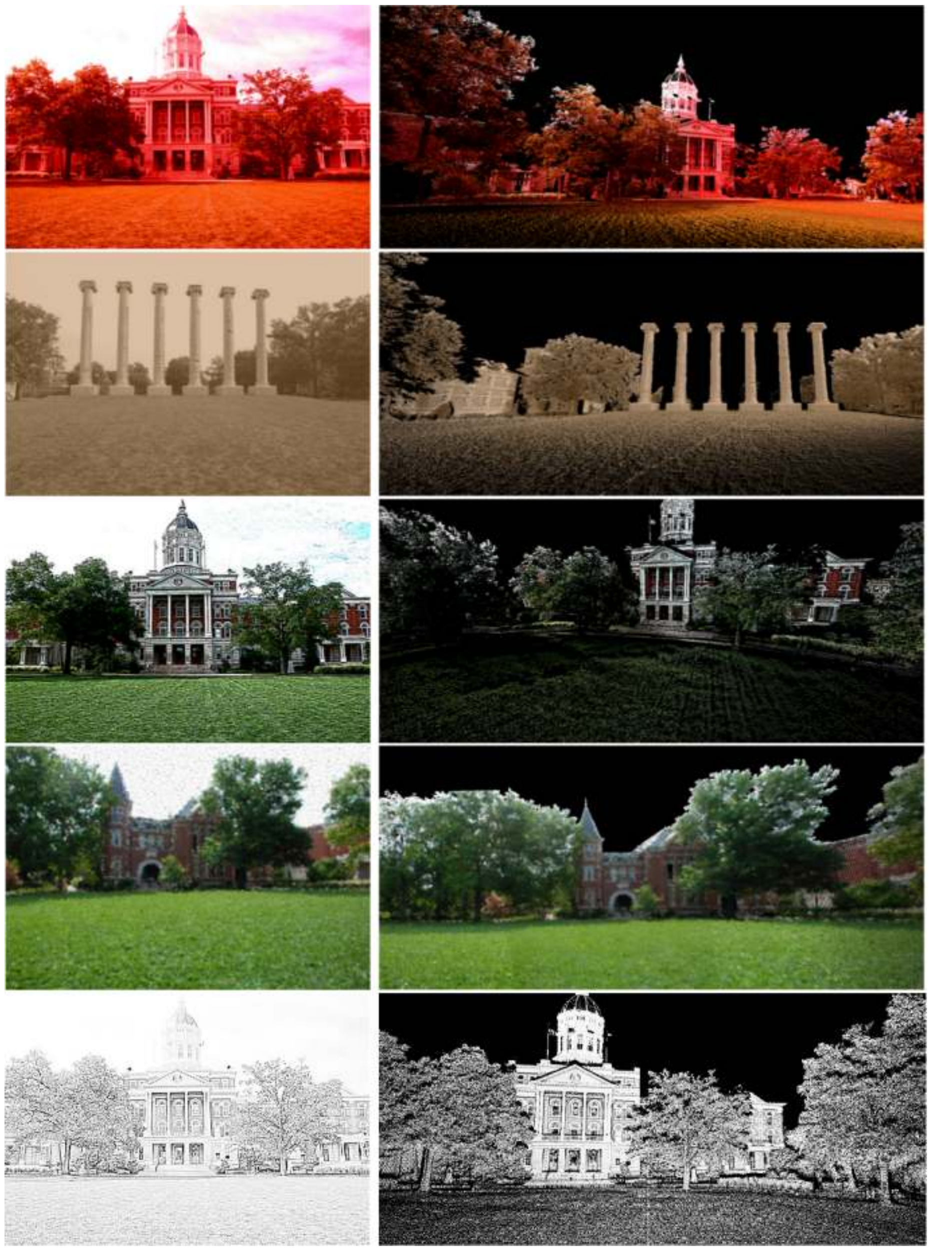


Figure 5.18: Five photos rendered non-photorealistically and projected onto scans. *Top to bottom:* 1) effect of changing color balance, 2) vintage photograph style, 3) cartoon style, 4) Cubist-era painting style, 5) outlines.

Chapter 6

3D Display

Once a set of images has been registered with the 3D scan, we can replace sections of the LIDAR point cloud that can be represented with planes with texture mapped 3D polygons as shown in Figure 6.1. We use high-resolution photographs to texture map these polygons, creating a smoother, more detailed model than the original point cloud. We have already segmented the LIDAR scan into a set of planes (Section 5.1) and aligned photographs with the scan (Section 5.3), so at this point we just need to determine which portions of photographs correspond to which 3D planes, break down the planes into sets of triangles, and texure map the triangles with sets of photographs. Our main steps for creating the final 3D planar model are described below:

1. Register image with scan and project 3D segments to 2D image plane (Section 6.1)

2. Fill in holes in segmentation image (Section 6.2)

3. Stitch together images of the same plane captured by different photographs (Section 6.4)

4. Break plane down into triangles for texture mapping (Section 6.5)

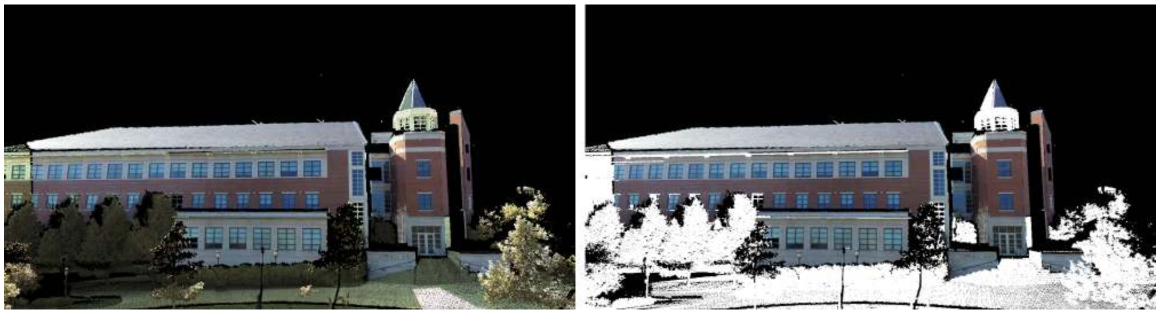


Figure 6.1: 3D texture mapped planes used to replace planar portions of the LIDAR point cloud. *Left:* Planes displayed with colored 3D point cloud. *Right:* Planes displayed with colorless 3D point cloud.

6.1 2D-3D Image Registration

Each photograph that will be used for the final 3D texture mapping result is registered with the scan using our standard procedure. Once the photo has been registered, we can get a 2D segmentation image by projecting all the 3D points down onto the 2D image plane and coloring each pixel according to what 3D planar segment was projected onto it. We get the 3D segments using the method described in Section 5.1. An example is shown in Figure 6.4. There are several peices of information we save when a photograph is registered that are used in later stages for the 3D texture mapping. These include:

- The 3D segmentation image (Figure 6.4).

- The photograph's 3D silhouette image (Figure 6.2). This masks all pixels that have a matching 3D LIDAR point. This information is used for hole filling in Section 6.2.

- A homography for each visible plane relating the 2D pixels that match to the 3D points that lay on the plane (Section 6.3).

- The relative angles between the camera's viewing angle and each visible plane's 3D normal. These relative angles are used to determine which photograph captures the most front facing view of a plane in Section 6.4.

- Two masked images for each plane that show only the pixels laying on the plane. One is a binary image where all pixels matching the plane are drawn in white and one is a color image that copies the color pixels from the original photograph (Figure 6.3).

- A list of 2D-3D matches for each plane. These matches will be used in Section 6.5 to determine the 3D location for the final texture mapped plane.

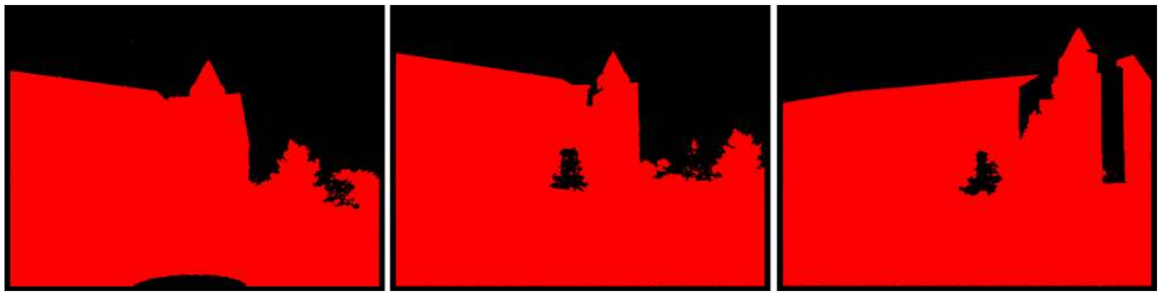


Figure 6.2: 3D silhouette images.

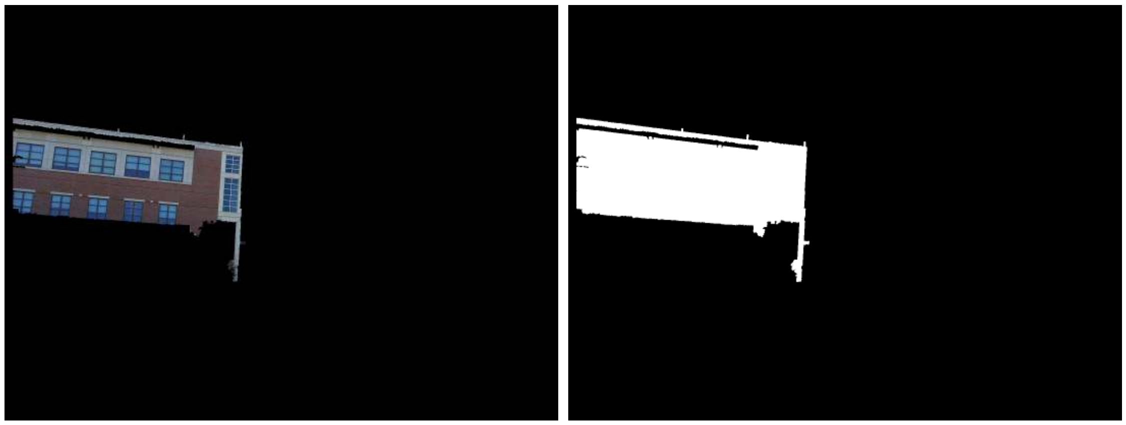


Figure 6.3: Color and binary masked images for individual planes.

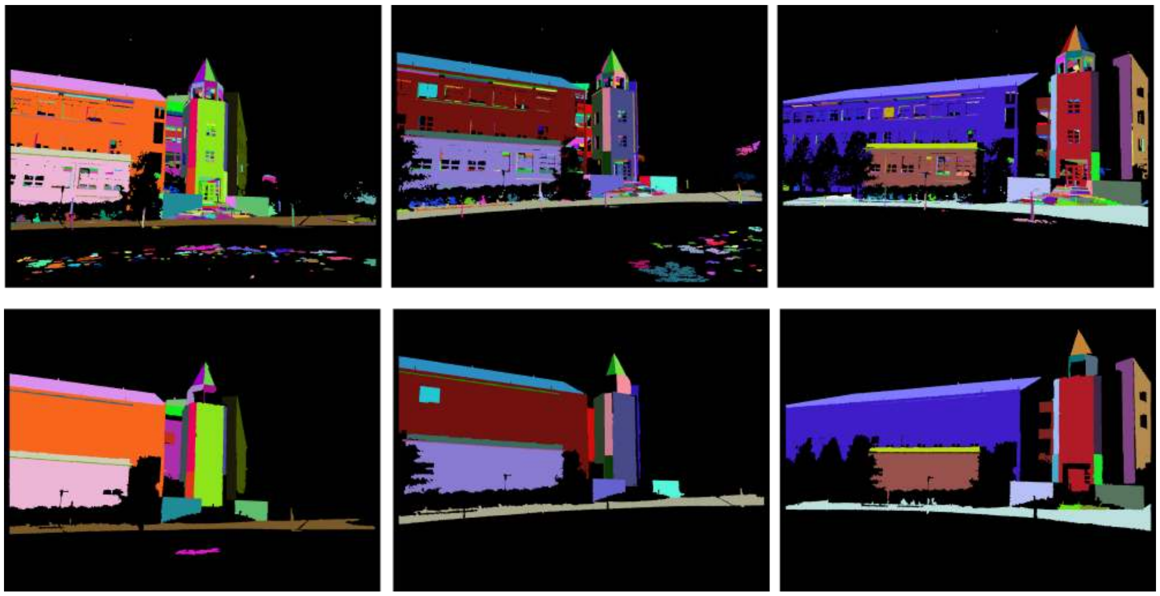


Figure 6.4: 3D segmentation images and our hole filling results.

6.2 Filling in Holes in 3D Segmentation Images

The 3D segmentation images contain holes where the point cloud has missing data and occlusions, where small parts of the point cloud are assigned to their own plane and later removed because they are considered to be noise, or where the data is incorrect. The main places the data is incorrect is where the depths assigned to windows are wrong due to the laser not being reflected as expected. We try to fill in all of these areas by assigning each component to a major plane that we are confident has been segmented and identified correctly. By filling in these holes, we can have more complete looking textures for our final 3D display.

To fill in holes, we identify connected components of empty areas in the segmentation image and merge each component with the plane that touches the majority of its perimeter. Figure 6.4 shows an example of a 3D segmentation image with the holes filled in using this approach.

6.3 2D-3D Homography Calculation

Once we have a set of matches between 2D pixels and 3D points that all lay on the same plane, we can compute a homography relating the two because the 3D points can be transformed so that they all have the same depth following a similar approach to Zhang's camera calibration work [122]. We transform the set of 3D points on the initial 3D segment (from Section 5.1's segmentation results) with a translation, t, to take the center of the plane to the origin, and with a rotation R that aligns the plane with the x-y axis. All the points on the plane have a depth (or z) value of nearly zero. Dropping all the z values, the 3D points can be represented with homogeneous coordinates $(x, y, 1)$ and a homography, H, can be calculated that relates the 2D pixels $(x_0, y_0, 1)$ to the 2D homogeneous plane points $(x, y, 1)$.

In Section 6.4, we will need the 3D points that match to particular 2D pixels to define texture coordinates. The 2D pixels are transformed to the 3D plane using H and then transformed back to the original LIDAR coordinate system using R^{-1} and $-t$.

6.4 Stitch Images of 3D Plane

To get the final image that will be used to texture map a 3D plane, we stitch together the masked images of the plane that we obtained from registering the original photographs with the LIDAR scan (Figures 6.5 and 6.6). These images are the 5th item mentioned in the list in Section 6.1. For each photograph containing the plane, we have a homography that relates its 2D pixels to the 3D plane points, as described in Section 6.3. For example, if we have three photographs that captured some image of the plane, we'll have three homographies, H_1, H_2, and H_3. We can use these homographies to transform all of the photograph's masks to one photograph's coordinate system. The set of masked images are transformed to the coordinate system of the photograph whose relative angle between the camera viewing angle and the 3D plane's normal is the lowest, indicating the most front facing view of the plane. Let's say in our example, photograph 2, with homography H_2, has the most front facing view of the plane. To transform masked image 1, with homography H_1, to photograph 2's coordinate system, we use Equations 6.1 and 6.2. They transform a pixel in image 1 (x_1, y_1) to the transformed 3D plane (X, Y, Z) and then transform the the 3D point to a pixel in image 2's coordinate system (x_2, y_2). λ is a constant that is used to normalize the 2D homogeneous coordinates.

$$H_1 * \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{6.1}$$

$$H_2^{-1} * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \lambda * \begin{pmatrix} x_2 \\ x_y \\ 1 \end{pmatrix} \tag{6.2}$$

As the masked images are stitched together, they need to be blended to avoid being left with visible seams along the image boundaries. We follow the multi-band blending pipeline described in the panorama stitching paper by Brown et. al [123]. To do this, we first create a Laplacian pyramid of each of the original photographs being used for stitching (Figure 6.7). Then we create binary masks for each image that mark which of its pixels will contribute the most to the final blended result. When more than one image has a pixel that is mapped to the same pixel in the stitched image, preference is given to the pixel from the most front facing photograph. Figure 6.5 shows a sample set of these masks. A Gaussian pyramid is constructed for each mask. The final blended result is created by combining images at the most blurred level and moving down the pyramid continuing to combine the images at each level. The final blended photograph is the sum of all of these combined levels. At each level, images are combined by summing the different photo's Laplacian images at the corresponding level of the pyramid. This is a weighted sum where the weight used for each pixel comes from the Gaussian pyramid of the masked images. Figure 6.8 demonstrates this process.

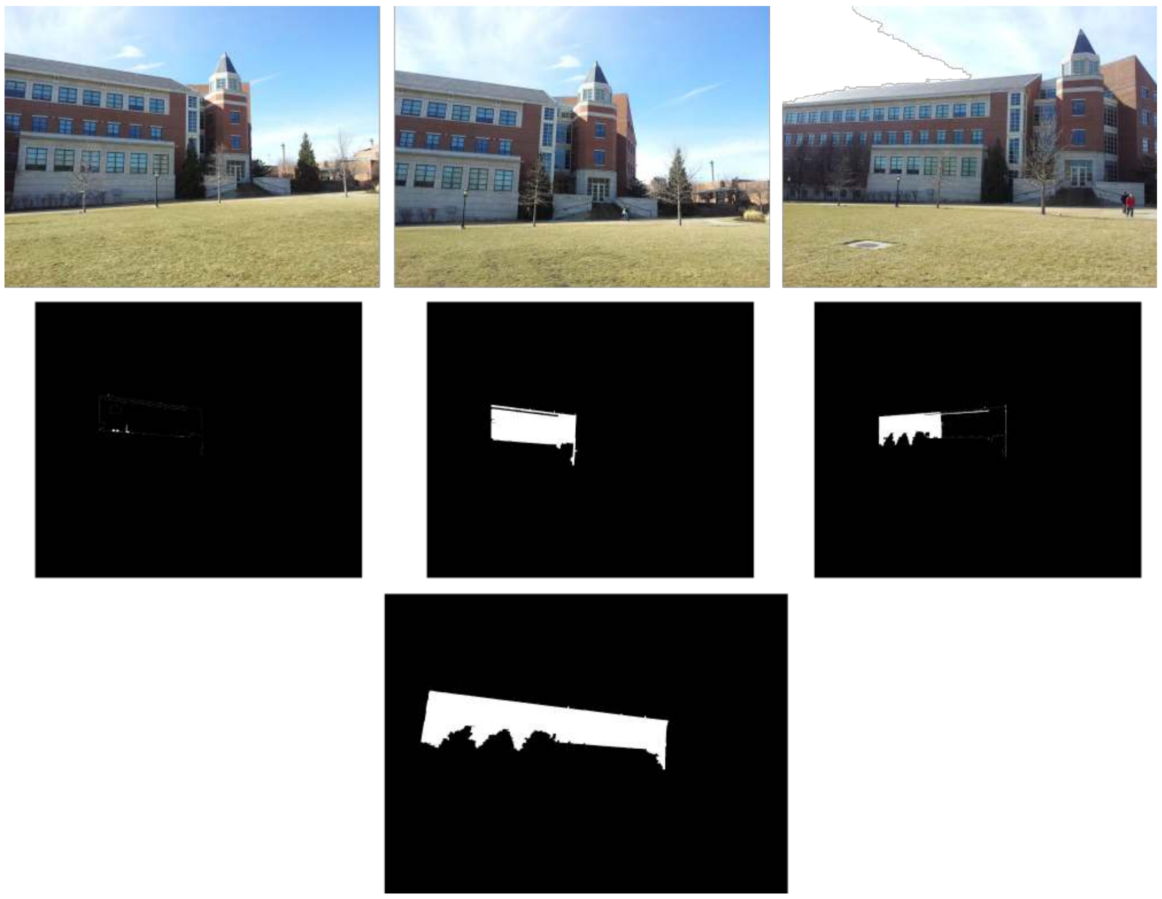


Figure 6.5: Three photographs that contain the same 3D plane. The second row shows the images masked by which pixels from which image have the most front facing normal. The bottom image shows all three masks stitched together using the three pixel to plane homographies.

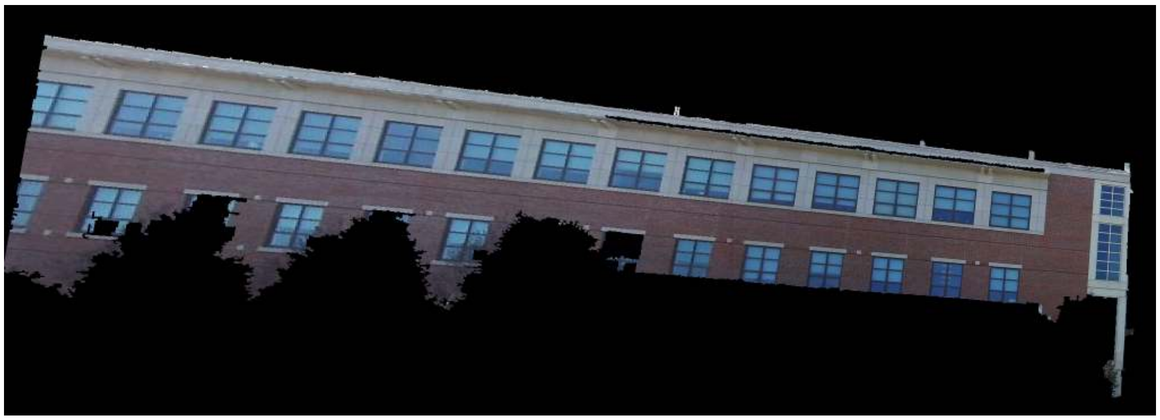


Figure 6.6: The final blended texture using all three images from Figure 6.5.

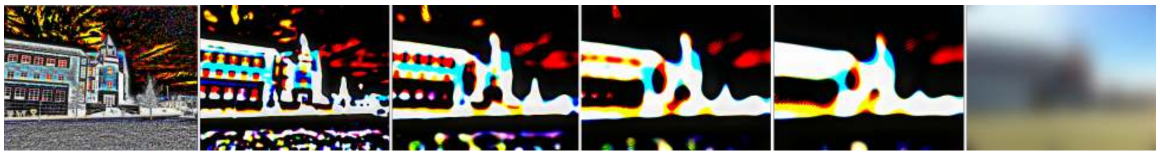


Figure 6.7: Laplacian pyramid of one image.

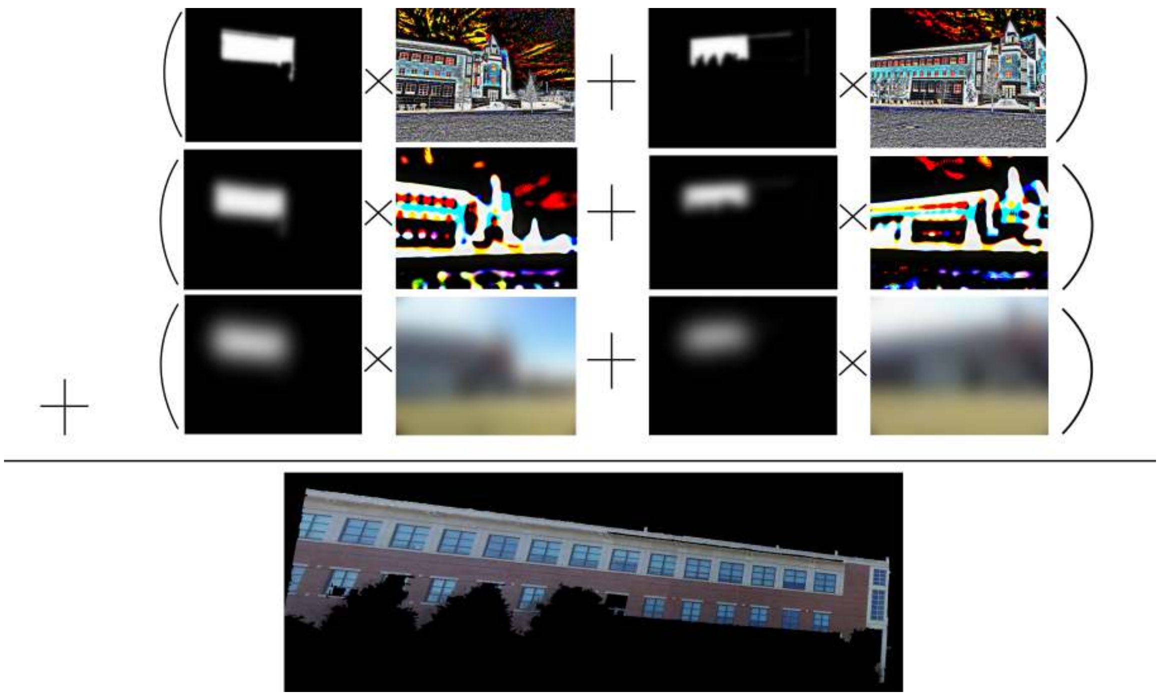


Figure 6.8: Using the Gaussian Pyramids of the image masks and the Laplacian Pyramids of the original photographs to perform multi-band blending.

6.5 Triangulate Plane and Compute 3D Texture Mapping Coordinates

For the final 3D texture mapped planes, we use a set of 3D triangles to display each plane. These triangles break down the entire visible plane into smaller segments. For each triangle we need the 2D image pixels and the matching 3D points on the plane for the three vertices on the triangle.

To triangulate the plane, we use Jonathan Shewchuk's program, "Triangle" [124]. Given a set of edges, this program can run different versions of Delaunay Triangulation and outputs a set of triangles as shown in Figure 6.10. To get the edges of the plane used during the triangulation, we run OpenCV's findContours function on a binary mask of the stitched plane texture to find the plane's outer boundary. We then use the Ramer-Douglas-Peucker algorithm to break the boundary down into a set of line segments. These line segments make up one set of edges that we input into the "Triangle" program.

Since these triangles only connect points along the plane's edges, some of them may still be a bit too big for OpenGL to texture map correctly. When really large segments are texture mapped in OpenGL, it breaks them down into smaller triangles and may not interpolate the texture correctly causing some visual inaccuracies in the final result. To avoid this, we just sample points through the interior of the plane and run region growing up to a set size region. If no boundaries are found within this region, we add new vertices and edges around the boundary to break up the triangulation. Figure 6.9 shows the plane boundary's Ramer-Douglas-Peucker line segments, the end points of each of the these line segments, as well at the vertices we insert through the interior of the plane.

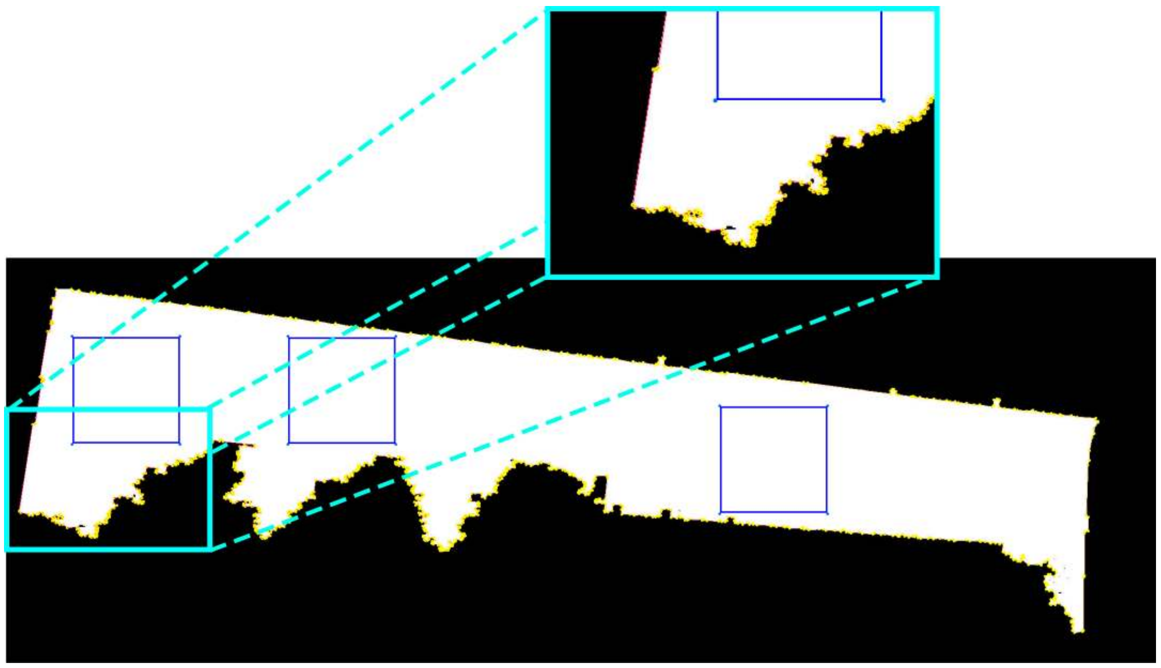


Figure 6.9: Edges and vertices put into "Triangle" for Delaunay Triangulation. The Ramer-Douglas-Peucker lines are drawn around the edge of the mask. The colors of these edges transition from yellow to red. The end points of the segments are shown with yellow circles. The vertices and edges we create through the plane's interior are shown in blue.

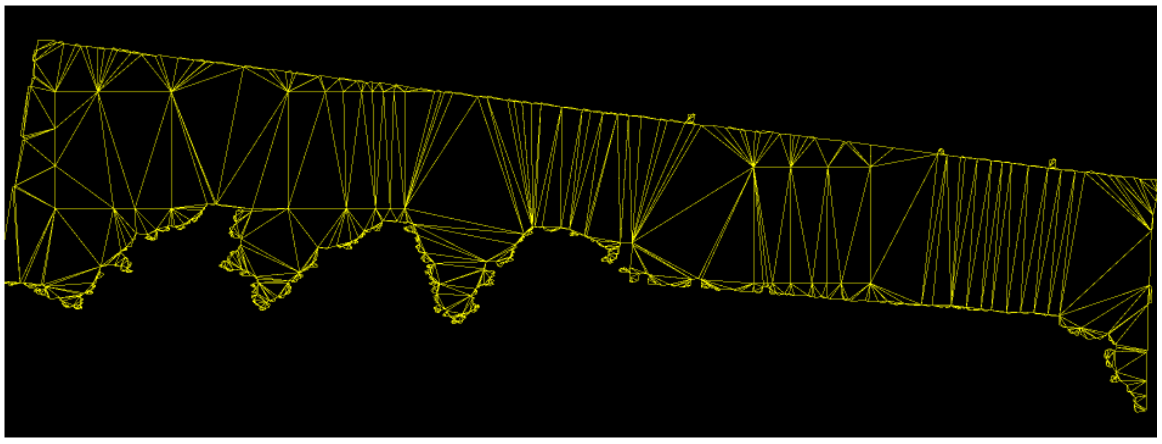


Figure 6.10: Result of Delaunay Triangulation. These are the triangles that will be texture mapped in 3D for our final display.

