TOWARDS VISION BASED ROBOTS FOR MONITORING BUILT ENVIRONMENTS

BY

JOSEPH DEGOL

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Associate Professor Derek Hoiem, Chair
Associate Professor Timothy Bretl
Associate Professor Mani Golparvar-Fard
Professor David Forsyth
Dr. Sudipta Sinha, Microsoft Research

Abstract

In construction, projects are typically behind schedule and over budget, largely due to the difficulty of progress monitoring. Once a structure (e.g. a bridge) is built, inspection becomes an important yet dangerous and costly job. We can provide a solution to both problems if we can simplify or automate visual data collection, monitoring, and analysis. In this work, we focus specifically on improving autonomous image collection, building 3D models from the images, and recognizing materials for progress monitoring using the images and 3D models.

Image capture can be done manually, but the process is tedious and better suited for autonomous robots. Robots follow a set trajectory to collect data of a site, but it is unclear if 3D reconstruction will be successful using the images captured by following this trajectory. We introduce a simulator that synthesizes feature tracks for 3D reconstruction to predict if images collected from a planned path will result in a successful 3D reconstruction. This can save time, money, and frustration because robot paths can be altered prior to the real image capture. When executing a planned trajectory, the robot needs to understand and navigate the environment autonomously. Robot navigation algorithms struggle in environments with few distinct features. We introduce a new fiducial marker that can be added to these scenes to increase the number of distinct features and a new detection algorithm that detects the marker with negligible computational overhead.

Adding markers prior to data collection does not guarantee that the algorithms for 3D model generation will succeed. In fact, out of the box, these algorithms do not take advantage of the unique characteristics of markers. Thus, we introduce an improved structure from motion approach that takes advantage of marker detections when they are present. We also create a dataset of challenging indoor image collections with markers placed throughout and show that previous methods often fail to produce accurate 3D models. However, our approach produces complete, accurate 3D models for all of these new image collections.

Recognizing materials on construction sites is useful for monitoring usage and tracking construction progress. However, it is difficult to recognize materials in real world scenes because shape and appearance vary considerably. Our solution is to introduce the first dataset of material patches that include both image data and 3D geometry. We then show that both independent and joint modeling of geometry are useful alongside image features to improve material recognition. Lastly, we use our material recognition with material priors from building plans to accurately identify progress on construction sites.

Acknowledgments

I had the pleasure of working closely with Derek Hoiem, Timothy Bretl, and Mani Golparvar-Fard. Thank you Derek for your guidance. You taught me how to think strategically about research, provide thorough experiments, and present results effectively. Moreover, your intuition and support ensured that our research progressed smoothly. Thank you also Tim. You taught me how to build a research plan and precisely communicate technical concepts. Furthermore, some of my most enjoyable moments were discussions with you concerning ChromaTag detection. Thank you Mani. You taught me to dream big and your vision provided the motivation for all my research. You also opened doors for me with business that few graduate students get to experience. I am profoundly grateful for the opportunity to work with each of you and proud of our accomplishments thus far.

In addition, thank you to my other committee members, David Forsyth and Sudipta Sinha. Thank you David for your great ideas to highlight the strengths of my work. Thank you Sudipta Sinha for our many 3D reconstruction discussions and for guiding my work at Microsoft Research.

I am also very grateful to have worked with many great student colleagues, including Miles Johnson, David Hanley, Navid Aghasadeghi, Kevin Karsch, Ryan Musa, Xinke Deng, Aadeel Akhtar, Bhargava Manja, Kevin Han, Jae Yong Lee, Rajbir Kataria, Jacob Lin, and Daniel Yuan. Thank you also to my many friends and colleagues at the University of Illinois and elsewhere; in particular, the computer vision group, Bretl robotics group, and RAAMAC lab.

Thank you to the many research mentors I had as an undergraduate. Thank you Pinaki Das for starting this journey by encouraging me to consider graduate school. Thank you Scott McCrickard for teaching me to think deeply about research problems and the research method. Our work with PIC-UP was my first publication. Thank you Patricio Vela, Ryan Eustice, John Hannan, and Robert Collins for your guidance during my research with each of you. From you, I learned the

fundamentals of machine learning, computer vision, and effective presentations. Thank you Myra Nam, Mike Zyskowski, and Chris Buehler for your guidance during my graduate school research internships. The projects I did with each of you have broadened my experiences and shaped my research goals.

Finally, thank you to my parents, Lisa and Joe, for your many years of love and encouragement. Parent engagement is crucial for a child's social and educational success. I am fortunate to have two loving parents that were actively involved in my schooling, provided financial support for college, and cultivated an active, social lifestyle.

Table of Contents

Chapter 1

Introduction

In this dissertation, we describe several contributions that improve autonomous collection and analysis of image and video data. One motivation for this work is to enable autonomous monitoring and inspection of construction sites, bridges, stadiums, and many other built environments. The potential gain from an autonomous monitoring system for construction is immense. The construction industry is more than a $1.1 billion industry [18] with 25-50% waste in coordinating labor and equipment, and in managing, moving, and installing material [130, 110]. In fact, the National Research Council of the National Academies [35] and the Construction Industry Institute [29] have identified improving the efficiency of construction as a key national need, citing progress monitoring tools as the solution. Safety is also a concern, with more than 900 construction workers killed in 2016 [137]. Almost half of these fatalities were from falling or being struck by an object, two cases that autonomous monitoring could mitigate through proactive detection of potential hazards.

Several companies (e.g. Reconstruct [143], Pix4D [140], and Drone Deploy [52]), provide 3D mapping solutions that can be used to track construction progress over time. These solutions use drone images to build 3D models of construction sites and align them to the 4D building information model (4D BIM) [103, 55]. The 4D BIM is a 3D model and schedule that serves as a blueprint for what should be built and when. By comparing the 3D reconstructions to the 4D BIM, these companies can infer which parts of the project are behind schedule. However, many challenges still exist with data capture, 3D reconstruction, and analysis that prevent these platforms from achieving fully autonomous solutions for both indoor and outdoor environments.

1.1 Simulating Data Collection to Improve 3D Reconstruction

One challenge with data capture is that the images captured by the drones (or any robot or handheld camera) are not guaranteed to result in a successful 3D model. Current approaches rely on intuition and standard lawn mower paths for image capture. These paths work okay for capturing relatively flat terrain, but they are not good for capturing buildings, bridges, or interiors. Moreover, there is a trade off between time to capture images and the total number captured. Ideally, just enough images are captured to cover the entirety of the relevant area with enough density such that there is significant overlap between the content of each image and neighboring images. Furthermore, the only way to know if the captured images will be useful is to run 3D reconstruction on them. If the images do not result in a nice 3D model, then another data capture needs to be done. This is time consuming, costly, and frustrating when failure occurs.

Chapter 2 details our solution to this image collection problem. Specifically, we simulate 3D geometry to predict if a collection of captured images will result in a successful reconstruction [47]. We use 3D models of a scene and a camera trajectory as the starting point. Then, for each image capture location along that camera trajectory, we model (1) image feature noise as 3D point locations are projected onto the image plane and (2) image feature matching between image pairs as 3D locations are viewed from two different perspectives. With our simulator, we can predict that an image collection will result in a failed reconstruction, which enables editing the path prior to the real data capture to improve the likelihood of a successful 3D reconstruction.

1.2 Fast Marker Detection to Improve Robot Navigation

While data capture can be done using handheld cameras, the process is tedious and better suited for autonomous robots. For a robot to navigate an environment, it must understand the scene (i.e. have a map), and know where it is located in that map (localization) [5, 151]. GPS enables autonomous navigation of drones and other robots in outdoor environments because it provides the

location of the robot (localization) in the world (map). However, GPS can have significant errors on the order of meters and does not work indoors, making it insufficient for precise navigation. This is especially important in dynamic scenes like construction sites where equipment and workers are present and the structure is continuously evolving. Simultaneous localization and mapping (SLAM) [98, 125, 56] is a 3D reconstruction approach that processes video frames in real time to align the current position of the camera to a continuously updating map. The real time nature of SLAM makes it ideal as the mapping solution for robot navigation, and indeed, SLAM has been the standard mapping method in robotics for many years [120, 173, 54, 40, 99, 129, 171, 125]. However, SLAM algorithms often fail in scenes with few distinct features such as sparsely textured surfaces (e.g. plain walls), repetitive structure (e.g. door frames, brick walls), and reflective surfaces (e.g. windows, metal surfaces), which are common characteristics of construction sites.

Our solution, presented in Chapter 3 is ChromaTag, a new fiducial marker that can be detected at over 700 frames per second [42]. ChromaTag uses both color and grayscale to best effect to achieve reliable detection at speeds orders of magnitude faster than state of the art methods. Fiducial markers are artificial features specifically engineered for reliable detection. They can alleviate challenges with 3D reconstruction because they (1) add unique features to scenes to supplement the number of features in plain environments and (2) have unique signatures to differentiate repetitive environments. An ideal fiducial marker for use in SLAM can be detected reliably to overcome feature tracking errors and quickly to minimally effect the runtime of the other computationally intensive steps of the pipeline. Unfortunately, other state of the art markers run at or below camera frame rate, often requiring a GPU and/or multiple CPU threads to achieve those speeds [22, 133, 184, 8]. With ChromaTag though, marker detections can be integrated to improve SLAM robustness without significantly increasing the computational burden.

1.3 Improved Structure from Motion Using Marker Matching

SLAM algorithms are ideal for mapping environments in real time from video, but are typically not designed for processing collections of images. Images are typically higher resolution and include metadata such as GPS and IMU information. Both the higher resolution images and metadata are useful in improving the accuracy of the 3D model and image alignments. In particular, the higher resolution images can capture additional fine grain detail in the 3D models, and the metadata can be used as priors for the image alignment positions and orientations. Structure from motion (SfM) [12, 2, 36, 123, 135], while similar to SLAM, is an alternative 3D reconstruction algorithm that is ideal for processing image collections and can take advantage of image metadata [117]. SfM is the algorithm of choice by many of the current construction site progress monitoring solutions. However, SfM algorithms also have trouble with the same challenging scene characteristics as SLAM because SfM also relies on distinct scene features.

In Chapter 4, we show how the detection and matching of fiducial markers can be used to improve SfM [43]. We introduce a new dataset of challenging indoor scenes (with fiducial markers placed throughout) and show how state of the art SfM algorithms (e.g. OpenSfM [135] and MarkerMapper [126]) fail on them. Our method matches markers across images and uses the matches to limit alignment confusions and to dictate the order in which images are aligned. With our method, we successfully reconstruct all the datasets that other SfM algorithms fail to reconstruct, providing promising results for indoor mapping for progress monitoring.

1.4 Geometry Informed Material Recognition

On a construction site, asset tracking is important. For example, it is useful to monitor a store of bricks being used in the correct location and the number of bricks still remaining. In addition, the material properties of a column can be used as an indicator of progress; for example, form work and rebar are placed before concrete. Material recognition work has been successful for swatch

datasets using only images [107, 181, 109]; however, use in real world scenes remains a challenge because the visual appearance changes considerably for different objects, lighting, and viewing direction [31, 7]. Since our solution reconstructs 3D models from collected images, we have both aligned images and 3D geometry at our disposal. Intuitively, it seems like 3D information would assist in material recognition; for example, marble tends to be flat and smooth, which is drastically different from a rough surface like dirt, or a distinct rectangular pattern like brick and mortar. However, datasets providing aligned 3D geometry and images are sparse and little work has been done to explore how they can be used together to improve material recognition.

In Chapter 5, we introduce a new GeoMat (geometry, materials) dataset that provides material patches with aligned images, normals, depths, and 3D camera locations [44]. We also provide a large construction site scene with labeled materials. Using this data, we train material classifiers for 19 common construction site materials and integrate 3D geometry by using the 3D surface normals of the materials as additional features. With our method we achieve markedly higher recognition rates than 2D methods.

1.5 Using Geometry for Improved Progress Monitoring

Large construction projects are behind schedule more than half the time and are over budget more than 60% of the time. Some of the main factors for this include inconsistency in progress reporting among workers and managers and infrequent reporting of actual progress by project teams. Automatically detecting progress would go a long way in alleviating some of these challenges. Several works have used images [163, 77, 76, 82] and laser scanners [174, 175, 15, 97] to build 3D models of sites, align them to BIM models, and detect progress over time. However, these approaches have yet to truly automate progress monitoring because of many challenges (e.g. occlusions, errors in 3D reconstruction and alignment, and discrepancies between plans and what was actually built).

In Chapter 6, we propose a system for detecting progress on construction sites. Our approach

uses dense point clouds from 3D reconstruction aligned to the 3D BIM model to identify which BIM elements have been built. Then, we use our material recognition paired with prior material information provided by the BIM elements to classify the material of the built elements; which allows us to reason about the progress of concrete columns (i.e. wood/formwork implies that construction is in progress and concrete implies construction is complete). We test our approach on a real hotel project and show that we detect all of the BIM elements and classify their material correctly more than 90% of the time.

1.6 Additional Work not in Thesis

During my doctoral studies, I completed several projects outside the scope of this thesis. With Myra Nam at MIT Lincoln Lab, we tracked moving objects in video captured from a moving aerial camera [48]. At Microsoft Research, I worked with Mike Zyskowski and Sudipta Sinha and created a simulation environment for quadrotor control and an RGB-D visual odometry system for estimating 6 DoF camera pose. I also worked with Chris Buehler, Michael Cohen, and Neel Joshi at Microsoft research on content aware hyperlapse that used learned saliency to adjust the speed of the hyperlapse video (e.g. moves fast during boring content and slows down for interesting content). With Timothy Bretl, I designed a passive (i.e. no power required to actuate) mechanism that allows a quadrotor to relocate cameras on steel beams of construction sites (and other metal surfaces) [46]. Also with Timothy Bretl, I used convolutional neural networks to control a prosthetic hand to automatically select grasps for manipulating objects [41].

Chapter 2

FEATS: Synthetic Feature Tracks for Structure from Motion Evaluation

We present FEATS (Feature Extraction And Tracking Simulator), a simulation environment that synthesizes feature matches (rather than images) using camera poses and scene geometry (e.g. CAD, laser, multi-view stereo) to (1) predict whether 3D reconstruction on real images will succeed and (2) evaluate structure from motion (SfM) using controlled noise and perfect ground truth. While it is difficult to synthesize images that are realistic enough for evaluation [182], synthesizing geometry is easier and has been shown to be effective for training and evaluation [162]. Moreover, SfM algorithms are independent of the images given extracted and matched features (i.e., "tracks" or "match graphs"). Synthesizing feature tracks also enables direct comparison among SfM algorithms (as feature detection/matching is held constant), and we can systematically vary trajectories, feature noise, and matching outliers to better understand the strengths and weaknesses of the algorithms. Lastly, our simulator provides a way to generate datasets with precise ground truth and this makes it possible to evaluate SfM algorithms in a way that was not possible before.

FEATS is motivated by two challenges. **The first challenge** is wanting to know if a image acquisition path will result in a successful reconstruction. For 3D mapping of construction sites [81, 80, 57, 62, 52, 140, 143], repeated (e.g. weekly) drone flights collect image data for 3D mapping the scene over time. For each flight, a path is planned using software before being exported to a drone that collects the images. These images are processed using an SfM algorithm which takes hours (sometimes days) to create a sparse point cloud model (often referred to as a 3D map). This process is time consuming, costly, and frustrating when the reconstruction fails. FEATS offers a solution because FEATS can use previous 3D maps and simulate feature extraction and matching (which are input to SfM) to evaluate (and fix) planned paths prior to data collection. **The second challenge** is obtaining aligned camera pose and 3D point ground truth data (both,

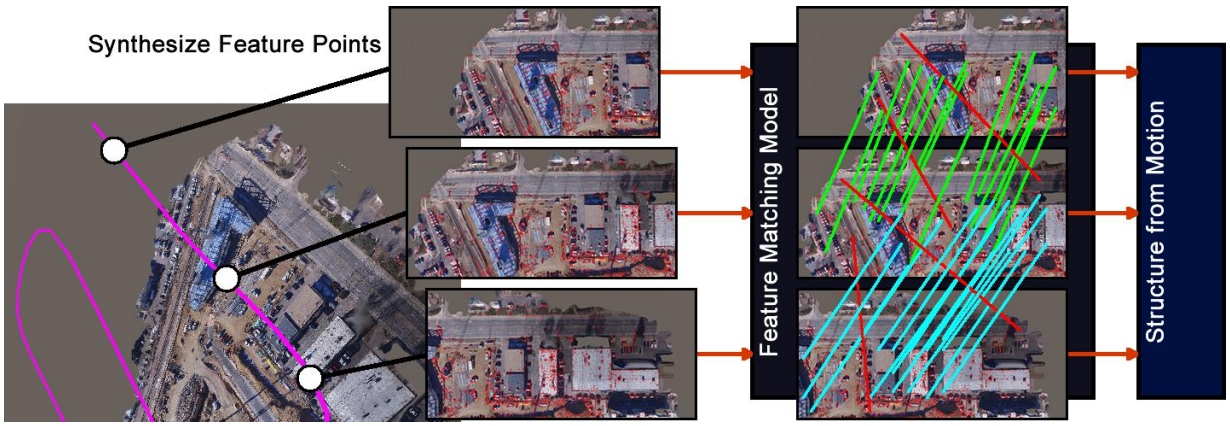


Figure 2.1: FEATS (Feature Extraction And Tracking Simulator) synthesizes feature tracks from camera motion through 3D scenes. The feature tracks are controlled through noise model parameters and input into SfM algorithms. We use new real world data (with ground truth) to validate that simulated tracks are predictive of real world tracks. We then show how several SfM algorithms perform for varying noise and calculate 3D point error on the resulting reconstructions.

together) of large environments for evaluating SfM algorithms. Currently, capturing ground truth is challenging and costly (see Section 2.1), but necessary because (1) large environments are a common SfM application [36, 2, 67, 185, 12, 165, 170]; and (2) accurate estimates of camera pose do not guarantee accurate estimates of 3D points [85] (reinforced in Section 3.4). FEATS provides ground truth camera pose and 3D point locations for arbitrarily large scenes.

We introduce a new dataset of image collections with ground truth camera pose and use FEATS to generate synthetic equivalents. We use FEATS to create synthetic versions of this new dataset and show that these synthetic equivalents are predictive of SfM results from processing real images. We then demonstrate two new benchmarks to evaluate SfM that are enabled by FEATS. In particular, we vary noise parameters for synthetic tracks to evaluate the robustness of SfM algorithms, and we calculate error between ground truth 3D points and reconstructed point clouds to evaluate accuracy of 3D structure (previous SfM evaluations focus on estimated camera extrinsic parameters).

In summary, our **contributions** are: (1) FEATS, a simulator to synthesize feature tracks (with ground truth pose and 3D point locations) for evaluation of SfM algorithms; (2) a new dataset (images and ground truth) that focuses on current pitfall scenarios for SfM (e.g. pure rotation, looming

motion, etc.); (3) experiments verifying that FEATS produces synthetic tracks that represent real world data; and (4) two new evaluations of current SfM algorithms using synthetic features.

2.1 Related Work

Simulation and Synthetic Data: Synthetic data has become popular for solving computer vision problems because of increasingly powerful computer graphics tools and the need for large amounts of ground truth data. Kaneva et al. [96] and Butler et al. [20] generate data for image feature and optical flow evaluation respectively. Battaglia et al. [6] use simulation data to study human interaction with objects. Taylor et. al. [172], Marin et al.[115], Stark et al. [169], and Hattori et al.[88] use synthetic data for training object classifiers. The main challenge is using synthetic data to achieve results that transfer to real data. Some have addressed this challenge by mixing both synthetic and real data. These works include Gaidon et al. [70], Fisher et al. [66], Handa et al. [84, 83], Ros et al. [148], Vazquez et al. [183, 188], and Shotton et al. [162] whom all train models using a mix of synthetic and real data to achieve state-of-the-art results on labeling and classification.

There are also a few similar simulators for 3D reconstruction. Handa et al. [85] uses two virtual 3D scenes to generate RGB-D images from RGB and depth noise models. CARLA (Dosovitskiy et al. [51]) and AirSim (Shah et al. [159]) each provide a small number of highly detailed virtual 3D worlds with moving vehicles and synthetic images and depth maps. FEATS is different because it enables unlimited scenes and camera paths. Moreover, all previously mentioned work synthesizes images. Vaudrey et al. [182] showed that results on synthetic images do not easily transfer to real world results because synthetic images have crisp image boundaries and consistent pixel intensity values. Alternatively, Shotten et al. [162] shows that synthetic geometry can effectively transfer to real world results. We follow in the footsteps of this work and provide the first simulation environment that synthesizes image feature tracks.

Real 3D Reconstruction Data: Collecting ground truth camera pose and scene geometry is difficult, costly, and few datasets exist that provide both. GPS tagged images can have meter level accuracy and are not ideal for ground truth camera pose. Real Time Kinematics (RTK) GPS systems are more accurate, providing centimeter level accuracy. Datasets such as Malaga [14], Rawseeds [142], KITTI Driving [74] and the Cornell Quad [36] all use RTK GPS to provide ground truth camera pose. However, none of these datasets provide ground truth geometry because they cover large outdoor scenes. For small workspaces, motion capture systems can produce millimeter accuracy ground truth camera pose. The TUM RGB-D dataset [171] and EuRoC dataset [19] provide tens of trajectories in small indoor and outdoor scenes with camera pose ground truth from a motion capture system. The EuRoC dataset also provides geometry ground truth using a laser scanner for one indoor scene. Tanks and Temples [101] and ETH3D [156] use laser scanners to provide 3D geometry ground truth of mid-sized scenes (e.g. rooms, courtyards, warehouses, etc.). They align the 3D points estimated by SfM to the 3D models to estimate camera pose ground truth.