We find the 3-D coordinates for all of the triangle vertices using the process described in Section 6.3 and use the edge information from "Triangle" to draw sets of texture mapped 3D triangles in OpenGL to display the final results. Figure 6.11 shows some texture mapped planes and Figure 6.12 shows all the 3D triangles rendered with a solid color to visualize what areas have corresponding 3D planes drawn.

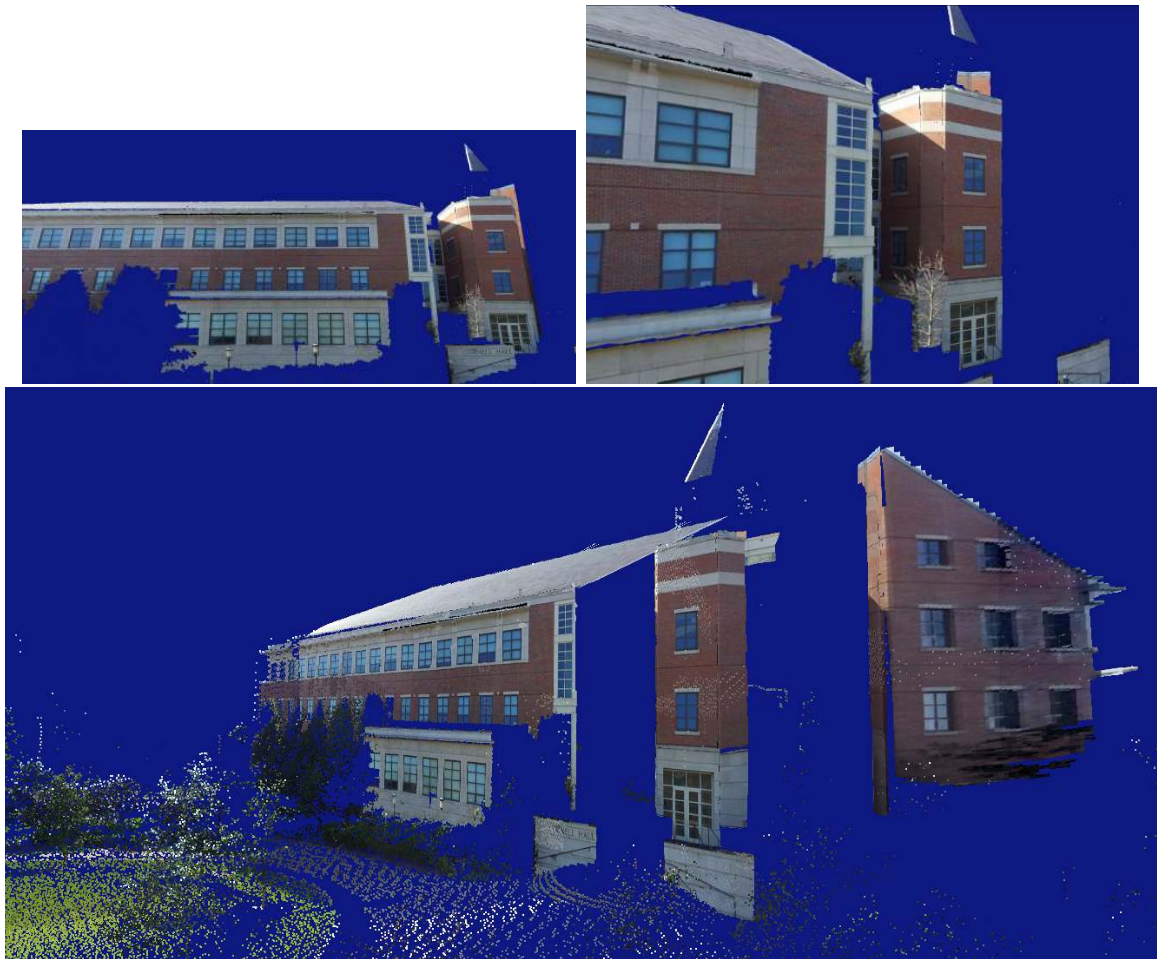


Figure 6.11: Final texture mapped planes shown from different perspectives.

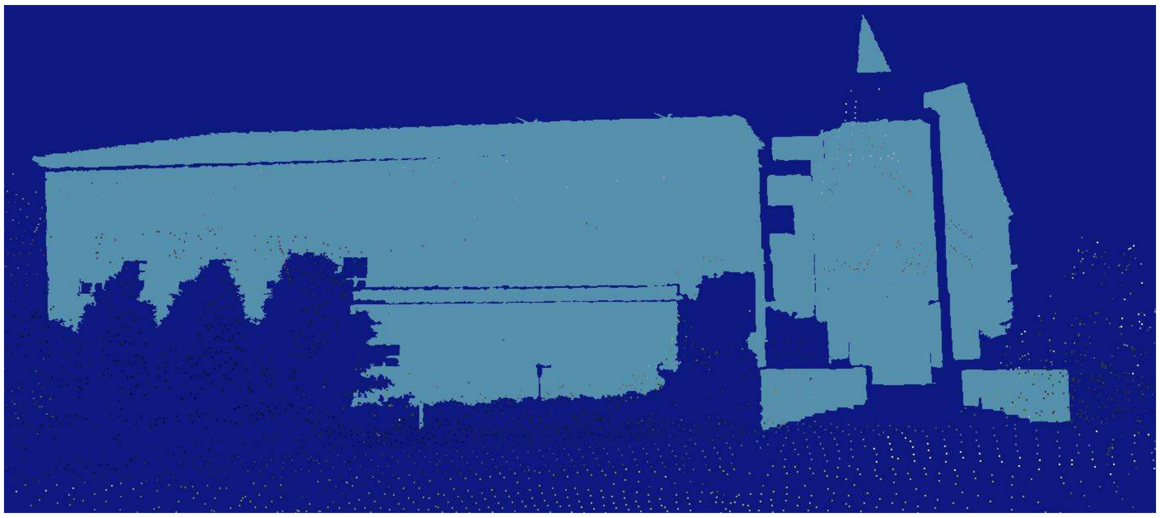


Figure 6.12: All 3D triangles displayed with solid color. This shows what areas of the point cloud are replaced with polygons.

6.6 Range Image

Another option for visualizing our fused 2D-3D registration results is to map an entire range scan onto a plane so that the full 360 degree scan is visible at once. 3D points, $(X,\ Y,\ Z)$, can be mapped to polar coordinates, (ρ, θ). 3D points are normalized so that each point has a magnitude of 1 using Equation 6.3.

$$\begin{aligned} x &= \frac{X}{mag} \\ y &= \frac{Y}{mag} \\ z &= \frac{Z}{mag} \end{aligned} \tag{6.3}$$

where

$$mag = \sqrt{X^2 + Y^2 + Z^2}. \tag{6.4}$$

The two dimensions of the plane that 3D points are projected onto represent 1) the projections onto the x-y plane and 2) the rotation along the z axis. To calculate the 2D coordinates, we use Equations 6.5 and 6.6.

$$\rho = arctan(y/x) \tag{6.5}$$

$$\theta = arcsin(z) \tag{6.6}$$

One important detail in creating a range image is determining the resolution of the scan. This should be set to the angular distance the scanner moves inbetween detecting point locations. The ρ and θ dimensions of the range image are discretized into $360/resolution$ and $90/resolution$ bins respectively. The 2D coordinate is divided by these bin sizes and rounded to the nearest integer to find the pixel coordinate corresponding to the 3D point.

Using this approach, all points will be projected onto one half of the range image. To handle this problem, we essentially divide the scan into half and project all 3D points with positive X values onto the left half of the image and shift all 3D points with negative x values onto the right half of the image.

We incorporate range images into our interface for viewing and interacting with our fused 3D models. The range image is displayed at the bottom of the interface, allowing the user to see the whole scan in one glance. We know from our mapping process what 3D points in the scan correspond to each of the pixels in the range image. The user can click anywhere on the range image and the 3D view of the fused model will be changed so that the virtual camera is looking at the selected point. A sample range image is shown in Figure 6.13.

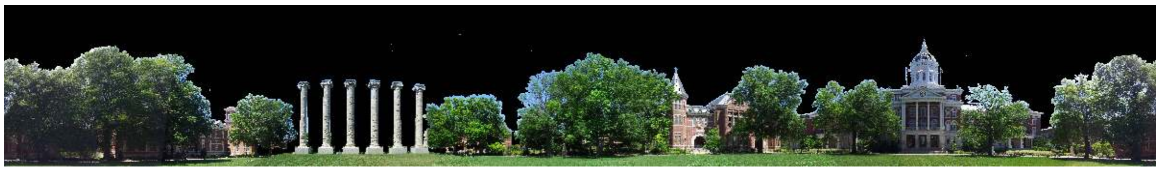


Figure 6.13: Range image from a LIDAR scan of Francis Quadrangle.

Chapter 7

Applications

Using our 2D-3D registration pipeline, we can expand our work into many exciting applications in the fields of graphics, computer vision, and visualization. These include rendering dynamic objects captured in 2D videos in our 3D environment, 3D video authoring, creating 3D stereo videos, taking geological measurements, and creating virtualizations of disaster scenarios.

7.1 Displaying Videos in 3D Environments

Video frames can be registered with a range scan using the approach described in Chapter 5. However, when an object that was not scanned is present in a video, such as a person walking around, it will be projected onto an incorrect location in the 3D space because there is no structure that corresponds to it. Though the visual result of an image's registration may look fine when the scene is viewed from the camera location, these errors are very apparent when the user starts changing perspectives as is demonstrated in Figure 7.1 Bottom Left. In order to handle many of these cases, we propose segmenting the motion in videos and adding 3D planes to the virtual environment to “catch” the projection of these new entities. We present a simple but effective approach here for modeling moving objects that are touching the ground and recorded by a single, stationary camera.

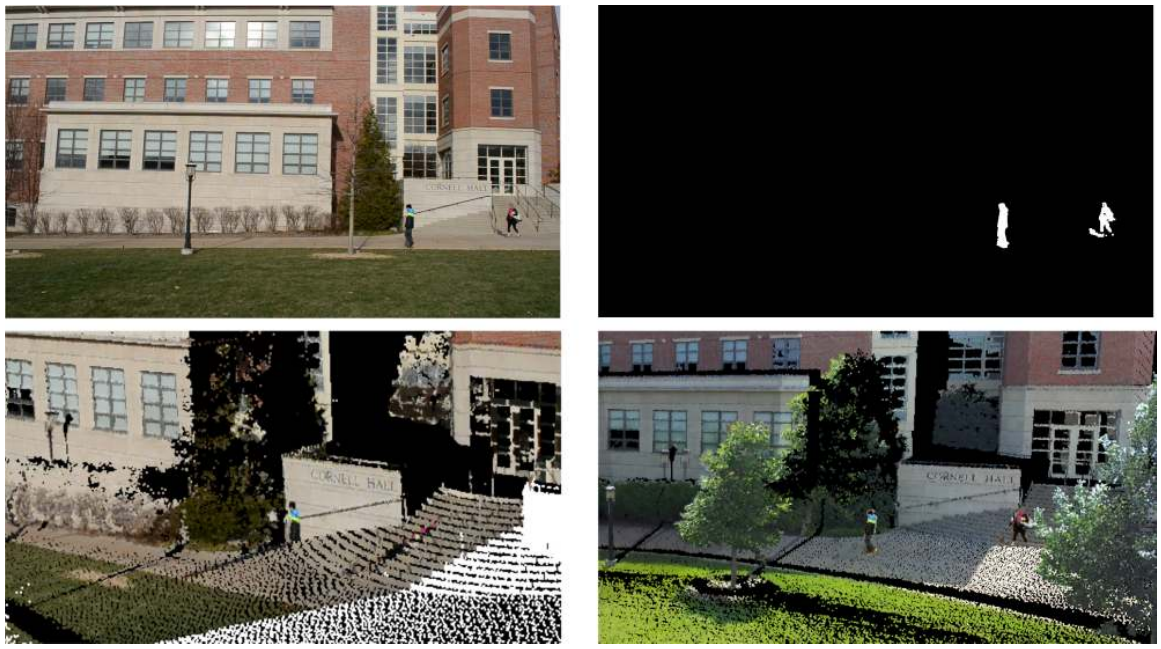


Figure 7.1: Displaying videos in 3D environments. *Top Left:* Original video frame. *Top Right:* Mask showing dynamic foreground (people) captured in video. *Bottom Left:* Video frame projected onto range scan without using our method for modeling moving objects. Notice how people are projected incorrectly onto the ground and stairs. *Bottom Right:* 3D planes constructed for moving objects identified in video using our modeling method.

In order to identify moving objects in the video stream, we use the Mixture of Gaussians (MOG) algorithm [125], though other background segmentation algorithms could be used in its place such as [126]. This yields a binary image with the motion segmented from the background as shown in Figure 7.1 Top Right. The connected components algorithm is applied to the MOG image to create cohesive segments. We scan this image starting from the bottom row of pixels to find the lowest points in each moving segment. We then identify the 3D points in the range scan that match to these low points when the video frame is registered with the range scan. Assuming that the moving object is touching the ground, these 3D points are the correct locations for the bottom of the segmented objects. New 3D points with the same depth as the bottom points and varying heights are created and projected onto the MOG image. If they fall within the segmented portion of the image, they correspond to a moving object that was not scanned and are added into the 3D space with the corresponding color information from the original video frame. The result of performing these steps can be viewed in Figure 7.1 Bottom Right. Figure 7.2 shows more examples of our video to LIDAR registration results.

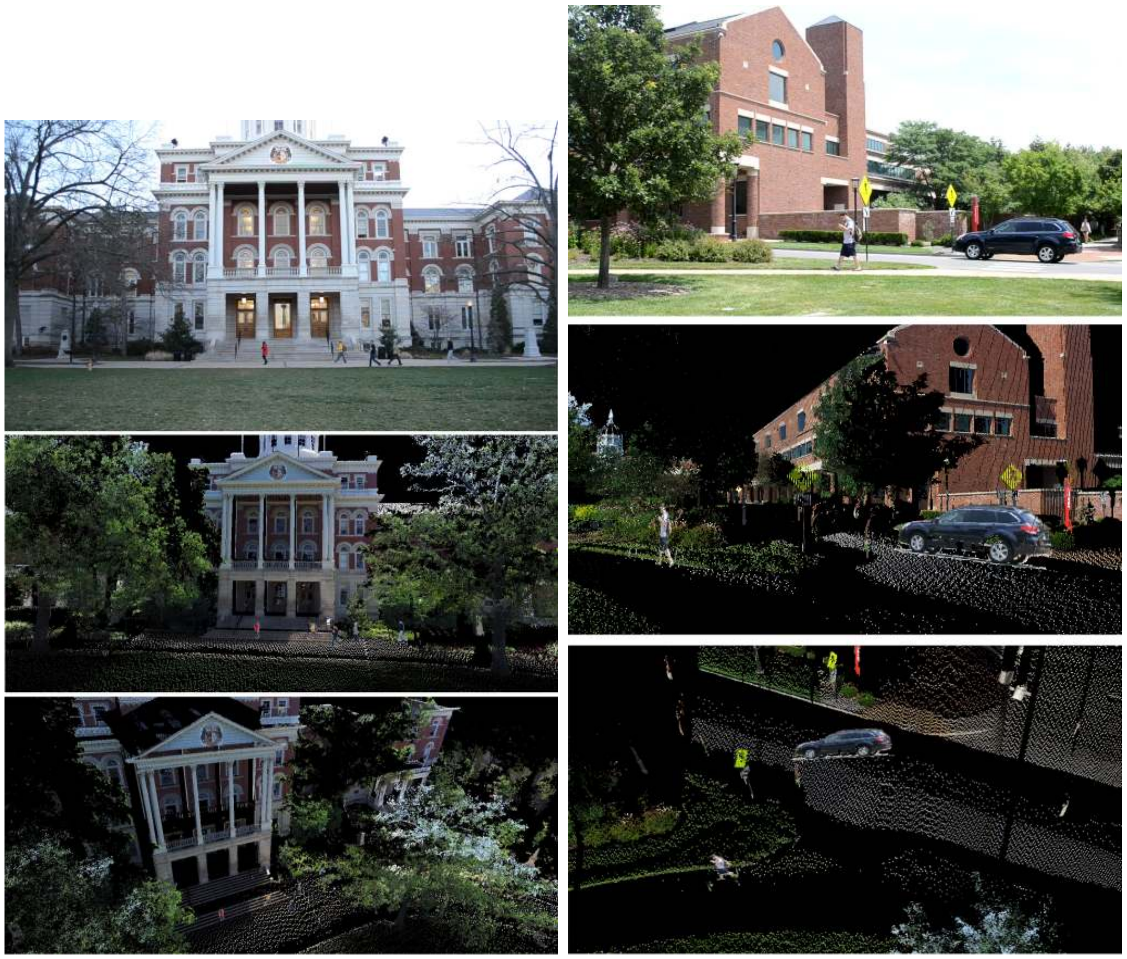


Figure 7.2: Additional results of displaying videos in 3D environments. *Top row:* Original video frames. *Bottom rows:* Resulting 3D environment after registration shown from multiple perspectives.

Figure 7.3 demonstrates how the new 3D points are calculated. They are chosen along a plane that is parallel to the image plane corresponding to the video, the orientation of which can be determined from the projection matrix. To find this plane, we look to the principal plane which is parallel to the image plane and goes through the camera center. Any 3D point lying on this plane will have a depth of 0, as is shown in Equation 7.1.

$$P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \tag{7.1}$$

For this to hold true, 3D points on the principal plane multiplied by the third row of the projection matrix will always yield a result of 0. We can describe the principal plane using the normalized vector representing the projection matrix's third row, n. The angles (θ_x, θ_y) between this vector and the x and y axes are given in Equation 7.2.

$$\begin{aligned} \theta_x &= arccos(n_x) \\ \theta_y &= arccos(n_y) \end{aligned} \tag{7.2}$$

The x and y coordinates of the new 3D points new_x, new_y are created using Equation 7.3. *new* is the 3D point, *bot* is the determined 3D location of the motion blob, l represents the number of steps taken along the plane so far, and *stepSize* is the distance between steps. The distance between the 3D points corresponding to the left and right most pixels of the motion blob dictates how far along the plane from *bot* new points are created.

$$\begin{aligned} new_x &= bot_x + l * stepSize * cos(\theta_x) \\ new_y &= bot_y + l * stepSize * cos(\theta_y) \end{aligned} \tag{7.3}$$

An important aspect of this method is having a depth value for each pixel in the registered video frame. If there is no information for pixels along the "bottom" of the motion blob, a 3D location for the object cannot be determined. Unfortunately,

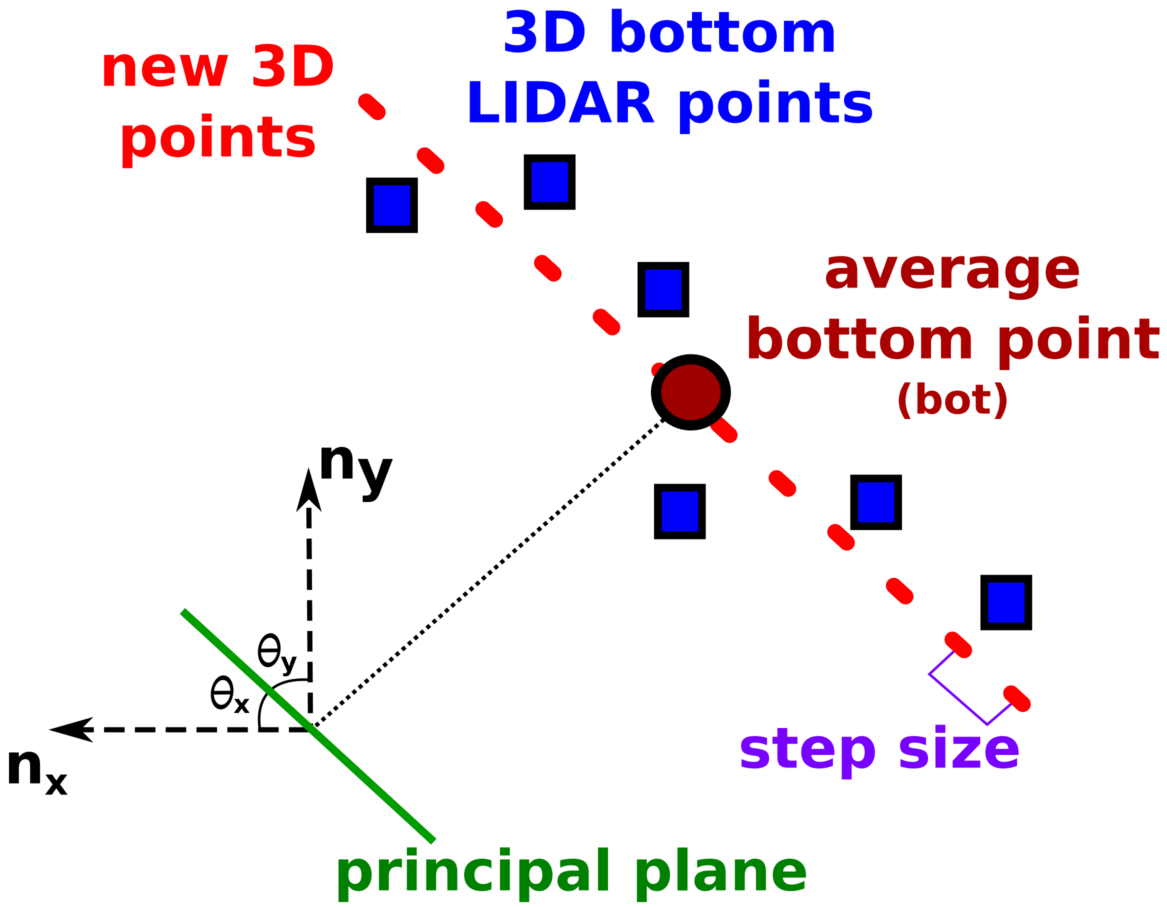


Figure 7.3: Diagram showing how new 3D points for videoed objects are calculated. Notation refers to that explained in Equations 7.2 and 7.3.

projecting each LIDAR point onto a video frame does not guarantee that every pixel will have a matching 3D point. The density of the scan may be low enough that there are noticeable gaps between 3D points. There may also be occlusions between the scanner and the ground area where the new object is moving, caused by trees, light posts, and elevation changes along the ground. To ensure that a video frame has continuous depth information, we include a hole filling step after registration. For each pixel in a "hole," the neighboring pixels with known 3D correspondences are used to determine the missing depth data. A 20x20 neighborhood is averaged using a linear filter with points closest to the pixel of interest given higher weights than those closer to the neighborhood boundaries. Doing so produced satisfactory results in our experiments, but more sophisticated methods may be employed such as using inpainting with depth gradients [93].

We also want to ensure that consistent bottom points are chosen between frames and that the 3D trajectory of the motion blob is smooth. For every frame, the lowest 10 pixels in each segment are stored. The 3D locations corresponding to these pixels are averaged after removing any outliers from the set. The averaged 3D point is used as the bottom point and the basis for the new 3D plane. To handle potential noise in the 3D data and to avoid the object jumping too far from the smooth trajectory during individual frames, we also include a prediction step for where we believe the 3D object should be. A line is fit through the bottom locations of the previous n frames ($n = 10$ in our experiments) using the least-squares method. The predicted location is set to the speed of the object over prior frames multiplied by the direction of the fit line. If the predicted location differs too greatly from the actual value computed for the frame, the predicted value is used.

7.2 3D Video Authoring

We can also insert dynamic virtual objects into the 3D scene that either follow the trajectory of an object/person captured on video (see Section 7.1) or a user-defined path. For the former case, the object is placed using the same bottom point found for the motion segment in each video frame. If the user instead wants to specify a path, he or she can define control points on the 3D point cloud outlining the shape of the route. When the user clicks on the point cloud, we use the OpenGL function "gluUnProject" to identify the chosen 3D coordinate. The user can then manipulate the x, y, and z values of each control point to modify the path. A B-spline is fit to the final set of points and the virtual object is displayed moving along this curve. The B-spline, P, is calculated using Equations 7.4 and 7.5, where c_k are the set of user-defined control points and $B_{k,d}$ are polynomials that are summed together to construct the final curve. u is the paramaterized range of the B-spline and d is the degree parameter, or the number of control points that influence a section of the curve between two control points [127].

$$\begin{gathered} P(u) = \sum_{k=0}^{n} c_k B_{k,d}(u), \\ u_{min} \leq u \leq u_{max} \\ 2 \leq d \leq n+1 \end{gathered} \tag{7.4}$$

$$
\begin{aligned}
B_{k,1}(u) &= \begin{cases} 1 & \text{if } u_k \le u \le u_{k+1} \\ 0 & \text{otherwise} \end{cases} \\
B_{k,d}(u) &= \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) \\
&\quad + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u)
\end{aligned} \tag{7.5}
$$

One additional step is required in placing the virtual object. We want the object to be oriented such that it is facing the direction in which it is moving. To accomplish this, we first assign an orientation for each frame by fitting a line through the object positions of the neighboring 50 frames (about 2 seconds worth of video). The orientation is set to be the angle of the line along the x-y plane. We include an averaging step to ensure that the object's orientation does not change drastically between frames. Figures 7.4 and 7.5 show several frames of virtual objects moving along user-specified paths. Figure 7.6 shows a virtual car following the trajectory of a walking person that has been captured on video. These examples are rendered in OpenGL using an environment that the user is free to navigate and study from different perspectives.

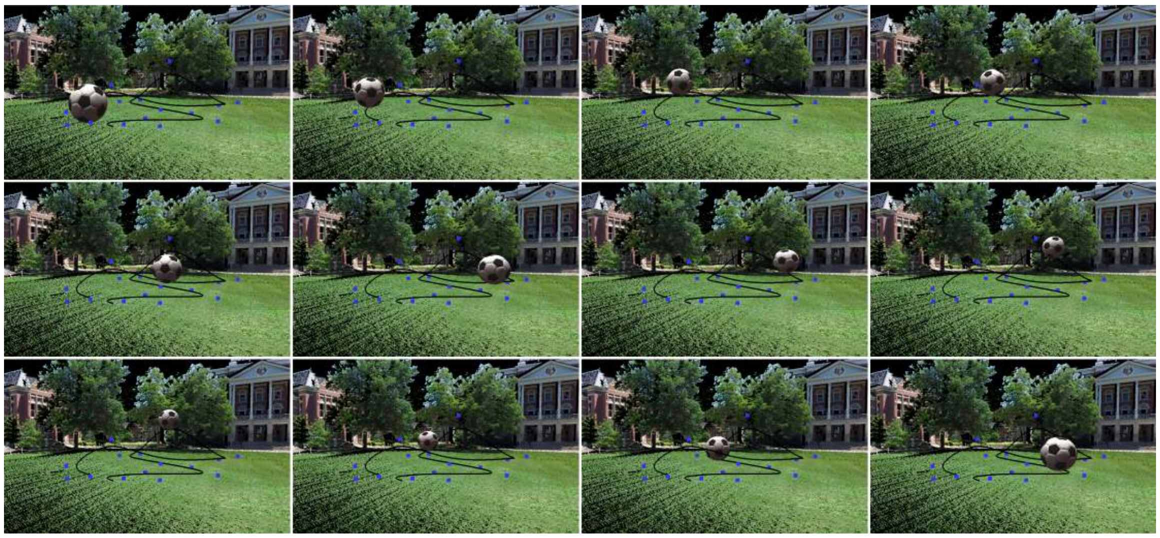


Figure 7.4: Inserted soccerball moving along user-defined path. Blue points represent user-specified control points. Black curve shows the B-spline curve.

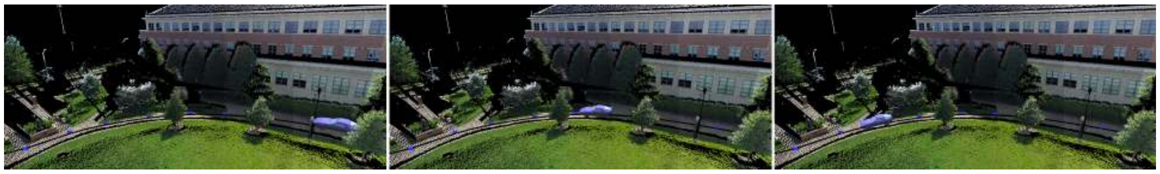


Figure 7.5: Inserted car moving along user-defined path. Blue points represent user-specified control points. Black curve shows the B-spline curve.

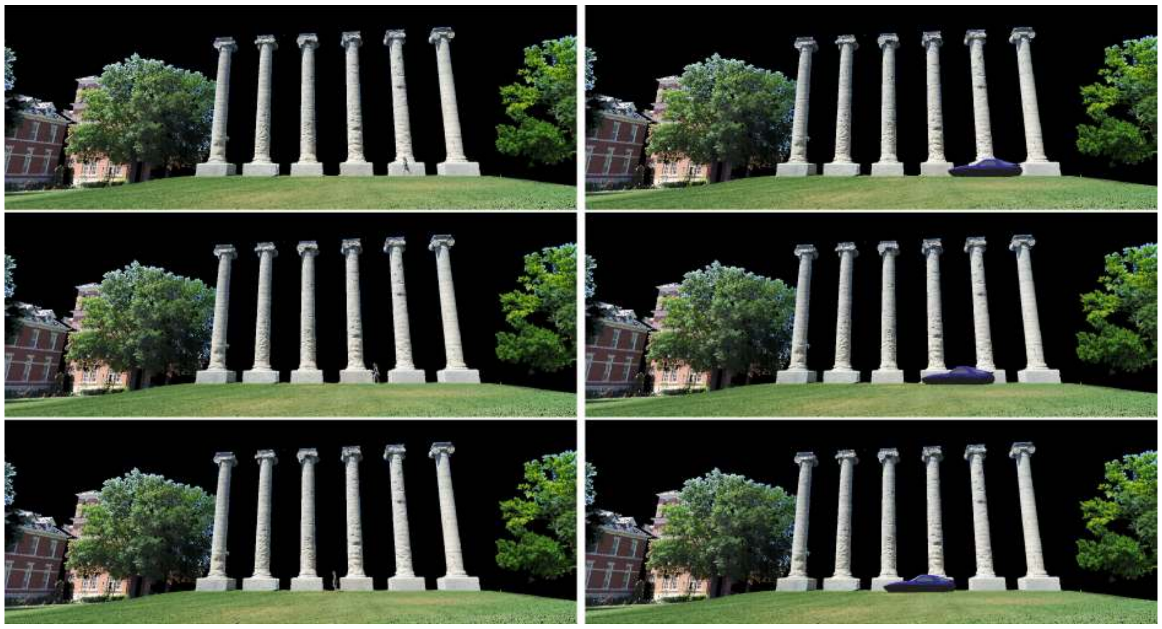


Figure 7.6: Dynanmic virtual object. *Left:* Three frames of walking person from registered video shown in the 3D environment. *Right:* Virtual object replacing moving person from video.