Only one dataset (EuRoc) uses both a laser scanner for ground truth geometry with a motion capture system (or RTK GPS) for accurate ground truth camera pose. Also, all previously mentioned datasets are from a ground perspective. Moreover, ground truth camera pose in large indoor scenes is limited, yet accurate camera localization and geometry estimation are particularly important for indoor robot navigation. FEATS mitigates these shortcomings because it provides ground truth camera pose and 3D point locations (enabling 3D point accuracy evaluation) for any scene. We also provide new real world data and camera pose ground truth that focuses on the fundamental motions that are challenging for SfM (e.g. pure rotation, looming motion, etc.).

Planning Trajectories: Works by Dunn et al. [53], Schmid et al. [154], Hollinger et al. [92], Fan et al. [59], Mostegel et al. [121], and Roberts et al. [147] represent an ongoing effort to plan paths for robots for 3D mapping. These works are a complement to FEATS because they can use the current scene to plan a path, then FEATS can predict if the path will be a success.

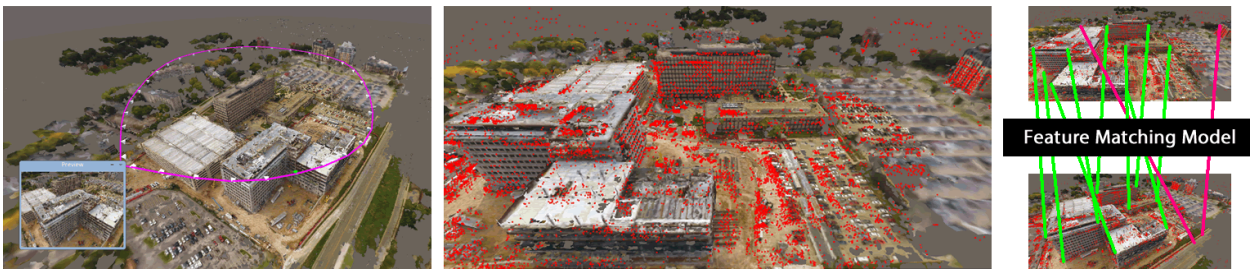


Figure 2.2: Left: Meshes and point clouds set the 3D scene and cameras (white frustums) set the trajectory (magenta line). Middle: 3D points are back projected onto the cameras as 2D image features (red dots). Right: 2D image features are matched across images using our feature matching model.

2.2 Synthesizing Feature Tracks

FEATS is implemented using the Unity3D game engine [176]. FEATS provides tools for a user to create 3D scenes (from imported meshes and point clouds) and define camera trajectories within those scenes (Section 2.2.1). With the 3D scene and a trajectory created, 2D feature and matching noise models are used to synthesize feature tracks with real world characteristics (Section 2.2.2).

2.2.1 Setting up a 3D Scene

FEATS imports both point clouds and meshes to provide a set of 3D points that can be back-projected as 2D keypoints as the camera moves through the scene. Point clouds generated using SLAM/SfM algorithms are ideal since (1) the 3D points are generated by tracking 2D point locations across images, and (2) the sparsely tracked points encode the feature density of the scene (i.e. as opposed to meshes that are dense surfaces). Meshes provide features for tracking if a point cloud is not provided, but are better suited for occlusion detection (i.e. we do not back project 3D points through walls). Ideally, both a point cloud and aligned mesh are imported together (easy to do using SfM and multi-view stereo [68]), providing both 3D points to track and surfaces for occlusion. Note that incorrect points are okay because they become perfectly accurate ground truth points for future reconstructions.

Trajectories are imported or edited/created by placing cameras. As each camera is placed, an

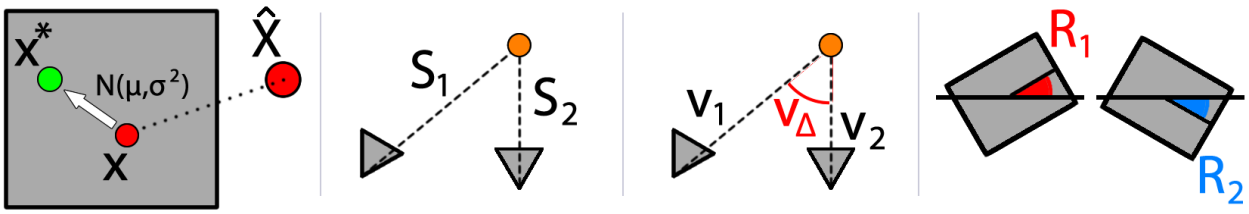


Figure 2.3: Farthest left: 2D noise being added to backprojected 3D points. Middle left: S_1,S_2 for scale match probability. Middle right: v_1,v_2 for viewing direction match probability. Farthest right: R_1,R_2 for rotation direction match probability.

interpolated path is created connecting the cameras sequentially. Cameras can be selected for position and rotation tweaks and a preview window shows the camera's current view. Figure 2.2 provides a depiction of setting up the 3D world and camera placement.

2.2.2 Synthesizing Feature Tracks and Ground Truth

FEATS provides options to dictate the density of frames to capture for a camera trajectory. For each frame, feature tracks are synthesized by first finding the 3D points within the viewing area of that camera and backprojecting them to the image plane of the frame according to our 2D feature noise model. Those points are then matched to points in all subsequent frames based on our matching model (Figure 2.2).

2D Feature Noise: 3D points are back projected to the image plane of a frame using the projective camera model:

$$x_{ij} = K_j[R_j\ t_j]\hat{X}^i \tag{2.1}$$

where K_j is the intrinsic camera matrix for image j, $[R_j\ t_j]$ is the extrinsic camera matrix for image j, $\hat{X}^i$ is position of point i in 3D, and x_{ij} is the 2D position of 3D point $\hat{X}^i$ projected onto image j. In the simulator, all $[R_j\ t_j]$ and $\hat{X}^i$ are known. In our experiments, we use the approach from VisualSfM [185] to estimate the focal length of the camera (f) of K as $f = 1.2 * max(W, H)$ and we assume that the principal point (cx, cy) of K is $cx = W/2$ and $cy = H/2$. W is frame

width and H is frame height.

Noise is then added to each 2D point on the frame (Figure 2.3):

$$x^* = x + \mathcal{N}(\mu,\, \sigma^2) \tag{2.2}$$

where x^* is final 2D position of the synthetic feature point, and $\mathcal{N}(\mu,\, \sigma^2)$ is the normal distribution with mean μ and standard deviation σ. We use $\mu = 0$ and $\sigma^2 = 1$ for all results in this chapter (except σ^2 varies for Section 2.4.1).

Matching Model: A probability model is used to dictate whether a feature point matches across two frames. The probability is calculated based on the difference in scale, viewing direction, and rotation (Figure 2.3). These equations are inspired by the feature matching experiments of Mikolajczyk et al. [119, 118]. The scale match probability (P_{scale}) is defined as:

$$P_{scale} = P^S_{max} * \exp\left(-\left|\frac{S_\Delta}{\alpha_S}\right|\right) \quad \text{and} \quad S_\Delta = \frac{\max\left(S_1, S_2\right)}{\min\left(S_1, S_2\right)} - 1 \tag{2.3}$$

where P^S_{max} is the max probability, α_S is a tuning parameter, and S_1 and S_2 are the distance from the 3D point to each frame's camera center. This model decreases the chances of a match as the difference in scale increases.

The viewing direction match probability (P_{view}) is defined as:

$$P_{view} = P^V_{max} * \exp\left(-\left|\frac{V_\Delta}{\alpha_V}\right|\right) \quad \text{and} \quad V_\Delta = \arccos\left(v1 \cdot v2\right) \tag{2.4}$$

where P^V_{max} is the max probability, α_V is a tuning parameter, and v_1 and v_2 are unit vectors from camera centers of each frame to the 3D point. This model decreases the chances of a match as the difference in viewing direction increases.

The rotation direction match probability (P_{rot}) is defined as:

$$P_{rot} = P^R_{max} - \frac{\alpha_R}{\pi} * R_\Delta \quad \text{and} \quad R_\Delta = \pi - ||R_1 - R_2| - \pi| \tag{2.5}$$

where P^R_{max} is the max probability, α_R is a tuning parameter, and R_1 and R_2 are the camera roll rotations (i.e. rotation about the look-at vector). This model decreases the chances of a match as the difference in orientation increases. The decay is slower because we found in practice that scale and viewing direction differences effect matching probability more than orientation.

The final probability of a match is:

$$P_{final} = P_{scale} * P_{view} * P_{rot}\,. \tag{2.6}$$

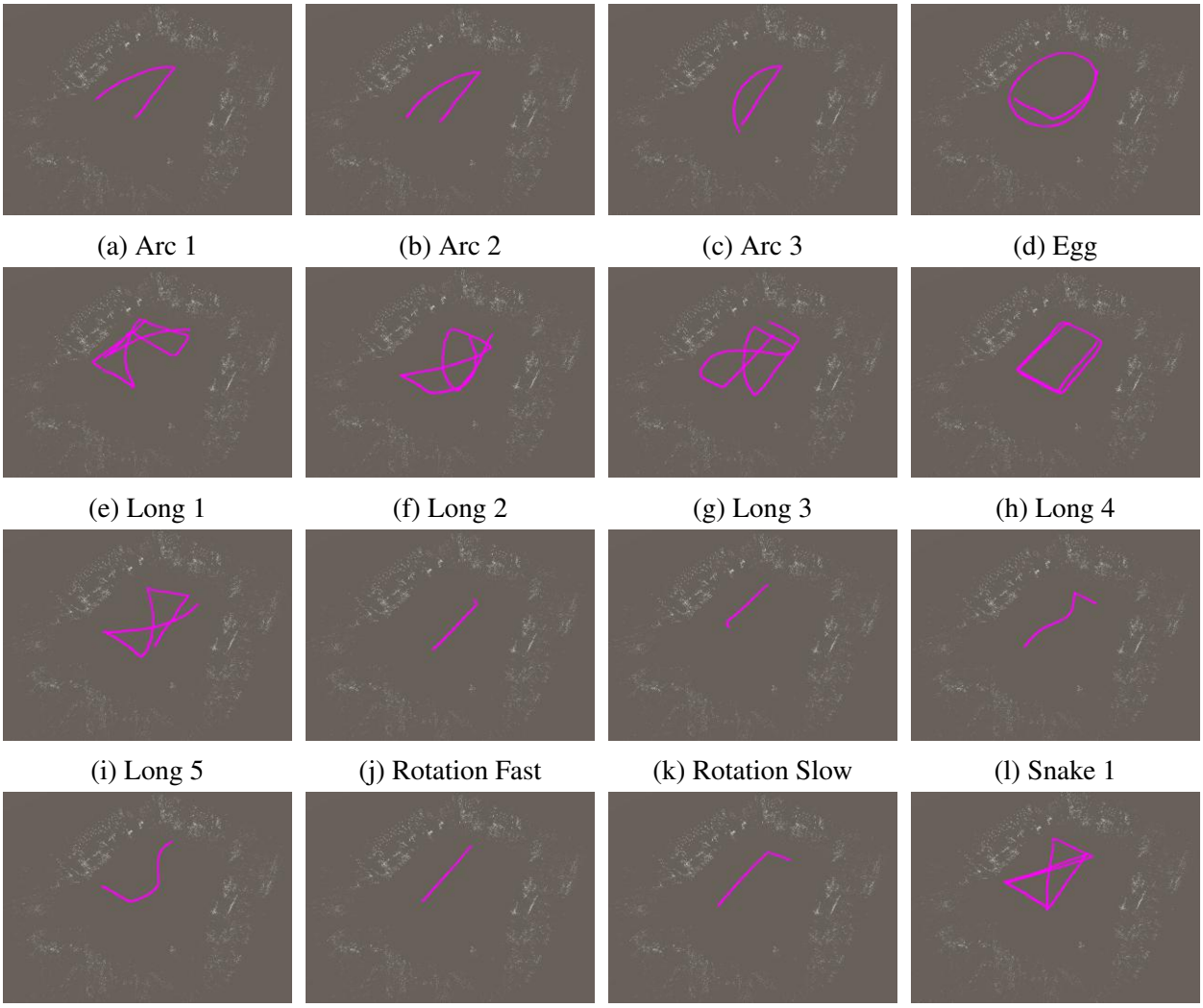
P_{final} is calculated for each feature in each pair of frames. For each feature in a pair of images, if P_{final} is larger than a randomly generated number, then that feature is a match in those images; otherwise it is not a match. Once all matches for a pair of frames are found, $N_{drop}\%$ of the matches are dropped. Lastly, $N_{bad}\%$ of incorrect matches are also added.

We use $P^S_{max} = 0.9$, $\alpha_S = 2$, $P^V_{max} = 0.9$, $\alpha_V = 6$, $P^R_{max} = 1.0$, $\alpha_R = 0.1$, $N_{drop} = 2\%$, and $N_{bad} = 1\%$ for all results in this chapter (except N_{bad} varies for Section 2.4.1). These values were chosen based on the experimentation in Section 2.3.2 and the results in [119, 118].

The final output is (1) a feature file for each frame with the noisy u,v locations and true X,Y,Z 3D positions of each feature point; and (2) a match file for each frame listing all matches to other frames.

2.3 Comparing to Real Data

In this section, we present a new dataset (Section 2.3.1) and use it to show that our matching model is a realistic representation of image matching (Section 2.3.2); and when SfM packages process our synthesized feature tracks, the pose and geometry outputs are representative of the



(a) Arc 1 (b) Arc 2 (c) Arc 3 (d) Egg

(e) Long 1 (f) Long 2 (g) Long 3 (h) Long 4

(i) Long 5 (j) Rotation Fast (k) Rotation Slow (l) Snake 1

(m) Snake 2 (n) Straight 1 (o) Straight 2 (p) X

Figure 2.4: The 16 datasets are depicted. The trajectory of the dataset is depicted as a magenta line. The ORB-SLAM2 map of the motion capture arena is shown as the white point cloud.

results that these SfM packages produce on real data (Section 2.3.3). Comparisons in the remainder of the chapter use three state of the art SfM algorithms: COLMAP [155], OpenSfM [135], and VisualSfM [185].

2.3.1 Trajectories with Pose Ground Truth

In total, 17 datasets (image collections with ground truth 6DoF pose) are collected in a motion capture arena (OptiTrack system [136]). The first dataset is strictly used to generate a point cloud

Figure 2.5: Example images from our 16 new image collections.

Track Names	Number of Images	Track Names	Number of Images
Arc1	107	Long5	254
Arc2	98	Rotation Fast	59
Arc3	123	Rotation Slow	82
Egg	205	Snake1	72
Long1	328	Snake2	74
Long2	302	Straight1	60
Long3	325	Straight2	72
Long4	275	X	276

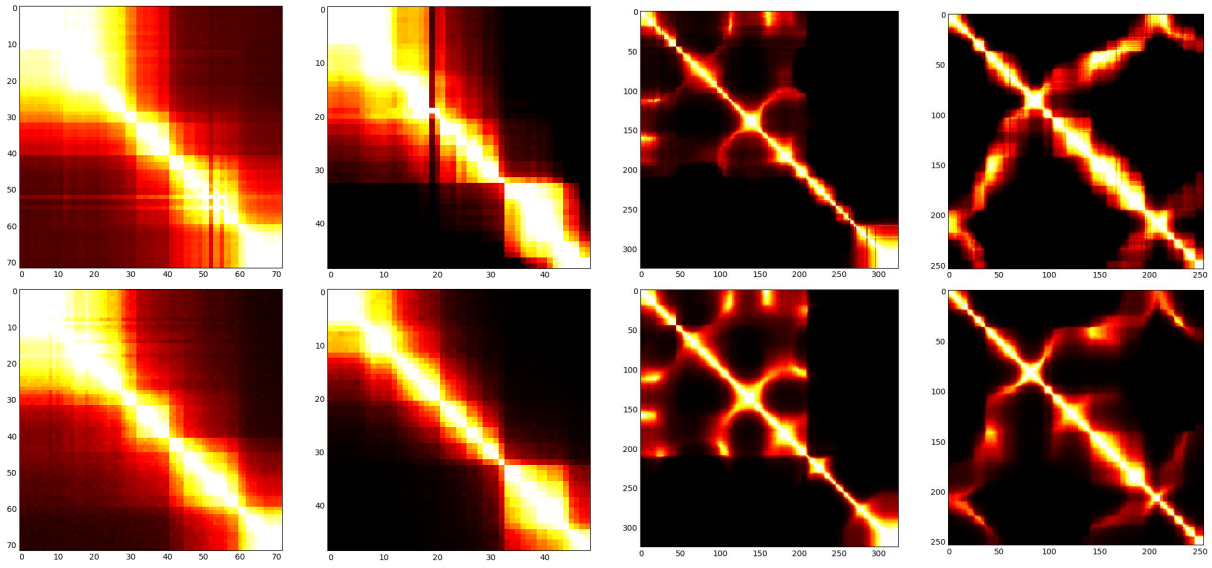
Table 2.1: Number of images in each image collection.

of the motion capture arena using ORB-SLAM2 [125]. ORB-SLAM2 is appropriate because it works well for indoor mapping and uses different features (ORB [149]) than those of SfM pipelines (SIFT [112]). We use coherent point drift [168] to align the ORB-SLAM2 trajectory to the ground truth trajectory and apply that transformation to bring the map into the coordinate frame of the motion capture arena.

The other 16 trajectories are for comparison. Each of these trajectories is meant to focus on specific types of motion (some of which are often difficult for SfM). For example, the Arc datasets add progressively more rotation, the straight datasets have sideways and looming translation, and the long datasets close loops. Figure 2.4 shows each of the 16 trajectories (magenta lines) in the ORB-SLAM2 generated map (white points). The images (colored 752x480 resolution) for these datasets are from a Matrix Vision mvBluefox-200wc camera [116]. Figure 2.5 shows example images from the datasets and Table 2.1 provides the number of images for each trajectory.

2.3.2 Verifying the Match Model

FEATS matching is highly predictive of matching on real images: Figure 2.6 shows comparisons (top: real, bottom: simulated) of the percentage of matches between image pairs for the matching step of COLMAP, OpenSfM, and VisualSfM. In each plot, the x and y axis are sequential frames. For example, reading across row 1 shows the percentage of matches for image 1 compared to each other image. Continuing with this example, the match percentage for row 1, column 10 (i.e. image 1 matching to image 10) is the number of feature matches between image 1 and 10 divided by the number of features in image one. On the other hand, the match percentage for row



(a) Straight2 OpenSfM (b) Rot. Fast COLMAP (c) Long3 OpenSfM (d) Long5 VisualSfM

Figure 2.6: Match percentages (top: real, bottom: synthetic) are shown for the matching steps of COLMAP, VisualSfM, and OpenSfM. All plots are in the same scale (black = 0%, white = 70%). The Pearson correlation between match percentages for "Straight 2" is 0.98 (0.98), "Rotation Fast" is 0.94 (0.92), "Long 3" is 0.93 (0.91), and "Long 5" is 0.88 (0.84). The value in parenthesis is ignoring cells where both match probabilities are below 1%. For all correlation calculations, the diagonal is not used. These Pearson correlation r-values indicate strong positive correlation.

10, column 1 is the number of feature matches between image 10 and 1 divided by the number of features in image 10. The scale of all plots is the same (black = 0% and white = 70%).

Note that the match percentages accurately reflect effects due to trajectory. For example, the "Straight 2" trajectory has very little rotation and mostly looming motion. Thus, we see decreasing match percentages because the scale difference increases (i.e. the chances of matches decreases as the scale difference increases). Similarly, "Rotation Fast" has shifting translation followed by significant out-of-plane rotation, causing the viewing angle difference to increase drastically and reduce the matching percentage around frame 30. For longer datasets with significant translation and rotation (i.e. "Long 3", "Long 5"), we see qualitatively that the synthetic match percentages represent the real match percentages well.

Table 2.2 provides the Pearson correlation r-values between each real and simulated match percentage matrix. For all calculations, the diagonal is ignored because each diagonal cell is an image

	COLMAP	OpenSfM	VisualSfM
Arc 1	0.84 (0.82)	0.91 (0.89)	0.78 (0.78)
Arc 2	0.87 (0.85)	0.92 (0.90)	0.80 (0.80)
Arc 3	0.86 (0.83)	0.96 (0.95)	0.93 (0.90)
Egg	0.88 (0.82)	0.91 (0.88)	0.84 (0.81)
Long 1	0.87 (0.82)	0.91 (0.89)	0.84 (0.78)
Long 2	0.90 (0.88)	0.91 (0.90)	0.86 (0.84)
Long 3	0.90 (0.87)	0.93 (0.91)	0.84 (0.84)
Long 4	0.90 (0.88)	0.93 (0.92)	0.88 (0.86)
Long 5	0.89 (0.86)	0.91 (0.89)	0.88 (0.84)
Pure rotation fast	0.94 (0.92)	0.94 (0.92)	0.93 (0.91)
Pure rotation slow	0.89 (0.87)	0.91 (0.88)	0.89 (0.87)
Snake 1	0.91 (0.91)	0.96 (0.96)	0.83 (0.83)
Snake 2	0.83 (0.83)	0.92 (0.92)	0.74 (0.74)
Straight 1	0.85 (0.85)	0.95 (0.95)	0.74 (0.74)
Straight 2	0.94 (0.94)	0.98 (0.98)	0.82 (0.82)
Trajectory X	0.90 (0.90)	0.95 (0.95)	0.88 (0.88)

Table 2.2: Pearson correlation r-values for the match percentages of real and synthetic data for the matching step of COLMAP, OpenSfM, and VisualSfM. The value in parenthesis is the correlation value ignoring cells of the matrix where both probabilities are below 1% (i.e. close to 0%). All values are above 0.74, indicating a strong positive correlation.

matching with itself. The values in parenthesis are the Pearson correlation r-values with the "zero" probabilities excluded. Specifically, if, for a given cell of the matrix, both the real and simulated match percentages are below 1%, then that value is excluded. For all trajectories for all three SfM methods, the r-values are 0.74 or greater. This indicates a strong positive correlation between the real and synthetic match percentages (values above 0.5 are typically considered a strong correlation [34]), providing evidence that the FEATS matching model represents matching of real images by COLMAP, OpenSfM, and VisualSfM.