In Figure 7.4, a ball is displayed rolling along the B-spline path. To achieve this effect, the orientation of the ball is updated at every frame so that it is rotating about an axis perpendicular to direction of motion, and along the x-y plane. The amount of rotation corresponds to the distance the ball moves in space between frames using $\theta = \frac{S}{r}$. θ is the angle the ball rotates, S is the distance the ball moves in space, and r is the radius of the ball.

7.3 Creating 3D Stereo Videos

Creating 3D videos requires only a few extra steps after a video has been registered with a scan. The depth information for every pixel in a video frame is known at the end of our 2D-3D fusion. This information can be used to create anaglyph images as shown in Figure 7.7 using a technique such as that discussed in [128].

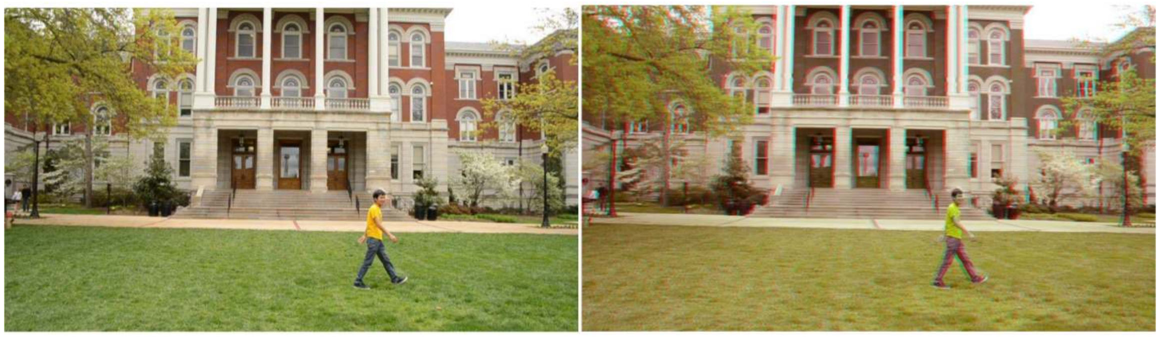


Figure 7.7: Creating stereo video from 2D-3D Registration. *Left:* Original video frame. *Right:* After video is registered with range scan, the 3D location of video pixels are known. Images for the left and right eyes can be created and combined to create an anaglyph image. (Best viewed with 3D anaglyph glasses.)

7.4 Virtual Geological Engineering Measurements

In mountainous and hilly regions, roadways commonly pass alongside and between tall walls of rocks that sit very close to the road. This poses many potential obstacles and dangers for drivers, road workers, and engineers who travel through these areas or who are responsible for building and maintaining the road infrastructure. Portions of rocks can detach from the walls and fall or slide down into the roadways, blocking passages or injuring or killing motorists. This type of geological behavior is influenced by discontinuities in the rocks [129] which oftentimes cause rock mass to break off along planar cracks that occur either naturally or as a result of engineered rock cutting during the road construction process.

By using analytical tools, the arrangement and orientations of single cracks or groups of cracks can actually be used to study rock stability and predict detachment events [130, 131]. Rock cracks or discontinuities tend to be naturally clustered in terms of their orientations, typically into three or more sets, which are often mutually orthogonal. A very popular approach for analyzing these properties is the stereonet projection method [132] which assigns each data point, consisting of a normal vector in relation to an individual discontinuity plane, to a discontinuity set using cluster analysis [133, 134, 135, 136, 137]. The rock orientations used in such methods have traditionally been measured manually using compass and clinometer methods. However, these methods are slow, tedious and cumbersome, and, in some cases, dangerous because of potentially falling rock [138]. They are often only able to be employed in easily accessible locations like the base of a slope without the added danger of accessing a high, steep, rocky area to take thorough measurements. They may also be inaccurate due to the sampling biases that occur when measurements are restricted to accessible areas only [139, 140].

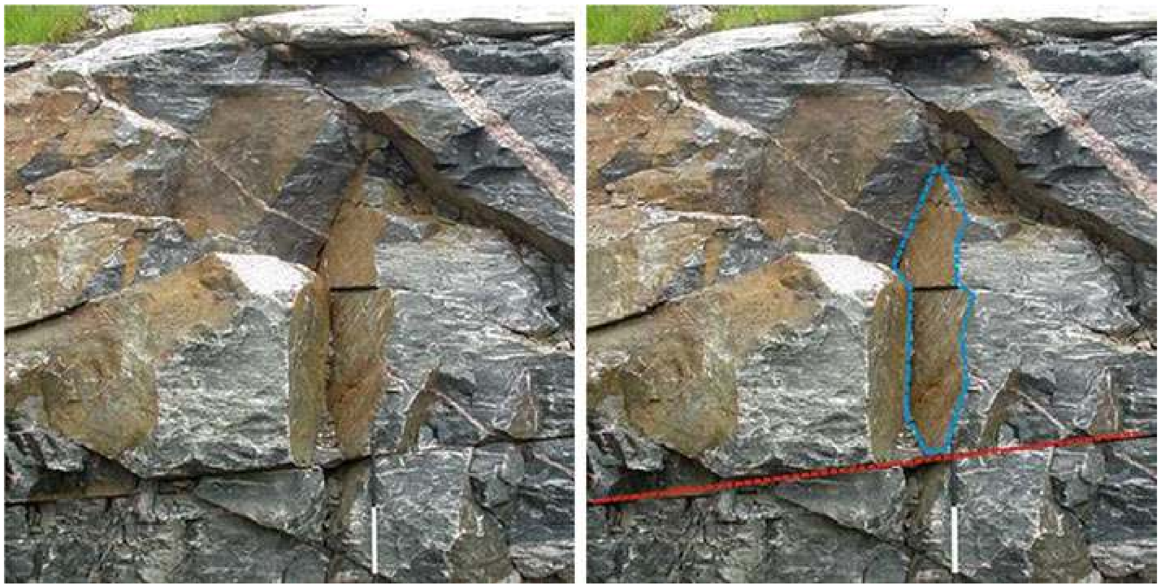


Figure 7.8: Rock cut containing both fracture traces (red line) and facets (cyan polygon) [13].

To ensure public safety, engineering projects in rocky terrains need to be carried out in such a way that rock behavior is predictable in order to increase the odds of rocky walls remaining stable, especially after blasting techniques are used during the highway construction process. The challenges in obtaining the needed measurements and limits on construction budgets can make achieving this goal difficult. Instead, photographs and LIDAR scans of rocky terrains can be used to obtain these measurements more quickly and safely than traditional techniques, and at a lower cost and with greater accuracy. Using digital data in this manner is demonstrated in the work of Kemeny et al. [131] who explored studying rock fracture properties using digitial photographs that have a known relationship to rock faces to estimate 3D measurements. Terrestrial LIDAR scanners provide highly accurate 3D point clouds of surfaces that can be used to extract geometric information of planar regions which are 3-D expressions of discontinuities. Using both 2D and 3D data in this context has been explored, but when used individually the full advantages of using these mediums for measurement purposes can not be exploited. The cracks or discontinuities in the rock manifest themselves either as fracture traces (a linear element) on a planar rock cut or as fracture surfaces or facets on an irregular rock cut, examples of which are shown in Figure 7.8. Measurements on optical images can yield apparent 2D orientations of the trace on the sampling plane and measurements made on LIDAR scans can be used to determine 3D orientations of fractures on irregular surfaces [141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 13, 152, 153, 138, 154]. Planar polygonal models can be generated from LIDAR scans yielding 3D orientations of rock faces. This approach can be complicated by the presence of vegetation or blasting and weathering-induced fractures which need to be identified by sophisticated filters or with the aid of human interaction. These algorithms automatically extract planar features (discontinuities) from the scan, calculate orientations, cluster the orientations, and present them on a stereonet. Typically, LIDAR methods require survey control to tie the LIDAR to the true north grid. The drawbacks of these approaches are that they require continuously exposed rock throughout the scanned images and sophisticated algorithms that are frequently incapable of distinguishing between discontinuity-like joints as opposed to other planar elements such as blast-induced fractures or erosional surfaces.

Despite the abundance of information that can be extracted from 2D or 3D data separately, neither technology, on its own, is sufficient for making measurements on both planar and irregular surfaces. Since it is common for rocks to contain discontinuities on both types of surfaces, we have decided to use our registration work to aid the measurement process by fusing photographs and LIDAR scans to perform a single analysis using both types of data. Using computer vision techniques, we are able to register photographs directly with 3D point clouds so that linear trace measurements that are easily taken on a 2D photograph automatically contain information about 3D facet orientations based on the 3D point cloud's normals.

With the the guidance of arbitrarily oriented line segments, our registration algorithm is able focus on matching rocky areas and not the unstable vegetation that may have been captured in the imagery. This section describes how our registration methodology can be successfully applied to imagery of rocky terrain as well as overviews our developed software, LIDAR Fusion, which allows engineers to interactively explore the 2D and 3D data and to select 2D trace and 3D facet measurements.

7.4.1 2D-3D Registration

We can apply our registration techniques discussed in Sections 5.1 through 5.3 to natural imagery of rocky walls as shown in Figure 7.9. Our segmentation and semantic labeling techniques can be applied in the same manner used on our typical datasets of buildings and foliage. The only difference in this context is that instead of labeling identified planes as buildings, we label them as rock faces. An example of applying our segmentation and labeling process to rock images is shown in Figure 7.10.

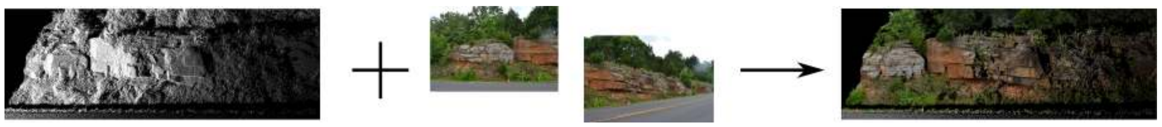


Figure 7.9: 2D-3D registration for rock imagery. Our registration pipeline can be used to fuse 3D LIDAR scans (*Left*) and 2D photographs (*Center*) with natural subjects together (*Right*).

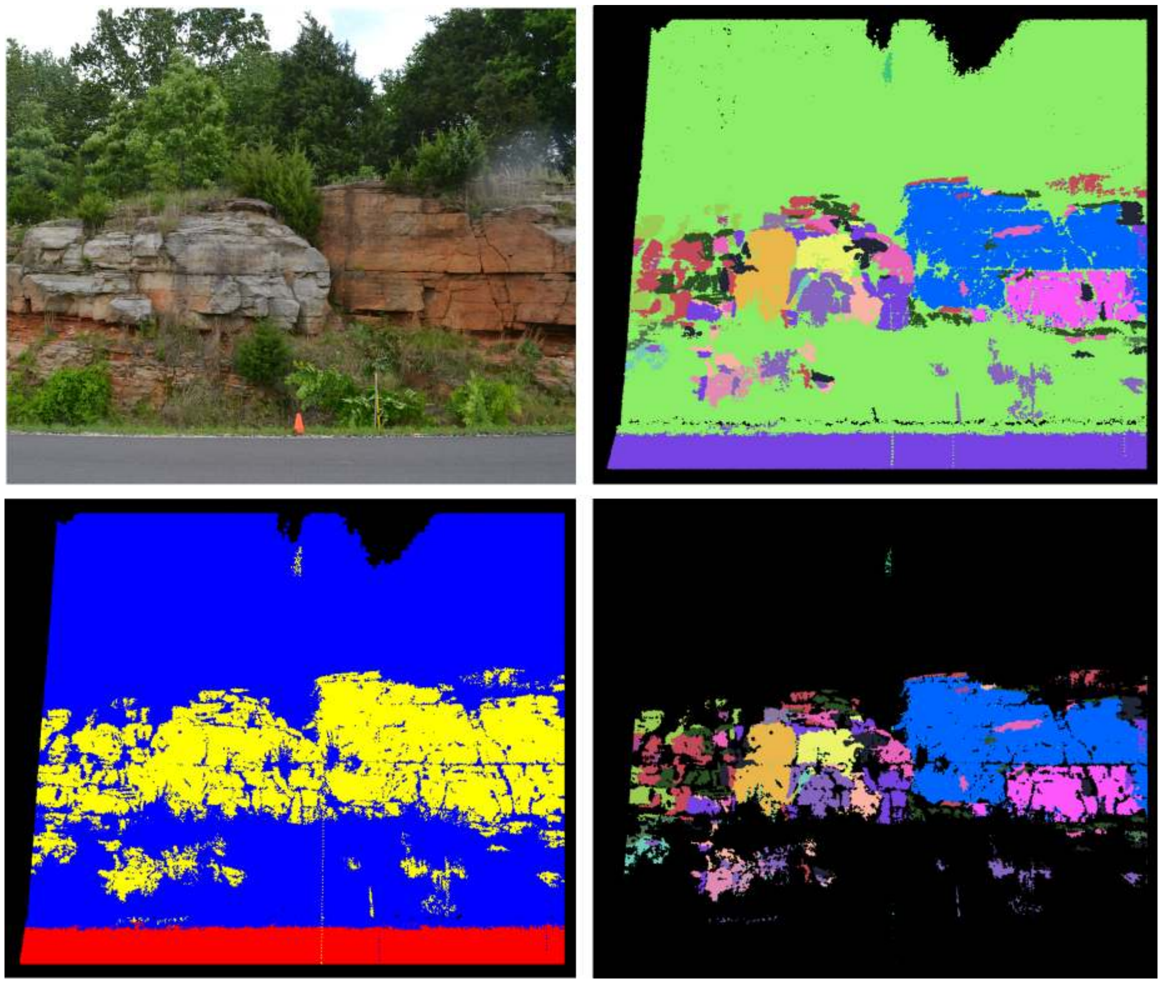
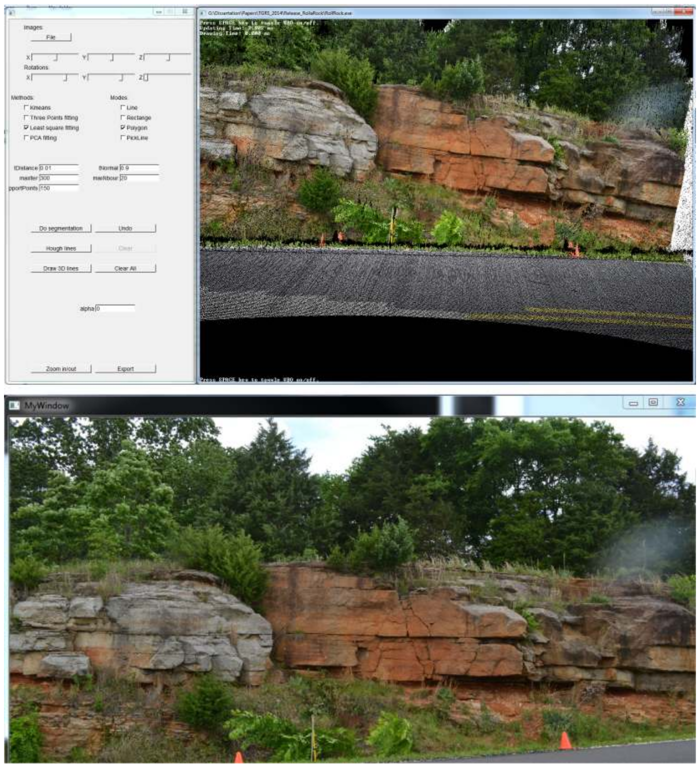


Figure 7.10: 3D segmentation of rock imagery. *Top Left:* 2D photograph of scanned site. *Top Right:* 3D segmentation result of portion of 3D scan visible from this camera. *Bottom left:* 3D scan labeled as ground (red), vegetation (blue), and planes/rocks (yellow). *Bottom right:* 3D segments used for matching after ground and vegetation have been removed.

7.4.2 LIDAR Fusion User-Interaction

Discontinuities or cracks in the rock mass, when exposed in an outcrop or cut, manifest themselves in one of two ways. On flat planar rock cuts, the intersection of the plane of the discontinuity and the planar rock cut results in a visible line fracture trace that lies on both planes (Figure 7.8). On rock cuts that are irregular, the actual faces of the discontinuities are exposed. These fracture surfaces are similar to the facets on a cut precious stone (Figure 7.8). Often times, both types of discontinuities are visible on the same exposure. We have developed a software system based on the algorithm described in Chapter 5 that is designed to aid in measuring these discontinuities with minimal user effort and a high degree of accuracy by fusing 2D and 3D data. The user is able to to make selections with ease on a regular photograph, and through our registration process they have access to the very detailed 3D LIDAR scan. Using the 3D scan, we are able to calculate the orientation of different planes on the rock using the 3D LIDAR data. This program is written in C/C++ and uses FLTK and OpenGL for the graphical user interface (GUI). The program along with the 2D and 3D data the user views and interacts with are displayed in Figure 7.11.



Methods:
- [ ] Kmeans
- [ ] Three Points fitting
- [✓ Least square fitting]
- [ ] PCA fitting

Modes:
- [ ] Line
- [ ] Rectangle
- [✓ Polygon]
- [ ] PickLine

tDistance 0.01
tNormal 0.9
maxIter 300
maxNbour 20
pportPoints 150
alpha 0

Figure 7.11: Interface for our LIDAR fusion software. *Top:* User command window and interface for working with 3D LIDAR data. *Bottom:* Interface for working with 2D photograph.

Linear Trace Measurement

Our system supports both interactive and automatic 2D linear trace extraction from 2D images and we display the result on the 3D view of the LIDAR data. First, the user selects a region of interest by checking the "Rectangle" button on the operation panel and marking the top left and lower right corners of the rectangle on the photograph with two mouse clicks. Figure 7.12 shows this process. The user can click the "Hough lines" button on the operation panel and the program then extracts the linear traces inside the region of interest (the blue rectangle) using Hough line transform [155].

The extracted lines are shown as red lines in Figure 7.12 Top and Center. The user can next select relevant linear traces by changing the program's mode by checking the "Pick Line" option and clicking on line segments in the photograph. Selected lines are shown in cyan in Figure 7.12 Center. From our 2D-3D registration algorithm, we know where the 2D lines appear in the 3D point cloud. When the user clicks the "Draw 3D lines" button in the operation panel the different lines appear in 3D in different colors as shown Figure 7.12. The 3D coordinates for the endpoints of each line are displayed on the screen and if the user selects the "Export" button they are written to a saved file.

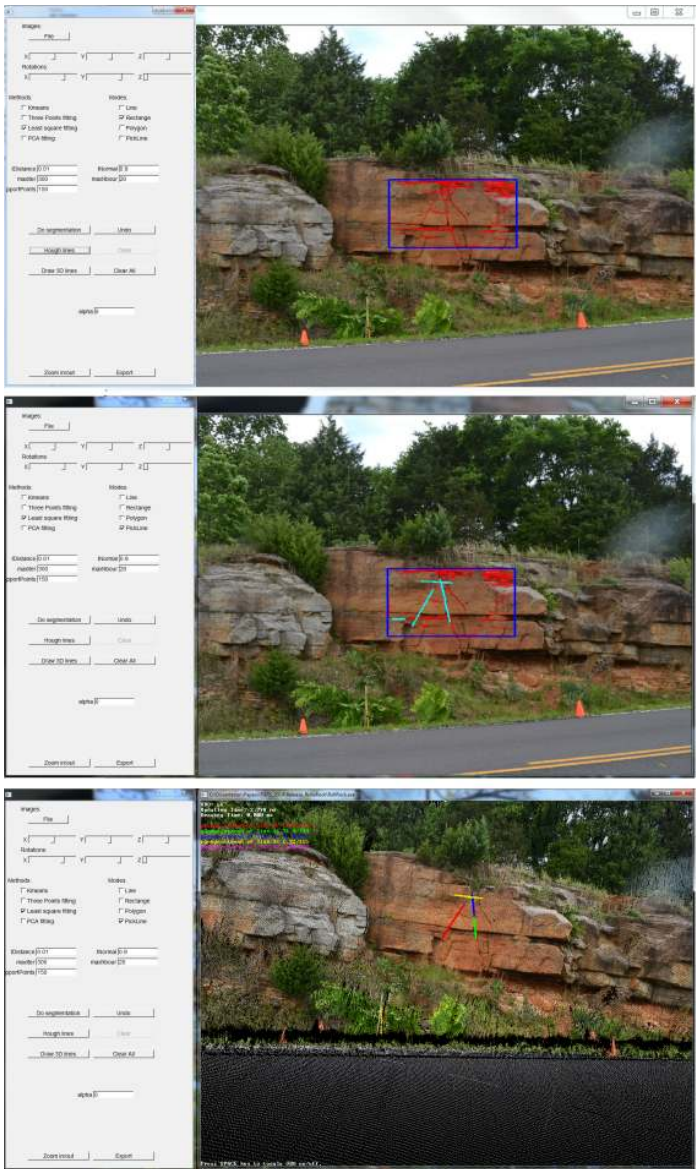


Figure 7.12: Interface for 2D linear trace extraction on 2D photographs. *Top:* The user selects a region of interest (blue) by checking the “Rectangle” button on the operational panel and clicking the top left and lower right corners of the region on the photograph. Hough lines are extracted (red) by clicking the “Hough lines” button. *Center:* The user selects relevant traces (cyan) by checking the “Pick Line” button and clicking on interest points in the image. *Bottom:* Corresponding 3D lines are shown on the LIDAR scan after the “Draw 3D Lines” button is clicked.

Facet Orientation Computation

Our 3D facet orientation computation, which is demonstrated in Figure 7.13, begins the same way as the linear trace extraction function with the user selecting a rectangular region of interest. We automatically conduct plane fitting on the 3D LIDAR points located within the marked rectangle using one of four methods. The user can specify that plane fitting be performed with either k-means clustering, three point fitting [156], least square fitting, or principle component analysis (PCA) fitting [118]. Different planes are displayed with different colors as seen in Figure 7.13 Right.

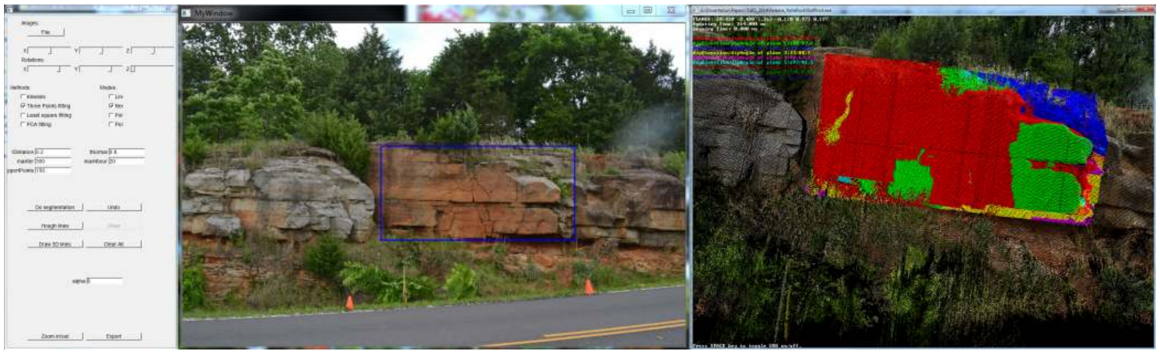


Figure 7.13: Interface for 3D facet orientation computation. *Left:* User selects rectangle surrounding region of interest on 2D photograph. *Right:* Different planes are shown in 3D LIDAR data as different colors. We provide four different ways to conduct the plane fitting: k-means clustering, three points fitting, least square fitting, and PCA fitting.

3D Orientation Measurements

Our program calculates the angles and orientations of 3D planes and lines of interest for engineers studying a rock site. Here, we detail how we measure the dip and dip directions of rock faces and the plunge and trend of rock discontinuities in a virtual environment. Diagrams of these measurements are shown in Figure 7.14. In the following equations, $\overrightarrow{N_p} = (n_1, n_2, n_3)$ denotes the normal of the plane representing a rock face and $\overrightarrow{N_h} = (0, 0, 1)$ is the normal of the horizontal plane.

We use Equation 7.6 to compute the strike of a rock face, which is the mean azimuth of the face. It is defined as the direction of the intersecting line between the rock face plane and the horizontal plane. The direction perpendicular to the strike, the dip, is calculated using Equation 7.7.

$$\begin{aligned} Strike &= N_p \times N_h \\ &= (n_1, n_2, n_3) \times (0, 0, 1) \\ &= (n_2, -n_1, 0)^T \end{aligned} \tag{7.6}$$

$$\begin{aligned} Dip &= Strike \times N_p \\ &= (n_2, -n_1, 0) \times (n_1, n_2, n_3) \\ &= (-n_1 n_3, -n_2 n_3, n_1^2 + n_2^2)^T \end{aligned} \tag{7.7}$$

Equation 7.8 gives us the dip angle, the angle between the dip of the rock cut (often close to 90 degrees) and the horizontal plane.

$$DipAngle = \frac{\pi}{2} - acos(\frac{dip \cdot N_h}{\|dip\|}) \tag{7.8}$$

To find the direction of the dip, we must project the dip onto the horizontal plane. The angle between the projection and the north pole, or the azimuth, is the dip direction. Since the dip has a direction of $(n_1 n_3, n_2 n_3, n_1^2 + n_2^2)$, the projection of the dip on the horizontal plane has the direction $(-n_1 n_3, -n_2 n_3)$ or $(-n_1, -n_2)$.

These values are used in Equation 7.9 to give us the dip direction.

$$DipDirection = acos(\frac{DipProjection \cdot NorthPole}{\|DipProjection\|}) \tag{7.9}$$

We follow a similar set of steps to quantify a rock discontinuity in 3D space which we represent as a line and define in terms of its plunge and trend. A 3D line is initially defined by fitting a line to the LIDAR points corresponding to the selected 2D pixels in a photograph. It can be represented using a point (x_o, y_o, z_o) and a direction is (a, b, c) following Equation 7.10.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} + t \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{7.10}$$

We first compute the plunge of the 3D line, which similarly to the dip angle, is the angle between the 3D line and the horizontal plane and can be calculated using Equation 7.11, where *LineDir* is the direction (azimuth) of the 3D line.

$$Plunge = \frac{\pi}{2} acos(LineDir \cdot N_h \|Line\|); \tag{7.11}$$

The second measurement we gather for a 3D line is the trend which is the angle between the north pole and the projection of the 3D line onto the horizontal plane. Since the 3D line has the direction (l_1, l_2, l_3), the trend's direction is $(l_1, l_2, 0)$. The trend angle is calculated using Equation 7.12.

$$Trend = acos(\frac{TrendDirection \cdot NorthPole}{\|TrendDirection\|}) \tag{7.12}$$

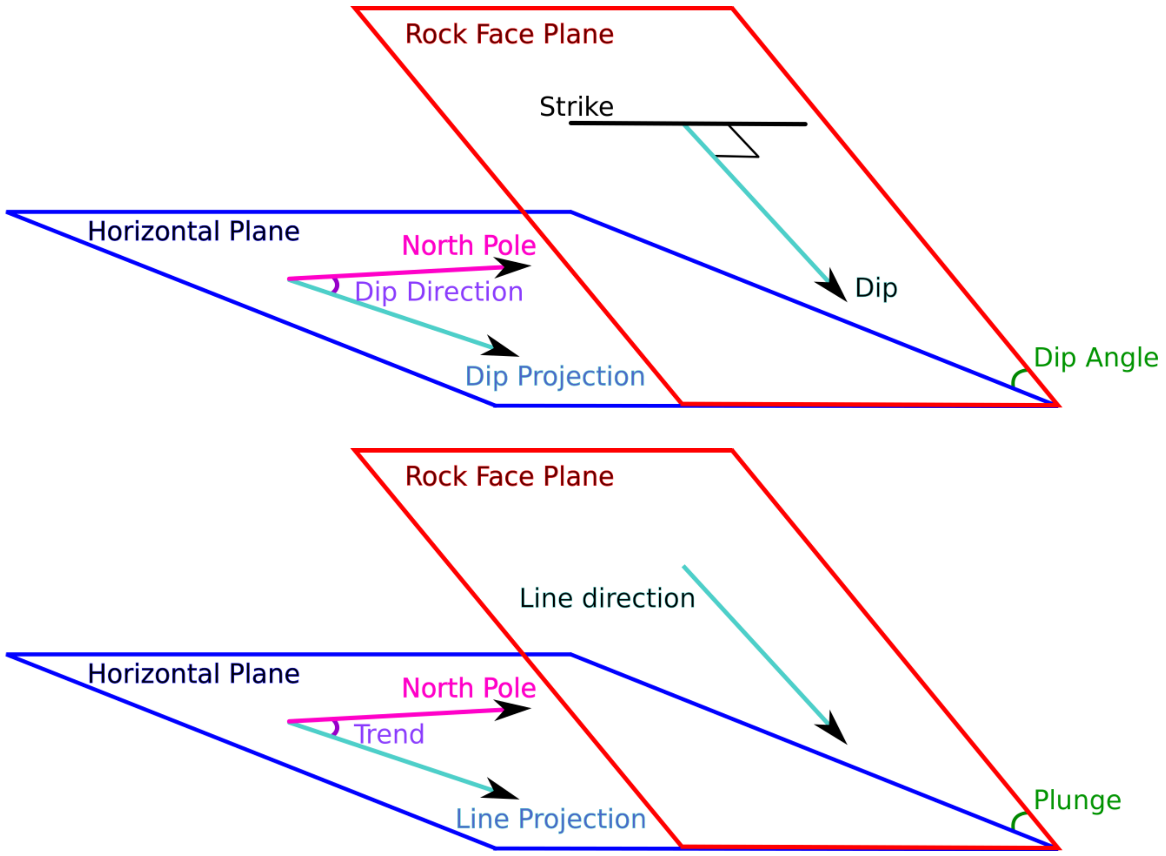


Figure 7.14: Measuring orientations of rock faces and discontinuities. *Top:* Diagram of strike and dip of a planar rock face. *Bottom:* Diagram of plunge and trend of a linear rock discontinuity.