We also calculate the average L2 distance in Table 2.3 between each real and simulated match percentage matrix. The total average L2 distance is less than 0.10% for all algorithms and all trajectories. These numbers show that there is no significant scaling or translation between the match percentages, and provide additional evidence that the match model accurately represents real matching for COLMAP, OpenSfM, and VisualSfM.

2.3.3 Verifying the Synthetic Tracks

Figure 2.7 outlines our experimental setup. There are two flows from left to right: one for real data (shown with green boxes and lines) and one for synthetic data (shown with blue boxes and

	COLMAP		OpenSfM		VisualSfM	
Arc 1	0.134	(0.163)	0.084	(0.107)	0.226	(0.231)
Arc 2	0.132	(0.173)	0.088	(0.115)	0.244	(0.245)
Arc 3	0.162	(0.231)	0.060	(0.125)	0.082	(0.168)
Egg	0.052	(0.112)	0.037	(0.081)	0.111	(0.134)
Long 1	0.032	(0.074)	0.019	(0.049)	0.042	(0.095)
Long 2	0.031	(0.044)	0.023	(0.034)	0.040	(0.058)
Long 3	0.030	(0.056)	0.021	(0.042)	0.067	(0.070)
Long 4	0.044	(0.060)	0.024	(0.034)	0.051	(0.069)
Long 5	0.033	(0.058)	0.027	(0.048)	0.038	(0.066)
Rotation Fast	0.234	(0.325)	0.138	(0.241)	0.259	(0.359)
Rotation Slow	0.230	(0.332)	0.132	(0.236)	0.264	(0.368)
Snake 1	0.228	(0.228)	0.129	(0.129)	0.321	(0.321)
Snake 2	0.252	(0.252)	0.131	(0.131)	0.318	(0.318)
Straight 1	0.638	(0.638)	0.273	(0.273)	0.328	(0.330)
Straight 2	0.243	(0.243)	0.131	(0.131)	0.369	(0.369)
X	0.047	(0.063)	0.021	(0.030)	0.055	(0.074)
Total Avg L2	0.050	(0.081)	0.032	(0.055)	0.068	(0.099)

Table 2.3: Comparing the average L2 distances between the match percentages of real and synthetic data for the matching step of COLMAP, OpenSfM, and VisualSfM. The total average error is calculated by summing up all L2 distances for all dataset pairs (real and synthetic) and dividing by the total number of L2 distances. Values in parenthesis ignore cells where both real and synthetic match percentages are below 1%. This table provides additional evidence that the synthetic matching is similar to real matching for these three algorithms.

lines). For clarification, ground truth pose (GT Pose) is real data, but is a blue box because it is used to set the trajectories in the simulator 3D scene (Section 2.2.1).

For the real data, images are input into the SfM pipeline and the output is pose and geometry. For the synthetic data, the ORB-SLAM2 point cloud of the motion capture arena and ground truth pose (GT Pose) are input into the simulator for each dataset. The simulator uses the GT Pose to define the camera trajectory. FEATS uses the feature noise and match model to generate synthetic feature tracks that are input to SfM. The output is pose and geometry. We align the real and synthetic pose estimates to the ground truth pose using Horn's method [93].

This process is repeated for all 16 trajectories and using three state of the art SfM pipelines: COLMAP [155], OpenSfM [135], and VisualSfM [185]. We manually inspect each set of trajectories and tabulate in Table 2.4 whether the reconstructions are successful or not ("F" denotes failure). Examples of the aligned trajectories (both successful and not) are shown in Figure 2.8.

FEATS successfully predicts success and failure: From Table 2.4, we see that the three SfM algorithms fail 8 times in total on the real data and all of those failures are predicted by the syn-

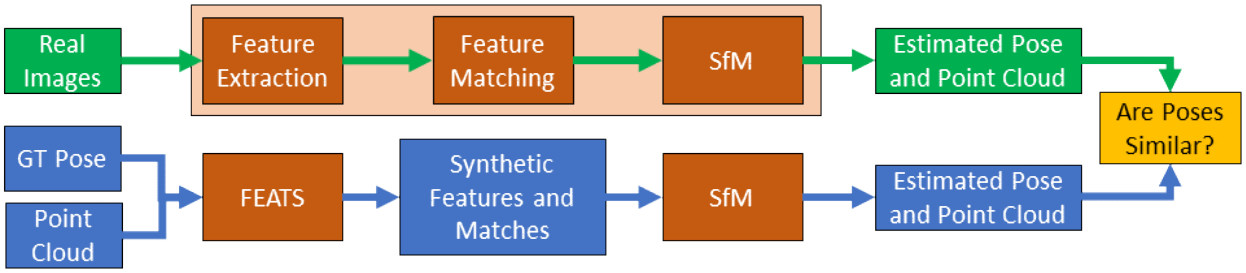


Figure 2.7: We show that synthetic tracks are predictive of results from real data using the following approach. There are two flows from left to right: real images (green boxes and arrows) and simulated tracks (blue boxes and arrows). Real images are input into SfM and pose and geometry are output. The ground truth (GT) pose for the real images and motion capture arena point cloud are input to FEATS. The GT Pose is used by the simulator to generate a camera trajectory through the point cloud. Simulated tracks are output and run with SfM to generate pose and geometry estimates. The output pose from the synthetic and real data are compared to show that the simulated tracks represent real data.

thetic data (recall of 100%). The simulated data predicts failure 10 times in total (precision is 80%). For classifying success, recall is 100% and precision is 95%. **Overall, the accuracy is 96% (46/48).** These results are evidence that the synthetic data is an effective representation of the real data. Moreover, these results provide evidence that FEATS can reliably predict that a reconstruction will fail for a certain input, which can be used to test and adjust planned data captures before the time and effort is expended to collect the data.

2.4 SfM Evaluations Enabled by FEATS

In this section, we demonstrate two new methods to evaluate SfM that are enabled by FEATS: (1) comparing SfM as 2D point noise and bad match percent vary and (2) calculating 3D point error.

2.4.1 SfM Robustness to Feature Noise and Bad Matches

We use FEATS to generate 99 trajectories of Arc1 while varying the percent of bad matches and σ^2 in Equation 2.2. We vary the percent of bad matches between 0% and 10% by increments

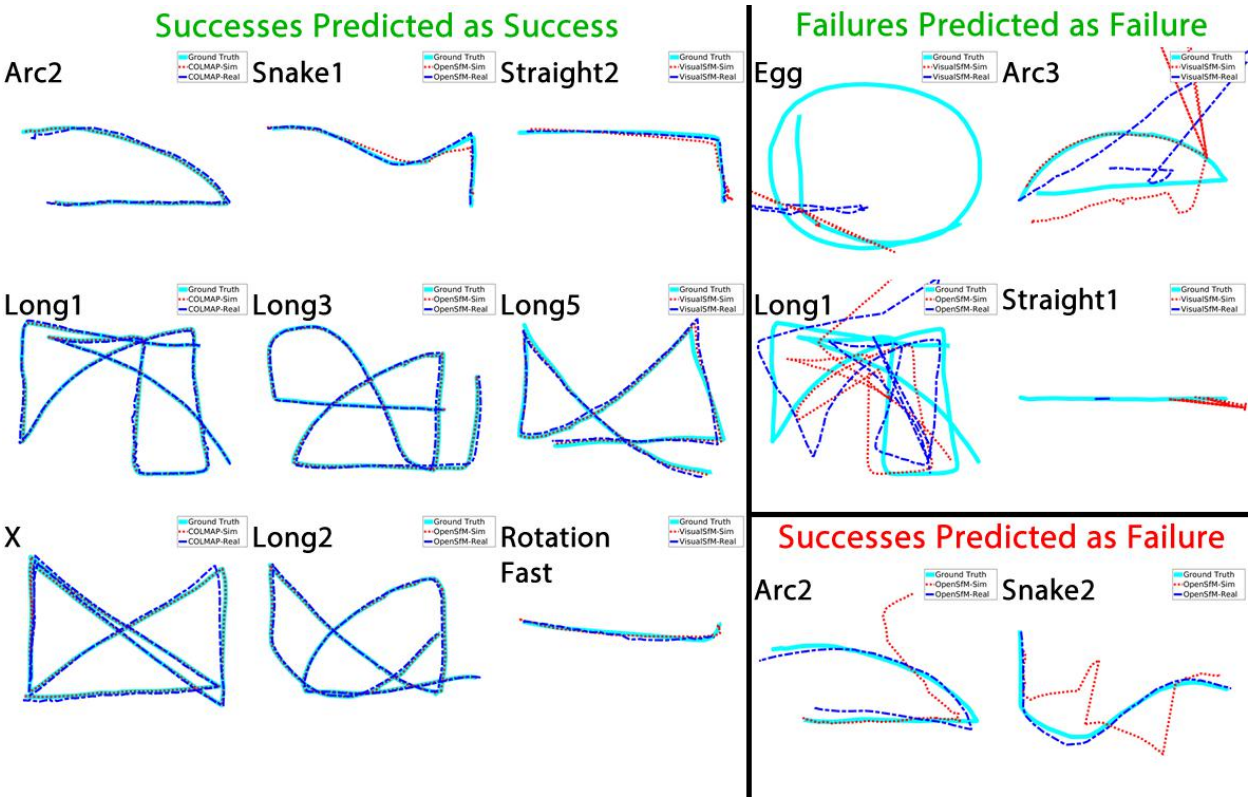


Figure 2.8: Trajectories for synthetic (red dotted) and real (blue dashed) data are aligned to ground truth (cyan solid). The left section shows successful reconstructions that the synthetic data also predicts as successful (column 1 for COLMAP, column 2 for OpenSfM, column 3 for VisualSfM). The top right section is failure reconstructions that the synthetic data also predicts as failures. The bottom right section is successful reconstructions that are predicted as failures.

of 1%. We vary σ^2 between 0 and 4 by increments of 0.5. We then process these 99 trajectories with COLMAP, OpenSfM, and VisualSfM. The percent of localized images for each algorithm is shown in the left 3 plots of Figure 2.9 (yellow = 100%, dark blue = 0%).

COLMAP is robust to both types of noise, registering all images. OpenSfM is sensitive to bad matches. On the other hand, VisualSfM is less sensitive to bad matches, but more sensitive to 2D feature noise. Comparing COLMAP to OpenSfM provides evidence that COLMAP's scene graph augmentation and new triangulation approach [155] (implementations that differ for OpenSfM) are effective in overcoming 2D noise and bad matches. For VisualSfM, the noise causes inconsistent results, which OpenSfM and COLMAP may not exhibit because COLMAP and OpenSfM have additional outlier filtering and retriangulation methods used during bundle adjustment. In all cases,

	COLMAP			OpenSfM			VisualSfM		
Arc 1		/			/			/	
Arc 2		/			/	F		/	
Arc 3		/			/		F	/	F
Egg	F	/	F		/		F	/	F
Long 1		/		F	/	F	F	/	F
Long 2		/			/			/	
Long 3		/			/			/	
Long 4		/			/			/	
Long 5		/			/			/	
Rotation Fast		/		F	/	F		/	
Rotation Slow		/		F	/	F		/	
Snake 1		/			/			/	
Snake 2		/			/	F		/	
Straight 1		/			/		F	/	F
Straight 2		/			/			/	
Trajectory X		/			/			/	

Table 2.4: Blank entries indicate that the reconstruction is correct. The notation is (real/synthetic). "F" indicates failure. There are 8 failures on real data and the synthetic data predicts 8 of them (recall of 100%). The synthetic data incorrectly predicts 2 failures that do not occur (precision of 80%). The overall accuracy is 96% (46/48). This is evidence that the synthetic data represents the real data well and helps predict failure reconstructions.

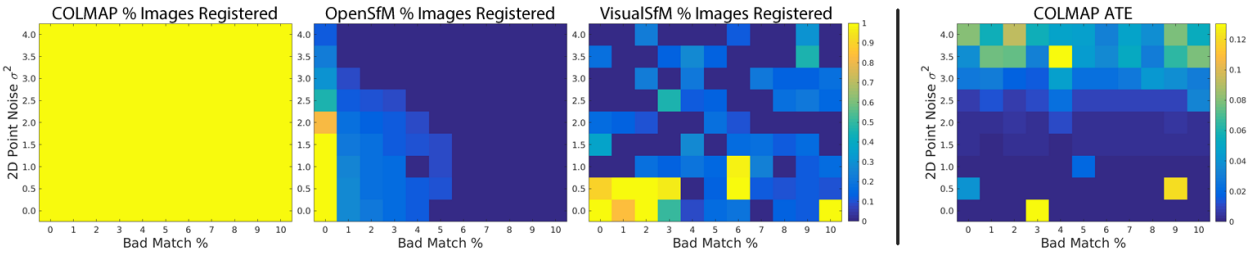


Figure 2.9: The left three plots show the percent of localized images as 2D gaussian noise σ^2 and bad match percent vary (yellow = 100%, dark blue = 0%). COLMAP registers all images for all parameters, demonstrating that COLMAP is robust to bad matches and feature point noise. The right plot shows that absolute trajectory error [171] generally increases as noise increases for COLMAP.

failed reconstructions are marked as 0% and incorrectly registered images are not counted.

Since COLMAP registers all images for all trajectories, we calculate the absolute trajectory error (ATE) [171] in the right plot of Figure 2.9. It is interesting to see how the ATE increases significantly as the 2D noise approaches $\sigma^2 = 4$.

2.4.2 Calculating 3D Point Error

It is challenging to evaluate the accuracy of a 3D reconstructed point cloud quantitatively. Instead, qualitative measures are used (i.e. does the point cloud look "correct","clean", and "good").

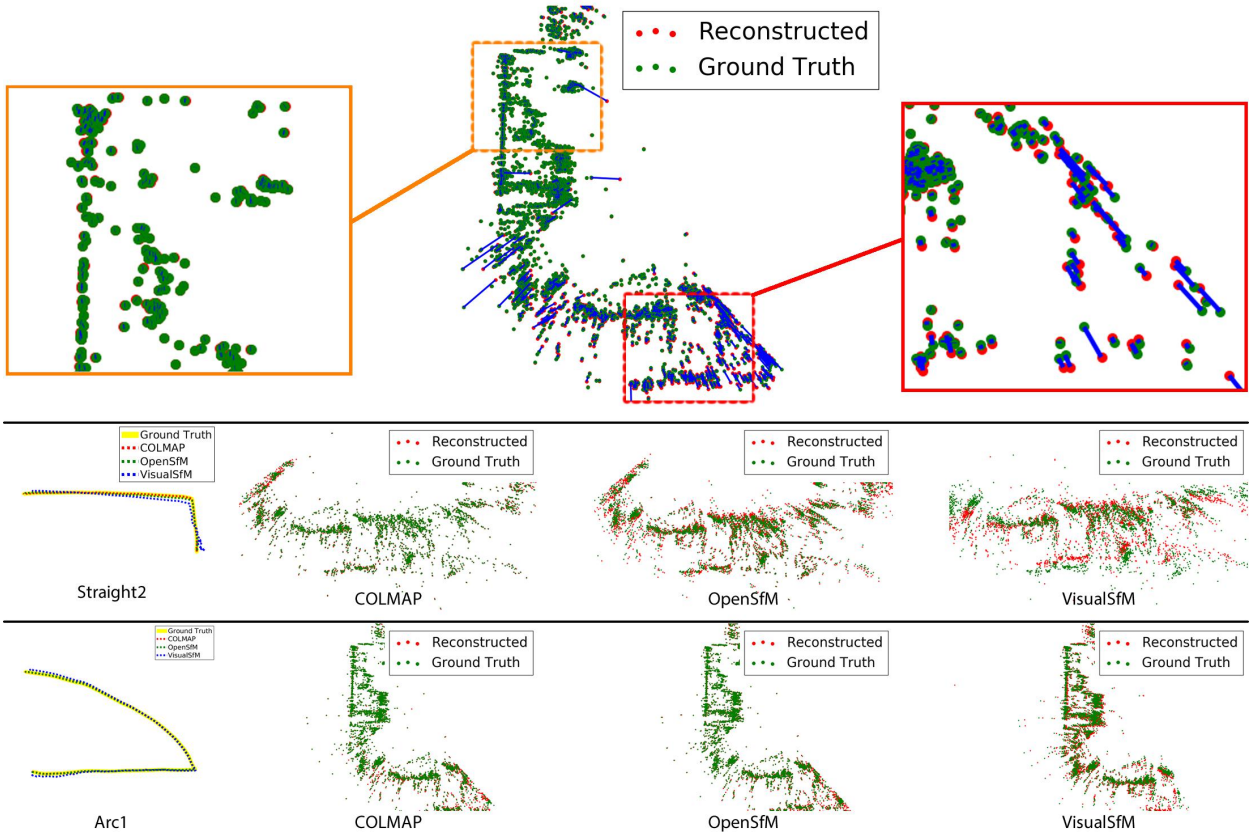


Figure 2.10: The ground truth point cloud (i.e. the map points used by FEATS) is shown in green. The reconstructed point cloud (i.e. from synthetic feature tracks processed by SfM) is shown in red. In the top figure, blue lines depict the errors between the reconstructed points and the ground truth points. Calculating this 3D point error is enabled by FEATS. In the bottom two figures, the trajectories are plotted, showing that pose error is low; however, the 3D point errors are much different for each algorithm.

Even when ground truth geometry is provided by a CAD model or laser scanner (typically a mesh), it is difficult (or impossible) to find the corresponding point on the ground truth mesh for each 3D reconstructed point. The typical approach is to fit the point cloud to the mesh and find the shortest distance between each point and the mesh. This is not an ideal scenario for quantitative evaluation of the quality of the reconstructed point cloud.

With FEATS, for each reconstructed 3D point, the corresponding ground truth 3D point coordinates are known (see Figure 2.10). This makes it easy to calculate the 3D point error for 3D reconstructions. We calculate the average L2 point error for each reconstructed point cloud and

	COLMAP	OpenSfM	VisualSfM
Arc 1	**21.5**	26.0	84.7
Arc 2	**23.0**	-	110.7
Arc 3	**16.8**	22.2	-
Egg	-	**20.2**	-
Long 1	**8.3**	-	-
Long 2	**9.0**	12.9	461.2
Long 3	10.7	**10.6**	138.8
Long 4	9.3	**7.3**	100.3
Long 5	**5.1**	7.3	70.1
Pure rotation fast	**33.2**	-	208.5
Pure rotation slow	**10.8**	-	212.1
Snake 1	**18.6**	271.7	155.2
Snake 2	**31.1**	-	171.9
Straight 1	**20.6**	171.0	-
Straight 2	**27.2**	118.1	246.8
Trajectory X	**7.0**	11.7	109.1

Table 2.5: Average 3D point error is calculated (in millimeters) between the ground truth 3D points and the 3D points reconstructed from synthetic data. The lowest error for each trajectory is bold. Failure reconstructions are denoted by "-". Points more than 10 meters from their ground truth position are not included. Note that some errors are quite high despite reasonable trajectory estimates. This error measurement is not possible without correspondences between the ground truth 3D points and the reconstructed 3D points, which FEATS provides.

ground truth point cloud using the following equation:

$$\frac{1}{N}\sum_{i=1}^{N} ||X_i - X_i^*|\,|_2 \tag{2.7}$$

where N is the number of points in the reconstruction, X_i is 3D point i from the reconstruction, and X_i^* is the ground truth 3D point corresponding to X_i. The resulting average point errors (in millimeters) are provided in Table 2.5. Failed reconstructions are not included and egregious (10 meters or more) outlier points are also not included in the calculation.

Table 2.5 shows that COLMAP and OpenSfM provide the most accurate point clouds. This is evidence that the retriangulation and bundle adjustment additions in these approaches [155] are effective at improving point cloud accuracy over previous methods (e.g. VisualSfM). Figure 2.10 provides examples of aligned trajectories and points for all three SfM approaches for Arc1 and Straight2. These plots show that the camera localizations are accurate for all three pipelines, yet the point error is noticeably different (i.e. tens vs hundreds of millimeters in error according to Table 2.5). This indicates that accurate pose estimates do not necessarily mean accurate 3D point locations, reinforcing the need for new quantitative metrics for measuring 3D point error.

2.5 Conclusions

In this chapter, we present the design and implementation of FEATS, a simulation environment that models feature noise and matching to generate feature tracks from camera trajectories in virtual 3D scenes. We show the synthetic tracks are representative of real world data by comparing the percentage of matches between image pairs of real and synthetic data and by using the simulated data to predict reconstruction successes and failures. We then use synthetic data to show (1) COLMAP is quite robust to 2D feature noise and bad matches; and (2) accurate camera localizations do not guarantee accurate point clouds, reinforcing the need for ground truth 3D points.

Chapter 3

ChromaTag: A Colored Fiducial Marker and Fast Detection Algorithm

In this chapter, we introduce ChromaTag , a new colored fiducial marker and detection algorithm that is significantly faster than current fiducial marker systems. Fiducial markers are artificial objects (typically paired with a detection algorithm) designed to be easily detected in an image from a variety of perspectives. They are widely used for augmented reality and robotics applications because they enable localization and landmark detection in featureless environments. However, because fiducial markers often complement large real-time systems (e.g. Camera Tracking [24, 4], SLAM [189] and Structure from Motion [22]), it is important that marker detection runs much faster than 30 frames per second. Figure 3.1 shows the run time of several state of the art markers, of which only ChromaTag achieves processing times significantly faster than 30 frames per second (all processing uses a 3.5 GHz Intel i7 Ivy Bridge processor on 752x480 resolution images).

ChromaTag achieves an order-of-magnitude speedup with good detection performance through careful design of the tag and detection algorithm. Previous marker designs (Figure 3.2) typically use highly contrasting (black to white) borders [63, 134, 28, 21, 22, 8] for initial detection, but black-white edges are common in images and result in many initial false detections. IDs are decoded from the tags to verify detections, but decoding is the last step in the pipeline, so most of the time is spent rejecting false tags. Tags with distinctive color patterns can be used to limit initial false detections, but color consistency and the reduced spatial resolution of color channels (Bayer grid) create challenges for ID encoding and tag localization.

ChromaTag uses each channel of the LAB opponent colorspace to best effect. Large gradients between red and green in the A channel, which are rare in natural scenes, are used for initial detection. This results in few initial false detections that can be quickly rejected. The black-white border takes advantage of high resolution of the L channel for precise localization. The B channel

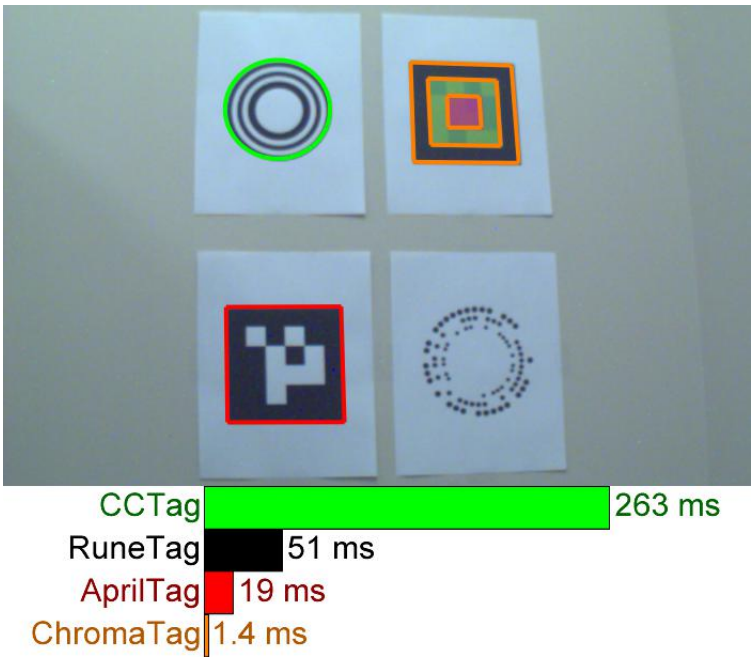
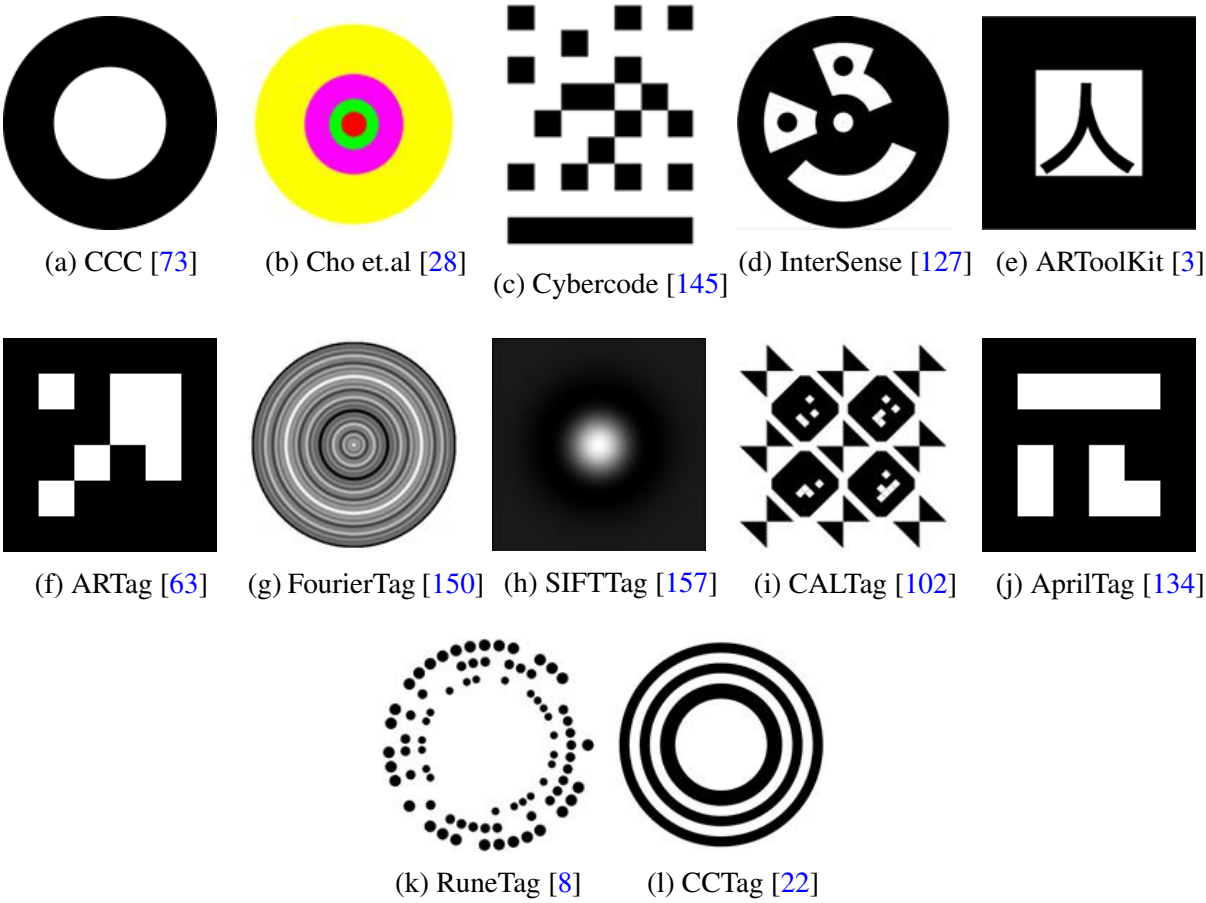


Figure 3.1: ChromaTag is a colored fiducial marker and detection algorithm that is significantly faster than other markers. Shown are successful detections of CCTag [22], AprilTag [184], and ChromaTag (RuneTag [8] was unsuccessful), with the time required by each tag's detection algorithm. The images in this chapter are best viewed in color.

is used to encode the tag ID. Our ChromaTag detection algorithm finds initial detections, builds a polygon on the borders, simplifies to a quadrilateral, and decodes the ID. Robustness to variations in lighting is achieved by using differences of chrominance and luminance throughout detection and localization. The algorithm is fast and robust and achieves precise tag localizations at more than 700 frames per second.

We collect thousands of images with ChromaTag and state-of-the-art tag designs in a motion capture arena. We use this data to demonstrate that our tag achieves significantly faster detection rates while maintaining similar or better detection accuracy for varying camera perspectives and lighting conditions. We tabulate which steps in the ChromaTag detection algorithm most often fail and consume the most time. We also evaluate how image tag size and camera viewing direction affect detection accuracy. Note that an unlabeled training video was used during algorithm design and parameter setting; the parameters are not tuned for the held out test sets.

In summary, the **contributions** described in this chapter are: (1) we present ChromaTag, a col-



(a) CCC [73] (b) Cho et.al [28] (c) Cybercode [145] (d) InterSense [127] (e) ARToolKit [3]

(f) ARTag [63] (g) FourierTag [150] (h) SIFTTag [157] (i) CALTag [102] (j) AprilTag [134]

(k) RuneTag [8] (l) CCTag [22]

Figure 3.2: Existing Fiducial Marker Designs

ored fiducial marker and detection algorithm; (2) we demonstrate that our tag achieves accurate detections faster than current state-of-the-art markers using thousands of ground truth labeled image frames with different lighting and varying perspectives; and (3) we show how ChromaTag and other fiducial markers perform as a function of image tag size and viewing direction.

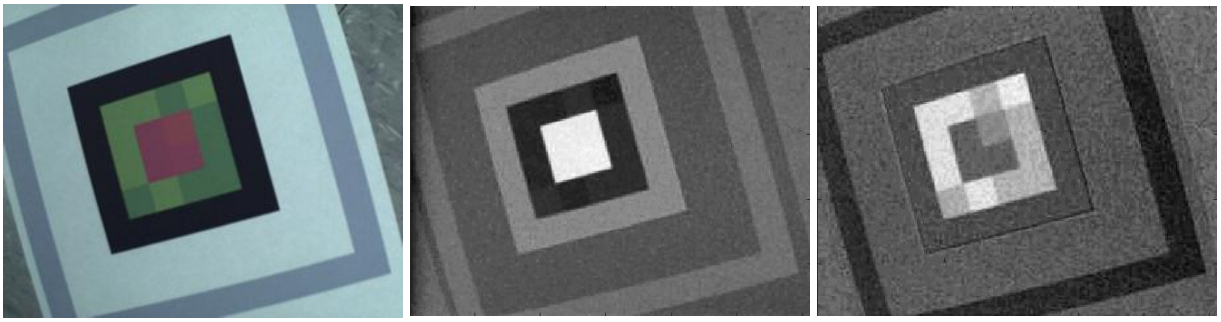
3.1 Related Work

Figure 3.2 shows many of the tags discussed in this section. Among the earliest fiducial markers is the concentric contrasting circle (CCC) of Gatrell et al. [73] which consists of a white inner circle surrounded by a black ring with an outer white border. CCC has no signature to differentiate markers. Cho et al. [28] adds additional rings to improve detection at different depths and color

to each ring as a signature. CyberCode (Rekimoto et al. [145]) is a square tag with a grid of square white and black blocks to encode signatures. Naimark et al. [127] combines the ring design of CCC with the block codes of CyberCode to create a circular marker with inner block codes. ARToolKit [3] and Fiala's [63, 64] ARTag are the first tags with widespread use for augmented reality [24]. Both tags borrow from past success with square designs by using a white square border around a black inner region. ARToolKit uses different symbols to differentiate markers while ARTag uses a grid of white and black squares. ARTag also uses hamming distance to improve false positive rejection.

Fourier Tag, proposed by Sattar et al. [150] and improved by Xu et al. [187], is a circular tag with a frequency image as the signature which results in graceful data degradation with distance. Schweiger et al. [157] uses the underlying filters of SIFT and SURF as the design motivation for their markers, which look like Laplacian of Gaussian images and are specifically designed to trigger a large response with SIFT and SURF detectors. Checkerboard-based markers increase the number of corners for improved camera pose estimation: CALTag by Atcheson et al. [102] is a checkerboard with inner square markers for camera calibration and Neto et al. [38] adds color to increase the size of the signature library and remove the perspective ambiguity of checkerboard detection. Herout et al. [90] create a marker field consisting of varying sized grid spaces with different shades of gray that are arranged so that the edges of the grid spaces meet at two vanishing points; enabling accurate, fast detection despite blur and steep viewing directions.

Olson and Wang's AprilTag [134, 184] is a faster and more robust reimplementation of ARTag. Garrido-Jurado et al [71, 72] uses mixed integer programming to generate additional marker codes for the square design of ARTag and AprilTag and provide their codes with the ArUco fiducial marker library [4]. Another state-of-the-art tag (RuneTag) comes from the work of Bergamasco et.al[9, 10, 8] which uses rings of dots to improve robustness to occlusion and provide more points for camera pose estimation. Lastly, CCTag by Calvet et al. [21, 22] uses a set of rings like that of Prasad et al. [141] to increase robustness to blur and ring width to encode marker signature. These tags (AprilTag, RuneTag, and CCTag) represent the current state-of-the-art for fiducial



(a) Original Image (b) A Channel (c) B Channel

Figure 3.3: The original image is converted to the LAB color space and the A and B channels are shown. In the A channel, the different shades of red look the same and make a bright square region; and the different shades of green look the same and make a dark ring. This makes it easy to detect the red to green border in the A channel. In the B channel, the different shades of red and green become differentiable, making it possible to read the binary code for the tag.

markers. They have demonstrated accurate detection and are commonly used. ChromaTag uses a new detection approach (taking advantage of color on the markers) to achieve processing times significantly faster than other tags while still maintaining similar detection accuracies.

Several works perform planar object detection using machine learning approaches. Early work by Claus et. al [32, 33] employs a cascaded Bayes and nearest neighbor classification scheme for marker detection. More recent work of Ozuysal et al. [138] and Lepetit et al. [106] uses randomized forests to learn and detect planar objects. In practice, these algorithms do not achieve detection accuracies on par with detection algorithms specifically designed for marker detection. With the increased popularity of deep learning based detection approaches, it is possible that better detection can be achieved; however, ChromaTag, which does not require a GPU, is faster than current deep learning approaches which do require a GPU [146, 144].

3.2 ChromaTag Design

Figure 3.4 provides an example of a ChromaTag. The inner red square and green ring are used for rejecting false positives (Section 3.2.1) and the outer black and white rings are used for precise localization of the tag (Section 3.2.2).

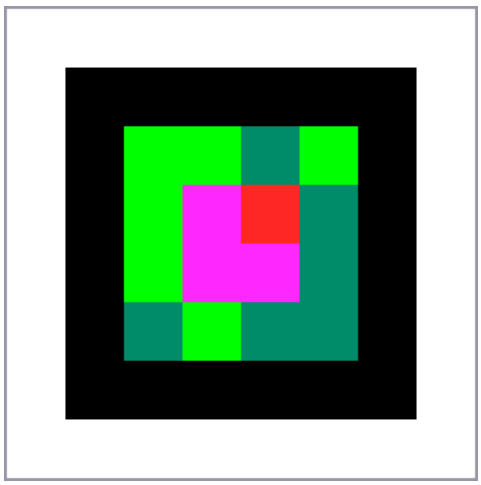


Figure 3.4: An example ChromaTag. The red and green regions are easily detectable and limit false positives. The two shades of red and two shades of green are used to embed the tag code. The outer black-white border provides full spatial resolution for accurate localization of the tag corners.

3.2.1 Efficient False Positive Rejection

Opponent color spaces offer large gradients between opposing colors in each channel. Figure 3.5 depicts the LAB color space, where red and green are opposing colors in the A channel and blue and yellow are opposing colors in the B channel. Figure 3.5 also depicts how two (or more) shades of green or red can have the same value in the A channel, but different values in the B channel. ChromaTag's color configuration is designed based on these properties.

The red center surrounded by the green ring has a large gradient in the A channel (Figure 3.3b), which rarely occurs in natural scenes (empirically validated in Figure 3.11). Thus, we can quickly detect tags with high precision and recall by scanning the image in steps of N pixels and thresholding the A channel difference of neighboring steps.

ChromaTag encodes the binary code in the B channel (Figure 3.3c), which has little effect on the A channel intensity (Figure 3.3b). Since every tag includes low and high B values (encoding 0 and 1) in both the green and red area, the thresholds are adapted per tag to account for variations due to lighting or printing. The tag detection is verified based on the code value, enabling high precision.

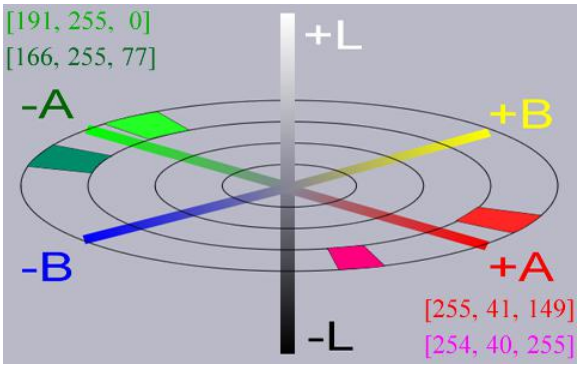


Figure 3.5: The LAB colorspace. ChromaTag uses red to green borders because they have a large image gradient in the A space. ChromaTag uses two shades of red and two shades of green as representations of 0 and 1 to embed a code in the tag. These different shades of red and green are only differentiable in the B channel, so detection is done in the A channel and decoding is done in the B channel. RGB values for the different red and green colors are shown.

We found LAB to be more robust to color printer and lighting variations than other color spaces such as YUV. We use hashing to speed computation of LAB values.

3.2.2 Precise Localization

Precise localization is required to decode the tag and recover camera pose. Chrominance has lower effective resolution than luminance due to the Bayer pattern filters used in common cameras. ChromaTag is designed with outer black and white concentric rectangles to provide high contrast and high resolution borders for precise corner localization.

3.3 ChromaTag Detection

Algorithm 3.1 outlines ChromaTag detection. Substantial effort was required to make each step of the detection computationally efficient (700+ FPS on 752x480 resolution images!) and robust to variable lighting and perspective.

FindADiff: The first step is to find potential tag locations. Assuming that tag cells are at least $N/2$ pixels wide, we search over pixels on a grid of every N^{th} row and column. If a sampled pixel at location (i, j) is in an already-detected area (*InPreviousDetectionArea(i,j,Dets)*), j is moved to

Algorithm 3.1: ChromaTag Detection

Input:
Im = Input RGB image
N = Stepsize in pixels (4 in our experiments)
ADiffThresh = Threshold for initial detection (25 in our experiments)

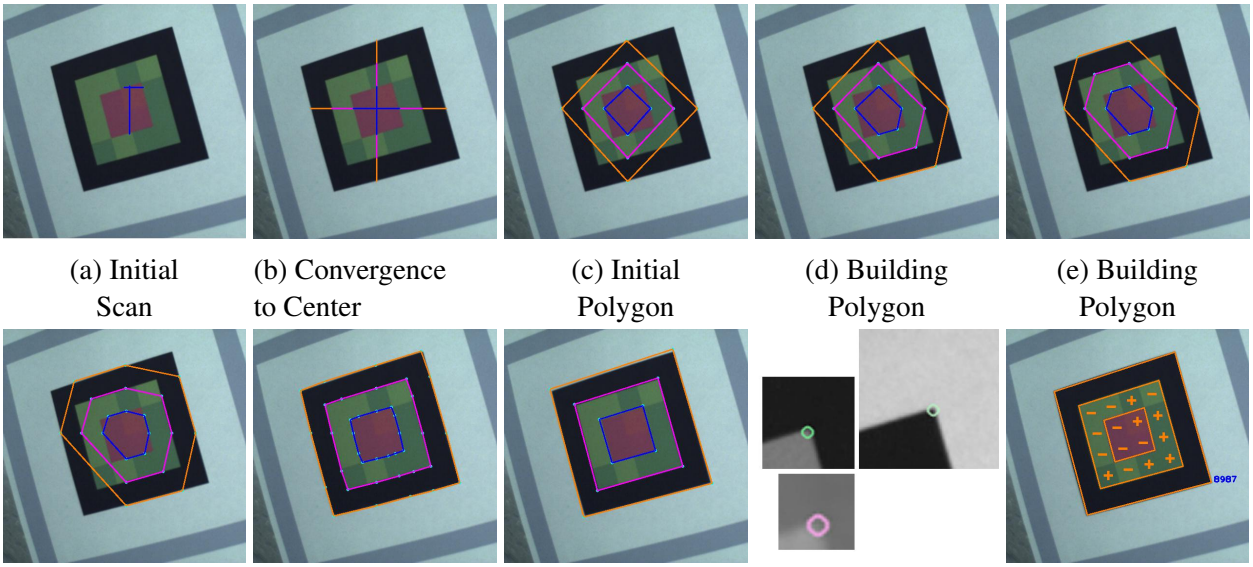
Output:
Dets = Struct to hold detections
TmpDet = Holds detections as they are built

```
for i=0; i < Im.Rows(); i+=N do
    OldA = ConvertToLAB( Im(i,j) )
    for j=0; j < Im.Cols(); j+=N do
        if InPreviousDetectionArea(i,j,Dets) then
            j = MoveJToEndOfDetection(i,j)
            OldA = ConvertToLAB( Im(i,j) )
            continue

        [L,A,B] = ConvertToLAB( Im(i,j) )
        if A - OldA > ADiffThresh then
            if InitialScan(Im,i,j,A,TmpDet) then
                if BuildPolygon(Im,TmpDet) then
                    if PolyToQuad(Im,TmpDet) then
                        if Decode(Im,TmpDet) then
                            Dets.Add(TmpDet)
        OldA = A
```

the next grid location outside the tag detection (*MoveJToEndOfDetection(i,j)*). Otherwise, the pixel is converted to the LAB space (*ConvertToLAB(Im(i,j))*), and the A channel intensity is compared to the previous grid location's A channel intensity (*A - OldA > ADiffThresh*). If the difference is greater than *ADiffThresh*, detection commences and the A value is set as *ReferenceRed*; otherwise, the loop continues to the next sampled pixel. In our experiments, *N=4* and *ADiffThresh=25*.