Integrating 2D Image-Based Linear Trace Measurement with 3D LIDAR-Based Facet Orientation

We can combine these measurements of 3-D facet orientations and 2-D trace orientations to obtain valuable information about the structure of a vertical rock cut. Our motivation for doing so is based on the fact that both 3-D facet orientations, measured on LIDAR data, and 2-D trace orientations, measured optically, can be clustered into groups based on their mean orientations (dip direction/dip angle and azimuth/plunge). All 2-D traces lie on, and define the intersection of two planar discontinuities, the plane of the rock cut face and the discontinuity plane associated with that trace. We can test geometrically if a given trace vector is parallel to a given discontinuity surface cluster. If so, it belongs to the same set, and vice versa.

We conduct K-Means clustering on the 2D linear trace directions and the 3D facet orientations, respectively. Since we have registered the 2D image with the 3D LIDAR, we can find the corresponding 3D line direction for each 2D linear trace, which in turn can be used to determine whether there are 2D linear traces that are parallel to some 3D facets computed from the LIDAR data. In Figure 7.15, we have color coded the 2D linear traces and 3D facets. For example, the blue colored linear traces are parallel to the blue colored 3D facet. The same applies to the sets of green and yellow traces and facets.

7.4.3 Experiments

We selected a vertical rock cut along the outer road adjacent to Highway 144 in Rolla, Missouri as our study site. The goal of our tests was to see if rock measurements obtained by a trained engineer using traditional techniques are equivalent to the measurements obtained with our software. A geological engineer took manual measurements using a Brunton compass of various facets and traces on a rock cut corresponding to the features that are highlighted in Figure 7.15. We treat these measurements as our ground truth during evaluation. We selected the same areas using our LIDAR fusion software and compared them to the manually collected values. Sets of plunge and trend measurements obtained manually and by our program are shown in Table 7.1 and a comparison of dip and dip direction results are shown in

Table 7.2. We can see from these charts that the software and geological engineer's measurements of both facets and traces tend to be quite consistent.

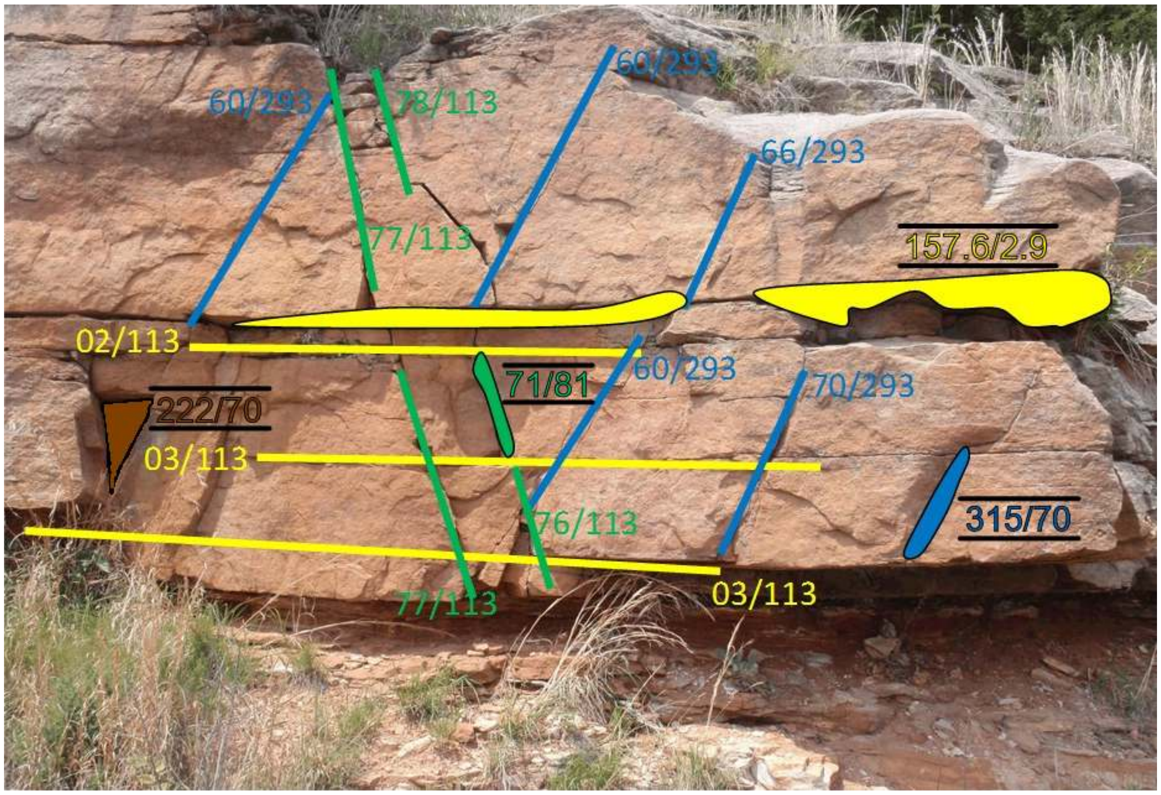


Figure 7.15: A vertical rock cut in Rolla showing traces (blue, green, yellow lines) and facets (blue, green, brown and yellow polygons). Trace measurements are displayed in the format of "plunge/trend" and facet measurements are in the format "dip direction/dip angle". Facet measurement are bounded in black. These measurements were taken manually by a geological engineer and we treat them as our ground truth during evaluation.

Table 7.1: Plunge and Trend (Traces) - Manual measurements vs. LIDAR Viewer measurements

Line Color	Manual Plunge	Manual Trend	Software Plunge	Software Trend
Blue	70	293	65	273
Blue	66	293	70	273
Blue	60	293	51	273
Blue	60	293	61	285
Blue	60	293	59	275
Green	77	113	73	126
Green	78	113	76	121
Green	77	113	74	137
Green	76	113	75	123
Yellow	2	113	1	117
Yellow	3	113	1	102
Yellow	3	113	1	110

Table 7.2: Dip and Dip Direction (Facets) - Manual measurements vs. LIDAR Viewer measurements

Plane Color	Manual Dip Direction	Manual Dip	Software Dip Direction	Software Dip
Blue	315	70	337	74
Green	71	81	101	76
Yellow	157.6	2.9	160	3.21
Brown	222	70	235	81.2

7.5 Cloud Computing for Disaster Scenario 3D Visualizations

During natural or man-made disasters, videos from many perspectives are collected by security cameras and civilian observers. This abundance of data can be helpful for officials and emergency responders who need to quickly ascertain the state of affairs during an incident. When the normal infrastructure starts breaking down, it can be difficult for the authorities deciding how to respond to access, observe, and determine the extent of the damage. However, it is becoming increasingly common for civilians to record videos of unfolding events on cell phones, making crowd-source information available in addition to video feeds from mounted security cameras. Unfortunately, handling such large collections of videos and performing this type of surveillance on a traditional 2D grid display can be quite challenging. It is difficult for users of such systems to perceive the spatio-temporal relationships between different video streams, understand the geographical context of the situation, track and analyze events as they unfold, and predict possible outcomes. On the other hand, we can create *3D virtualizations of scenes* using our registration techniques discussed in Chapter 5 and Section 7.1 to fuse a set of videos with a single 3D environment. Doing so can help a great deal in obtaining greater spatial-awareness such information, especially in disaster scenarios [157]. Allowing a model of a 3D scene to be viewed from remote locations by sending it over available networks to officials can aid in response organization. Large scale 3D models can be created for these purposes using 3D LIDAR scans [158]. LIDAR data has been shown to be useful for assessing the aftermath of natural disasters since highly accurate scans can be obtained very quickly [159]. However, LIDAR data by nature can be large and computationally expensive to process so adequate resources must be available to take advantage of this rich source of information. Leveraging the elasticity of cloud computing and the resources of high-speed networks provides new opportunities for using LIDAR data for life-saving applications.

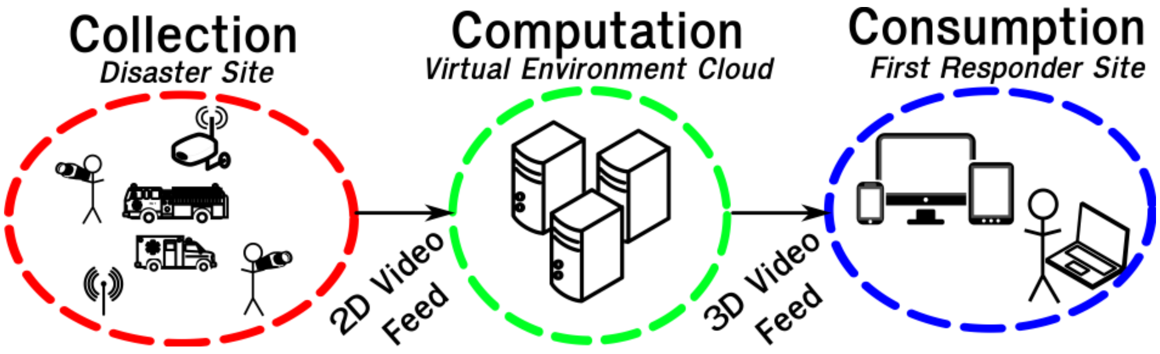


Figure 7.16: Diagram showing overview of system. Videos are collected at disaster scene and transfered wirelessly to the server where their 3D poses are estimated. The virtual 3D environment is transferred from the server to thin clients for data consumption.

In this section, we study how these opportunities can be realized by working with a dynamic interactive 3D scene visualization system in which videos are captured at a disaster site, transferred to a server for on-demand processing and model construction, and viewed remotely with a thin-client. We look at provisioning options such as elastic compute to carry out these tasks. This overall process of *collection, computation, and consumption* is outlined in Figure 7.16. By seamlessly rendering dynamic video data from multiple cameras on top of a LIDAR point cloud, the system allows the users to view the recorded action in the context of a global 3D model from the viewpoints of any virtual camera. Many groups have developed methods for making visually rich information for scene understanding available as was discussed in Section 2.4. However, many of these techniques are computationally intensive and can benefit performance-wise from high-speed networking and cloud computing resources. By transferring collected 2D and 3D data to a remote server for real-time computation and processing and transferring the final fused results to mobile devices for consumption, a whole new range of use cases for LIDAR data in disaster scenarios becomes possible.

We test running our proposed system over various application-driven overlay networks using concepts of software defined networking (SDN) [160, 161] to identify the network configuration requirements for processing and viewing 3D video and delivering high Quality of Service (QoS). We use a wireless overlay network (WON) to represent a standardly available network and a higher-speed network that simulates a fast overlay network made available on top of existing infrastructure in a disaster scenario that meets special needs, which we refer to as a disaster overlay network (DON). Videos must be able to be uploaded to the server over DON without encountering network congestion so that they can be processed on-demand in the cloud infrastructure and transferred via the same network to thin-client protocols for first responders to view.

Research on how to set up mobile networks in a disaster scenario has been carried out that propose systems for transferring and sharing data amongst emergency response crews, administrative officials, and medical professionals for planning and rescue purposes. However, to the best of our knowledge, none of these studies have looked at using LIDAR-based 3D models and simulations to organize the data being sent through these networks. Several groups have investigated ways in which wireless networks can be set up and utilized for communication in the event of an emergency [162]. Chissungo et. al [163] have studied using Wireless Mesh Networks to transfer medical information throughout disaster zones in situations where wired networks are damaged. Witkowski et. al [164] set up mobile ad hoc networks that allow humans to communicate with robots being used to explore the aftermath of a disaster. As these types of studies have become more popular, the speed of message delivery over wireless networks and the energy efficiency of these on-the-fly network setups has been prioritized and explored [165]. In our work, we study achieving similar goals of high-speed communication with DON.

7.5.1 Experimental Methodology

Experimental Setup

Our testbed setup consists of clients connected to WON (∼10Mbps) that represent a standardly available campus enterprise network for QoS priorities and a compute manager VMware Horizon View© connected over a higher-speed campus research network DON (∼600Mbps). We emulate a network made available in a disaster scenario in which these two networks can be used in parallel. By pooling resources, our collection, computation, and consumption steps can be employed effectively by first responders. We have a virtual server setup with 6vCPU (12GHz) and 16 GB of memory with Windows 2008R2 64bits installed. Our physical server has 2 processors Intel Xeon Processor E5-2640 v2, 8 cores each for a total of 16 cores. The clients have a Windows 7 Enterprise 64 bits O.S. installed. Our clients are able to stream data to the server by using `curl` Linux utility functionality that is authenticated by the FTP server in the virtual server.

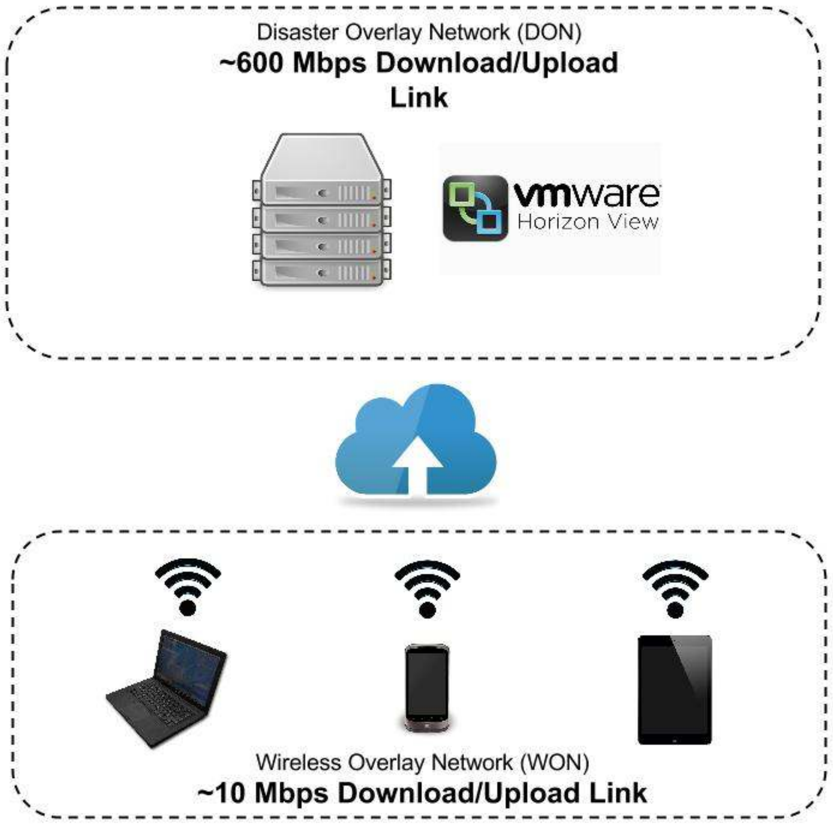

Disaster Overlay Network (DON)
~600 Mbps Download/Upload Link
vmware Horizon View
Wireless Overlay Network (WON)
~10 Mbps Download/Upload Link

Figure 7.17: Diagram of the two networks over which we test transferring data.

To obtain a 3D model for our location of interest, we use a Leica C10 HDS LIDAR scanner that provides a high-resolution point cloud of a scene and 2D images using a built-in camera. We also capture multiple video streams of people walking around the university campus with HD video cameras. This video data needs to be transferred in real-time to the HPC server for data processing.

Design of Experiments

We perform tests on our university campus after obtaining the LIDAR scan and several HD videos. We separately evaluate the performance for the three stages of our system shown in Figure 7.16, i.e. collection and transferring the 3D scan and video files, computing the 2D-3D data fusion, and consumption by the user to receive 3D scenes and multiple videos for virtual navigation and video analysis. The goal is to obtain real-time (or near real-time) responses for all of these tasks. We also experiment with scaling up the amount of data transferred to see how many videos we can handle and how large the 3D model data can be, depending on the hardware used.

To simulate the collecting and transferring of any number of real-time video streams, we first obtain several HD videos on campus. These videos are stored on a laptop and a varying number of duplicates are sent over the network simultaneously to tax the system. Our goal is to observe what happens to the system when one verses many videos become available and need to be viewed. Individual video frames are transferred sequentially to mimic real-time video capture. The 2D-3D registration and video motion analysis stages are performed on the server.

For the final consumption stage, the large 3D model only needs to be transferred over the network to the remote device one time when it is first requested. If the user wishes to a view a different location, a new model will need to be sent to the mobile device (laptop, tablet, cellular phone, etc.).

7.5.2 Study Results

We studied the collection, computation, and consumption sections of our pipeline individually. All of our tests were performed three times, and in this section we report the averages of these tests as our final results.

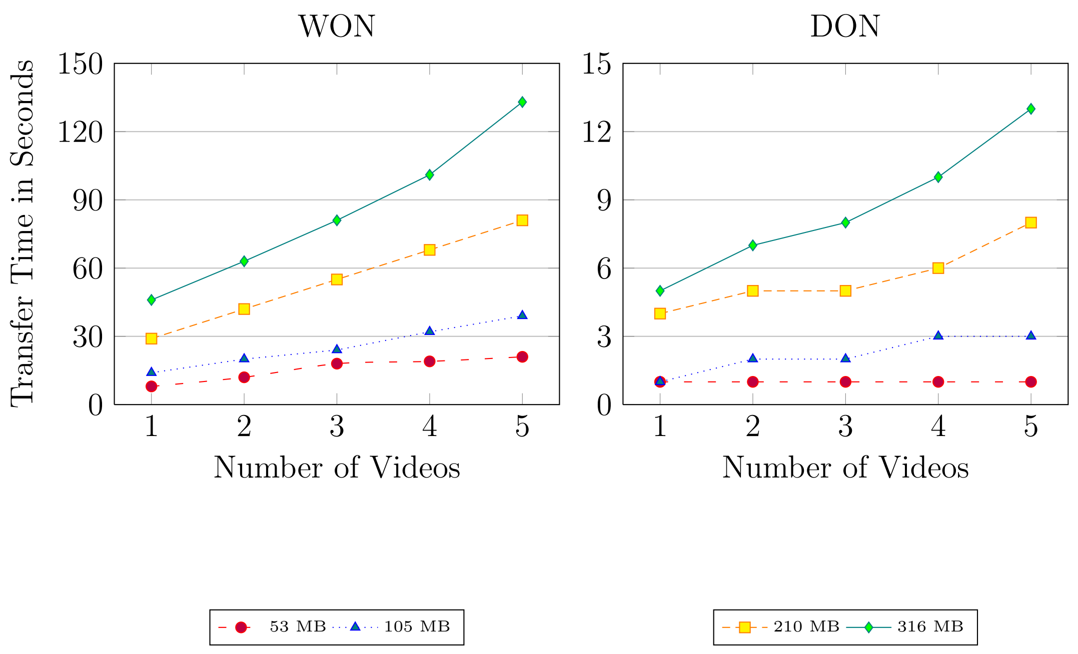


Figure 7.18: Collection stage transfer times for varying video sizes. *Left:* Transfer times for WON. *Right:* Transfer times for DON.

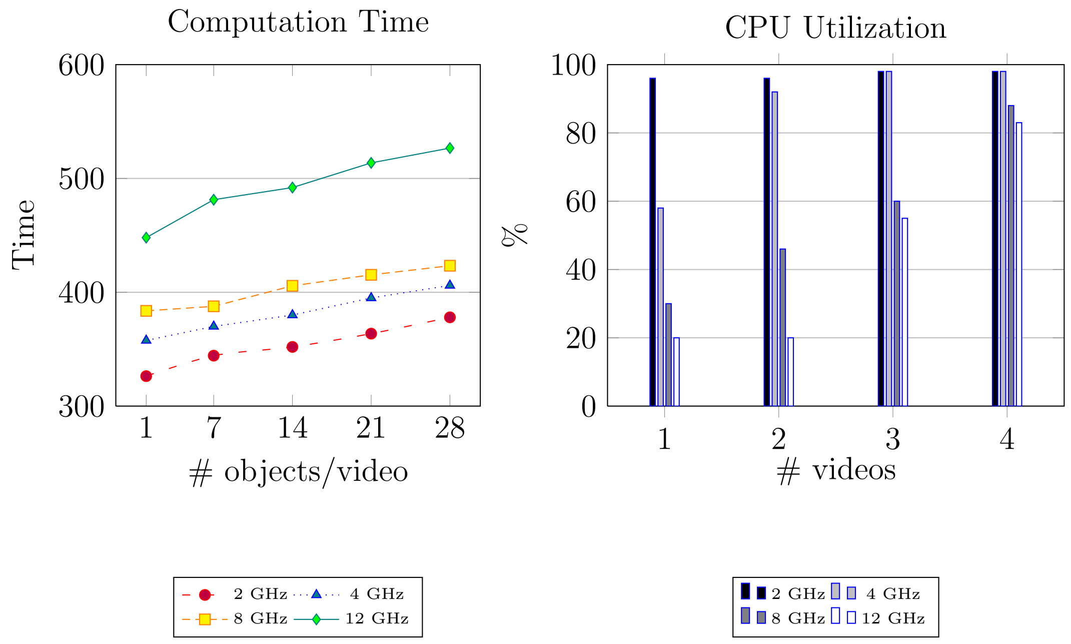


Figure 7.19: Performance during the computation stage. *Left:* Time measured in seconds required to compute 3D pose for increasing number of dynamic objects in videos on different server configurations. *Right:* CPU utilization on server during computation stage for different configurations.

Collection

We tested sending varying file sizes over the server (52, 105, 210, and 316 MB) to account for situations where different definition videos are streamed from client connected to WON and DON. For example, low-resolution, black and white security camera footage may be utilized compared to high definition video captured by a smartphone or hand held camera. The transfer times in seconds for these tests are shown in Figure 7.18. The maximum number of videos we tested sending at once is five because our server has six cores and cannot process more than that number of videos at once. Despite the fact that these are relatively small-scale tests, we still get a good sense from the charts how communication time will increase as the number of videos rises. These tests also show that DON is able to transfer data about 10 times faster than WON and would be very beneficial in a disaster scenario where timely information sharing is key.

Computation

We modified the vCPU capacity of our virtual server with 2, 4, 8 and 12 GHz, testing the processing times in seconds for videos containing 1-28 moving objects. Each dynamic object in every video needs to be identified, segmented from the static background, and modeled in 3D so we are interested in what happens to our overall performance as more objects are recorded. We stopped at 28 objects because this seems to be a reasonable limit on the maximum number of people that will be captured in a typical camera's field of view and be able to be separately identified and modeled as individual objects in 3D. We also tested the system's performance when processing 1 to 4 videos of 185MB each with the same content simultaneously and looked at the CPU percentage utilization. These results are all shown in Figure 7.19. We observe here that the system becomes saturated when processing four videos and can see what will happen as more videos are added to the system. We gain the greatest boost in performance when increasing from two to four videos.

Consumption

During the consumption stage, the clients need to download the files containing 3D information from the server. In the case that a client is connected to the server via WON, this process is time consuming compared with a virtual desktop accessed from a thin-client (hardware/software). For both cases, Teradici PCoIP protocol© is used for remote access. A comparison of file transfer times in seconds between a physical client connected to WON and using a virtual desktop setup on a server connected to DON is shown in Figure 7.20. We tested transferring 3D data files for between 377 and 3,496 individual moving objects simultaneously to significantly stress the system and to find out how much information can be processed in a timely manner if the disaster site is very congested with people and cameras. We can see that using DON, thousands of moving objects can be transferred and displayed in a matter of seconds, making this setup great for first responders needing to rapidly sift through vast information from the disaster scene.

Our final stage of testing looks at the actual user experience during the consumption stage. We evaluated the data transfer times in Kbps for running various programs over the thin-client. We compared the performance of our 3D video program to everyday programs that most people are familiar with such as Excel and Internet Explorer on four different types and speeds of networks, as shown in Figure 7.21. We can see that the 3D video program requires a tremendous amount of resources for processing because it contains vast amounts of rich visual information even after encoding. Ideally, emergency responders will be able to use thin-clients for 3D video analysis to avoid potentially long download times, 3D viewing software setup and speed up data acquisition.

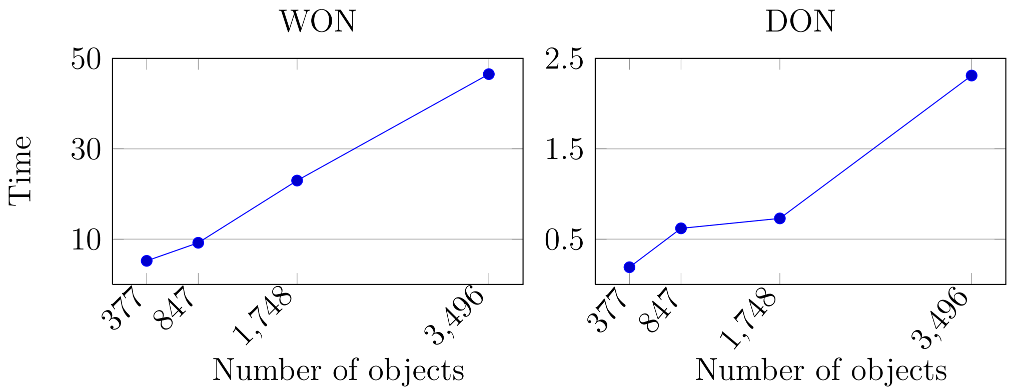


Figure 7.20: Consumption evaluation measured in seconds. Transfer times for dynamic 3D objects captured in videos to be sent to remote device for viewing. *Left:* Transfer times for WON. *Right:* Transfer times for DON.

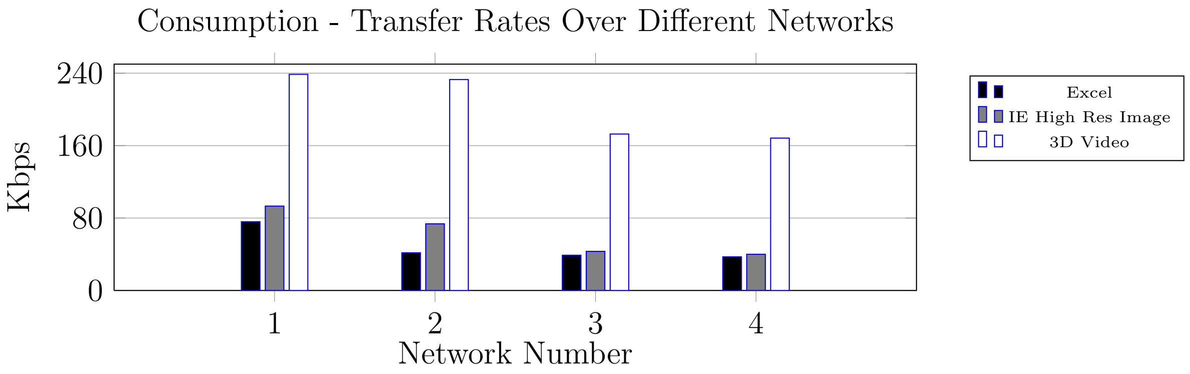


Figure 7.21: Comparing user experience of our program to other standard programs over the thin-client during the consumption stage. Network 1 the Campus Wireless with a bandwidth of 8-9 Mbps/10 Mbps. Network 2 is a Wired Lab with 7 Mbps/7 Mbps. Network 3 is a Wired Lab with 5 Mbps/3 Mbps, and Network 4 is a Wired Lab with 3 Mbps/1 Mbps.

//rolla section

Chapter 8

Summary and Concluding Remarks

8.1 Future Work

There are a few aspects of our current work that can be expanded to be more robust and new research avenues we can take with our general registration framework. Depending on the type of data used, our matching pipeline may face a few roadblocks. If only a small percentage of the identified correspondences, whether they be 2D-2D or 2D-3D matches, are actually inliers, it is very unlikely that we will find a correct alignment using RANSAC. To address this in the future, we will explore using techniques that have expanded on RANSAC to work with extremely noisy data such as [166]. One of the reasons we may encounter this problem is that fact that, currently, our pipeline does not incorporate affine invariant descriptors and is limited to relatively small baselines. One option for addressing this is to include an affine invariant descriptor such as ASIFT [47] to expand our work. Any keypoint matching technique or combination of techniques can be inserted into our pipeline.

We can also perform rectification on the images prior to our general contextual 2D matching so that they are all front facing. For photographs, this is very feasible in images of architecture where two or three main vanishing directions may be identified. We use the techniques described in [167, 168] to perform rectification. Figure 8.1 shows an image pair with a wide baseline, the keypoint matches we obtain by applying Contextual Harris-Laplacian to the images, the rectified photographs, and the new, more accurate set of keypoint matches we obtain with Contextual Harris-Laplacian. Performing rectification in this manner has the potential to especially improve the performance of our Pseudo Corner method which looks for line segment intersections on planar areas. Synthetic images of LIDAR scans can also be created such that the main planes are front facing since we can identify planes and their normals using our segmentation techniques.

In our work on aligning repetitive architectural images, the main bottle neck of our pipeline is the actual requirement that the image be rectified. While this is an acceptable assumption for urban images, we cannot guarantee that the method we use will also be able to perfectly rectify each image. Other repetition matching pipelines use an initial estimation of the repetition to refine rectification [36, 37]. We can apply a similar approach to our work. We also assume that the images do not have severe radial distortion. Images that do have this type of distortion may not be rectified correctly. A radial undistortion step could be applied in this case [105]. The effect that radial distortion has on our evaluation is discussed in greater detail in the Appendix C.