InitialScan(Im,i,j,A,TmpDet): The next step is to reject red locations that are not tags so that no further processing is done at that location. From the grid location *(i,j)*, we scan left, right, up and down as shown in Figure 3.6a. Each scan continues until M successive pixels have A



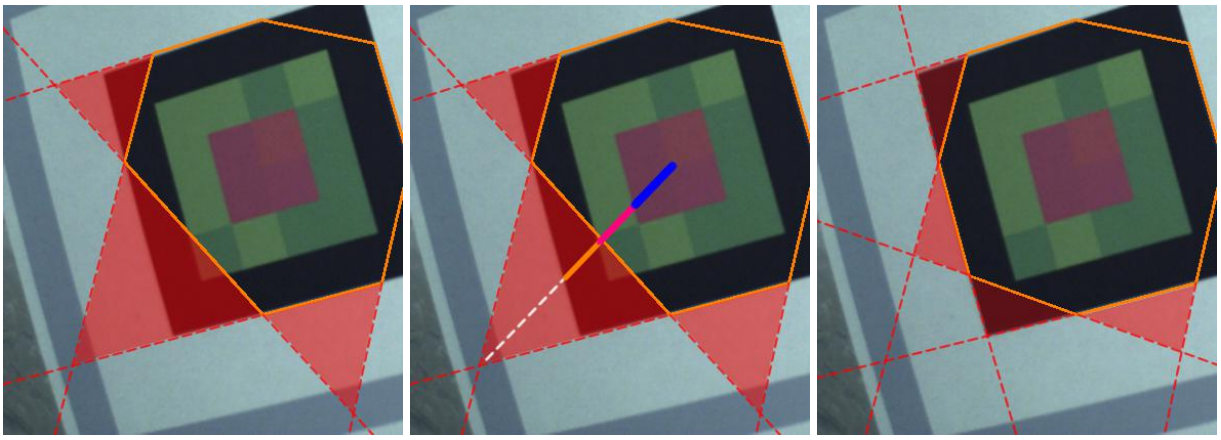
(a) Initial Scan (b) Convergence to Center (c) Initial Polygon (d) Building Polygon (e) Building Polygon

(f) Building Polygon (g) Polygon Convergence (h) Fit Quad to Polygon (i) Refine Corners (j) Final Detection

Figure 3.6: Scans up, down, left, and right are done to find the red-green border (Figure 3.6a). If the red-green border is found, the center point is adjusted and scans are repeated until the center point converges. Scans then continue to find all borders (Figure 3.6b). An initial polygon is built from the found border points (Figure 3.6c). Additional scans add new points to the polygon (Figures 3.6d, 3.6e, and 3.6f). After enough scans, the polygon converges to the tag quadrilateral (Figure 3.6g). Polygon edges are clustered into four edges of a quadrilateral (Figure 3.6h). Patches around each quadrilateral corner are searched for a precise corner location (Figure 3.6i). A homography is fit and a grid of pixel locations are sampled to decode the tag (Figure 3.6j).

channel differences greater than $BorderThresh$ when compared to *ReferenceRed*. If successive pixels are found, scanning has entered the green region and the border *(u,v)* location and pixel value (*ReferenceGreen*) are remembered. If successive pixels are not found, we return *false* and the grid location is abandoned. In our experiments, $M = 3$ and $BorderThresh = 5$.

If all four scans (up, down, left, right) find the green region, we average the *(u,v)* locations on the red-green border to estimate the center of the tag and repeat the scan. If any scan fails, we return *false* and the grid location is abandoned. When the center location converges, we continue the scans through the green region to find the green-black and black-white borders. These scans compare against a set *ReferenceGreen* or *ReferenceBlack* respectively and use the same M and $BorderThresh$ parameters. Center convergence and scans are depicted in Figure 3.6b. If all scans are successful, the *(u,v)* locations for each border are used to build three initial polygons as



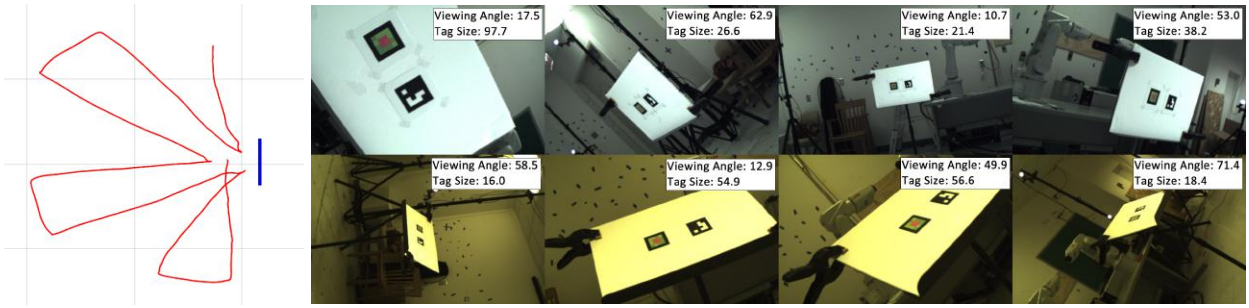
(a) Potential Area (b) Scanning (c) Adding an Edge

Figure 3.7: The potential areas (shown in red) are constructed from neighboring edges extending to an intersection point. The intersection point defines the apex of the triangle and the maximum area that can be added to the current polygon without violating convexity. The next scan moves towards the apex of the largest potential area and finds each border along the way. The new points are added to each border (only outer border is shown).

shown in Figure 3.6c.

BuildPolygon(Im,TmpDet): The next step is to expand the initial polygon to match the tag borders. Because squares project to quadrilaterals in the image (ignoring lens distortion), the polygon must remain convex. This limits the maximum potential area that can be added to the polygon with the addition of a new point. Figure 3.7 shows the maximum area associated with each edge. We build the polygon by greedily scanning in the direction of maximum potential area. The scanning procedure is the same as that described for *InitialScan*. Figures 3.6d, 3.6e, 3.6f demonstrate how iterative scans add points to the polygon along the tag border. Convergence is reached when the ratio of potential areas and polygon area is greater than $ConvThresh$; resulting in a polygon roughly outlining the tag border (Figure 3.6g). If a scan fails, *false* is returned and the grid location is abandoned. In our experiments, $ConvThresh = 0.98$.

PolyToQuad(Im,TmpDet): Next, a quadrilateral is fit to the polygon. We cluster the angles of the edges weighted by edge length using K-Means ($K = 4$). The cluster centers and outer-most point of each cluster defines four lines and their intersections form the four corners of the quadrilateral (Figure 3.6h). A patch around each corner is searched using GoodFeaturesToTrack [161],



(a) Data Collection Path (b) Example Data Under Two Lighting Conditions

Figure 3.8: The tag (blue) and the camera trajectory (red) are shown. Example images annotated with tag size (square root of area) and viewing angle are shown in Figure 3.8b (top row: white balance (WB); bottom row: no white balance (NWB)).

and the highest scoring point is saved as the new quadrilateral corner. Patch size is scaled by the size of the quadrilateral. Figure 3.6i shows an example patch and corner for each border.

Decode(Im,TmpDet): A homography matrix is estimated from the black-white border corners. A grid is fit to the black-white border using the homography and the pixel at each grid location is converted to the B channel. The red and green pixels are each clustered separately where the decision boundary is the midpoint of the max and min. The clusters represent the 1 and 0 values that define the tag signature. Figure 3.6j shows the grid of samples that were decoded in order to finalize detection of the tag. For clustering to work, both shades of red and both shades of green must be represented in the tag; we remove from established tag libraries any codes that do not satisfy this requirement. The decoded signature is identified as a match using a precomputed hash-table containing all the signatures as was done for AprilTag [184]. If a match is not found, *false* is returned and the grid location is abandoned.

3.4 Results and Discussion

We collect six datasets for comparison of ChromaTag against AprilTag [184], CCTag [22], and RuneTag [8]. Colored images are captured at 30 fps with a resolution of 752 x 480 using a Matrix Vision mvBluefox-200wc camera [116]. The camera is attached to a servo motor that continuously rotates the camera inplane between 0 and 180 degrees. For each dataset, ChromaTag and one of

	Average Frames Per Second		
	Total	> 0 Detections	0 Detections
ChromaTag	926.4	709.2	2616.1
AprilTag	56.1	56.3	49.0
CCTag	10.0	6.5	18.5
RuneTag	41.9	2.4	71.3

Table 3.1: ChromaTag has a faster average frames per second for all frames, frames with at least one detection, and frames with 0 detections.

Detection Step	Average Time Spent on Each Step
FindADiff	0.52 ms
InitialScan	0.03 ms
BuildPolygon	0.08 ms
PolyToQuad	0.74 ms
Decode	0.04 ms

Table 3.2: For frames with at least one detection, *PolyToQuad* and *FindADiff* dominate computation.

the comparison tags is placed side-by-side on a flat surface. Data is captured as the camera is moved around the scene. Both tags remain in the image frame during the entirety of the captured sequence. A similar trajectory is traversed three times as the pitch of the tag surface is adjusted between 0 degrees (vertical), 30 degrees, and 60 degrees. A motion capture system is used to capture the pose of the camera during the data collection. This collection is repeated twice for two different lighting conditions: white balance (WB) and no white balance (NWB). Figure 3.8 depicts the path that is walked and some sample images. The same ID from the 16H5 family was used for both ChromaTag and AprilTag. Tag locations are hand annotated in each image.

The implementations provided by the authors of AprilTag, CCTag, and RuneTag are used for all experiments. All processing uses a 3.5 GHz Intel i7 Ivy Bridge processor.

3.4.1 Detection Speed

ChromaTag's detection algorithm is faster than other state of the art tags. Table 3.1 provides the computation time for each fiducial marker. Table 3.3 provides the number of frames timed for each marker. The average frames per second (FPS) was calculated by dividing 1 by the

		Chroma	April	Chroma	CCTag
Frames	WB	10266		11238	
	NWB	10303		10891	
Precision	WB	96.9	46.0	96.3	100.0
	NWB	95.7	42.9	95.7	99.9
Recall	WB	64.0	96.4	64.5	45.7
	NWB	67.9	98.2	66.1	46.3

Table 3.3: Each dataset is a pairwise comparison between ChromaTag and another tag with white balance (WB) and no white balance (NWB). Precision and Recall are calculated for each dataset. ChromaTag achieves high precision and better recall than CCTag. AprilTag is successfully detected in almost every frame (high recall); however, many false positives are also detected (low precision).

mean of the computation time. From Table 3.1, we see that ChromaTag achieves an average FPS of 926.4, which is 16x, 92x, and 22x faster than AprilTag, CCTag, and RuneTag respectively.

The frames with detection often require more computation than those without. Thus, we provide the computation times for correct detections (true positives). Table 3.1 shows that ChromaTag has an Average FPS of 709.2 for true positive detections, which is 12x, 109x, and 295x faster than AprilTag, CCTag, and RuneTag respectively.

The frames without detections should require very little time because time spent on tagless images or failed detections is wasteful. We calculate computation time for frames without a detection (false negatives because all data contains the tags). ChromaTag has an average FPS of 2616.1, which is 37x faster than the next fastest tag (RuneTag).

Table 3.2 shows how much time each step of ChromaTag detection uses. Only frames where successful detection occurred are used for this breakdown. *PolyToQuad* (0.74 ms) and *FindADiff* (0.52 ms) are the most costly. *PolyToQuad* is costly because of K-Means clustering and corner localization, and *FindADiff* is costly because of scanning the image and converting pixels to the LAB color space.

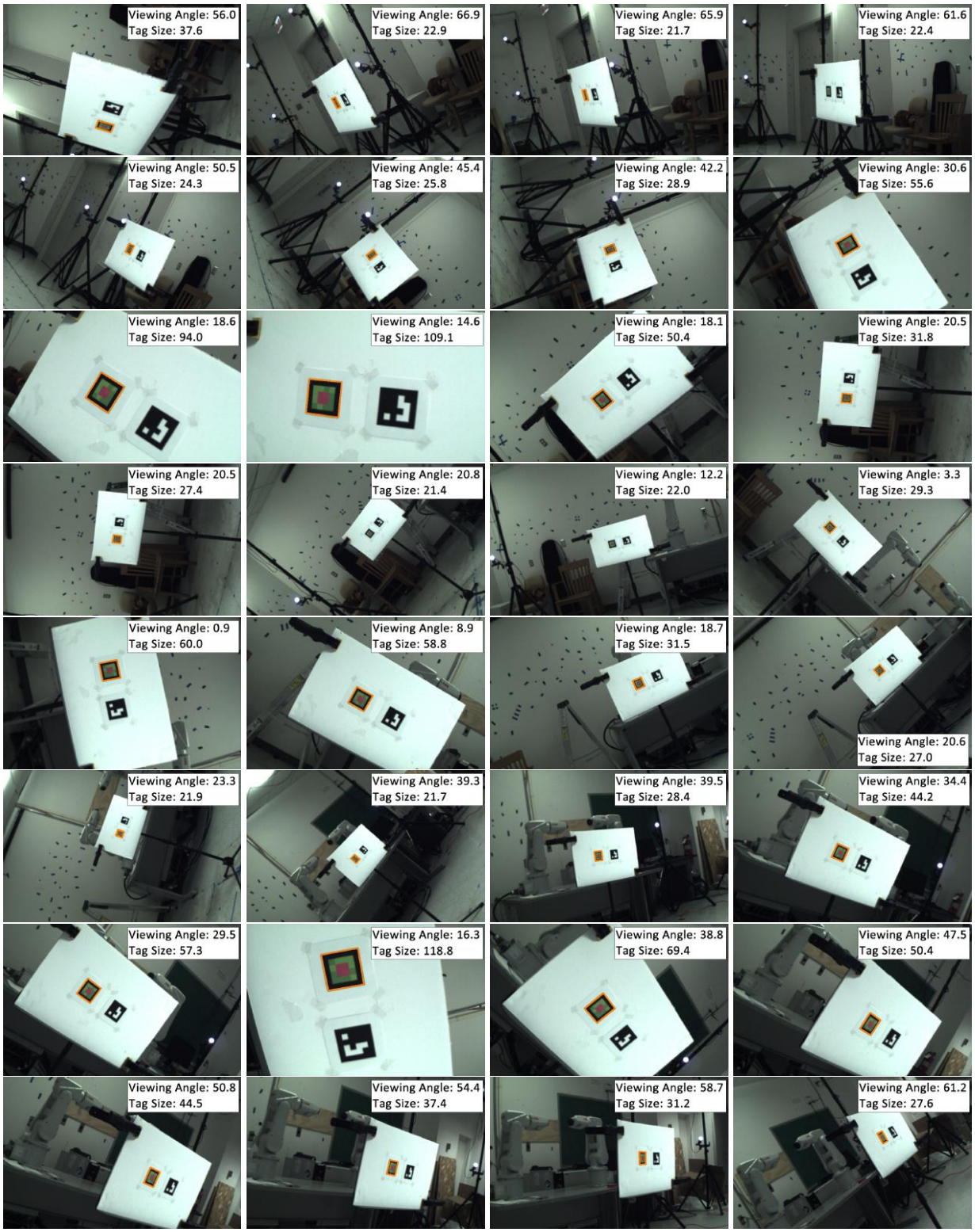


Figure 3.9: Example images processed by ChromaTag with labeled tag size and viewing angle.

Detection Step	Percent of Frames(%)
FindADiff	2.0
InitialScan	2.8
BuildPolygon	25.8
PolyToQuad	1.6
Decode	1.1

Table 3.4: For each step, the % of frames that failed is shown. *BuildPolygon* is most often the step where failure occurs.

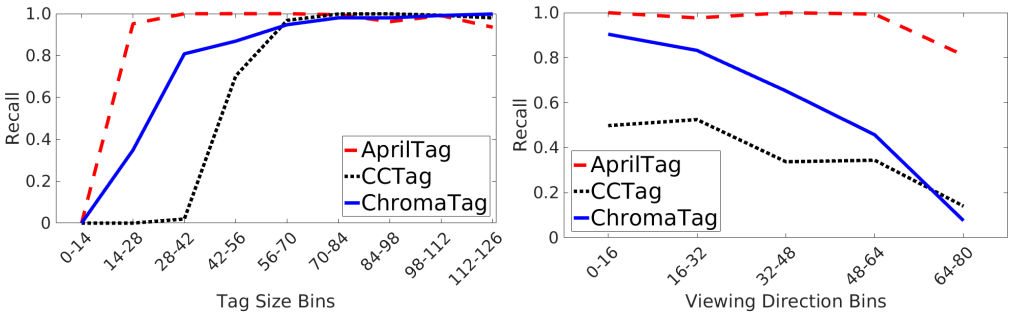
3.4.2 Detection Accuracy

Table 3.3 summarizes the detection results for ChromaTag compared to AprilTag and CCTag. We define true positives (TP) as when the tag was correctly detected in the image. This means locating the tag and correctly identifying the ID. Correctly identifying the tag is determined by having at least 50 percent intersection over union between the detection and the ground truth (though detections by all tags far exceeds this threshold). We define false positives (FP) as detections returned by the detection algorithms that do not identify the location and ID correctly. We define false negatives (FN) as any marker that was not identified correctly. Precision is $\frac{TP}{TP+FP}$ and recall is $\frac{TP}{TP+FN}$.

RuneTag is omitted from Table 3.3 because it was not detected in our dataset. The maximum size of any tag in the images is about 126x126 pixels. With additional data, we found that RuneTag requires larger tag sizes for detection. Figure 3.9 shows example detections and failures of ChromaTag for varying tag size and viewing angle.

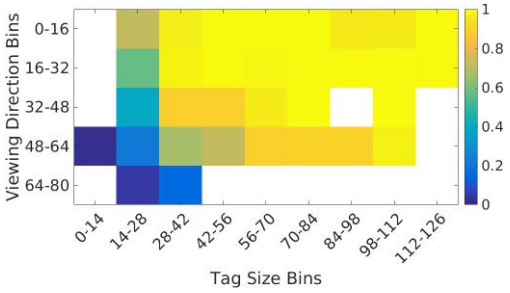
Table 3.4 breaks down how often each step causes failed detection. Most detection failures occur during the *Build Polygon* step (25.8%). Profiling failures is useful to emphasize areas for future improvements.

ChromaTag's initial false positive rejection improves precision. Table 3.3 shows that ChromaTag (96%) has a higher precision than AprilTag (44%). This is interesting because both tags are using the 16H5 family. ChromaTag is successfully rejecting many initial false positives that AprilTag identifies as tags. Note that the 36H11 family causes less false positives [184]; however,



(a) Tag Size vs Recall

(b) Viewing Direction vs Recall



(c) ChromaTag Recall for Tag Size vs Viewing Direction

Figure 3.10: Recall is high when tag size (square root of tag area in image) is large and decreases as tag size decreases (Figure 3.10a). Recall is high when the camera is frontally facing the tag (low values) and decreases as the angle increases (Figure 3.10b). Viewing angle is the angle between the normal vector of the tag plane and the camera viewing direction (tag plane center to camera center). ChromaTag achieves similar recall to AprilTag for tags larger than 70 pixels. Figure 3.10c shows the recall for ChromaTag as it depends on both viewing direction and tag size. Yellow means a high recall and blue means a low recall. We see from this plot that both viewing direction and tag size affect ChromaTag recall, though tag size has a larger direct effect.

less space on the tag is available for the border and inner grid squares (resulting in less recall). Thus, ChromaTag is able to take advantage of the larger grid and border areas of the 16H5 codes without suffering from false positives like AprilTag. ChromaTag and CCTag have similar precisions (greater than 95%).

The combination of high recall and fast detection makes ChromaTag a good choice for many applications. Table 3.3 shows that ChromaTag has lower recall than AprilTag (and higher recall than CCTag). However, figure 3.10a shows that for tag sizes (square root of tag area in the image) greater than 70 pixels, ChromaTag achieves similarly high recall. To put that in perspective, the resolution of the dataset images is 752x480 pixels, so a 70x70 tag area is less than 2% of the

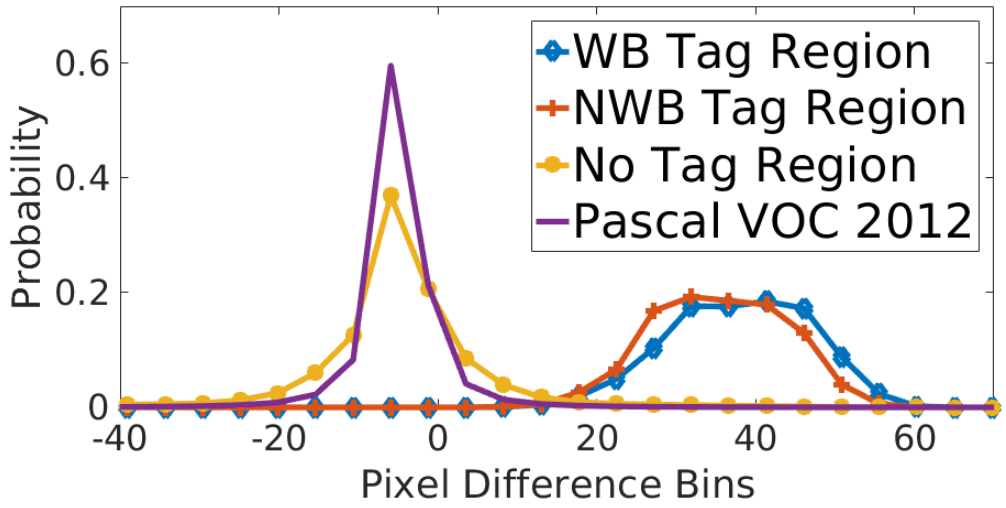


Figure 3.11: WB Tag Region (blue) and NWB Tag Region (orange) are the regions of the WB and NWB data respectively that include the tag. No Tag Region (yellow) is the tagless parts of the WB and NWB data. Each histogram bins A channel pixel differences from horizontally scanning the image (same approach as our detection algorithm). For WB and NWB Tag Region, only the max pixel difference is binned. The result is two modes highlighting how the pixel difference on the tag is easily differentiable from the pixel differences of natural images. The colors of WB and NWB are drastically different, yet still differentiable from natural images, showing that red and green pixel differences in the A channel are robust to color variation.

total available area in these images. Thus, for applications where detecting very small tags is not important, the 16x speed gain makes ChromaTag the better option. Compared to CCTag, ChromaTag achieves similar recall for large tags and higher recall as tag size decreases.