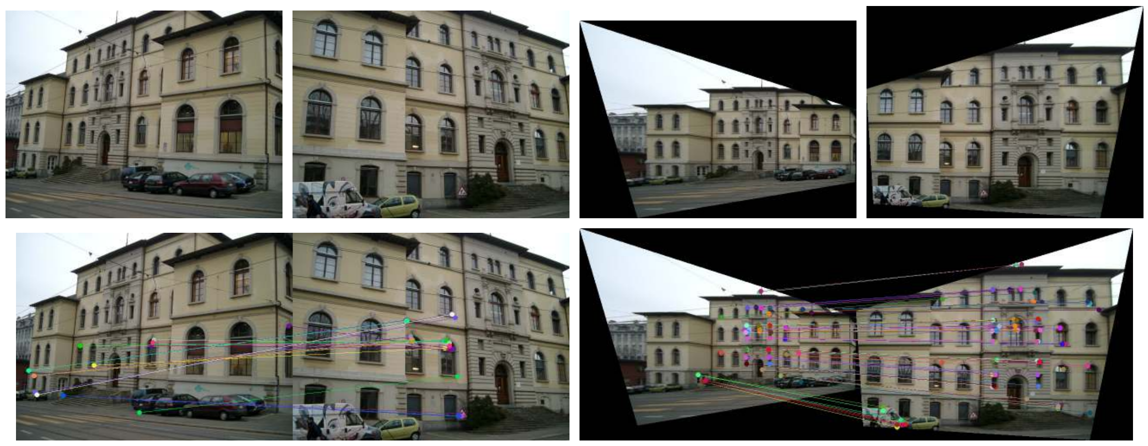


Figure 8.1: Using image rectification to match image pairs with wide baselines. *Top Row Left:* Original images with wide baseline. *Bottom Row Left:* Keypoint matches on original images using Contextual Harris-Laplacian. *Top Row Right:* Rectified images. *Bottom Row Right:* Keypoint matches on rectified images using Contextual Harris-Laplacian method.

The next area I will be exploring in this work is performing change detection between imagery sets and using this information to update the 3D point cloud used as a basis for visualization. In our 2D-3D work, we do not currently perform any explicit change detection between the 2D photographs and the 3D point cloud such as is discussed in [169]. If the content in these different types of imagery changes too much, our registration will not have a pleasing visual result. For instance, if a building is completely covered by trees in the summer that are rather far in front of it and then it becomes exposed in the winter, a winter image will not have any correct 3D information onto which it can be projected as is demonstrated in Figure 8.2. In the future, we will need to update the 3D information that has large changes to produce a more visually accurate result. We can do this by creating a structure from motion point cloud from the new set of photographs, identifying which areas to use from the SfM information and which is use from the LIDAR data, and combine the two. The general semantic information we gain from our 3D planar segmentation step can help in this decision.

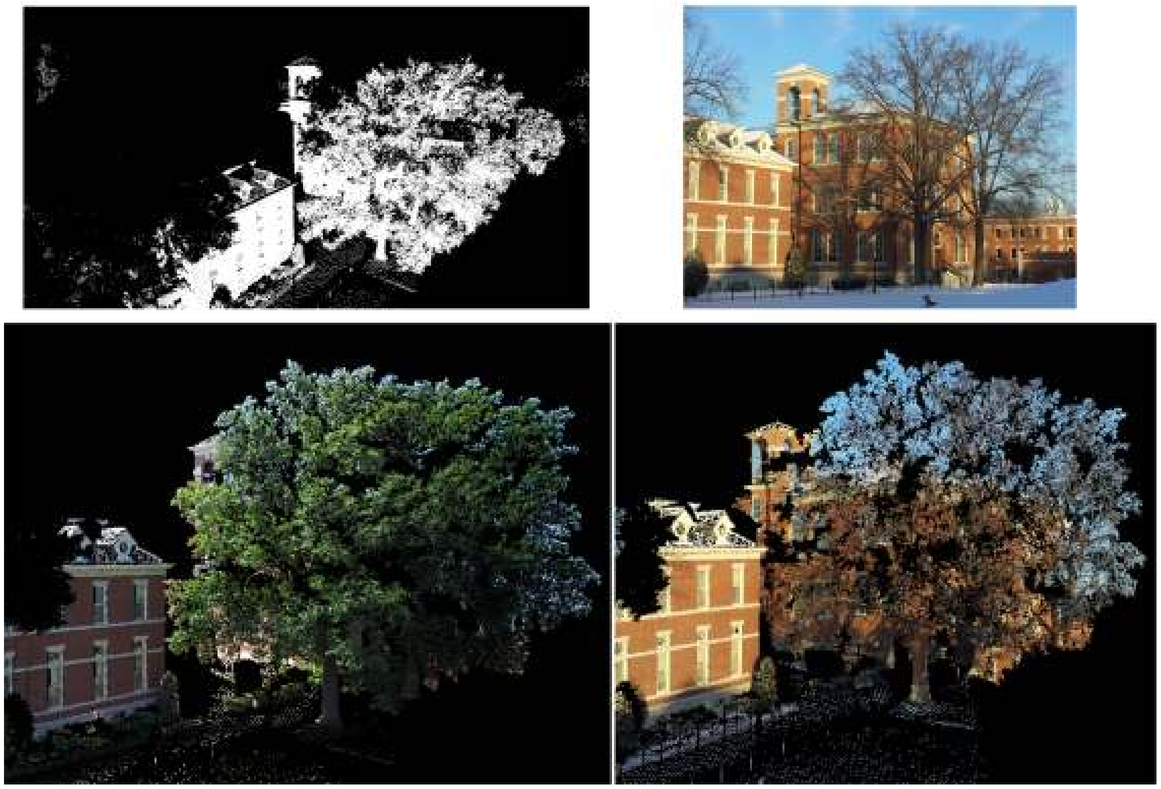


Figure 8.2: Limitation of not performing change detection when registering imagery from different seasons. *Top left:* Original LIDAR scan. *Top Right:* A photo is taken in the winter and registered with a scan captured in the summer. *Bottom left:* In the summer the right building is covered by trees. *Bottom right:* In the winter the building is exposed, but the summer scan does not have correct 3D information onto which the building can be projected. However, the building on the left is still correctly registered between seasons because its structure does not change.

We can also expand how much we use semantic labeling for matching. We have already labeled all the segments in the point cloud as being either buildings, trees, or ground and by extension have the semantic information for the LIDAR photographs. We can incorporate a scene parsing method such as [170] to obtain similar labels for the photographs and videos that need to be registered for the scan. By only matching sections of images and the scan with the same labels, we can reduce our search space for keypoint matching even further.

8.2 Conclusion

In this thesis, we have presented a system for registering photographs, videos, and 3D LIDAR range scans depicting a variety of scenes including both natural and architectural subjects. We have developed methods for visualizing and organizing multi-modality imagery with highly varying visual properties in a manner that allows us to align different datasets. We have incorporated a contextual framework for matching features both within and across dimensions that we use for describing and matching keypoints' neighborhoods. Our framework can be used to build upon and complement local keypoint matching techniques, creating an ensemble feature, or can be used independently to describe and match keypoints. In addition to potentially incorporating raw local keypoint matches provided by existing techniques, we can use various keypoint detectors as anchors to study larger regions surrounding keypoints in terms of salient line segments and texture-rich HOG information. We have studied, tested, and presented results for using our framework in combination with several different keypoint descriptors, including our newly proposed Pseudo Corners, to demonstrate that it is very flexible and not dependent on any one method or type of information. By using both local and regional descriptors, we are able to identify small unique regions surrounded by clutter and to clear up potential ambiguity during feature matching by looking at a larger subset of the image. This pipeline has been designed to combine these various feature extraction and matching methods in such a way that it is robust under lighting, content, and rendering changes. We have shown quantitatively that using this information can increase the accuracy of identified matches and image alignment over using state-of-the-art methods. Our technique can work on images containing a variety of architectural styles, man-made features, and natural content without making assumptions about the general structure of a scene and can handle different artistic representations of the same location.

We have also presented a specific method for registering images of architecture that contain repetitive features that have a tendency to create ambiguity for general keypoint matching methodologies. In contrast to several state-of-the-art approaches to registering urban data, our method does not try to fit an evenly-spaced grid to repetitive elements [36], considers both salient and repetitive regions simultaneously [37], and does not require that the same portion of a building's facade be captured in different images [38]. Our method takes a two-step approach to align images one dimension at a time to reduce the search space during each matching stage. We have discussed how we perform a dimension reduction amongst unorganized repetitive features to further remove ambiguity from the search space when matching regions across images. We have also proposed equations for determining the saliency of matching regions between an image pair without requiring machine learning techniques to learn what elements are salient on a large dataset [39]. We believe that by developing a pipeline without these constraints, our method is more robust than many other proposed registration approaches, allowing us to align challenging, ambiguous architectural photographs in a more general manner.

We have also shown how we can use the information obtained from these matching techniques to compute a camera's 3D pose in relation to an existing 3D model. Our pipeline includes two options for determining the 2D-3D matches necessary for computing the camera pose based on what data is available. The first uses contextual information to match photographs and videos to pre-registered images. The second matches edges between synthetic views of the planar-segmented scan and laser intensity visualization and our photographs to directly determine 2D-3D correspondences. Both paths use edge, line segment, and HOG information to handle difficulties presented by imagery obtained from different sensors and under different conditions by making the imagery appear more similar. Our work can be used to view multiple photographs and videos simultaneously in a 3D environment as well as for exciting applications such as creating 3D stereo videos and authoring 3D videos. We are also able to use our work for very practical, safety-related purposes such as assisting geological engineers measuring rocky-walls along roadways and emergency response coordinators needing to fully understand a disaster site. Through our research on fusing multi-modality data, we have offered new technical methods for imagery visualization and registration and, on a larger scale, have demonstrated how using computer vision and imagery alignment methods can positively impact several different fields.

Appendix A

Computing Camera Geometry

A.1 Homography Matrix

The homography matrix (H) is a $3x3$ matrix that maps a group of points that all have the same depth to a corresponding group of points which also all have the same depth. A pair of images (I_1 and I_2) can be thought of as planes in a 3D space and matching pixels ([x_1,y_1] and [x_2,y_2]) laying on these planes can be mapped back and forth between images using the homography transformation as shown in Equation A.1.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{A.1}$$

The homography matrix has the form shown below with 8 unknowns ($a - h$).

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

We can use the Direct Linear Transformation algorithm (which is explained in greater detail in Section A.3) to solve for the vales homography matrix. Each pair of matching points between images provides two constraints on H so at least four pairs of matching points are needed to solve for the values of H. We can rearrange Equation A.1 to the form given in Equation A.2. By taking the cross product in this manner we can expand Equation A.2 and separate all of the known values (matching pixel coordinates) from the unknown values (elements of the homography matrix), giving us Equation A.3. A is an 8x9 matrix consisting of 4 sets of b (shown in Equation A.4) stacked together. Each pair of matching points between the I_1 and I_2 gives us one matrix of the form b.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \times H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = 0 \tag{A.2}$$

$$AH = 0 \tag{A.3}$$

$$b = \begin{bmatrix} 0 & 0 & 0 & -x_1 & -y_1 & 1 & y_2 * x_1 & y_2 * y_1 & y_2 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_2 * x_1 & -x_2 * y_1 & -x_2 \end{bmatrix} \tag{A.4}$$

We can solve for H by running Singular Value Decomposition on A. The Eigen vector corresponding to the smallest singular value of A is used to build up H. The first eight values of the A's Eigen vector are scaled down by the ninth (and last value) of the vector so that it equals 1. The first eight values become that values of $a - h$ in the homography matrix.

A.2 Projection Matrix

The projection matrix P maps 3D points (X, Y, Z) to 2D image points (x, y) as shown in Equation A.5. We can follow a very similar process as was described in Section A.1 for computing a homography matrix to compute a projecion matrix. The cross product of a 2D point with the projection matrix multiplied by a 3D point will equal zero as displayed in Equation A.6. Using this property, we can create a matrix b for each 2D-3D correspondence using the format given in Equation A.7. Six b matrices are stacked into a 12x12 matrix A. The projection matrix is solved subject to Equation A.8 by performing Singular Value Decomposition on A. The smallest singular value of A is identified and its corresponding vector is unstacked into the 3x4 projection matrix, P.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{A.5}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0 \tag{A.6}$$

$$b = \left[\begin{smallmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & -X & -Y & -Z & -1 & yX & yY & yZ & y \end{smallmatrix}\right] \tag{A.7}$$

$$AP = 0 \tag{A.8}$$

This process is detailed further in Section A.3.

A.3 Direct Linear Transformation

The Direct Linear Transformation algorithm (DLT) is the general process we use for solving the linear equations we come across when computing homography and projection matrices. Here we show how to use DLT specifically to solve Equation A.5. We want an equation of the form shown in Equation A.9. Using **b** to represent the homogeneous 2D point and **c** for the 3D homogeneous point, we can rewrite Equation A.6 as Equation A.10. We can also break down the projection matrix P into rows as shown in Equation A.11.

$$XP = 0 \tag{A.9}$$

$$b \times Pc = 0 \tag{A.10}$$

$$P = \begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \tag{A.11}$$

If we expand Equation A.10 we arrive at Equation A.12.

$$\begin{pmatrix} yp^{3T}c - p^{2T}c \\ p^{1T}c - xp^{3T}c \\ xp^{2T}c - yp^{1T}c \end{pmatrix} = 0 \tag{A.12}$$

If we modify Equation A.12 to match the format of Equation A.9 we get Equation A.13.

$$\begin{pmatrix} 0 & -c & yc \\ c & 0 & -xc \\ -yc & xc & 0 \end{pmatrix} \begin{pmatrix} p^1 \\ p^2 \\ p^3 \end{pmatrix} = 0 \tag{A.13}$$

We can eliminate the third row since it is a linear combination of rows one and two. **P** has 12 entries, but 11 degrees of freedom. We need six pairs of matches to solve this equation since we get two equations from each match of the form in Equation A.13.

To help ensure the stability of DLT, the input values should be normalized so that they all fall within a limited range. Section A.4 covers how we normalize our data.

A.4 Normalization

Points in different images and dimensions will most likely be represented in different scales and coordinate systems. When these largely varying values are combined into a matrix for the DLT algorithm, entries may differ by several orders of magnitude.

This will create uneven errors across the projection matrix that is constructed from the Singular Value Decomposition stage [171][105]. The normalization algorithm we use consists of finding a transformation for a set of points such that the centroid of the new points is the origin, and the average distance from the origin is $\sqrt{2}$ [105].

An estimate of the camera calibration matrix K can also be used to normalize the image points, by approximating the focal length $\hat{f}$ as follows:

$$\hat{f} = k * D_x$$

k is in the range [0.5,2].

$$K = \begin{bmatrix} \hat{f} & 0 & D_x/2 \\ 0 & \hat{f} & D_y/2 \\ 0 & 0 & 1 \end{bmatrix}$$

A.5 Random Sample Consensus

RANSAC (Random Sample Consensus) [33] is a method for calculating the most accurate solution when a larger amount of data is available than what is needed to solve a problem. For example, when computing a homography matrix we may have 100 keypoint matches between an image pair, when technically only four matches are needed to calculate the relating transformation. Some of this data may be noisy, meaning not every sample is correct. RANSAC is designed to choose the solution that aligns with a larger portion of the data than any other potential solution. It works by performing an arbitrary number, n, of iterations during each of which a random set of data is chosen that is used to calculate the solution to the problem. This random set will generally be the minimum number of data needed to solve the problem in question (i.e. 4 samples to solve a homography matrix and 6 samples to solve a projection matrix). The accuracy of the set is determined by how closely the solution fits the data not included in the current set of samples. After n iterations, the solution that fits more of the data than any other solution is chosen. All of the inlying data, or the data points that agree with the solution, are used for a refitting stage. We can identify the inlying sample points for a given solution using the approaches described in Section A.6. The final solution is recalculated using all of these points, as opposed to previous iterations where only a minimal number of points are used to calculate a solution.

A.6 Geometric Verification

Feature matches between a pair of images can be verified geometrically by solving a transformation matrix between the two images such as the homography, fundamental, or projection matrices. Using RANSAC, a random set of points can be chosen from the set of matches to construct a potential transformation matrix. The matrix determined to align most accurately with the data according to RANSAC can be used to identify the inliers and outliers in the match set. The inliers will be those matches that yield low reprojection errors (Section A.6.1) or symmetric transfer errors (Section A.6.2) using the transformation matrix. Determining the threshold for determining inliers can be decided experimentally and based on the application. Snavley et al. [29] use a threshold of 0.6% of the maximum image dimension (width or height). We choose to use the same threshold in many of our experiments.

A.6.1 Reprojection Error

Reprojection error is one measurement of the accuracy of a 3D pose estimate. It can be calculated when a set of point correspondences are known between different coordinate systems. These coordinate systems can be defined around spaces such as image planes or 3D point clouds. A transformation matrix can be calculated to define the relationship between different coordinate systems. Projection, fundamental, and homography matrices are all examples of this type of relationship. When a point in one coordinate system is multiplied by a transformation matrix, the output should be its matching point in the related coordinate system. However, in practice this will generally not be the case. The transformed point will probably just be close in space to the matching point. The reprojection error measures how close the transformed and matching points actually are. The sum of the Euclidean distances between all transformed points and the correspondences in the new coordinate system is calculated as is shown in Equation A.14 where n is the number of matches, T is the transformation matrix, and x_i and x_j are matching points.

$$re = \sum_{i=0}^{n} |x_i - T * x_j| \tag{A.14}$$

Ideally, a low reprojection error signifies that the transformation matrix is an accurate representation of the relationship between two spaces. It can be misleading though because many other factors may be at play. For instance, if many of the feature correspondences are inaccurate, the reprojection error will obviously convey no information about the relationship of the pose estimates. Also, if feature points are clustered in one part of the data in a coordinate system the reprojection error will not represent the overall structure. For example, if all the feature matches between an image pair are near one side of the image planes, the rest of the image may be distorted from radial distortion or sharp perspective changes and not match up at all using the transformation matrix. However, the reprojection error will not account for any of this.

A.6.2 Symmetric Transfer Error

Symmetric transfer error is another measurement of the accuracy of an aligning transformation. The only difference between symmetric transfer error and reprojection error is that the error is calculated in both directions. For example, if we are measuring the accuracy of a homography, we will first compute the Euclidean distance between a point transformed from one image ($H * x_1$) and it's matching point in the the second image (x_2). We will then compute the Euclidean distance between x_1 and x_2 after x_2 has been transformed back to the first image and the sum the two distances. Equation A.15 shows the equation for computing symmetric transfer error where n is the number of matches, T is the transformation matrix, and x_i and x_j are matching points.

$$re = \sum_{i=0}^{n} |(x_i - T * x_j)| + |(x_j - T' * x_i)| \tag{A.15}$$

Appendix B

Supplemental Experiments and Discussion of 2D Contextual Matching Methodology

B.1 Image Alignment and Keypoint Matching Results

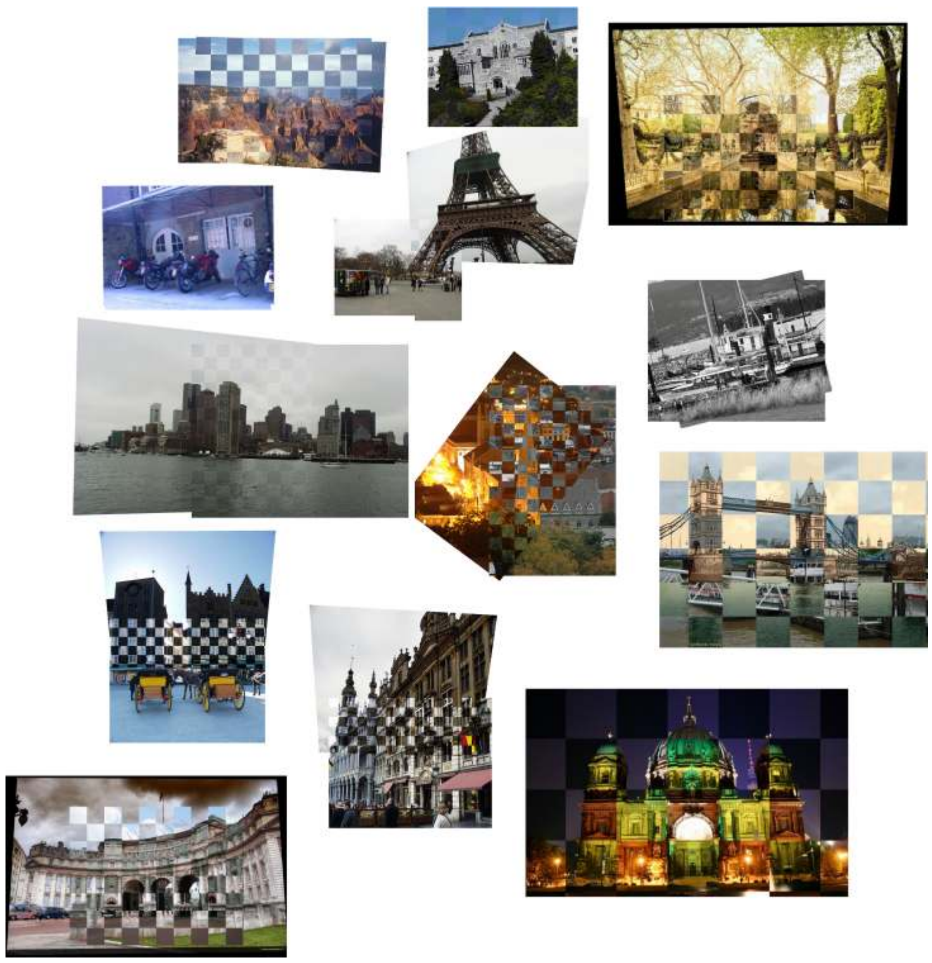


Figure B.1: Registration results after applying our pipeline to several public dataset image pairs presenting challenges including small image overlap, camera rotation, lighting changes, and .JPEG compression. Images provided courtesy of [2], [3], [5] and [7].

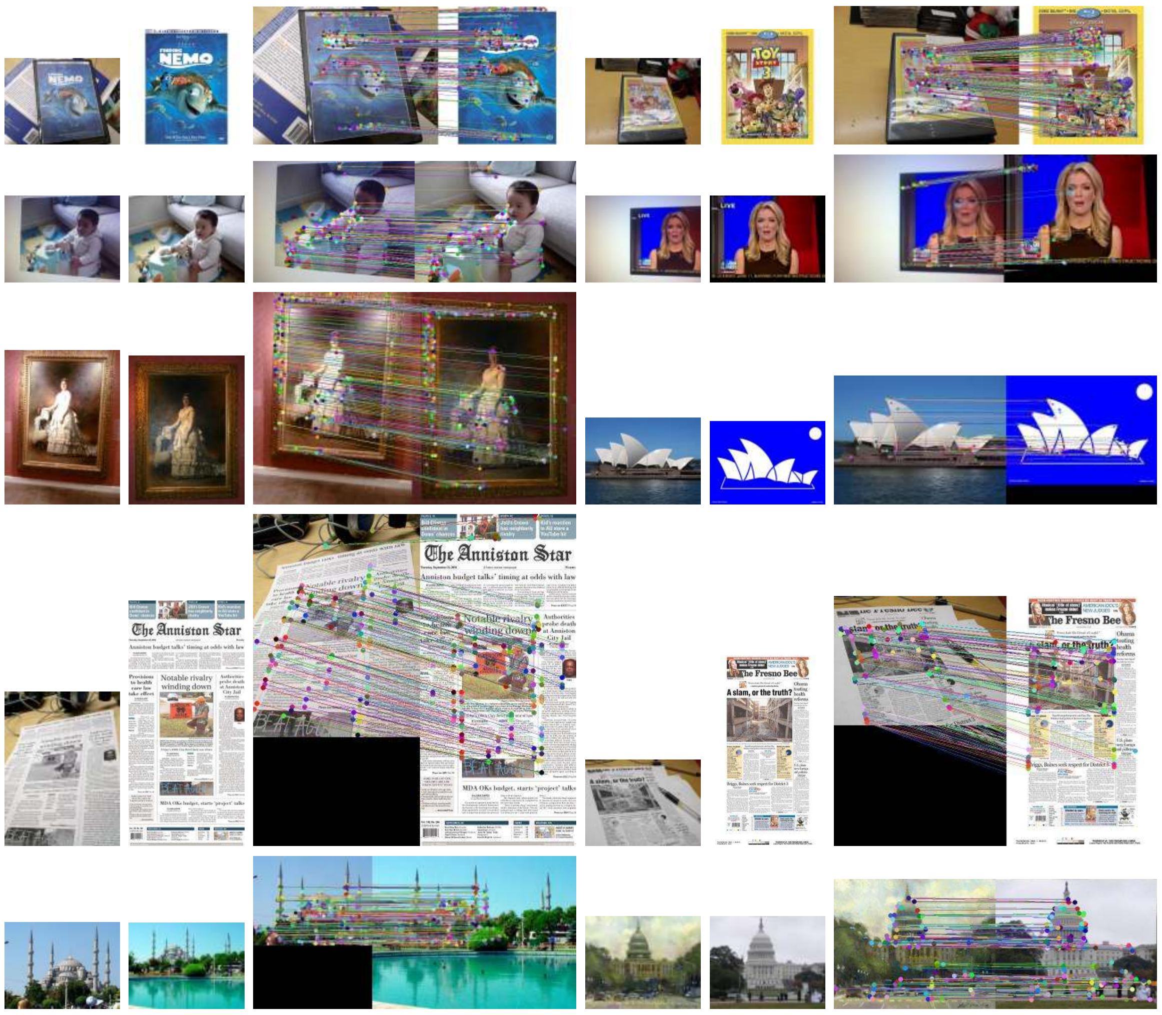


Figure B.2: Keypoint matches for images from Stanford Mobile Visual Search dataset [6] and the dataset provided in [5].

B.2 Comparison with Other Methods

We compared each of our three matching techniques to other matching methods including MSER with SIFT descriptors, SIFT Flow [8], and local symmetry features [7] on the symmetry dataset. Following the local symmetry paper, we tried matching the images using only symmetry descriptors and symmetry and SIFT descriptors concatenated. These results are shown in Figure B.4. A few visual examples from this test are displayed in Figure B.3.

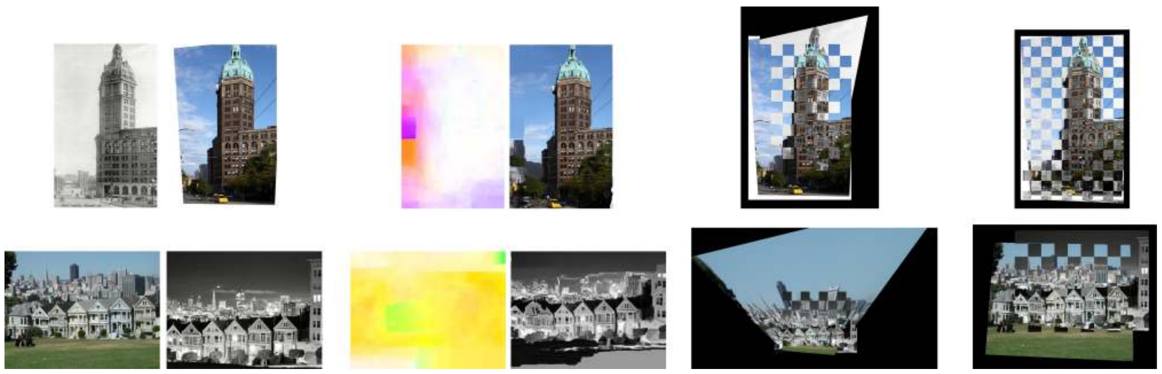


Figure B.3: Registration results after applying our pipeline to several symmetry database pairs. Images provided courtesy of [7]. Top: worldbuilding, Bottom: paintedladies18. *From left to right:* The original image pair, SIFT Flow results, stitched images using the SIFT+Symmetry method described in [7], and stitched images using our pipeline. For the SIFT Flow results, we show the colored pixel displacement map on the left and the I_2 warped to I_1's image plane on the right.

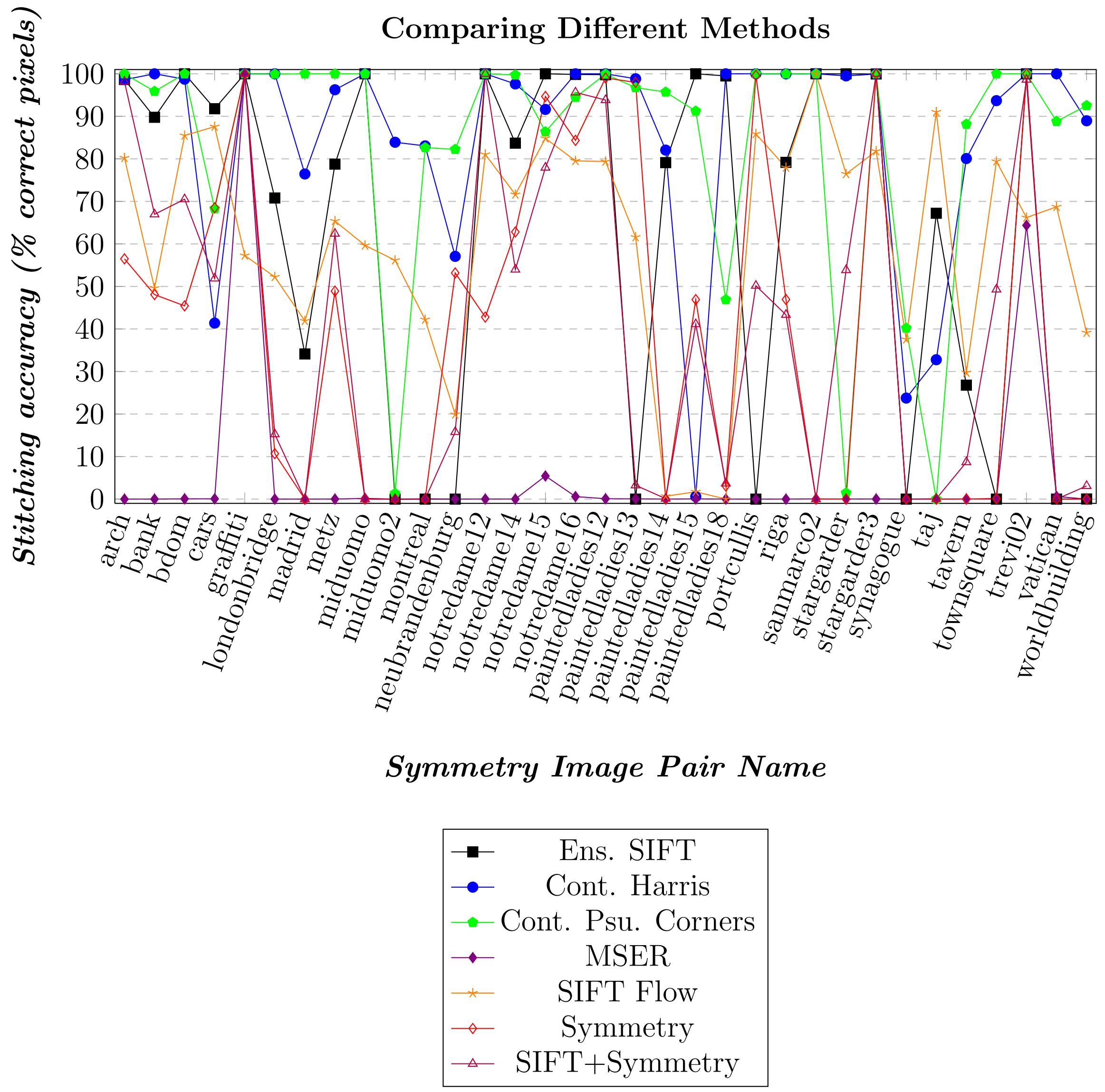


Figure B.4: Comparing our ensemble methods to other matching techniques. We show in this chart accuracy measures for aligned image pairs using Ensemble SIFT-based features, Contextual Harris Laplacian features, Contextual Pseudo Corners, MSER with SIFT descriptors, SIFT Flow [8], and Local Symmetry Features [7]. For the Local Symmetry Features method, we tested using just the symmetry descriptor and the symmetry descriptor concatenated with each feature point's SIFT descriptor as described in [7].