ChromaTag localizes corners precisely. Since use of tags for pose estimation depends on accurately localizing the corners, we evaluate accuracy for corner localization compared to hand-labeled corners that are locally refined with a Harris corner detector [86]. We found that ChromaTag localized 94.4% of the corners within 3 pixels of the ground truth corners and AprilTag localized 89.1% of the corners within 3 pixels (94.2% within 4 pixels). This demonstrates that our white-black border design and detection enables precise corner localization on par with that of AprilTag. On visual inspection, errors of within 5 pixels are attributable to reasonable variation.

ChromaTag is robust to color variation. Table 3.3 shows that ChromaTag achieves similar recall for both the WB and NWB datasets despite the colors being significantly different. Specifically, ChromaTag recall is 64.0% and 67.9% for WB and NWB on the AprilTag dataset and 64.5%

and 67.9% for WB and NWB on the CCTag dataset. The comparisons for ChromaTag are similarly close for precision between WB and NWB for each dataset. These results provide evidence that ChromaTag is robust to color variation.

ChromaTag is robust to color variation because initial detection and finding borders rely on LAB pixel differences, which the tag design ensures are consistently large. Figure 3.11 shows histograms of pixel differences in the A channel for the tag region of the WB data (blue), the tag region of the NWB data (orange), the tagless regions of WB and NWB (yellow), and the Pascal VOC 2012 dataset [58] (purple). Each histogram is created by binning A channel pixel differences from horizontally scanning the image with $N = 4$ subsampling (same approach as our detection algorithm). For the tag regions, only the max difference is counted since only one difference must be above threshold for detection. Despite large variation in color between WB and NWB, the A channel difference for tag regions is clearly differentiable from A channel differences in natural images, which shows why ChromaTag is robust to lighting and color variation and can reject false positives quickly.

3.5 Conclusions

We present a new square fiducial marker and detection algorithm that uses concentric inner red and green rings to eliminate false positives quickly and outer black and white rings for precise localization. We demonstrate on thousands of real images that ChromaTag achieves detection speeds significantly faster than the current state of the art while maintaining similar detection accuracy. We also show that ChromaTag detection fails at far distances and steep viewing angles and recommend AprilTag as a better option for applications that require detection in these conditions. Lastly, we provide evidence that ChromaTag detection is robust to color variation, and break down which steps of the detection algorithm take the most time and fail most often.

Chapter 4

Improved Structure from Motion Using Fiducial Marker Matching

Fiducial markers are often claimed to be useful for 3D reconstruction [13, 184, 42, 71, 65, 8, 22]. Markers provide highly detectable and identifiable features that 3D reconstruction can use to overcome challenging scene characteristics such as low-texture surfaces (e.g., blank walls), reflective surfaces (e.g., windows), and repetitive patterns (e.g., columns and door frames). Figure 4.1 shows an example of a dataset with exactly these challenging characteristics. Figure 4.1 also shows that approaches that treat markers as texture, only use them as additional tracks, or rely on them exclusively perform no better or even worse than if markers were ignored.

In this chapter, we present an incremental structure from motion (SfM) algorithm that significantly outperforms these other approaches when markers are present in the scene. We exploit that markers can be identified with very low false positive rates (e.g. AprilTag2 with 36h11 markers has a false positive rate of 0.000044% [184]) to create a reliable marker match graph that guides image matching and resectioning. We encode constraints on marker size, shape, and planarity in bundle adjustment to further improve results. Importantly, our approach benefits from any detected markers without sacrificing performance when markers are not detected, and can benefit from even a small number of markers.

To evaluate our method, we introduce a new dataset with 16 image collections of indoor scenes. The scenes present challenging circumstances for SfM (e.g. blank hallways, reflective glass facades, and repetitive brick walls). Each indoor scene has tens to hundreds (depending on scene size) of markers placed approximately uniformly throughout. We test our system and several cutting edge benchmarks on this data and show that our system performs favorably. We also selectively mask markers and show that performance gracefully degrades towards markerless SfM as the number of markers in the scene decreases.

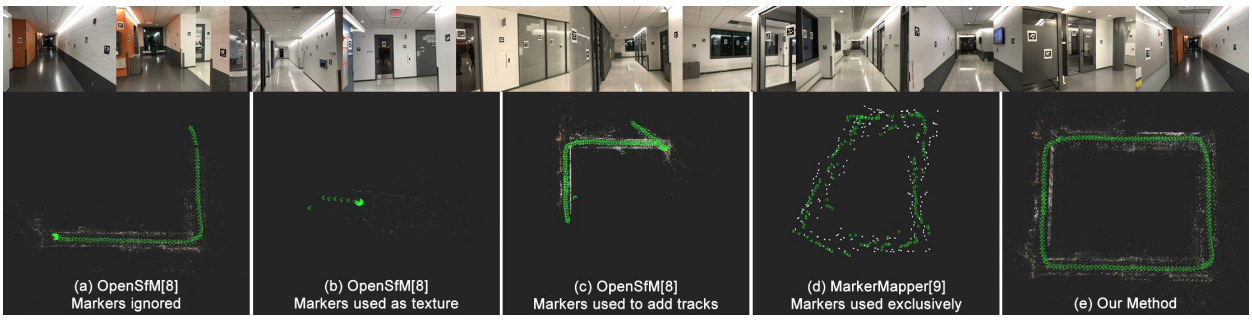


Figure 4.1: We introduce a new dataset of unordered image collections of challenging indoor scenes with markers placed throughout (example images along top row). We process the data using OpenSfM [135] with (a) markers ignored, (b) markers used as texture, and (c) markers used as additional tracks; with (d) MarkerMapper [126], which uses markers exclusively; and with (e) our approach, which uses markers to limit image matches, dictate resectioning order, and constrain bundle adjustment. Clearly, our method (e) outperforms the others. Moreover, the other approaches often perform worse than ignoring the markers, highlighting the importance of our method.

In summary, the **contributions** of this chapter are: (1) an SfM algorithm that uses both fiducial markers (when available) and interest point features for improved results; (2) a large, challenging dataset of indoor scenes with markers placed throughout; and (3) experiments showing the effectiveness of our approach, even when only a small number of markers are visible.

4.1 Related Work

Incremental SfM: Early works by Schaffalitzky and Zisserman [152] and Snavely et al. [166] establish the pipeline for feature extraction, matching, and incremental SfM for unordered image collections. Focus then turns to large image collections with work by Agarwal et al. [2] and Frahm et al. [117] who use appearance based clustering to limit potential image matches; enabling reconstructions of Rome from thousands of internet photos. Work by Wu [185] shows that preemptive feature matching and well timed global bundle adjustments can maintain high accuracy while reducing the runtime of SfM to roughly $\mathcal{O}(n)$. Recently, several new SfM algorithms are available including COLMAP [155] by Schönberger and Frahm and OpenSfM [135] by Mapillary. These impressive works provide the baseline for the work in this chapter.

Figure 4.2: Example images from the Neunert et al. [128] dataset: desk (top left), dataset1 (top right), cube (bottom left), and pavilion (bottom right). Experiments in Section 3.4 show that our method and current SfM methods perform well on this data, motivating our new dataset that offers new challenges and better distinguishes between approaches.

3D Reconstruction using Fiducial Markers: Early works using markers for 3D reconstruction focus on tracking the markers in simultaneous localization and mapping (SLAM) systems. Work by Klopschitz and Schmalstieg [100] tracks both feature points and marker matches in video frames to estimate the camera pose and triangulate the marker positions in 3D. Lim and Lee [108] add the estimation of ground robot camera pose and marker positions in 3D using an extended kalman filter (EKF) based SLAM. Similarly, Yamada et al. [189] also introduce an EKF-SLAM system for blimp navigation. Neunert et al. [128] integrate IMU measurements into the EKF-SLAM system to improve pose estimates during marker tracking. Feng et al. [61] proposes an incremental SfM approach to marker based 3D reconstruction. They use markers to create an initial reconstruction and add new images using marker matches. They also add geometry constraints to the bundle adjustment to enforce the square shape and planarity of markers. The work of Muñoz-Salinas et al. [126] introduces MarkerMapper. MarkerMapper overcomes the pose ambiguity problem [158] with planar marker pose estimation to create an initial proposal of 3D camera and marker locations. Then, MarkerMapper uses global bundle adjustment to refine the initial proposal. We compare to MarkerMapper in Section 3.4.

Only MarkerMapper [126] and Feng et al. [61] pursue 3D reconstruction from unordered image collections. However, neither method uses both image features and marker detections for 3D reconstruction. Experiments in Section 3.4 show that both image features and marker detections can be used together to achieve the best results, and, when few or no markers are available, our system performs no worse than non-marker based SfM.

Datasets: Datasets for testing marker based 3D reconstruction are limited. Only the dataset of Neunert et al. [128] is publicly available (all other aforementioned works use self contained datasets). This dataset consists of four video sequences of a table, room, office space, and two-story building with markers placed throughout each scene. Figure 4.2 provides snapshots from the four video sequences of this dataset. With only four sequences (two of which are of very small environments with only 1-3 markers), this dataset is no longer challenging for the current state of the art (e.g. in Section 3.4, we process this data with our method and other current SfM approaches, and all perform well). Our new dataset (Section 4.2) consists of 16 new image collections in environments with challenging characteristics for SfM (e.g. many low-texture walls and reflective glass). We hope our dataset will offer new challenges for future work on SfM both with and without marker assistance.

4.2 Indoor Image Collections with Fiducial Markers

We introduce 16 new unordered image sets for evaluating structure from motion for scenes containing fiducial markers. Each set is from one of three buildings: ECE, CEE, or MUF. Figures 4.3 and 4.4 provide floor plans for the sections of these buildings that are used to collect this data. Paths are drawn on each floor plan and the colors of the paths match the respective image sets in the figures (e.g. the green path on Floor 4 and 5 of ECE matches the ECE Floor5 Hall image set). For each set, fiducial markers are placed around the scene with enough density to see at least one in every image (and images are captured to satisfy this also). All images are captured with an iPhone7 camera and have a resolution of 4032x3024 pixels.

Note that some of the image collections are combinations of others. Specifically, *ECE Floor5* includes all the images of *ECE Floor5 Hall* and *ECE Floor5 Stairs*. *ECE Floor3 Loop* includes all the images of *ECE Floor3 Loop CW* and *ECE Floor 3 CCW*. *CEE Day* includes all the images of *CEE Day CW* and *CEE Day CCW* (plus some extra images). The nice thing about collecting data in this way is that we can test progressively larger datasets that present different circumstances that

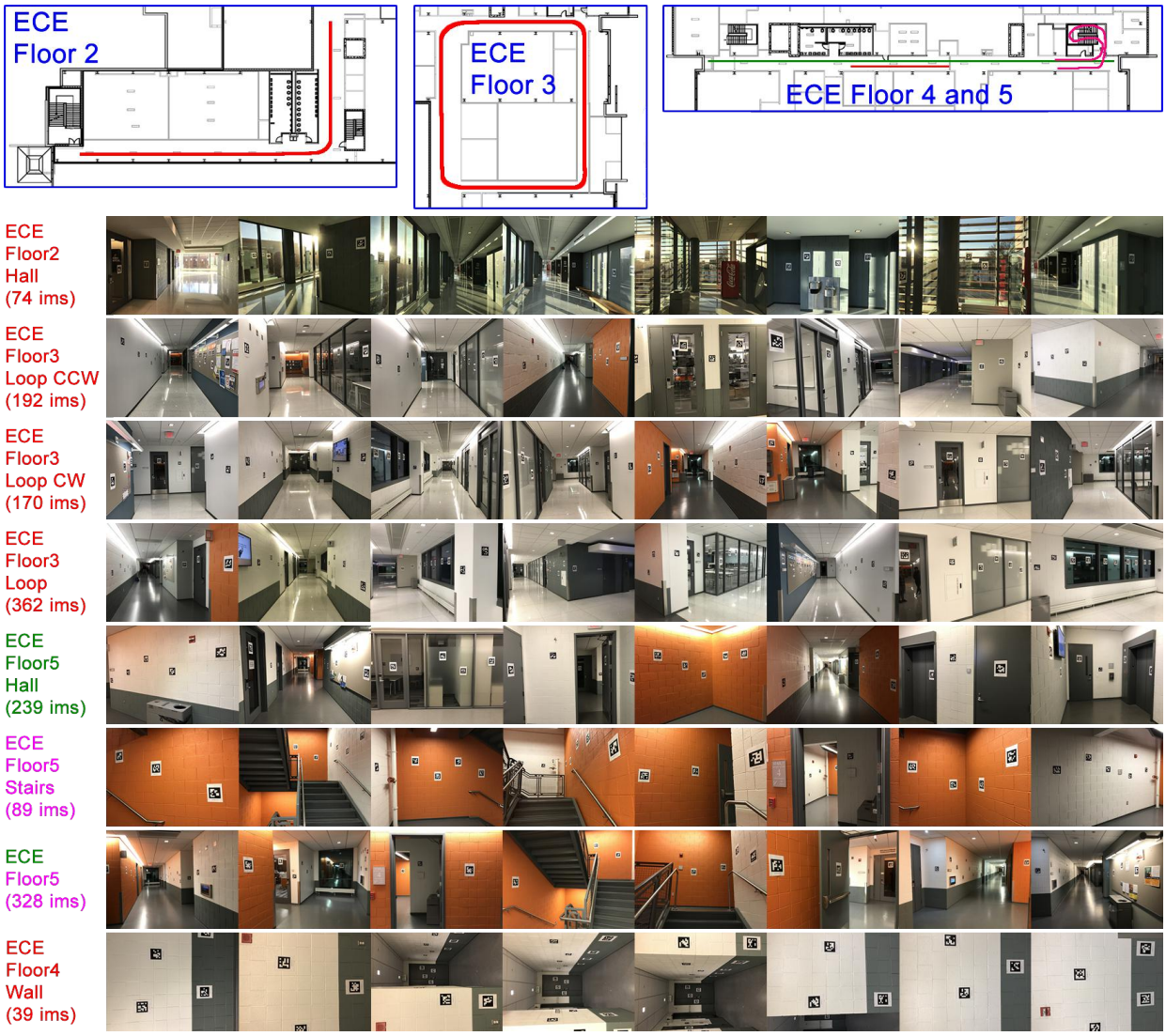


Figure 4.3: The top diagrams are floor plans of ECE. The paths for image collection are superimposed in red, green, and magenta. These colors correspond to the image set name and example images. For example, *ECE Floor5 Stairs* is shown in the *ECE Floor4 and 5* floor plan as a magenta line and the name with example images is also magenta.

may make the image set easier or more difficult. For example, the results in Section 3.4 show that *ECE Floor3 Loop CW* and *ECE Floor3 Loop CCW* are typically more difficult than putting them together into *ECE Floor 3 Loop*. This is most likely because of the additional overlap between images since all locations are now seen more often from more viewing directions.

We use ECE, CEE, and MUF because they are large indoor scenes with characteristics that are challenging for SfM (as shown in Section 3.4). Specifically, ECE has long plain hallways, large

Figure 4.4: The top diagrams are floor plans for CEE and MUF. The paths for image collection are superimposed in red. Image set names and example images are shown.

glass walls separating conference rooms, large exterior windows, and the hallways form a loop. CEE has a two-floor glass facade and repetitive brick walls. MUF is currently under construction and has large open spaces and limited texture.

4.3 Improving SfM with Markers

Figure 4.5 diagrams our marker assisted incremental SfM algorithm. The blue boxes represent the components of our algorithm that are different from typical state of the art incremental SfM approaches: detecting markers, filtering image pairs, resectioning images, and marker constraints for bundle adjustment.

4.3.1 Incremental SfM Overview

Incremental SfM takes a collection of images as input. For each image, focal length (and other priors) is estimated from metadata (or using heuristics when metadata is unavailable). Next, image features (e.g. SIFT features [112]) are extracted from each image. These image features are matched across image pairs. Matching is attempted between the set of all images pairs or a subset based on filtering criteria (e.g. GPS locations [117], Vocab Tree [2]). A fundamental matrix is estimated from the feature matches to filter bad matches and verify that each image pair is a good match.

After matching, reconstruction begins. Matches in two images are used to create an initial 3D reconstruction (pose of the two images with triangulated 3D points). Then, one at a time, a new image is added to the reconstruction (usually referred to as resectioning). This image is typically chosen based on the number of feature matches this image shares with the already reconstructed images. These shared feature matches are used to estimate the pose of this new camera and triangulate new 3D points. Bundle adjustment is run to optimize all camera poses and 3D point positions to minimize reprojection error. Lastly, outlier points are removed. Resectioning is repeated to add all images to the reconstruction. The final output is a point cloud and set of camera poses (one camera pose for each image that is successfully resectioned).

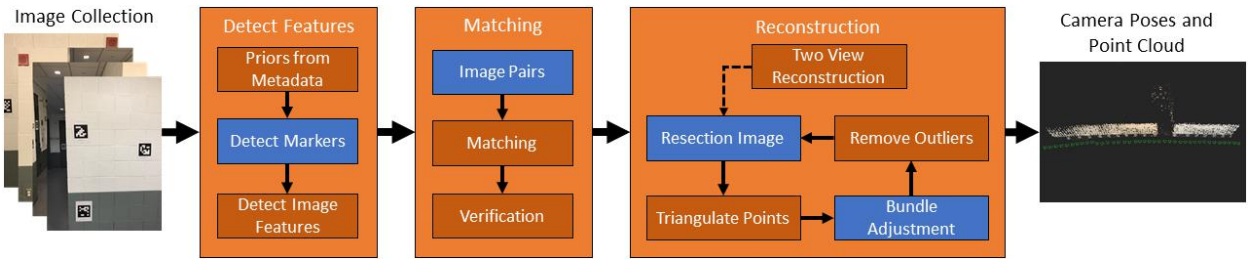


Figure 4.5: This diagram depicts the typical incremental SfM approach: extracting priors from metadata (e.g. focal length), detecting features, matching features, and reconstruction. The blue boxes are the areas we added or changed in our method.

4.3.2 Detect Markers

We run a square marker detection algorithm on each input image. The images are processed in parallel. Image name, marker id, corner locations, and corner pixel colors are saved for each detection. The four corner locations, if triangulated during 3D reconstruction, can then be used for additional constraints in bundle adjustment as described in section 4.3.5

4.3.3 Marker Informed Image Pairs

Prior to matching and verification, we create a set of candidate image pairs. We only attempt matching on the image pairs in this set. One approach is to add all possible image pairs; however, this greatly increases matching time and can lead to bad image matches that cause errors in the reconstruction. We apply three rules to use marker detections to dictate which images are added. **Rule 1:** we add an image pair if the same marker (at least one) is detected in both images. **Rule 2:** if an image does not share a detected marker with any other image, we add all possible pairs that contain that image. **Rule 3:** if the set of all added pairs do not form one connected component, we connect separate components by adding pairs for each image in the separate component to each image not in the separate component.