B.3 True Positive/False Positive Rates

We created several non-matching image pairs from various categories of the Stanford Mobile Visual Search dataset [6] to test the true positive and false postive rates of our methods. We used images from the museum paintings, dvd covers, print, landmarks, and video frames sub-datasets. We chose two pairs of query and reference images from each dataset and matched each query image to the ten selected reference images. If the most geometrically verified matches (using a solved homography) were found between the correctly matching query and reference pair, the image was marked as being a true positive. If a non-matching pair had more matches than a correct matching pair with the same query image, it was marked as a false positive. The results of testing our contextual methods in this manner are shown in Table B.1. We repeated this test focusing on architectural buildings from the MPEG CDVS Landmark Dataset. We chose 60 pairs of images from the the Oxford Buildings Dataset [4], the Zurich Building Image Dataset [172], and the Stanford Mobile Visual Search dataset [6]. Thumbnails of the query and reference image pairs we tested are provided in Figure B.5. These images present obstacles for matching including changes in lighting, perspective, scale, rendering style, and image blurring. Table B.2 shows the the true and false positive rates we obtained after matching each of the 60 selected query images to each of the 60 reference images using our three different contextual approaches.

Table B.1: True Positive/True Negative Rates of Ensemble Methods on Stanford Mobile Visual Search Dataset

-	True Positive	False Positive
Ensemble SIFT	1.0	0.0
Cont. Harris-Lap.	0.7	0.111
Cont. Psu. Corners	0.8	0.1

Table B.2: True Positive/False Positive Rates of Ensemble Methods on Landmark Dataset

-	True Positive	False Positive
Ensemble SIFT	0.883	0.027
Cont. Harris-Lap.	0.833	0.056
Cont. Psu. Corners	0.833	0.084

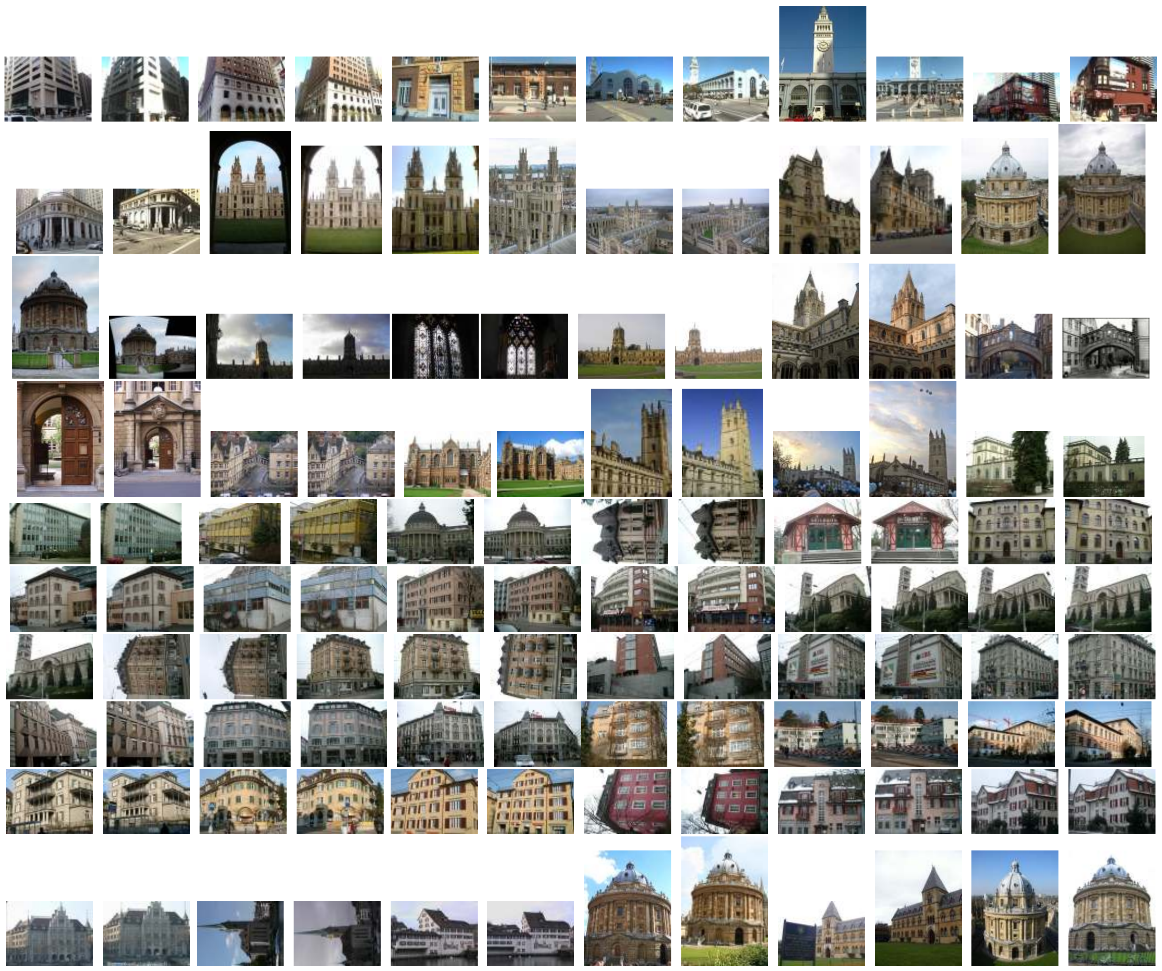


Figure B.5: Image pairs from Landmark dataset that are used for True Positive/False Positive tests in Table B.2.

B.4 Complexity

In Table B.3 we have outlined the complexity of different stages of our contextual matching algorithm. We present the complexity in terms appropriate for each stage whether they be number of pixels, lines, features, or matches involved in the method. The complexity of assigning pixels to MSER-based neighborhoods is linear in terms of the number of pixels in the image as each is assigned to a region in addition to the same complexity of actually extracting MSER's [173]. Each pixel is visited once to choose from which level of the Gaussian Pyramid it should choose an MSER. The complexity of extracting line segments using LSD is proportional to the number of pixels in the image as described in [35]. When we extract Pseudo Corners using line segments, we check the intersection of each pair of line segments within a defined distance from each other. The complexity of doing this is at most $O(l^2)$ in terms of the number of lines. To assign line segments to nearby keypoints to create contextual linear descriptors we must compute the distance of each line l to each keypoint feature f. Matching contextual linear descriptors is that of Hungarian Graph Matching which is $O(l^3)$. To construct contextual HOG descriptors, we need to traverse every pixel within a defined neighborhood surrounding each keypoint. The complexity of these descriptors depends on the approach being used for matching. For Ensemble SIFT, contextual descriptors are compared for each potential SIFT match that falls within a specified ratio range, making the matching linear in terms of the number of SIFT matches. For our contextual corner methods, each corner feature f in I_1 is compared to each corner feature g in I_2, giving us a complexity of $O(f * g)$. For our ICP refinement, each feature in I_1 is compared to every feature in I_2 within a specified ratio, giving this stage, at worst, also a complexity of $O(f * g)$. For our global

Levenberg-Marquardt refinement, we check the gradient magnitude at each pixel in I_2 that has a corresponding line feature in I_1, making this stage linear in terms of the number of features whose gradients are checked.

Many of these stages are dependent on each other. For instance, the number of Pseudo Corners we extract for Contextual Pseudo Corner matching and the time it takes is related to the number of line segments we have identified in an image pair. The time for extracting line segments, and likely the number of line segments we have for finding corners, is guided by the number of pixels in, or resolution of, the images.

As the resolution of the image increases, it is likely that we will find more line segments and more keypoints for all three of our Contextual Matching approaches and for image alignment refinement. Since the main stages of our algorithm are dependent on the number of pixels and features in an image, it will generally take our program longer to perform matching on higher resolution images. This increase in runtime and number of features with higher resolution images can be observed in Table B.4. Since higher resolution images tend to contain more features and finer details, our matching pipeline has more information to perform accurate matching than it generally does with lower resolution images.

Table B.3: Complexity of Different Stages of Our Pipeline

Stage	**Paper Section with Explanation**	**Complexity**
Assigning pixels to MSER-based neighborhoods	II. A.	$O(p) + O(p \log \log p)$
Extracting line segments (LSD)	II. B. and II. F.	$O(p)$
Extracting pseudo-corners	II. F.	$O(l^2)$
Creating linear contextual descriptors	II. B.	$O(f * l)$
Matching linear contextual descriptors (Ensemble SIFT)	II. B.	$O(l^3)$
Creating HOG contextual descriptors	II. C.	$O(f * q)$
Matching HOG contextual descriptors (Ensemble SIFT)	II. D.	$O(m)$
Matching contextual descriptors (Contextual Corners)	II E. and II. F.	$O(f * g)$
Refining with ICP	III.	$O(f * g)$
Refining with Levenberg-Marquardt	III.	$O(f)$

Notation: **p - number of pixels in image, q - number of pixels in local neighborhood, l - lines, f - features (in image 1), g - features (in image 2), m - SIFT matches**

B.5 Runtimes

Table B.4 shows the runtimes of various components of our pipeline. All tests were performed using a Dell Optiplex 7010 desktop with 7.7 GB of memory and a 3.2 GHz Intel Core i5-3470 processor running the Ubuntu 12.04 64-bit operating system. We ran all of our contextual methods on the Symmetry Dataset [7]. The timing for each method and stage in Table B.4 is the average runtime in seconds to complete the stage on all of the images in the dataset. The resolutions of these images range from 640x350 to 889x1221. The top left quadrant of Table B.4 shows the timings of stages of our pipeline that are used regardless of the descriptor and matching technique used. The top right shows timings for stages that are used specifically by our Ensemble SIFT method, and the bottom two quadrants show the timings for our Contextual Harris-Laplacian and Pseudo Corner methods. In our Ensemble SIFT method, we compare contextual descriptors of all potential SIFT matches within a specified distinctiveness ratio. The contextual descriptors for a keypoint are only compared to the contextual descriptors for its potentially matching keypoint. For this reason, in the Ensemble SIFT section of the table, we report the average pair-wise times to compare two linear contextual descriptors with each other and two HOG contextual descriptors with each other. We also include the total average time to match all of the SIFT features using SIFT descriptors and contextual descriptors according to our ensemble approach. For the Contextual Corner methods, each corner in I_1 is compared to all the corners in I_2. For both Contextual Harris-Laplacian and Contextual Pseudo Corners, we have included the average time to find the closest match for a single keypoint in I_1 in I_2. We have also documented the total time it takes to complete the matching, including matching all corners in I_1 to all corners in I_2. Notice the average times vary between the two Contextual Corner methods in relation to how many corners are identified. We use the OpenCV library [9] to extract SIFT features and to perform matching and compute homographies for all three versions of our contextual work. To extract Harris-Corners, we use the VLFeat library [10]. We have also included the runtimes of publicly available code for several of the methods we compare our work to throughout this section in Table B.5.

Table B.4: Average Runtimes in Seconds and Number of Features for Different Stages of Our Pipeline on Symmetry Dataset

General Steps	-	**Ensemble SIFT**	-
MSER Neighborhood Assignment	8.188724	SIFT Feature Detection	0.403183
LSD Extraction	0.827009	*Number of SIFT Features*	*4574*
Number of Lines	*1128*	SIFT Feature Matching	1.104771
Individual Linear Descriptor Construction	0.003683	Pair-wise Linear Descriptor Comparison	0.000004
Individual HOG Descriptor Construction	0.004568	Pair-wise HOG Descriptor Comparison	0.004069
ICP Optimization	447.459381	Total Ensemble SIFT Matching	54.007973
LM Optimization	34.251392	Homography Estimation	0.021649
Contextual Harris-Laplacian	-	**Contextual Pseudo Corners**	-
Harris-Laplacian Corner Extraction	2.413574	Pseudo Corner Extraction	0.826131
Number of Harris-Laplacian Corners	*1310*	*Number of Pseudo Corners*	*625*
HOG Descriptor Matching	2.381177	HOG Descriptor Matching	0.450085
Linear Descriptor Matching	31.772003	Linear Descriptor Matching	6.271157
Total HOG Contextual Harris-Laplacian Corner Matching	6.870990	Total HOG Contextual Pseudo Corner Matching	2.647096
Total Linear Contextual Harris-Laplacian Corner Matching	76.265503	Total Linear Contextual Pseudo Corner Matching	28.250336
Homography Estimation	0.072110	Homography Estimation	0.067457

Table B.5: Average Runtimes in Seconds and Number of Features for Other Methods on Symmetry Dataset

SIFTFlow	-
Dense SIFT	0.250315
SIFT Flow Matching	18.869501
Symmetry	-
Intensity-Based Detector	6.924392
Number of Intensity-Based Features	*1625*
Gradient-Based Detector	14.648718
Number Gradient-Based of Features	*309*
Symmetry Descriptor Constructor	11.240939

Table B.6 shows runtimes and the number of features found for individual image pairs in the Symmetry dataset. This is a more detailed version of the average runtimes presented in the above table and correspond to the same set of tests. Table B.5 shows the average runtimes for the same set of images using publicly available code for some of the methods we compared our work to in the evaluation section in Chapter 3.

Table B.6: Runtimes in Seconds and Number of Features for Our Method on Symmetry Dataset

Image Pair Name	Resolution	LSD Runtime	Number Lines Found	SIFT Detection Time	Number SIFT Found	Ensemble SIFT Runtime	Harris Laplacian Detection Time	Number Harris Corners Found	Contextual Harris Runtime	Pseudo Corner Detection Time	Number of Pseudo Corners	Contextual Pseudo Runtime
arch	760x457	0.578489	831	0.303273	3640	33.340401	1.460975	957	19.519281	0.449897	775	13.299746
bank	503x688	0.540794	1504	0.233044	2405	22.738800	1.390905	806	13.697970	1.144807	873	10.557721
bdom	1000x667	0.929965	755	0.403952	3806	67.593399	2.428175	1109	37.255890	0.469433	952	25.434511
cars	921x614	0.662509	923	0.288606	2248	14.823300	2.168925	400	8.437840	0.583067	411	6.138016
graffiti	800x640	0.924307	2403	0.344087	3216	55.935699	1.414420	1086	31.359785	2.156555	1137	22.710155
londonbridge	826x611	0.681589	905	0.349185	3776	31.878901	2.284855	1072	19.290495	0.488789	852	13.084978
madrid	478x703	0.629675	1112	0.293336	3480	31.383900	1.802265	1166	19.395626	0.621716	994	13.526447
metz	657x916	0.877384	1137	0.455136	4944	87.614197	2.809455	1328	48.179680	0.479380	1005	32.156914
miduomo	816x640	0.623738	775	0.307380	2864	67.268898	1.290895	795	35.929741	0.453963	661	24.267370
miduomo2	1000x667	1.134530	695	0.560695	6389	54.511299	3.500555	2878	39.406860	0.358904	2034	25.410034
montreal	484x698	0.577507	1270	0.286014	3462	32.031200	1.734345	1055	19.271891	0.826298	917	13.455502
neubrandenburg	750x1000	1.231780	943	0.799051	10566	44.487900	3.580110	2722	33.382317	0.431724	1879	21.285355
notredame12	460x600	0.477276	1039	0.255082	3085	51.568401	1.305980	976	28.745270	0.582321	844	19.722622
notredame14	460x600	0.441070	996	0.230541	2683	37.765900	1.231155	826	21.274130	0.533563	732	14.735154
notredame15	460x600	0.442573	1032	0.232740	2717	53.311600	1.233215	839	29.088530	0.588377	753	20.012655
notredame16	460x600	0.460450	1069	0.237539	2802	52.370098	1.299830	987	29.208084	0.622654	864	20.087568
paintedladies12	618x468	0.433365	1194	0.210036	2392	38.959202	1.462385	844	21.946136	0.703804	817	15.565760
paintedladies13	618x468	0.448844	1187	0.214946	2452	31.414301	1.513165	946	18.556835	0.707964	890	13.296378
paintedladies14	618x468	0.503865	1620	0.236839	2819	22.109699	1.673410	1227	14.959524	1.194898	1167	11.273854
paintedladies15	618x468	0.484457	1454	0.241695	2789	32.071201	1.606995	1118	19.545431	0.989491	1057	14.166041
paintedladies18	618x468	0.484548	1240	0.233896	2765	25.764601	1.615500	1105	16.340805	0.739267	991	11.723385
portcullis	767x449	0.555083	1418	0.340527	4195	37.308102	1.651170	769	20.814186	1.079212	794	14.966271
riga	914x674	0.855716	1320	0.402044	3756	76.622902	2.761095	1128	41.795372	0.981426	1022	28.786160
sanmarco2	1035x775	1.447300	1458	0.738580	8331	105.390999	3.856660	2130	60.683716	0.977927	1743	41.180325
stargarder	889x1221	1.889775	810	1.469655	18271	140.912994	5.913735	3739	87.999245	0.493061	2574	57.050255
stargarder3	849x927	1.657560	1171	1.039945	14261	120.709000	5.314855	4501	84.398499	0.667714	3123	54.107925
synagogue	722x1000	0.986614	1408	0.501303	5520	33.129501	2.568010	1071	19.554756	1.018296	977	13.932168
taj	786x1000	0.887520	532	0.359371	2586	61.190399	2.547490	693	32.520771	0.346009	551	21.852446
tavern	496x698	0.583837	1614	0.245502	2629	25.937700	1.720255	1511	17.980274	1.230275	1379	13.344784
townsquare	645x440	0.502801	903	0.230761	2600	27.007200	1.377270	1116	16.968115	0.513886	907	11.640694
trevi	1024x768	1.389885	1288	0.645386	6665	186.410995	2.362840	573	94.774231	0.924814	637	64.223030
vatican	640x350	0.279302	617	0.120421	1088	18.425600	0.760408	272	9.925074	0.332963	286	6.987846
worldbuilding	630x1000	0.873211	590	0.473391	5716	60.274700	2.541705	1488	35.011494	0.347677	1091	23.220135

B.6 Quantitative Registration Results

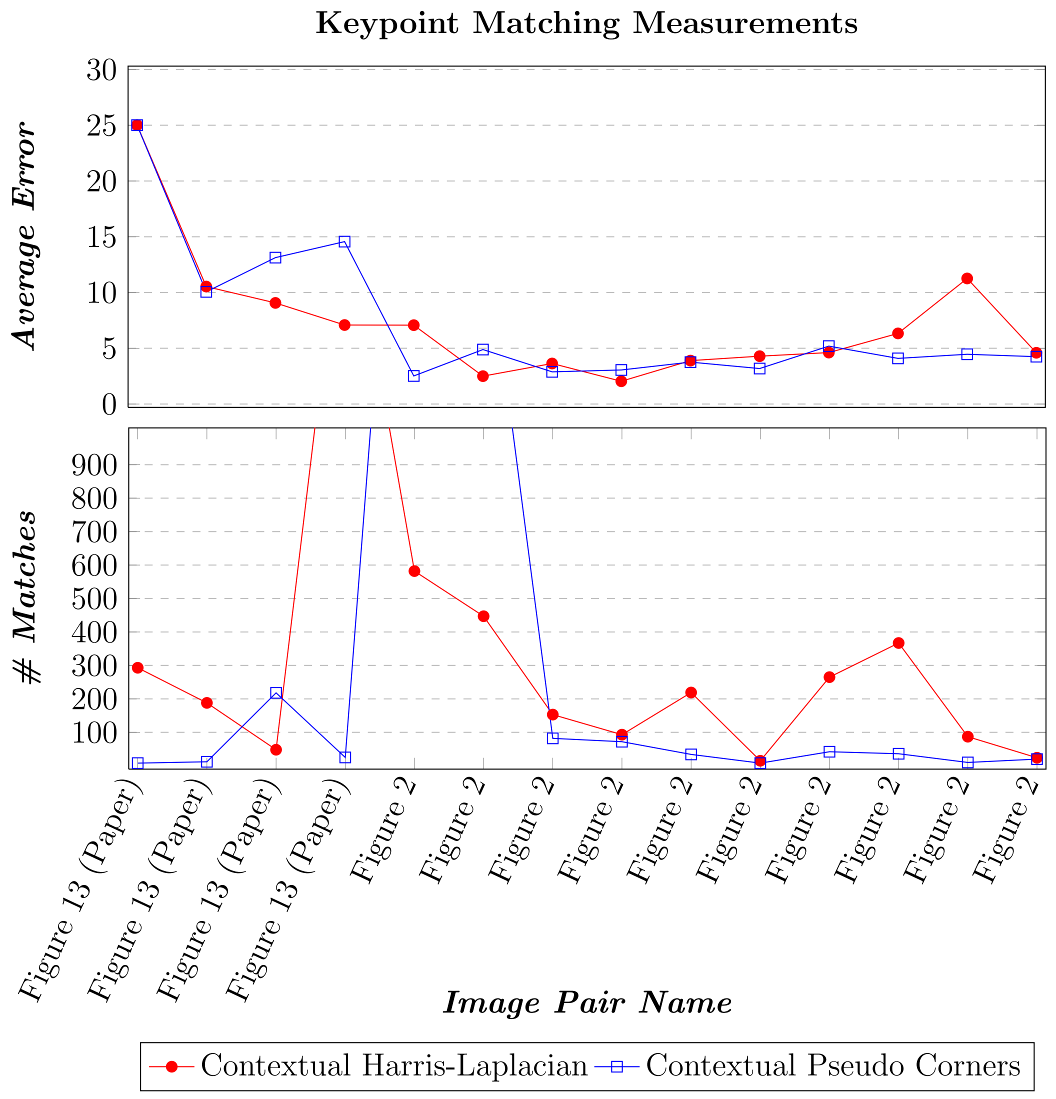


Figure B.6: Image pair alignment accuracy and number of matches found using Contextual Harris-Laplacian and Contextual Pseudo Corners on various datasets. The results for Figure 13 correspond to the images shown in Figure 13 in Chapter 3 from left to right and come from the dataset provided by [5]. The results for Figure B.2 correspond to the images in Figure B.2 in this document from left to right and from top to bottom. These images come from the datasets provided by [5] and [6].

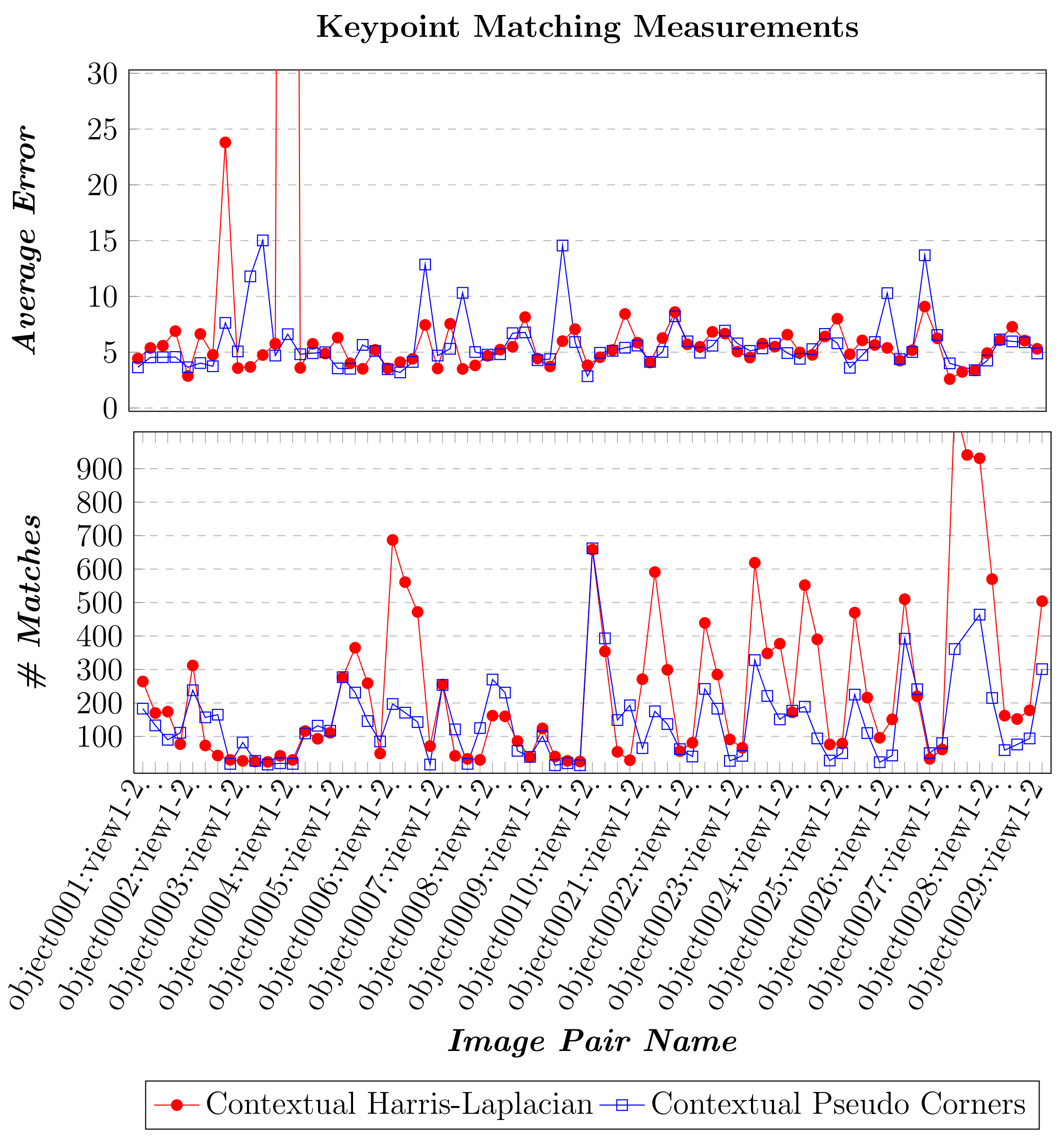


Figure B.7: Image pair alignment accuracy and number of matches found using Contextual Harris-Laplacian and Contextual Pseudo Corners on images from the png-ZuBuD dataset [1]. Images object0001-view1:2 through object0029-view1:5 are images of buildings in Zurich from various view points. Each building is assigned an object number and there are five views of each building. We matched view 1 to view 2, view 1 to view 3, and so for each building set for objects 1-10 and 21-29 in this dataset.

Appendix C

Supplemental Experiments and Discussion of Matching Repetitive Data

This section contains the tests described in the Results section of Chapter 4 on additional datasets. We have also included additional data about our test set and the ground truth used for evaluation.

- Figures C.1 and C.2 show our tests demonstrating the usefulness of dimension reduction.
- Figures C.3 and C.4 show the results of our two-step approach.
- Figures C.5 and C.6 show the image pair alignment accuracies, keypoint match precision values, and numbers of matches for the specified images.
- Figures C.7 and C.8 show thumbnails of our test datasets.

- Figure C.9 discusses how rectification errors affect our ground truth and evaluation.

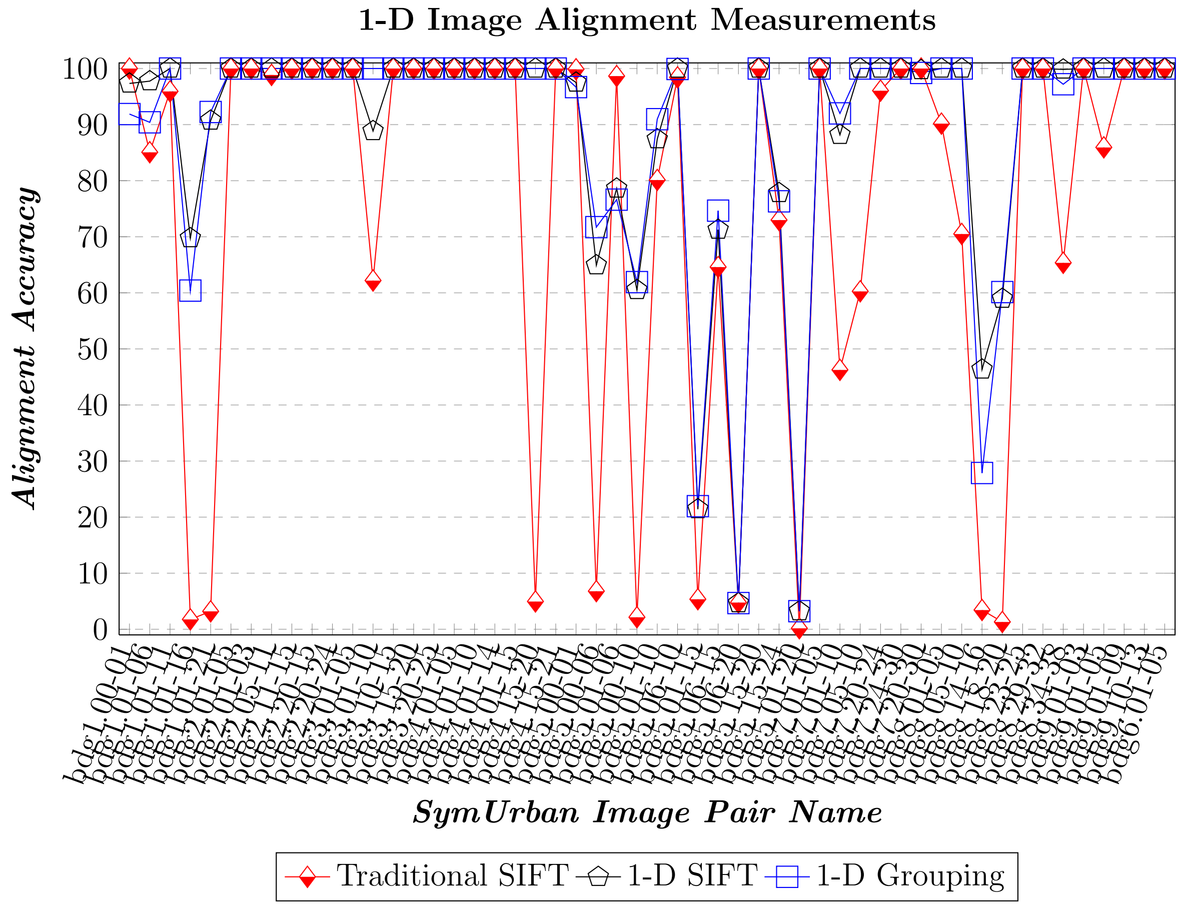


Figure C.1: Test of our proposed dimension reduction. Here we compare the 1-dimensional accuracy (along the y-axis) of different methods. The first is using traditional SIFT matches with the 8-point algorithm to compute a full 8 degree of freedom homography. The second approach is to use traditional SIFT matches, but compute a 1-D, 2 degree of freedom homography (as described in Chapter 4) using just the y-coordinates of the SIFT matches. The third approach is the grouping method described in Chapter 4 to compute a 1-D vertical homography.