As an example, consider the top left diagram in Figure 4.6. Each lettered box represents an image, and each numbered edge represents the number of marker matches those images share. Applying rule 1, we add the following possible image pairs (A,B), (A,C), (B,C), (B,D), (C,E),

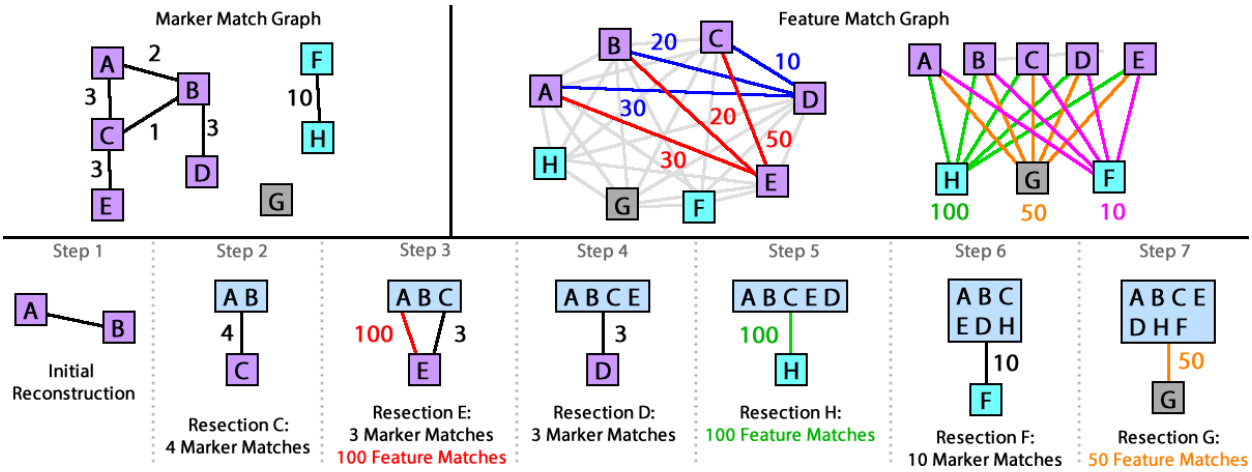


Figure 4.6: The top left diagram depicts images as lettered boxes with edges representing the number of matched markers between image pairs. The top middle and top right diagrams depict the number of common feature matches between images. The bottom diagram depicts the resectioning order of images A to G based on two rules: (1) add the image that shares the most marker matches with the reconstruction; (2) break ties using most shared feature matches.

and (F,H). No pair is added that includes G, so based on rule 2, we add (G,A), (G,B), ..., (G,H). Lastly, since (F,H) is a separate component (rule 3), we add (F,A), (F,B), ..., (F,E) and (H,A), (H,B), ..., (H,E). We show in the results that this strategy can greatly speed up processing and eliminate many bad image matches. Note that other filtering approaches (e.g. Vocab Tree [2]) can be used in conjunction with our approach to add or filter image pairs.

4.3.4 Marker Informed Resectioning

Resectioning is the process of adding a new image to the existing reconstruction. The order in which images are added is important because poorly registered images can propagate errors that result in failure. One approach is to choose the image to resection that shares the most feature matches with the images in the reconstruction. This approach works well when image features are distinct and plentiful; however, for the challenging scenes we are targeting, failure can occur. Instead, we apply two rules to use marker detections to dictate resectioning order. **Rule 1:** the next image to resection shares the most marker matches with the current reconstruction. **Rule 2:** if multiple images share the same number of marker matches with the current reconstruction, choose

the image that shares the most feature matches.

For example, consider the diagrams in Figure 4.6. In the top left diagram, each edge represents the number of marker matches those images share. In the top middle and top right diagram, each numbered edge represents the number of image feature matches those images share. The bottom diagram depicts the resectioning procedure. First, images A and B are used for the initial reconstruction (step 1). The next image that is resectioned is C because it shares 4 (3 with A and 1 with B) marker matches with the current reconstruction (step 2). After that, image E is added because E and D both share 3 marker matches with the reconstruction, but E shares 100 feature matches and D only shares 60 (step 3). Image D is then added (step 4). No remaining images share marker matches with the current reconstruction, so image H is added based on shared image feature matches (step 5). F is added next (step 6) because it now shares marker matches with the reconstruction (because H was added). Lastly, G is added (step 7).

4.3.5 Marker Constraints for Bundle Adjustment

In bundle adjustment, we solve for camera poses $\vec{P}$ and 3D points $\vec{X}$ that optimize the following:

$$\min_{\vec{P},\vec{X}} \left[w_R E_R\left(\vec{P},\vec{X}\right) + w_S E_S\left(\vec{V}\right) + w_O E_O\left(\vec{V}\right) \right] . \tag{4.1}$$

$\vec{V}$ is the set of vectors formed between neighboring 3D corners on each marker (i.e. there are four vectors for each marker detection because there are four corners detected on each square marker). w_R, w_S, and w_O are weights. Reprojection error [87] is

$$E_R\left(\vec{P},\vec{X}\right) = \sum_{i=1}^{C}\sum_{j=1}^{N} L\left(x_{ij}, \vec{P}_i\left(\vec{X}^j\right)\right) \tag{4.2}$$

where C is the number of cameras, N is the number of 3D points (both marker and feature points), L is a loss function, x_{ij} is the 2D location in image i of 3D point $\vec{X}^j$, and $\vec{P}_i$ is the projection function of camera i. Similar to [61], we also include error terms for marker scale (E_S, Equation 4.3)

and marker orthogonality (E_O, Equation 4.4).

Marker Scale: the distance between marker corners in the reconstruction should match the known marker size. We define this error as $E_S\left(\vec{V}\right) =$

$$\sum_{i=1}^{T}\left(\left\|\vec{V}_{12}^{i}\right\|_2 - S\right)^2 + \left(\left\|\vec{V}_{23}^{i}\right\|_2 - S\right)^2 + \left(\left\|\vec{V}_{34}^{i}\right\|_2 - S\right)^2 + \left(\left\|\vec{V}_{41}^{i}\right\|_2 - S\right)^2 \tag{4.3}$$

where $\vec{V}_{NM}^{i}$ is the 3D vector from the 3D point of corner N to the 3D point of corner M on marker i, T is the number of markers, and S is the marker size.

Marker Orthogonality: adjacent sides of the marker should be perpendicular. We define this error as $E_O\left(\vec{V}\right) =$

$$\sum_{i=1}^{T}\left(\vec{V}_{12}^{i}\cdot\vec{V}_{23}^{i}\right)^2 + \left(\vec{V}_{23}^{i}\cdot\vec{V}_{34}^{i}\right)^2 + \left(\vec{V}_{34}^{i}\cdot\vec{V}_{41}^{i}\right)^2 + \left(\vec{V}_{41}^{i}\cdot\vec{V}_{12}^{i}\right)^2 \tag{4.4}$$

where $\vec{V}_{NM}^{i}$ is the 3D vector from the 3D point of corner N to the 3D point of corner M on marker i, and T is the number of markers.

4.3.6 Implementation Details

We implement our approach on top of OpenSfM v0.1.0 [135]. We use default parameters, which work well for unordered image collections. We use AprilTag2 [184] to detect markers. For all experiments, we use a soft L1 loss for L; cost weights of $w_R = 62500$, $w_S = 100$, and $w_O = 100$; and marker size $S = 0.21$ m. In principal, our approach works with any square marker detector and can be integrated with any incremental or global [124, 122] (except resectioning) SfM method.

	# Images	# Registered			% Registered		
		No MIM	No MIR	Full	No MIM	No MIR	Full
ECE F3 Loop CCW	192	139	191	191	72.4%	99.5%	99.5%
ECE F3 Loop CW	170	135	170	170	79.4%	100.0%	100.0%
ECE F3 Loop	362	-	-	360	-	-	99.4%
ECE F5 Stairs	89	46	89	89	51.7%	100.0%	100.0%
ECE F4 Wall	39	21	22	39	53.8%	56.4%	100.0%
CEE Day CW	63	33	42	62	47.8%	60.9%	89.9%
CEE Day CCW	120	60	120	119	50.0%	100.0%	99.2%
CEE Night CCW	79	-	-	77	-	-	97.5%
CEE Night	170	158	157	170	92.9%	92.4%	100.0%

Table 4.1: Ablation Study. We provide the number of images registered and the percent registered for our method without marker informed matching (denoted as *No MIM*), our method without marker informed resectioning (denoted as *No MIR*), and our full method (denoted as *Full*). The next closest method is OpenSfM with markers masked, which has an average percent registered of 42.3%. Thus, marker informed matching and marker informed resectioning both help, but are better when used together.

4.4 Results

We process our new dataset using: (1) OpenSfM [135], an open source state of the art SfM algorithm that is actively used and maintained by Mapillary [114]; (2) OpenSfM, but with all feature points on markers masked; (3) MarkerMapper [126], a state of the art algorithm for marker based SfM; (4) OpenSfM with the four marker corners used as tracks in reconstruction; and (5) our method. Table 4.2 provides quantitative results on the number of images localized, number of points, and reprojection errors. Failure reconstructions are denoted by a "-". Figures 4.7 and 4.8 provide qualitative results of the 3D reconstructions. The green pyramids are the camera locations. The floor plans in Figures 4.3 and 4.4 provide guidelines for how each reconstruction should look (e.g. ECE Floor3 Loop should be a rectangle). Because of the challenging nature of these datasets, the algorithms often fail or have large, noticeable mistakes; therefore, we focus more on the qualitative results because they illustrate the improvements clearly.

We also process the Neunert et al. [128] dataset. Since it is video data, we subsample the frames by a factor of 5 to simulate an unordered image collection. All OpenSfM methods and our method successfully reconstruct all image sets. MarkerMapper has trouble with this dataset because there are few (often only one) markers in each image. Qualitative results are shown in Figure 4.9.

	# Ims	# Registered					# Points					Avg. Rep. Error [px]				
		[135]	[135]*	[126]	MT	Ours	[135]	[135]*	[126]	MT	Ours	[135]	[135]*	[126]	MT	Ours
ECE F2 Hall	74	-	70	-	-	71	-	15.9K	-	-	16.4K	-	3.1	-	-	2.8
ECE F3 Loop CCW	192	-	-	190	-	191	-	-	808	-	61K	-	-	200.8	-	2.8
ECE F3 Loop CW	170	-	-	166	-	170	-	-	736	-	58K	-	-	358.1	-	2.7
ECE F3 Loop	362	-	-	356	-	360	-	-	920	-	105K	-	-	324.0	-	2.8
ECE F5 Hall	239	230	230	213	223	231	50K	45K	736	47K	63K	2.8	2.7	141.0	2.7	2.7
ECE F5 Stairs	89	52	51	-	45	89	20K	20K	-	14K	43K	1.9	1.7	-	1.9	1.8
ECE F5	328	313	315	-	-	327	79K	73K	-	-	109K	2.3	2.3	-	-	2.3
ECE F4 Wall	39	21	18	39	18	39	13K	9K	204	9K	28K	1.1	1.1	25.8	1.2	1.2
CEE Day CW	63	55	52	-	52	62	24K	20K	-	28K	30K	1.6	1.6	-	1.6	1.6
CEE Day CCW	120	65	116	-	116	119	30K	52K	-	56K	64K	1.6	1.5	-	1.6	1.5
CEE Day	252	-	251	238	103	246	-	89K	768	398	104K	-	1.7	204.8	0.2	1.8
CEE Night CW	96	96	96	96	-	96	48K	44K	548	-	51K	1.7	1.6	164.0	-	1.7
CEE Night CCW	79	-	-	79	-	77	-	-	580	-	40K	-	-	116.6	-	1.5
CEE Night	170	-	166	170	-	170	-	61K	760	-	77K	-	1.6	181.4	-	1.6
MUF F2	896	883	514	-	885	882	224K	133K	-	151K	251K	2.5	2.5	-	2.1	2.9
MUF F3	361	343	-	-	324	358	84K	-	-	55K	89K	2.8	-	-	2.4	2.8
cube [128]	327	327	327	-	327	327	99K	101K	-	100K	99K	0.8	0.8	-	0.8	0.8
dataset1 [128]	91	91	91	3	91	91	31K	30K	8	31K	33K	0.9	0.9	0.6	0.9	0.8
pavilion [128]	585	585	585	-	585	583	178K	168K	-	186K	178K	0.8	0.7	-	0.7	0.7
table [128]	80	80	49	38	80	80	7K	5K	12	7K	6K	0.9	1.0	0.3	0.9	1.0

Table 4.2: Reconstruction results for OpenSfM [135], OpenSfM with markers masked (denoted by [135]*), MarkerMapper [126], OpenSfM with marker tracks (denoted by MT), and our method. Failure reconstructions (Figures 4.7 and 4.8) are blank because the numbers can be misleading (e.g. all cameras localized to one spot). Our method achieves similar or better results for number of registered images and points for all reconstructions.

Reconstruction and timing results are reported in Tables 4.2 and 4.3 respectively.

We do an ablation study (Table 4.1) with marker informed matching (Section 4.3.3) and marker informed resectioning (Section 4.3.4). For each dataset and method we calculate the percent of images localized. The average percentages of localized images are 98% (our full method), 68% (no marker informed resectioning), 50% (no marker informed matching), and 42% (OpenSfM with markers masked — the next best method). These percentages show that both marker informed matching and resectioning are useful individually, but most effective when used together. We also test our method without marker scale (E_S, Eqn. 4.3) and orthogonality (E_O, Eqn. 4.4) constraints and find that the only noticeable gain from E_O and E_S is that the point cloud is correctly scaled (verified by measuring the length of neighboring reconstructed 3D marker points).

All experiments use an Intel Xeon E5-2620 V4 2.1GHz 16 cores (32 virtual cores) processor with 128 GB of RAM. No graphics card is used.

Using markers as texture often makes reconstructions worse. Masking the markers shows

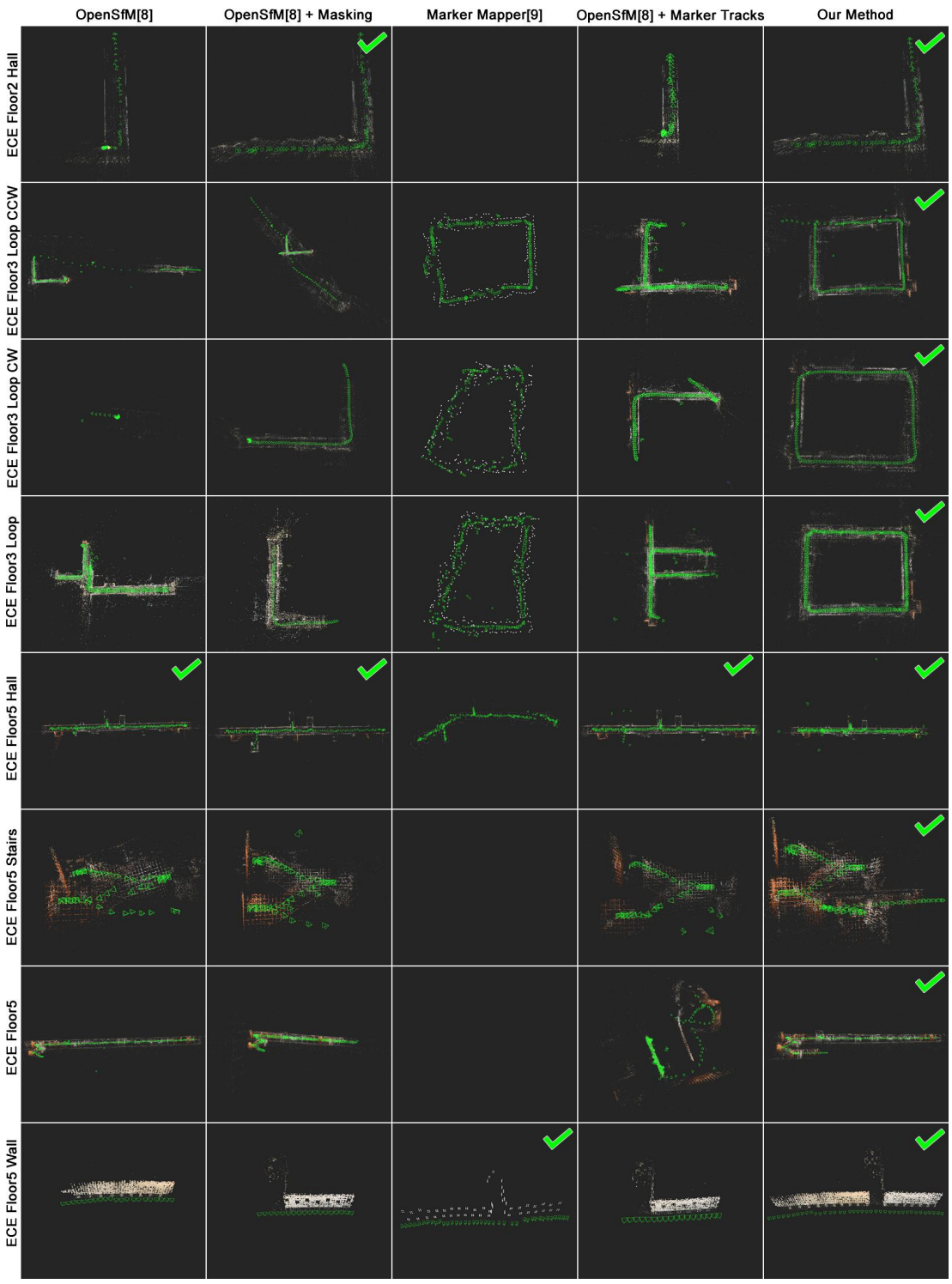


Figure 4.7: Reconstructions for OpenSfM, OpenSfM with markers masked, MarkerMapper, OpenSfM with marker tracks, and our method on the ECE image collections. Using the markers as texture often produces worse results (e.g. *ECE Floor2 Hall*, *ECE Floor3 Loop CW*, *ECE Floor3 Loop*, and *ECE Floor5 Stairs*). Our method produces complete reconstructions that are as good or better than the other methods. The best results are denoted by a green check mark.

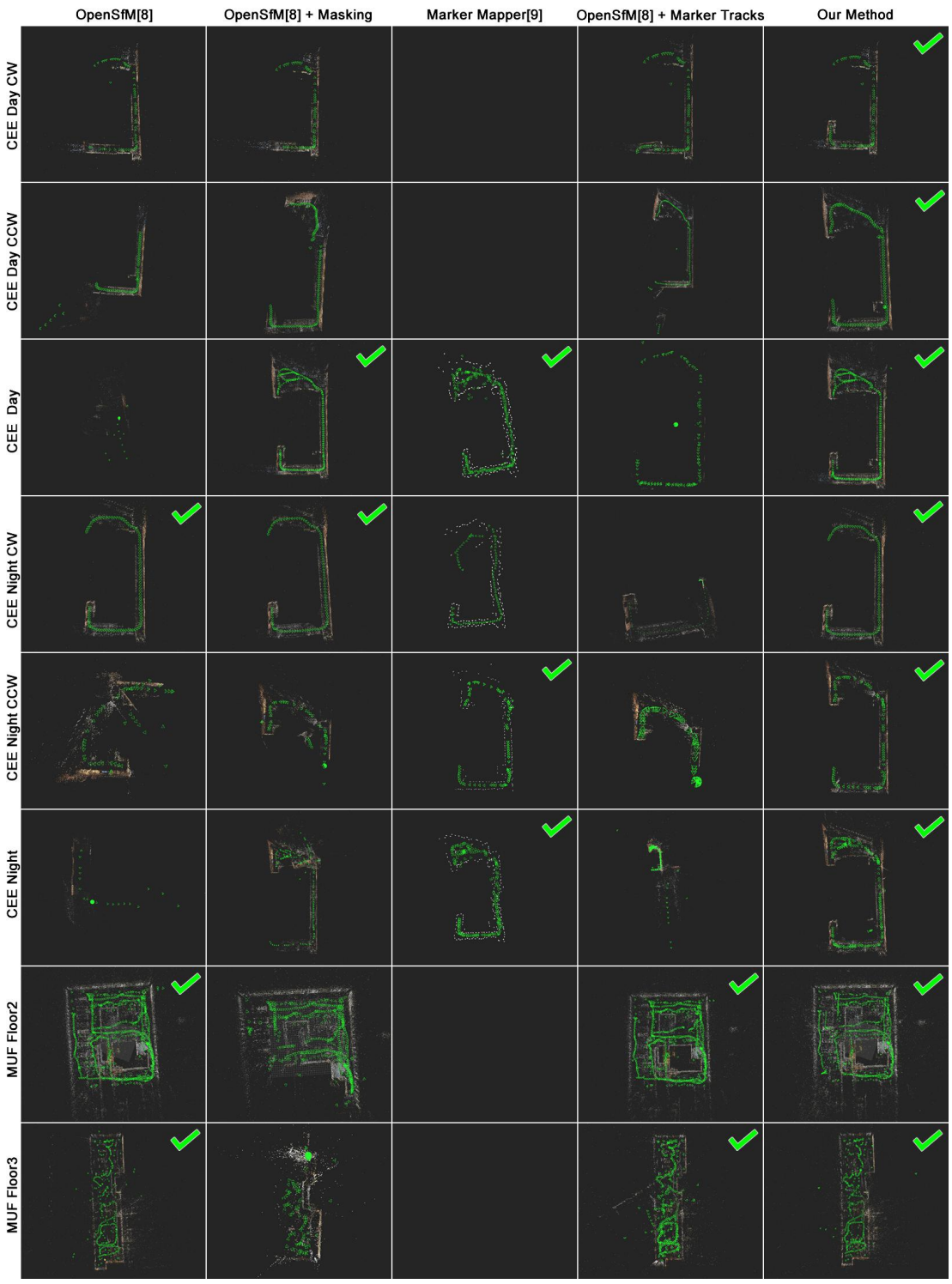


Figure 4.8: Reconstructions for OpenSfM, OpenSfM with markers masked, MarkerMapper, OpenSfM with marker tracks, and our method on the CEE and MUF image collections. Again, using the markers as texture often produces worse results (e.g. *CEE Day CCW*, *CEE Day*, and *CEE Night*). Our method produces complete reconstructions that are as good or better than the other methods for all image collections. The best results are denoted by a green check mark.