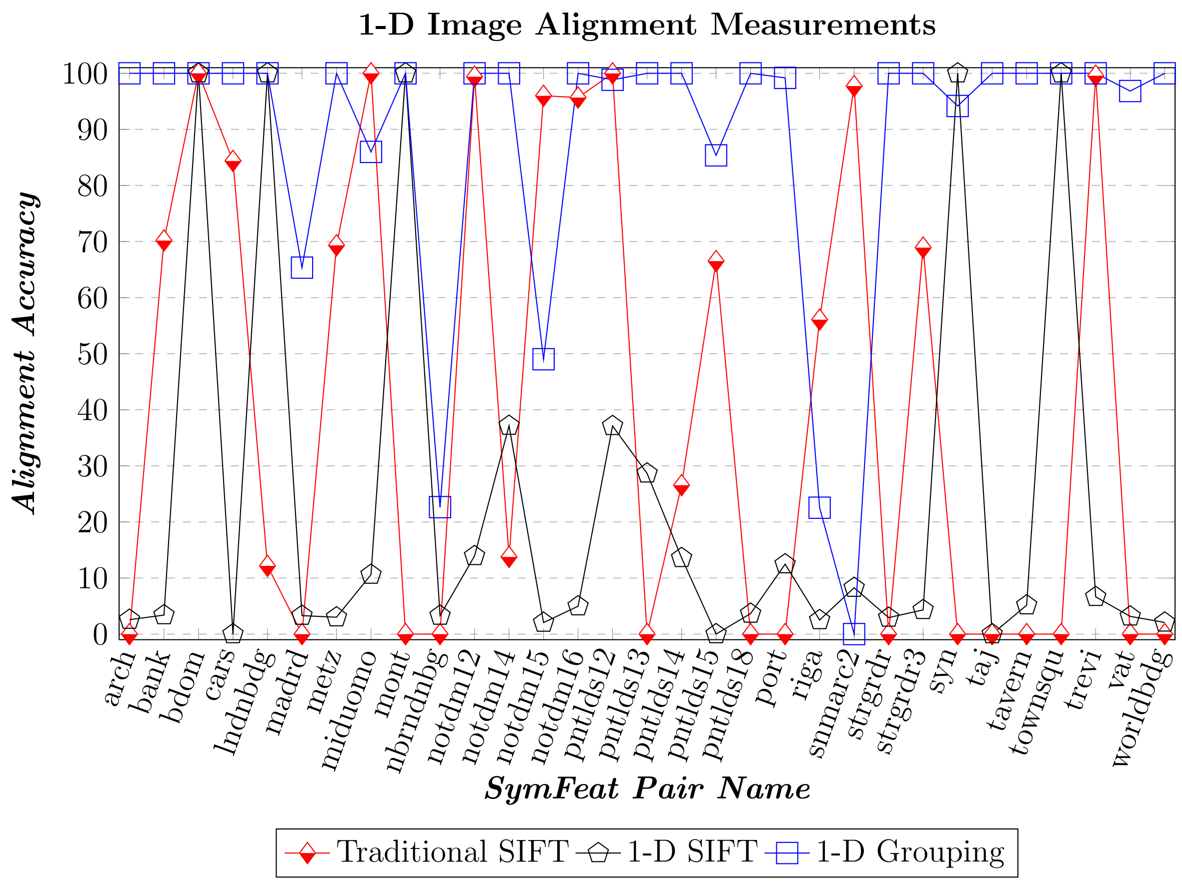


Figure C.2: Test of our proposed dimension reduction. Here we compare the 1-dimensional accuracy (along the y-axis) of different methods. The first is to use traditional SIFT matches with the 8-point algorithm to compute a full 8 degree of freedom homography. The second approach is to use traditional SIFT matches, but compute a 1-D, 2 degree of freedom homography (as described in Chapter 4) using just the y-coordinates of the SIFT matches. The third approach is the grouping method described in Chapter 4 to compute a 1-D vertical homography.

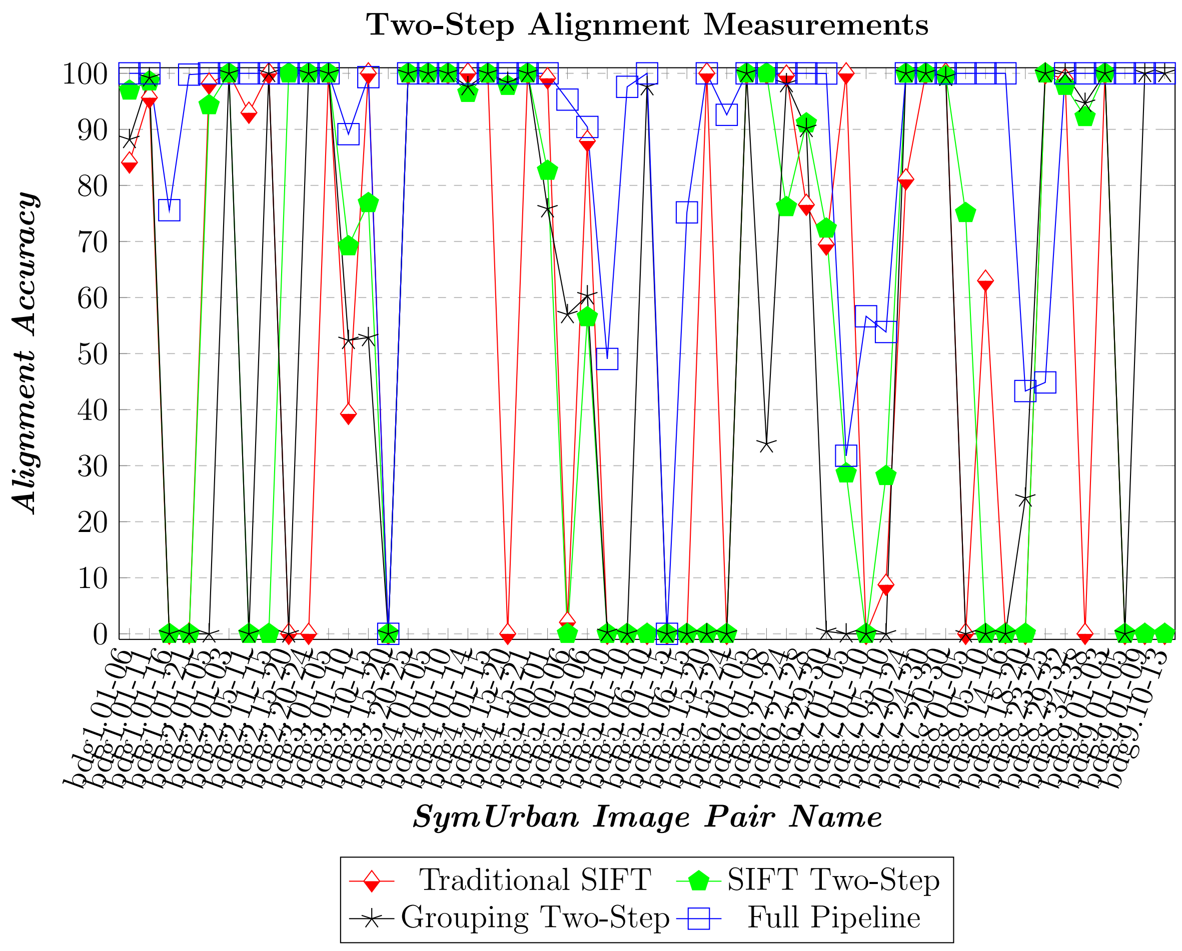


Figure C.3: Test of our proposed two-step method on the SymUrban dataset. As a baseline, we show the accuracy of using traditional SIFT matching to compute a standard 8 DOF homography (*Traditional SIFT*). We then show tests for using 1) SIFT features (*SIFT Two-Step*) and 2) grouping features (*Grouping Two-Step*) to compute a 1-D vertical homography. For both of these tests, we matched SIFT features along corresponding rows to vote for a horizontal 1-D homography. To show the strength of our full pipeline, we have also included in this chart the results of using the two-step approach with the intra-image saliency and pair-wise match saliency computations (*Full Pipeline*).

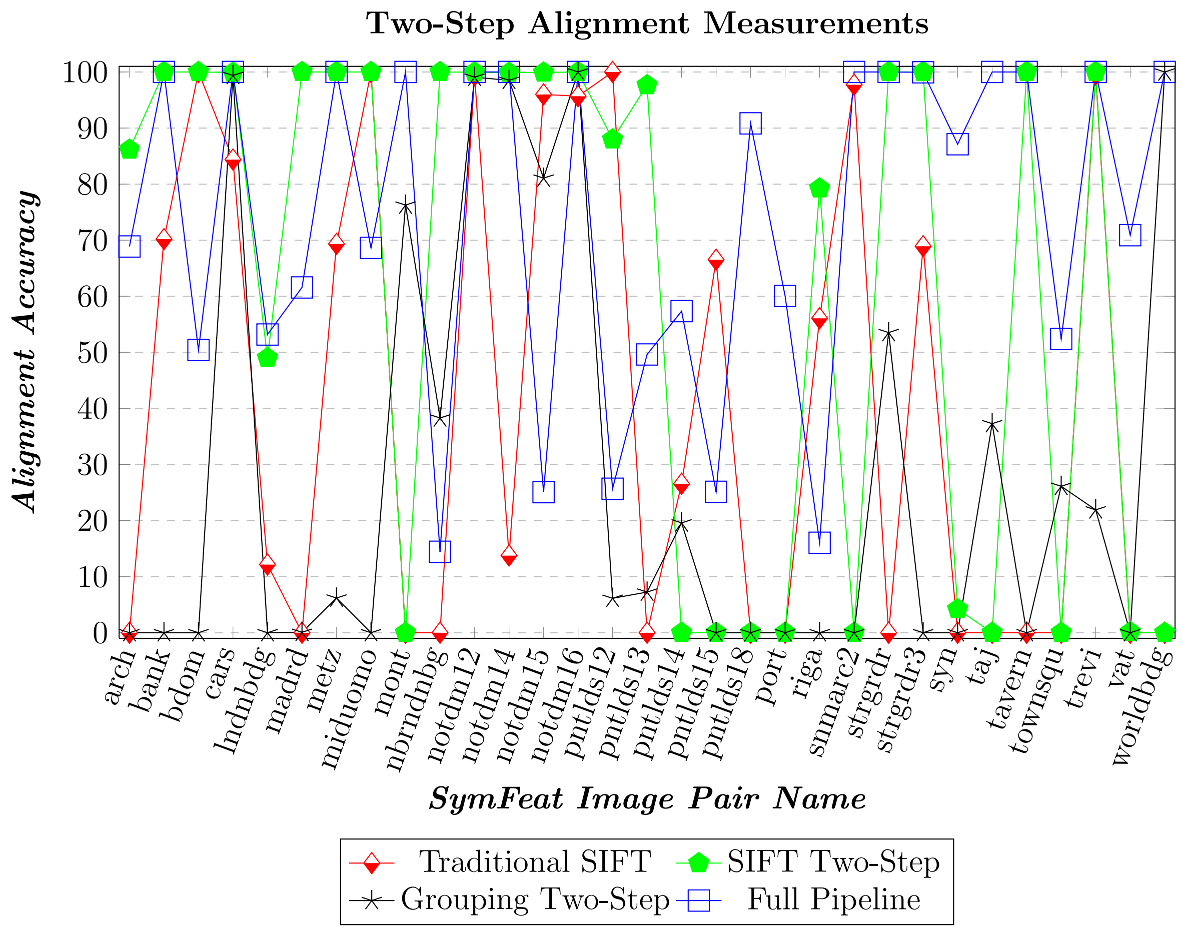


Figure C.4: Test of our proposed two-step on the SymFeat dataset. As a baseline, we show the accuracy of using traditional SIFT matching to compute a standard 8 DOF homography (*Traditional SIFT*). We then show tests for using 1) SIFT features (*SIFT Two-Step*) and 2) grouping features (*Grouping Two-Step*) to compute a 1-D vertical homography. For both of these tests, we matched SIFT features along corresponding rows to vote for a horizontal 1-D homography. To show the strength of our full pipeline, we have also included in this chart the results of using the two-step approach with the intra-image saliency and pair-wise match saliency computations (*Full Pipeline*).

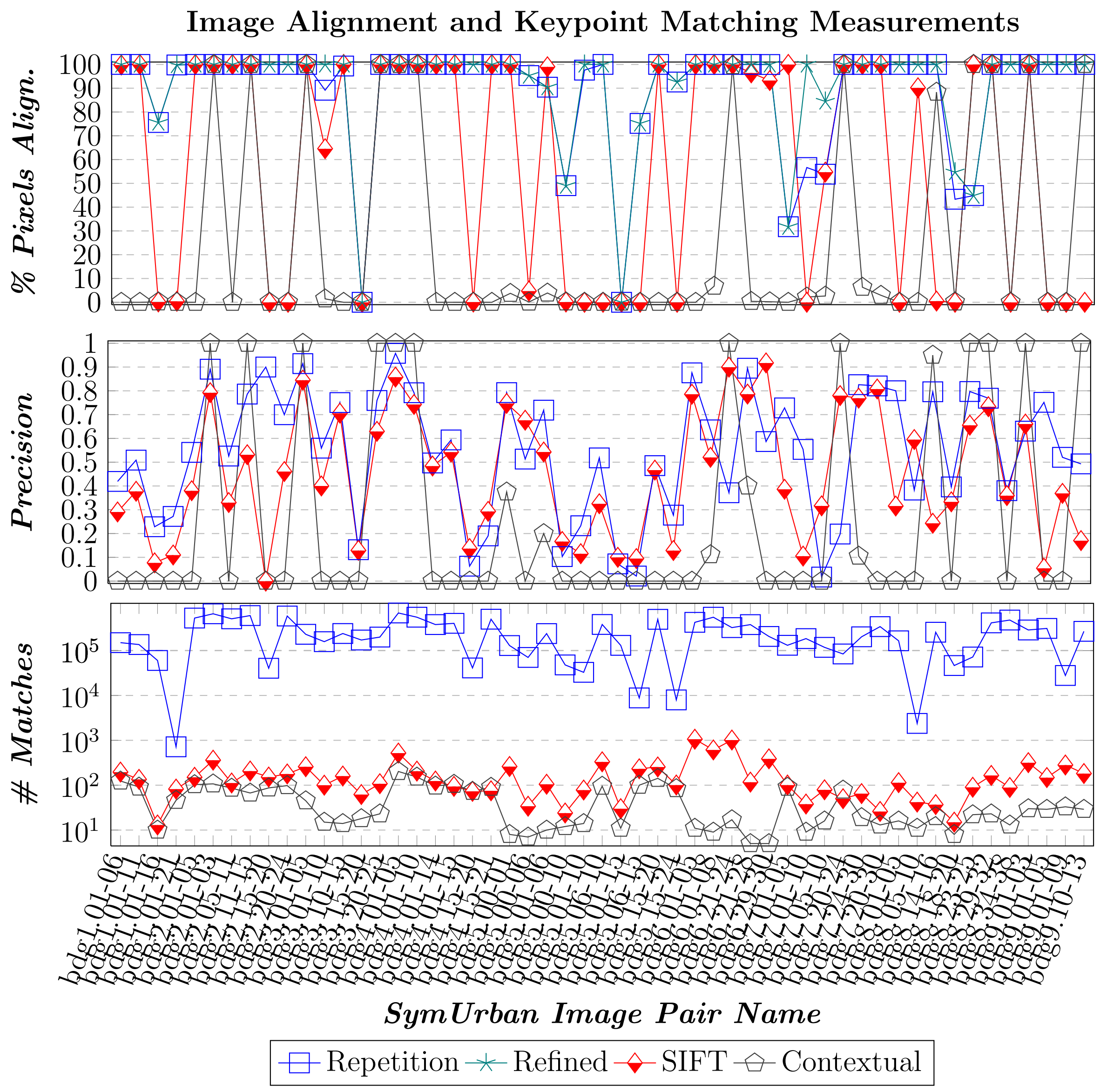


Figure C.5: Quantitative results for stitching image pairs. *Top:* Percentage of pixels in image plane that are correctly aligned after using solved homography. *Center:* Percentage of matches that are identified using different methods that are correct. *Bottom:* Number of matches identified using several different methods. Here we show the result of using 1) our proposed pipeline with Pseudo Corners as the original anchor points for grouping (*Repetition Pseudo*), 2) our proposed pipeline after refinement is applied (*Refined Pseudo*), 3) using traditional SIFT extraction and matching with the 8 point algorithm + RANSAC (*SIFT*), and 4) the contextual method mentioned in Chapter 4 (*Contextual*).

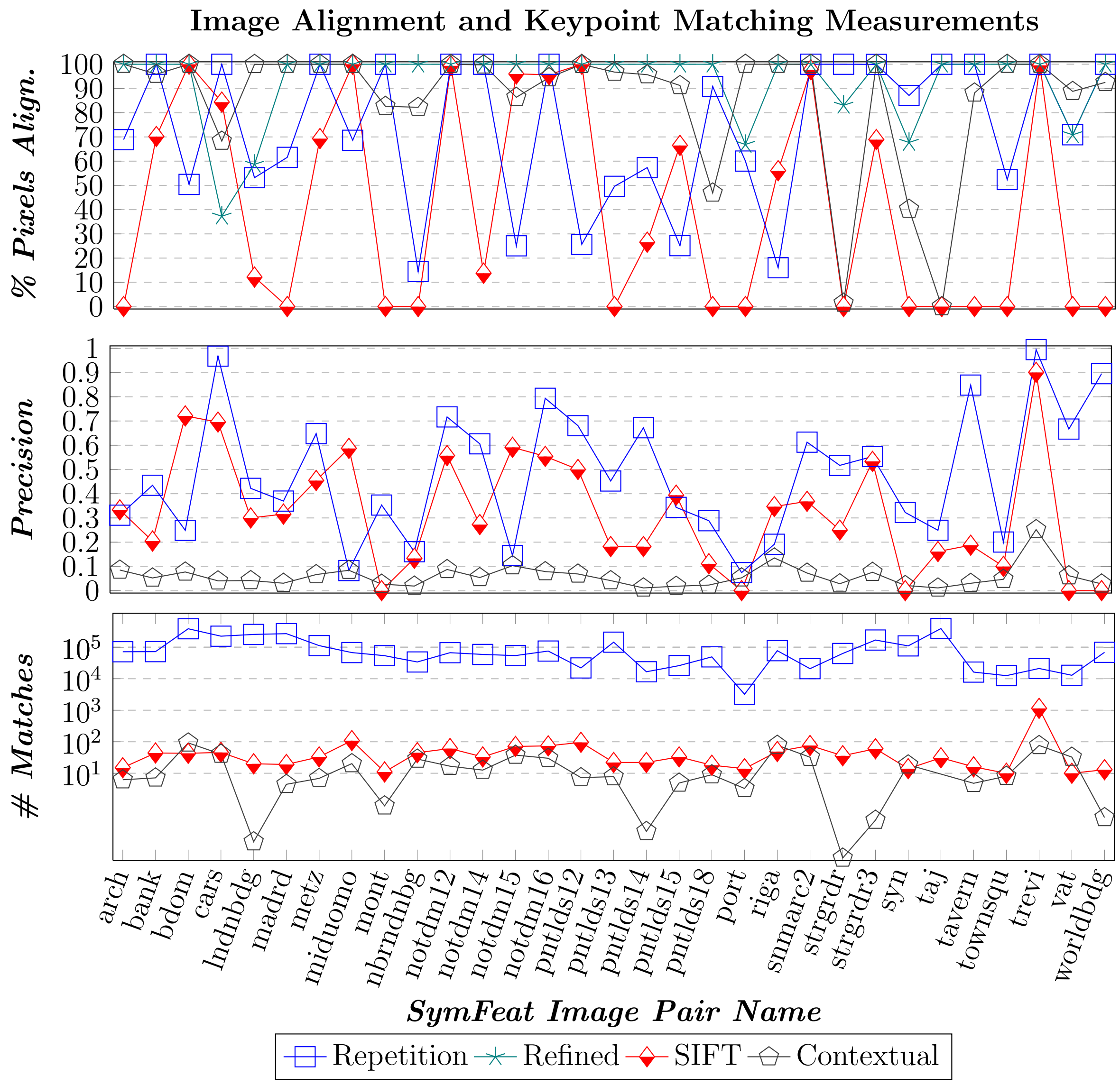


Figure C.6: Quantitative results for stitching image pairs. *Top:* Percentage of pixels in image plane that are correctly aligned after using solved homography. *Center:* Percentage of matches that are identified using different methods that are correct. *Bottom:* Number of matches identified using several different methods. Here we show the result of using 1) our proposed pipeline with Pseudo Corners as the original anchor points for grouping (*Repetition Pseudo*), 2) our proposed pipeline after refinement is applied (*Refined Pseudo*), 3) using traditional SIFT extraction and matching with the 8 point algorithm + RANSAC (*SIFT*), and 4) the contextual method mentioned in Chapter 4 (*Contextual*).

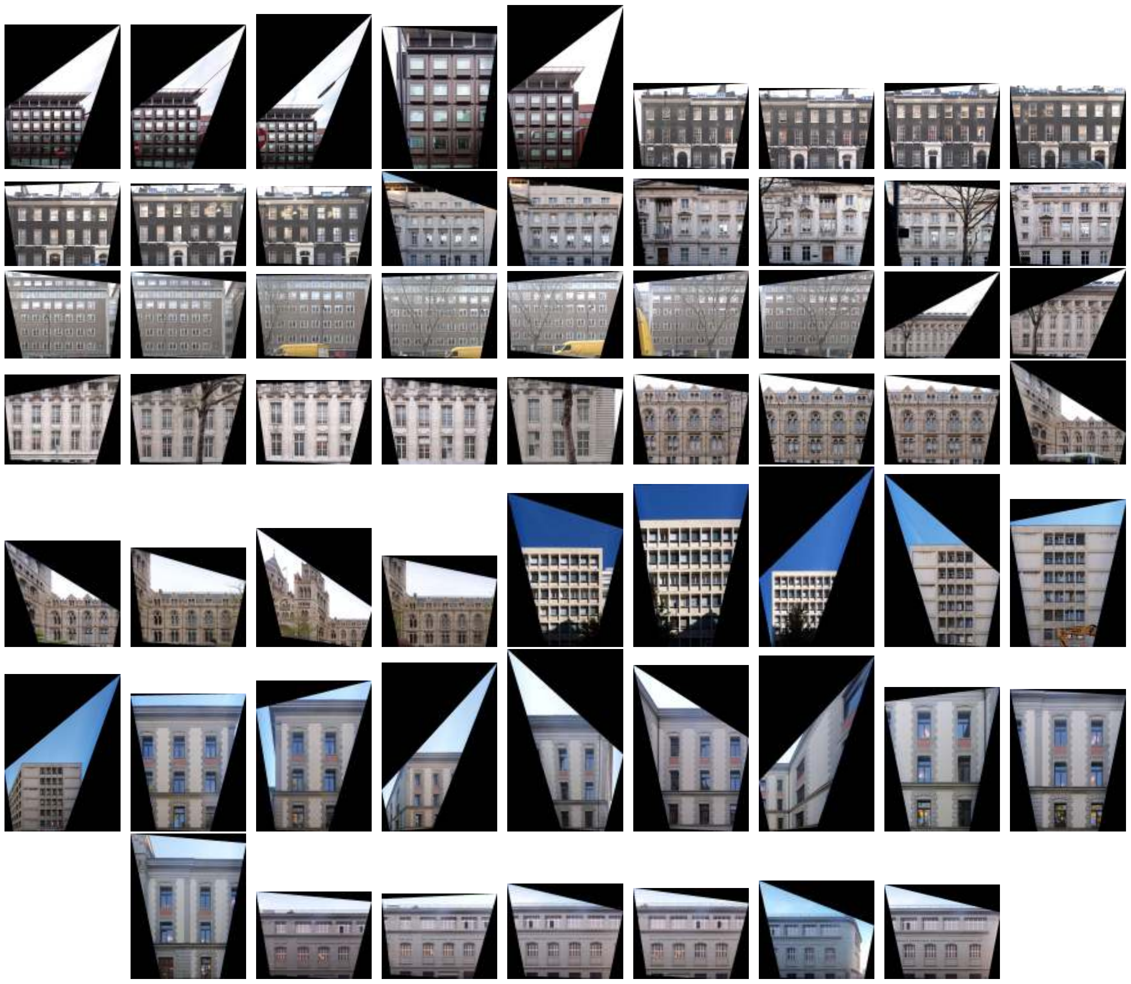


Figure C.7: Images we use for testing from SymUrban dataset discussed in Chapter 4.

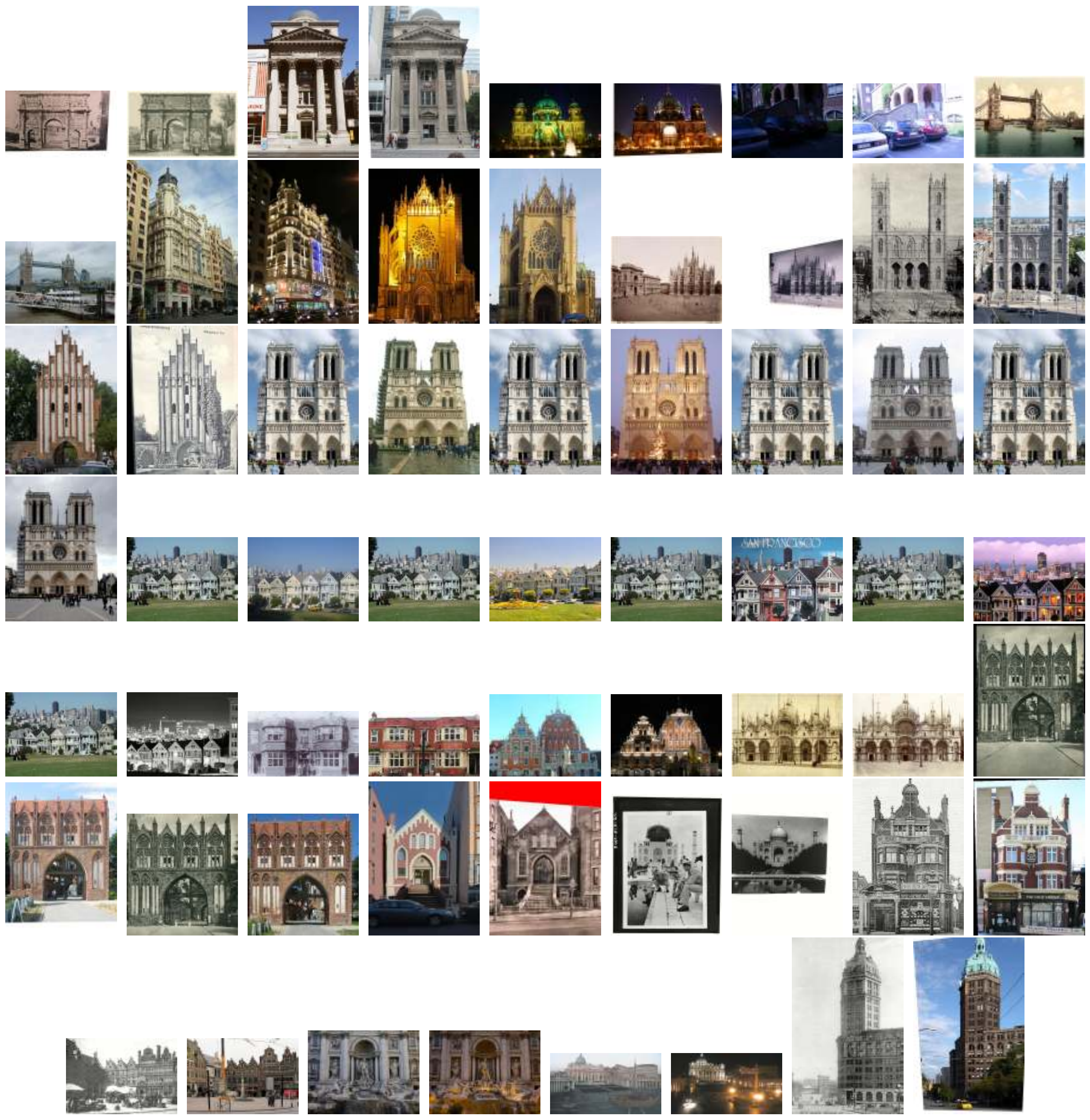
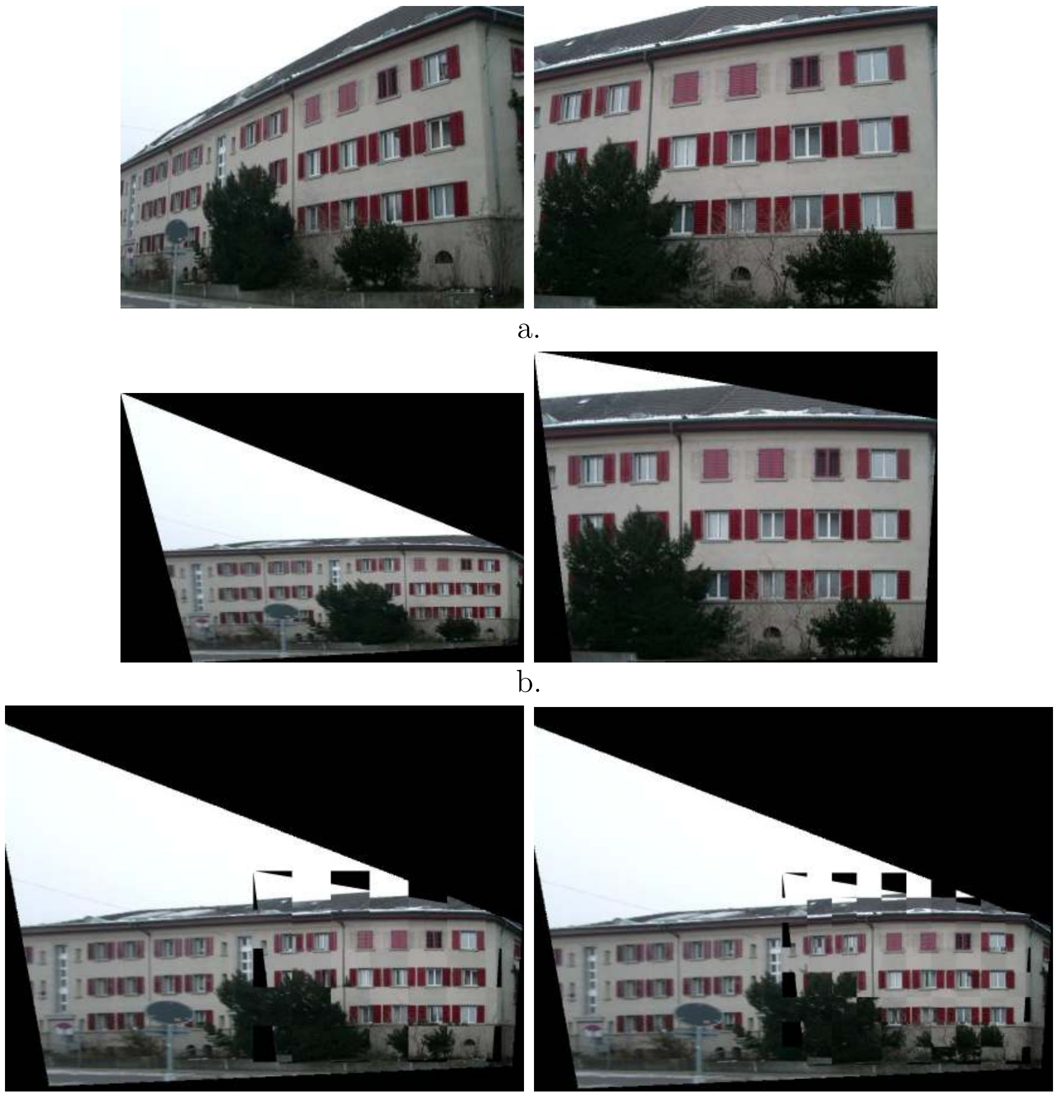


Figure C.8: Images we use for testing from SymFeat dataset discussed in Chapter 4.



a.

b.

c.

d.

Figure C.9: Example of radial distortion-induced errors. a) The images in this pair from the ZuBuD dataset (object0111:view01-view03) exhibit radial distortion which prevents from them being b) accurately rectified. With this type of warping, even the c) ground truth homography which is shown here is not 100% correct. This leads to some inaccuracies in our evaluation because the ground truth and all of the methods tested on radially distorted images suffer from homography errors due to this warping in addition to any keypoint or region match inaccuracies. d) We show the aligned result of our full pipeline to visually compare with the ground truth result.

Bibliography

[1] H. Shao, T. Svoboda, and L. Van Gool. Zubud-zurich buildings database for image based recognition. *Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland, Technical Report*, 260, 2003.

[2] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[3] G. Yang, C. V. Stewart, M. Sofka, and C. Tsai. Registration of challenging image pairs: Initialization, estimation, and decision. *Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1973–1989, 2007.

[4] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, 2007.

[5] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. In *Transactions on Graphics*, volume 30, page 154. ACM, 2011.

[6] V. R. Chandrasekhar, D. M. Chen, S. S. Tsai, N. Cheung, H. Chen, G. Takacs, Y. Reznik, R. Vedantham, R. Grzeszczuk, and J. Bach. The stanford mobile visual search data set. In *Multimedia Systems*, pages 117–122. ACM, 2011.

[7] D. Hauagge and N. Snavely. Image matching using local symmetry features. In *Computer Vision and Pattern Recognition*, pages 206–213. IEEE, 2012.

[8] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[9] G. Bradski. Opencv library. *Dr. Dobb's Journ. of Software Tools*, 2000.

[10] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of comp. vis. algorithms. %urlhttp://www.vlfeat.org/, 2008.

[11] D. Ceylan, N. J. Mitra, Y. Zheng, and M. Pauly. Coupled structure-from-motion and 3d symmetry detection for urban facades. *Transactions on Graphics*, 33(1):2, 2014.

[12] B. Morago, G. Bui, and Y. Duan. An ensemble approach to image matching using contextual features. *Transactions on Image Processing*, 24(11):4474–4486.

[13] J. N. Otoo, N. H. Maerz, L. Xiaoling, and Y. Duan. 3-d discontinuity orientations using combined optical imaging and lidar techniques. In *Proceedings of the 45th US rock mechanics symposium*, 2011.

[14] F. Durand, A. Agarwala, S. Bae, F. Durand, et al. Computational Re-Photography. *Transactions on Graphics*, 29(3), 2010.

[15] J. Xiao and Y. Furukawa. Reconstructing the world's museums. In *International Journal of Computer Vision*, volume 110, pages 243–258. Springer, 2014.

[16] Paul Haeberli. Paint by numbers: abstract image representations. In *SIGGRAPH*, volume 24, pages 207–214. ACM, 1990.

[17] G. Winkenbach and D. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH*, pages 91–100. ACM, 1994.

[18] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *Computer Graphics and Interactive Techniques*, pages 421–430. ACM Press/Addison-Wesley Publishing Co., 1997.

[19] H. Xu and B. Chen. Stylized rendering of 3d scanned real world environments. In *Non-Photorealistic Animation and Rendering*, pages 25–34, New York, NY, USA, 2004. ACM.

[20] H. Xu, N. Gossett, and B. Chen. Abstraction and depiction of sparsely scanned outdoor environments. In *Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, Computational Aesthetics, pages 19–27, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[21] A. Mastin, J. Kepner, and J. Fisher. Automatic registration of LIDAR and optical images of urban scenes. In *Computer Vision and Pattern Recognition*, pages 2639–2646. IEEE, 2009.

[22] Y. Li, Q. Zheng, A. Sharf, D. Cohen-Or, B. Chen, and N.J. Mitra. 2D-3D fusion for layer decomposition of urban facades. In *International Conference on Computer Vision*, pages 882–889. IEEE, 2011.

[23] U. Neumann, S. You, J. Hu, B. Jiang, and J.W. Lee. Augmented virtual environments (ave): Dynamic fusion of imagery and 3d models. *Proceedings on Virtual Reality*, pages 61–67, 2003.

[24] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2008.

[25] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai. Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes. *International Journal of Computer Vision*, 78(2):237–260, 2008.

[26] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Computer Vision and Pattern Recognition*, pages 3001–3008. IEEE, 2011.

[27] B. Micusik and J. Kosecka. Piecewise planar city 3D modeling from street view panoramic sequences. In *Computer Vision and Pattern Recognition*, pages 2906–2912. IEEE, 2009.

[28] S.N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *Proc. ICCV*, pages 1881–1888, 2009.

[29] N. Snavely, S.M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[30] G. Wan, N. Snavely, D. Cohen-Or, Q. Zheng, B. Chen, and S. Li. Sorting unorganized photo sets for urban reconstruction. *Graphical Models*, 74(1):14–28, 2012.

[31] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[32] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[33] M. A. Fischler and R. C. Bolles. 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.

[34] A. Irschara, C. Zach, J.M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Computer Vision and Pattern Recognition*, pages 2599–2606. IEEE, 2009.

[35] V. Gioi, R. Grompone, J. Jakubowicz, J. Morel, and G. Randall. Lsd: a line segment detector. *Image Processing On Line*, 2012.

[36] C. Wu, J. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In *European Conference on Computer Vision*, pages 142–155. Springer, 2010.

[37] M. Kushnir and I. Shimshoni. Epipolar geometry estimation for urban scenes with repetitive structures. *Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2381–2395, 2014.

[38] Y. Chung, T. Han, and Z. He. Building recognition using sketch-based representations and spectral graph matching. In *International Conference on Computer Vision*, pages 2014–2020. IEEE, 2009.

[39] M. Aubry, B. Russell, and J. Sivic. Painting-to-3d model alignment via discriminative visual elements. In *Transactions on Graphics*, volume 33, page 14. ACM, 2013.

[40] B. Russell, J. Sivic, J. Ponce, and H. Dessales. Automatic alignment of paintings and photographs depicting a 3d scene. In *International Conference on Computer Vision Workshops*, pages 545–552. IEEE, 2011.

[41] R. G. Von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.

[42] S. Singh and A. Guptaand A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, pages 73–86. Springer, 2012.

[43] G. Yang, J. Becker, and C.V. Stewart. Estimating the location of a camera with respect to a 3d model. In *3-D Digital Imaging and Modeling*, pages 159–166. IEEE, 2007.

[44] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Alvey Vision Conference*, volume 15, page 50. Manchester, UK, 1988.

[45] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of international Conference on Computer Vision*, volume 1, pages 525–531. IEEE, 2001.

[46] S. Zhong, J. Wang, L. Yan, L. Kang, and Z. Cao. A real-time embedded architecture for sift. *Journal of Systems Architecture*, 59(1):16–29, 2013.

[47] J. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *Journal on Imaging Sciences*, 2(2):438–469, 2009.

[48] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, volume 110, pages 404–417. Springer, 2006.

[49] C. Wu, F. Fraundorfer J.-M., Frahm, and M. Pollefeys. 3d model search and pose estimation from single images using vip features. In *Computer Vision and Pattern Recognition Workshops*. IEEE, 2008.

[50] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski. Non-rigid dense correspondence with applications for image enhancement. In *Transactions on Graphics*, volume 30. ACM, 2011.

[51] P. Forssen and D. G. Lowe. Shape descriptors for maximally stable extremal regions. In *International Conference on Computer Vision*. IEEE, 2007.

[52] R. Kimmel, C. Zhang, A. M. Bronstein, and M. M. Bronstein. Are mser features really interesting? *Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2316–2320, 2011.

[53] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[54] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[55] Q. Li, G. Qu, and Z. Li. Matching between sar images and optical images based on hog descriptor. In *Radar Confernce*, pages 1–4. IET International, April 2013.

[56] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, 1:886–893, 2005.

[57] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition*, pages 3304–3311. IEEE, 2010.

[58] L. Wang, U. Neumann, and S. You. Wide-baseline image matching using line signatures. In *International Conference on Computer Vision*, pages 1311–1318. IEEE, 2009.

[59] B. Fan, F. Wu, and Z. Hu. Line matching leveraged by point correspondences. In *Computer Vision and Pattern Recognition*, pages 390–397. IEEE, 2010.

[60] H. Bay, V. Ferrari, and L. Van Gool. Wide-baseline stereo matching with line segments. In *Computer Vision and Pattern Recognition*, volume 1, pages 329–336. IEEE, 2005.

[61] J. Matas, O. Chum, M. Urban, and T. Pajdla. Distinguished regions for wide-baseline stereo. *Center for Machine Perception-K333 CVUT, Praha, Technical Report*, 2001.

[62] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Comp. Graphics and Vis.*, 3(3):177–280, 2008.

[63] M. Park, S. Lee, P. Chen, S. Kashyap, A. Butt, and Y. Liu. Performance evaluation of state-of-the-art discrete symmetry detection algorithms. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[64] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Computer Vision and Pattern Recognition*.

[65] M. Bansal, K. Daniilidis, and H. Sawhney. Ultra-wide baseline facade matching for geo-localization. In *European Conference on Computer Vision Workshops and Demonstrations*, pages 175–186. Springer, 2012.

[66] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *Transactions on Graphics*, 28(3):24, 2009.

[67] C. Barnes, E. Shechtman, and D. Goldman. The generalized patchmatch correspondence algorithm. pages 29–43. Springer, 2010.

[68] D. Nistér. An efficient solution to the five-point relative pose problem. *Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[69] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry, volume =. In *Computer Vision and Pattern Recognition*, pages 652–659. IEEE.

[70] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *Transactions on Graphics*, volume 25, pages 835–846. ACM, 2006.

[71] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. In *Computer Vision and Pattern Recognition*. IEEE, 2007.

[72] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[73] Y. Furukawa and J. Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. In *Computer Vision and Pattern Recognition*. IEEE, 2008.

[74] Y. Furukawa and J. Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *International Journal of Computer Vision*, 84(3):257–268, 2009.

[75] Yasutaka Furukawa and Jean Ponce. Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[76] Y. Cao and J. McDonald. Improved feature extraction and matching in urban environments based on 3D viewpoint normalization. *Computer Vision and Image Understanding*, 2011.

[77] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *Computer Vision and Pattern Recognition*, pages 1450–1457. IEEE, 2012.

[78] L. Liu and I. Stamos. A systematic approach for 2d-image to 3d-range registration in urban environments. *Computer Vision and Image Understanding*, 116(1):25–37, 2012.

[79] D. Craciun, N. Paparoditis, and F. Schmitt. *Image-Laser Fusion for In Situ 3D Modeling of Complex Environments: A 4D Panoramic-Driven Approach.* InTech, 2012.

[80] W. Zhao, D. Nister, and S. Hsu. Alignment of continuous video onto 3d point clouds. *Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1305–1318, 2005.

[81] M. Ding, K. Lyngbaek, and A. Zakhor. Automatic registration of aerial imagery with untextured 3d lidar models. In *Computer Vision and Pattern Recognition.* IEEE, 2008.

[82] B. C. Matei, N. V. Valk, Z. Zhu, and H. Cheng. Image to lidar matching for geotagging in urban environments. In *Applications of Computer Vision*, pages 413–420. IEEE, 2013.

[83] Gaurav Pandey, James R McBride, Silvio Savarese, and Ryan M Eustice. Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information. *Proc. AAAI National Conference on Artifical Intelligence*, 2012.

[84] M. Corsini, M. Dellepiane, F. Ponchio, and R. Scopigno. Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties. In *Computer Graphics Forum*, volume 28, pages 1755–1764. Wiley Online Library, 2009.

[85] H. Alismail, L.D. Baker, and B. Browning. Automatic calibration of a range sensor and camera system. In *3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 286–292, 2012.

[86] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[87] M. Weinmann, L. Hoegner, J. Leitloff, U. Stilla, S. Hinz, and B. Jutzi. Fusing passive and active sensed images to gain infrared-textured 3d models. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39:71–76, 2012.

[88] W. Guan, S. You, and G. Pang. Estimation of camera pose with respect to terrestrial lidar data. In *Workshop on Applications of Computer Vision*, pages 391–398. IEEE, 2013.

[89] G. Sun, S. Wang, X. Liu, Q. Huang, Y. Chen, and E. Wu. Accurate and efficient cross-domain visual matching leveraging multiple feature representations. *The Visual Computer*, 29(6-8):565–575, 2013.

[90] J. Becker, C.V. Stewart, and R. J. Radke. LiDAR Inpainting from a Single Image. In *3-D Digital Imaging and Modeling*. IEEE, 2009.

[91] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N.J. Mitra. GlobFit: consistently fitting primitives by discovering global relations. In *Transactions on Graphics*, volume 30, page 52. ACM, 2011.

[92] M.Y. Yang, Y. Cao, and J. McDonald. Fusion of camera images and laser scans for wide baseline 3D scene alignment in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2011.

[93] D. Doria and R. J. Radke. Filling Large Holes in LiDAR Data by Inpainting Depth Gradients. In *Computer Vision and Pattern Recognition Workshops*. IEEE, 2012.

[94] Christian Früh, Siddharth Jain, and Avideh Zakhor. Data Processing Algorithms for Generating Textured 3D Building Facade Meshes from Laser Scans and Camera Images. *International Journal of Computer Vision*, (2):159–184.

[95] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, TU Darmstadt, D. Steedly, and R. Szeliski. Ambient point clouds for view interpolation. *Transactions on Graphics*, 29(4):95, 2010.

[96] D. Gallup, J.M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[97] D. Gallup, J.M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Computer Vision and Pattern Recognition*, pages 1418–1425. IEEE, 2010.

[98] A. Alharthy and J. Bethel. Detailed building reconstruction from airborne laser data using a moving surface method. *International Archives of Photogrammetry and Remote Sensing*, 35:213–218, 2004.

[99] S.N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3D architectural modeling from unordered photo collections. In *Transactions on Graphics*, volume 27, page 159. ACM, 2008.

[100] R. Grzeszczuk, J. Kosecka, R. Vedantham, and H. Hile. Creating compact architectural models by geo-registering image collections. In *International Conference on Computer Vision Workshops*, pages 1718–1725. IEEE, 2009.

[101] J. Yao, P. Taddei, M.R. Ruggeri, and V. Sequeira. Complex and photo-realistic scene representation based on range planar segmentation and model fusion. *The International Journal of Robotics Research*, 2011.

[102] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *International Conference on Computer Vision*, volume 1, pages 370–377. IEEE, 2005.

[103] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *Computer Vision and Pattern Recognition*, pages 25–32. IEEE, 2009.

[104] C. Stachniss. C implementation of the hungarian method, 2004.

[105] R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, Cambridge, United Kingdom, 2010.

[106] T. Vincent and R. Laganiére. Detecting planar homographies in an image pair. In *Proceedings of Image and Signal Processing and Analysis*, pages 182–187. IEEE, 2001.

[107] F. Fraundorfer, K. Schindler, and H. Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, 24(4):395 – 406, 2006.

[108] D. Santosh Kumar and C. V. Jawahar. Robust homography-based control for camera positioning in piecewise planar environments. In *Computer Vision, Graphics and Image Processing*, pages 906–918. Springer, 2006.

[109] W. Lin, S. Liu, Y. Matsushita, T. Ng, and L. Cheong. Smoothly varying affine stitching. In *Computer Vision and Pattern Recognition*, pages 345–352. IEEE, 2011.

[110] J. Zaragoza, T. Chin, M. S. Brown, and D. Suter. As-projective-as-possible image stitching with moving dlt. In *Computer Vision and Pattern Recognition*, pages 2339–2346. IEEE, 2013.

[111] Y. Lipman, S. Yagev, R. Poranne, D. W. Jacobs, and R. Basri. Feature matching with bounded distortion. *Transactions on Graphics*, 33(3):26, 2014.

[112] W. D. Lin, M. Cheng, J. Lu, H. Yang, M. N. Do, and P. Torr. Bilateral functions for global motion modeling. In *European Conference on Computer Vision*, pages 341–356. Springer, 2014.

[113] H. Yang, W. Lin, and J. Lu. Daisy filter flow: a generalized discrete approach to dense correspondences. In *Computer Vision and Pattern Recognition*, pages 3406–3413. IEEE, 2014.

[114] P. Besl and N. McKay. Method for registration of 3-d shapes. In *Proceedings of SPIE Sensor Fusion IV: Control Paradigms and Data Structures*, pages 586–606. International Society for Optics and Photonics, 1992.

[115] J. Lezama, R. Grompone von Gioi, G. Randall, and J. Morel. Finding vanishing points via point alignments in image primal and dual domains. In *Computer Vision and Pattern Recognition*, pages 509–515. IEEE, 2014.

[116] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[117] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[118] M. Pauly, M. Gross, and L.P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Visualization*, pages 163–170. IEEE, 2002.

[119] H. Lin, J. Gao, Y. Zhou, G. Lu, M. Ye, C. Zhang, L. Liu, and R. Yang. Semantic decomposition and reconstruction of residential scenes from lidar data. *Transactions on Graphics*, 32(4):66, 2013.

[120] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Computer Vision and Pattern Recognition*.

[121] J. Shan and C. K. Toth. *Topographic laser ranging and scanning: principles and processing*. CRC press, 2008.

[122] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, volume 1, pages 666–673. IEEE, 1999.

[123] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[124] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, volume 1148, pages 203–222. Springer, 1996.

[125] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*. IEEE, 1999.

[126] Marko M. Heikkila and M. Pietikainen. A texture-based method for modeling the background and detecting moving objects. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):657–662, 2006.

[127] D. Hearn and M. Baker. *Computer Graphics with OpenGL*. Pearson Prentice Hall, Upper Saddle River, NJ, 2004.

[128] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross. Stereobrush: interactive 2d to 3d conversion using discontinuous warps. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 47–54. ACM, 2011.

[129] J. A. Hudson. *Rock mechanics principles in engineering practice.* Butterworths, London, 1989.

[130] E. Hoek and J.W. Bray. Rock slope engineering. institution of mining and metallurgy. *London*, 360, 1981.

[131] J. Kemeny and R. Post. Estimating three-dimensional rock discontinuity orientation from digital images of fracture traces. *Computers & Geosciences*, 29(1):65–77, 2003.

[132] S. D. Priest. *Hemispherical projection methods in rock mechanics.* Allen & Unwin London, 1985.

[133] N. H. Maerz and W. Zhou. Multivariate analysis of bore hole discontinuity data. In *Rock Mechanics for Industry, Vail, Colorado*, volume 1, pages 431–438, 1999.

[134] N. Maerz and W. Zhou. Discontinuity data analysis from oriented boreholes. In *North American Rock Mechanics Symposium*, pages 667–674, Seattle, Washington, July,31-August,1 2000. American Rock Mechanics Association.

[135] W. Zhou and N. H. Maerz. Multivariate clustering analysis of discontinuity data: implementation and applications. In *Rock mechanics in the national interest. US Rock Mechanics Symposium, Washington, DC*, pages 861–868, Washington, D.C., July,7-10 2001.

[136] W. Zhou and N. Maerz. Implementation of multivariate clustering methods for characterizing discontinuities data from scanlines and oriented boreholes. *Computers & geosciences*, 28(7):827–839, 2002.

[137] N. H. Maerz and W. Zhou. Multivariate clustering analysis of the ecrb cross drift discontinuities, yucca mountain project. In *Alaska Rocks, US Rock Mechanics Symposium, Anchorage Alaska*, volume 10, June 25-29 2005.

[138] Matthew M. J. Lato and M. Vöge. Automated mapping of rock discontinuities in 3d lidar and photogrammetry models. *International Journal of Rock Mechanics and Mining Sciences*, 54:150–158, 2012.

[139] R. D. Terzaghi. Sources of error in joint surveys. *Geotechnique*, 15(3):287–304, 1965.

[140] M. J. Lato, M. S. Diederichs, and D. J. Hutchinson. Bias correction for view-limited lidar scanning of rock outcrops for structural characterization. *Rock mechanics and rock engineering*, 43(5):615–628, 2010.

[141] F. Bulut and Ş Tüdeş. Determination of discontinuity traces on inaccessible rock slopes using electronic tacheometer: an example from the ikizdere (rize) region, turkey. *Engineering geology*, 44(1):229–233, 1996.

[142] Q. Feng. *Novel methods for 3-D semi-automatic mapping of fracture geometry at exposed rock faces.* PhD thesis, Division of Engineering Geology, Royal Institute of Technology (KTH), Stockholm, Sweden, 2001.

[143] R. Post. Characterizing of joints and fractures in a rock mass using digital image processing. Master's thesis, University of Arizona, Tucson, AZ 105 pp, 2001.

[144] J. Donovan, J. Kemeny, and J. Handy. The application of three-dimensional imaging to rock discontinuity characterization. In *Alaska Rocks, US Rock Mechanics Symposium, Anchorage Alaska*, volume 7, June 25-29 2005.

[145] A. Strouth and E. Eberhardt. The use of lidar to overcome rock slope hazard data collection challenges at afternoon creek, washington. In *Methods for Rock Face Characterization Workshop US Symposium on Rock Mechanics*, pages 49–62, Golden, CO, June 17-21 2006. American Rock Mechanics Association.

[146] M. Jaboyedoff, R. Metzger, T. Oppikofer, R. Couture, M.H. Derron, J. Locat, and D. Turmel. New insight techniques to analyze rock-slope relief using dem and 3d-imaging cloud points: Coltop-3d software. In *Rock mechanics: Meeting Societys Challenges and demands*, volume 1, pages 61–68, 2007.

[147] W. Hanberg. Using close range terrestrial digital photogrammetry for 3-d rock slope modeling and discontinuity mapping in the united states. In *Bulletin of Engineering Geology and the Environment*, volume 67, pages 457–469, 2008.

[148] M. I. Olariu, J. F. Ferguson, C. L. V. Aiken, and X. Xu. Outcrop fracture characterization using terrestrial laser scanners: Deep-water jackfork sandstone at big rock quarry, arkansas. *Geosphere*, 4(1):247–259, 2008.

[149] M. Lato, J. Hutchinson, M. Diederichs, D. Ball, and R. Harrap. Engineering monitoring of rockfall hazards along transportation corridors: using mobile terrestrial lidar. *Natural Hazards and Earth System Sciences*, 9(3):935–946, 2009.

[150] M. Sturzenegger and D. Stead. Close-range terrestrial digital photogrammetry and terrestrial laser scanning for discontinuity characterization on rock cuts. *Engineering Geology*, 106(3):163–182, 2009.

[151] G. Gigli and N. Casagli. Semi-automatic extraction of rock mass structural data from high resolution lidar point clouds. *International Journal of Rock Mechanics and Mining Sciences*, 48(2):187–198, 2011.

[152] D García-Sellés, Oriol Falivene, Pau Arbués, Oscar Gratacos, S Tavani, and Josep Anton Muñoz. Supervised identification and reconstruction of near-planar geological surfaces from terrestrial laser scanning. *Computers & Geosciences*, 37(10):1584–1594, 2011.

[153] K. Khoshelham, D. Altundag, D. Ngan-Tillard, and M. Menenti. Influence of range measurement noise on roughness characterization of rock surfaces using terrestrial laser scanning. *International Journal of Rock Mechanics and Mining Sciences*, 48(8):1215–1223, 2011.

[154] A. Riquelme, A. Abellán, R. Tomás, and M. Jaboyedoff. Rock slope discontinuity extraction and stability analysis from 3d point clouds: application to an urban rock slope. In *Vertical Geology Conference*, 2014.

[155] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[156] J. N. Otoo, N. H. Maerz, X. Li, and Y. Duan. Verification of a 3-d lidar viewer for discontinuity orientations. *Rock mechanics and rock engineering*, 46(3):543–554, 2013.

[157] N. Schurr, J. Marecki, M. Tambe, P. Scerri, N. Kasinadhuni, and J. P. Lewis. The future of disaster response: Humans working with multiagent teams using defacto. In *AAAI Spring Symposium: AI Technologies for Homeland Security*, pages 9–16, 2005.

[158] B. Morago, G. Bui, and Y. Duan. Integrating lidar range scans and photographs with temporal changes. In *Computer Vision and Pattern Recognition Workshops*, pages 718–723, 2014.

[159] M. Kwan and D. M. Ransberger. Lidar assisted emergency response: Detection of transport network obstructions caused by major disasters. *Computers, Environment and Urban Systems*, 34(3):179–188, 2010.

[160] S. Seetharam, P. Calyam, and T. Beyene. ADON: Application-Driven Overla Network-as-a-Service for Data-Intensive Science. Master's thesis, University of Missouri-Columbia, 2014.

[161] P. Calyam, A. Berryman, E. Saule, H. Subramoni, P. Schopis, G. Springer, U. Catalyurek, and D. K. Panda. Wide-area Overlay Networking to Manage Science DMZ Accelerated Flows. IEEE, 2014.

[162] S. Kumar, R. K. Rathy, and D. Pandey. Design of an ad-hoc network model for disaster recovery scenario using various routing protocols. In *International Conference on Advances in Computing, Communication and Control*, pages 100–105. ACM, 2009.

[163] E. Chissungo, H. Le, and E. Blake. An electronic health application for disaster recovery. In *Symposium on Information and Communication Technology*, pages 134–138. ACM, 2010.

[164] U. Witkowski, E. Habbal, M. A. Mostafa, S. Herbrechtsmeier, A. Tanoto, J. Penders, L. Alboul, and V. Gazi. Ad-hoc network communication infrastructure for multi-robot systems in disaster scenarios. In *Proceedings of IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance*, 2008.

[165] N. K. Ray and A. K. Turuk. A framework for disaster management using wireless ad hoc networks. In *International Conference on Communication, Computing & Security*, pages 138–141. ACM, 2011.

[166] L. Moisan, P. Moulon, and P. Monasse. Automatic homographic registration of a pair of images, with a contrario elimination of outliers. *Image Processing On Line*, 10, 2012.

[167] Y. Chen, D. P. Robertson, and Roberto R. Cipolla. A practical system for modelling body shapes from single view measurements. In *British Machine Vision Conference.*

[168] J. Lezama, R. G. Gioi, G. Randall, and J. Morel. Finding vanishing points via point alignments in image primal and dual domains. In *Computer Vision and Pattern Recognition*, pages 509–515. IEEE, 2014.

[169] A. Taneja, L. Ballan, and M. Pollefeys. City-scale change detection in cadastral 3d models using images. In *Computer Vision and Pattern Recognition*, pages 113–120. IEEE, 2013.

[170] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision*, pages 352–365. Springer, 2010.

[171] Y. Ma, S. Soatta, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision.* Springer, New York, New York, 2004.

[172] H. Shao, T. Svoboda, and L. Van Gool. Zubud-zurich buildings database for image based recognition. *Comp. Vis. Lab, Swiss Federal Institute of Technology, Switzerland, Tech. Rep*, 260, 2003.

[173] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

VITA

Brittany Morago was born in White Plains, NY. She attended the University of Florida where she graduated with a B.S. Degree in Digital Arts and Sciences in 2010. In May 2016, she will be completing her Ph.D. in Computer Science at the University of Missouri. While at the University of Missouri, Brittany has also completed a College Teaching Minor. In 2010, Brittany began working with Ye Duan on computer vision-related research with her main project being multi-modality imagery registration. During graduate school, Brittany has been a recipient of the National Science Foundation Graduate Research Fellowship, the Graduate Areas in Assistance of National Need Fellowship, and the Robert E. Waterson Fellowship.

In August 2016, Brittany will be starting as an Assistant Professor in the Computer Science Department at the University of North Carolina-Wilmington.