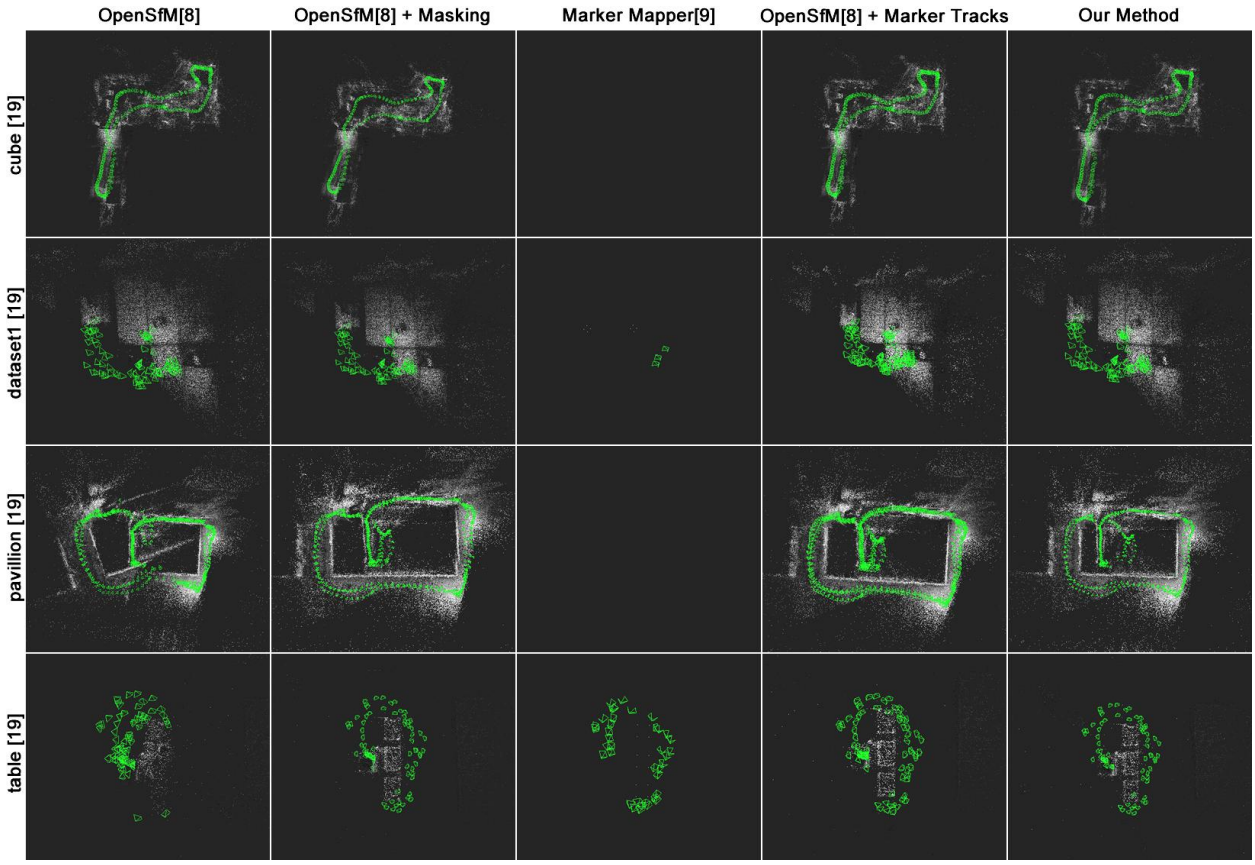


Figure 4.9: Qualitative results for Neunert et al. [19] dataset are shown. All algorithms tend to do well on these image collections. These datasets were originally video sequences, so for these results the frames are subsampled by 5 to simulate the data as unordered image collections.

how OpenSfM performs if the scenes have no markers. Comparing column 1 (OpenSfM) and column 2 (OpenSfM with masked markers) in Figures 4.7 and 4.8, shows that masking the markers often produces better results. For example, *ECE Floor2 Hall* should have an "L" shape, which OpenSfM with masked markers achieves, but OpenSfM does not. Other examples where masking markers is clearly better are *ECE Floor3 Loop CW*, *ECE Floor5 Stairs*, *CEE Day CCW*, *CEE Day*, and *CEE Night*.

Marker texture does not always produce bad results (e.g. *MUF Floor3*), but marker texture can cause bad feature matches because the appearance is similar between the markers (i.e. black and white squares). This reinforces the need for our approach which takes advantage of visible markers to improve results.

	# Images	Marker Detection [s]				Matching [s]				Reconstruction [s]				
		[135]	[135]*	MT	Ours	[135]	[135]*	MT	Ours	[135]	[135]*	[126]	MT	Ours
ECE F2	74	0	14	14	14	215	186	223	73	331	222	-	363	277
ECE F3 Loop CCW	192	0	32	32	32	1356	1160	1398	293	3433	2766	85	2282	3097
ECE F3 Loop CW	170	0	30	30	30	1071	888	1152	273	2797	2084	83	2430	2367
ECE F3 Loop	362	0	59	59	59	4568	3820	4675	876	9944	5082	195	9239	9704
ECE F5 Hall	239	0	40	40	40	1955	1650	1974	296	2810	2363	80	2774	3061
ECE F5 Stairs	89	0	16	16	16	307	258	317	55	425	278	-	347	658
ECE F5	328	0	57	57	57	3787	3195	3945	372	6341	5083	-	7268	5513
ECE F4 Wall	39	0	9	9	9	61	46	47	8	133	46	22	63	263
CEE Day CW	63	0	11	11	11	160	126	171	49	336	216	-	489	382
CEE Day CCW	120	0	21	21	21	535	437	570	139	1011	1377	-	2305	1809
CEE Day	252	0	41	41	41	2373	1919	2567	440	7137	5252	148	4102	4987
CEE Night CW	96	0	16	16	16	358	278	380	99	1083	793	25	1136	1010
CEE Night CCW	79	0	14	14	14	247	193	70	69	425	418	32	917	654
CEE Night	170	0	30	30	30	1093	873	1154	216	3232	2251	93	3287	2984
MUF F2	896	0	158	158	158	31180	25613	35844	5596	72055	40958	-	66095	60542
MUF F3	361	0	64	64	64	5094	4302	5205	758	8977	6903	-	5017	9090
cube [128]	327	0	6	6	6	3473	2724	2776	423	4066	5232	-	4244	4134
dataset1 [128]	91	0	1	1	1	351	305	304	207	847	593	1	619	595
pavilion [128]	585	0	7	7	7	9103	9239	9219	2100	22908	15931	-	21903	22594
table [128]	80	0	1	1	1	64	60	63	65	113	188	2	205	191

Table 4.3: Reconstruction timings for OpenSfM [135], OpenSfM with markers masked (denoted by [135]*), MarkerMapper [126], OpenSfM with marker tracks (denoted by MT), and our method. Using the markers to limit possible image pairs decreases the matching time significantly. Also, because more images are resectioned, the reconstruction time increases. Overall, our method produces better reconstructions in a shorter time.

Using marker detections as tracks has little effect. Comparing column 4 (OpenSfM with marker tracks) to columns 1 and 2 (OpenSfM with and without markers masked) of Figures 4.3 and 4.4 shows that the marker tracks rarely improve the reconstructions and sometimes make them worse (e.g. *ECE Floor5* and *CEE Night*). We suspect this is because the localization of the marker corners can be less accurate (e.g. off by 3-5 pixels [42]) than image features.

Our approach succeeds where others fail. From Figures 4.7 and 4.8, we see that our method produces a successful reconstruction for every image set. We also see that our method produces better results than the other methods on the challenging sets. Most notable are *ECE Floor3 Loop CW*, *ECE Floor3 Loop*, and *CEE Day CCW* because all other methods fail or have significant mistakes. For *ECE Floor5 stairs*, *ECE Floor5*, and *CEE Day CW*, other methods produce reasonable results, but our approach is more complete.

Our approach succeeds where others succeed. There are several image sets where all (or most) of the methods produce successful reconstructions (e.g. *ECE Floor5 Hall* and *CEE Night CW*). In these cases, our method also produces nice reconstructions. This is important because our algorithm improves on the challenging image sets without sacrificing accuracy on the easier image sets.

Using markers improves reconstruction time. Table 4.3 provides the run times for marker detection, matching, and reconstruction for all image sets. Timings for other parts of SfM are not included since they do not change between methods. Also, only the total run time of MarkerMapper is reported because it does not follow the same pipeline as the others. One main thing to note is that using markers to limit pairs for image matching can decrease run times significantly (e.g. for *MUF Floor2*, our method took 5596 seconds and the other OpenSfM approaches took 5-6 times longer). Time is added to detect markers in each image, but it is typically negligible compared to the time saved in matching. Another interesting point is that reconstruction time often increases. This is because more images are able to be registered with our method.

Few marker detections still improves reconstructions. Figure 4.10 demonstrates how marker density effects the reconstructions. In particular, the left six images show how the reconstruction of *ECE Floor3 Loop CCW* improves as the marker density increases. Here AMD stands for average marker detections per image (e.g. AMD = 0.0 means there are no markers detected, and AMD = 6.0 means that there were an average of 6 markers detected per image).

The plot in Figure 4.10 shows how the percent of localized images increases as the AMD increases for seven datasets. These datasets were chosen because our method achieves clear improvements over the other methods. The trend line is plotted in black. This plot shows that markers help even when AMD is less than 1 (sometimes even 100% of the images are localized). As AMD increases, the number of localized images increases towards 100%. Placing enough markers for an AMD of 6 will likely produce accurate, complete reconstructions with 90+% of images localized. However, markers are most useful in areas with challenging conditions for SfM, so placing more

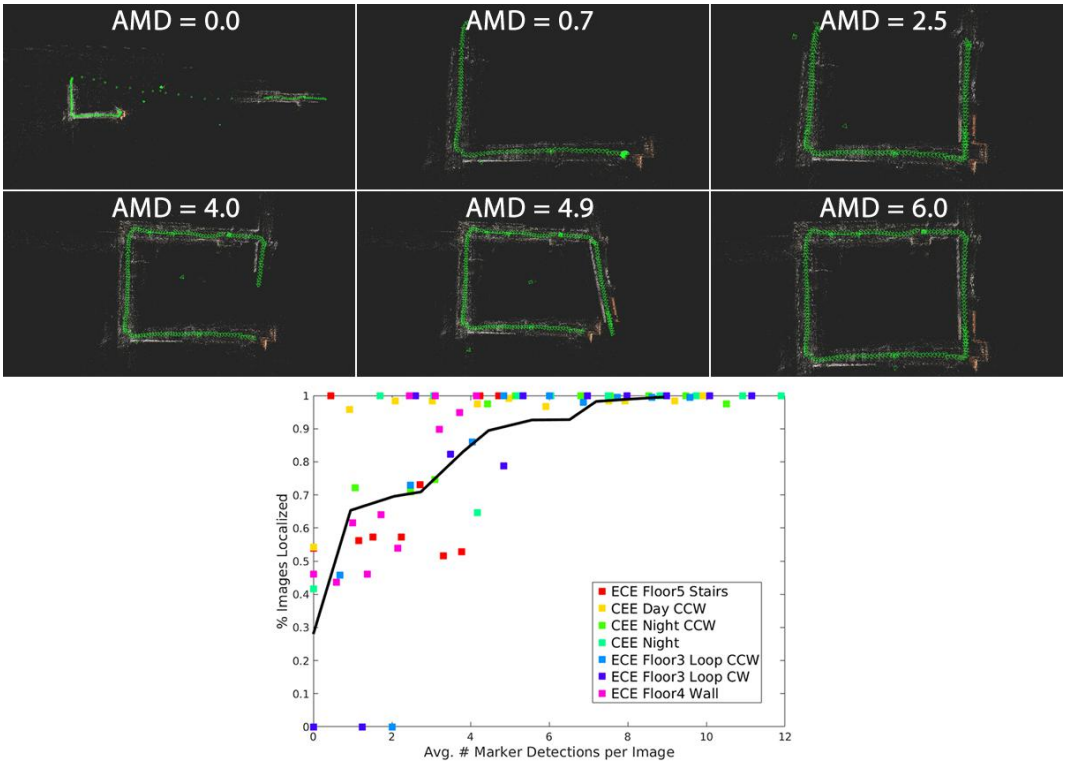


Figure 4.10: Top: six images show how the reconstruction of *ECE Floor3 Loop CCW* improves as AMD (average marker detections per image) increases. Bottom: the plot shows the percent of images localized as the AMD increases. Each color represents a different dataset. The trend line is shown in black. As the AMD increases, the percent of localized images increased to 100%.

markers in these challenging areas and fewer (or zero) markers in easier areas can help our method achieve accurate, complete reconstructions with drastically fewer total marker detections.

4.5 Conclusion

In this chapter, we present an incremental SfM method that significantly outperforms existing methods when fiducial markers are detected in the scene. We introduce a new dataset with 16 image collections of indoor scenes with square markers placed throughout. We use the unique marker IDs to improve image matching and resectioning order. We use the marker size and corner locations to add new constraints for bundle adjustment. Using our new dataset, we demonstrate how our method outperforms state of the art SfM and marker based SfM algorithms. Lastly, we show that even a small number of visible markers often improves reconstruction results.

Chapter 5

Geometry-Informed Material Recognition

Our goal is to recognize material categories using images and estimated 3D points. In prior material recognition research, surface geometry is a confounder, and much effort goes into creating features that are stable under varying perspective (e.g., scale and rotationally invariant features [178]) and lighting. Although the resulting systems often perform well for standard material/texture datasets [39, 89, 23, 131], their success does not always translate to improved categorization in natural objects or scenes [109, 49]. However, for many important applications, 3D surface geometry can be estimated, rather than marginalized, and used to improve performance. For example, a ground robot can estimate surface geometry from stereo when identifying navigable terrain. Likewise, when surveying progress in a construction site, 3D points from LiDAR or structure-from-motion can help distinguish between concrete and stone to determine if a facade is in place. In principal, geometric estimates should help with material classification by revealing surface orientation and roughness and disambiguating texture cues, but because surface texture and geometry interact in complex ways, it is not clear how best to take advantage of 3D points. Can local geometry cues be simply added to existing color/texture features, or do they need to be considered jointly? Are approaches to improve robustness of texture descriptors still helpful? Is it helpful to rectify the image based on surface geometry? This chapter aims to answer these questions and provide a material recognition approach that is well-suited to applications for which surface geometry estimates are available.

We introduce a new dataset of construction materials photographed in natural outdoor lighting (called "GeoMat" for geometry/materials). Many of the 19 material categories (Fig. 5.3) are highly confusable, such as "paving" vs. "limestone" or "smooth cement" vs. "granular stone" (Fig. 5.1), but these distinctions are important in a construction setting. For each category, several different

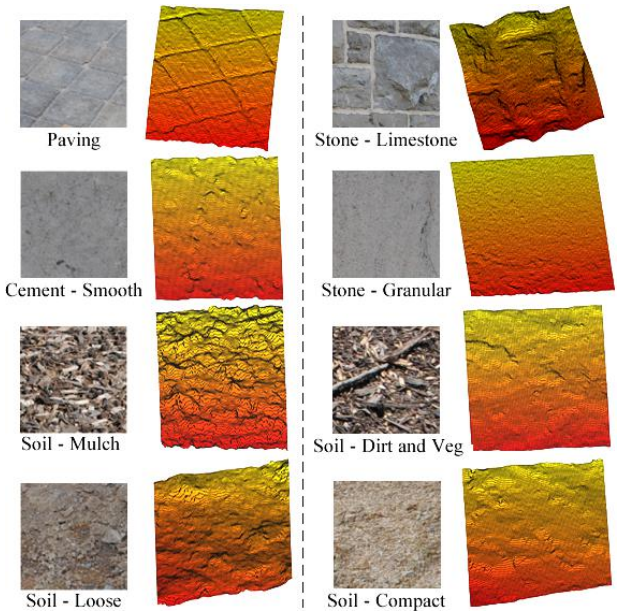


Figure 5.1: The material patches shown in column one were misclassified as the class shown in column three by [30] because the classes are visually similar. However, the geometry (column two and four) for these patches is different. This chapter investigates how to use differences in 3D geometry to improve material classification. We also contribute the GeoMat dataset consisting of images and geometry for material patches and a large scale construction site scene.

physical samples are photographed from a variety of orientations and positions, and structure-from-motion [167] and multi-view stereo [69] are used to estimate 3D points. We explore two test settings: individual 2D/3D patches of material samples and scene-scale images of construction sites with 3D point clouds.

Using our GeoMat dataset, we investigate how estimated 3D geometry can improve material classification in real world scenes. Surface orientation and roughness provide valuable cues to material category, and we model them with histograms of surface normals. Additionally, observed texture is due to a combination of surface markings, micro-geometric texture, and camera-relative surface normal. Our geometric detail is not sufficient to model micro-geometric texture, but by jointly representing camera-relative surface normal and texture response, we may reduce ambiguity of signal. Thus, we try jointly representing texture and normals. An alternative strategy is to frontally warp the image, based on surface normal, which would undo perspective effects at the cost of some resolution due to interpolation. Our main technical contribution is to investigate all

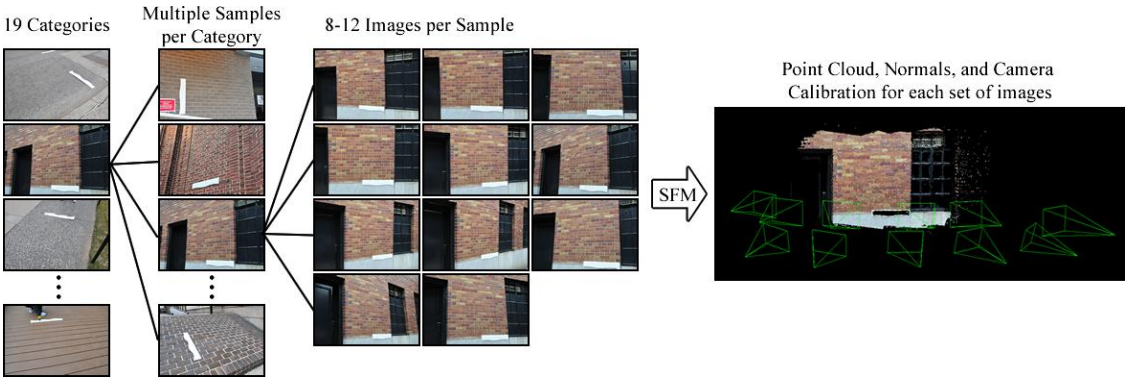


Figure 5.2: Each GeoMat material category is made from 3 to 26 different samples where each sample consists of 8 to 12 images at different viewpoints, a segmented point cloud, and normal vectors.

Figure 5.3: GeoMat represents 19 material categories.

of these strategies to determine which strategy or combination of strategies makes the best use of geometric information. We also investigate how performance of 3D-sensitive features varies with scale and surface orientation.

In summary, our **contributions** are: (1) we create the GeoMat dataset for studying material categorization from images supplemented with sparse 3D points; (2) we investigate several strategies for using 3D geometry with color and texture to improve material recognition; (3) we investigate effects of scale and orientation and application to images of construction sites.

5.1 Related Work

Features: Early methods for material classification used filter bank responses to extract salient statistical characteristics from image patches [39, 190, 177, 17, 16]. Leung and Malik [107] introduced the LM-filter bank and proposed "3D Textons", which, despite the name, are clustered

2D filter responses at each pixel without direct 3D information. The term "texton" was coined by Julez [95] twenty years earlier to describe elements of human texture perception, and "3D" conveys the goal of classifying 3D material textures. Varma and Zisserman [178] later proposed the "RFS" filter bank and an in-plane rotationally invariant (via max pooling) "MR8" response set. A string of subsequent work, led by Varma and Zisserman, replaced filter responses with more direct clusterings and statistics of intensities of small pixel neighborhoods [179, 132, 181, 27, 79, 160, 164, 113]. Liu et al. [109] explored a variety of color, texture, gradient, and curvature features for classifying object-level material images. It was recently shown by Cimpoi et al. [30, 31] that convolutional neural networks and fisher vectors with dense SIFT outperforms previous approaches for texture classification.

These works all explore purely 2D image-based features and, as such, aim to be robust to 3D surface variations by encoding texture for samples observed from various viewpoints and lighting. We show that directly encoding local surface geometry both jointly and independently with texture yields significant gains. We are the first, to our knowledge, to investigate how to integrate 3D geometric cues with texture representations for material classification. We note that object segmentation from RGB-D images is a commonly studied problem (e.g., Koppula et al. [104]), but because the image resolution is too low for texture to be an effective cue and the focus is on object rather than material categories, the problem is dissimilar (the same is also true of LiDAR classification approaches).

Datasets: The CUReT dataset created by Dana et al. [39] was the first large-scale texture / material dataset, providing 61 material categories, photographed in 205 viewing and lighting conditions. The KTH-TIPS dataset by Hayman et al. [89] added scale variation by imaging 10 categories from the CUReT dataset at different scales. For both datasets, all images for a category were from the same physical sample, so that they may be more accurately called texture categorization than material categorization datasets. Subsequently, KTH-TIPS2 by Caputo et al. [23] was introduced, adding images from four physical samples per category. Still, variation of material complexity within categories was limited, motivating Liu et al. [109] to create the Flickr Materials Database

containing images for ten categories with 50 material swatch images and 50 object-level images. Several recent datasets have focused on material recognition for applications. This includes the construction materials dataset by Dimitrov and Golparvar-Fard [49] which consists of 200x200 patches of 20 common construction materials, and the Describable Texture Dataset by Cimpoi et al. [30] which provides 5,640 texture images jointly annotated with 47 material attributes. Most recently, Bell et al. [7] contributed the Materials in Context Database, consisting of many full scenes with material labels.

While these datasets provide ample resources for studying image-based material classification, there does not yet exist a dataset that provides geometry information together with real-world material images. Our GeoMat dataset provides real world material images and geometric information in the form of point clouds, surface normals, and camera intrinsic and extrinsic parameters. Our dataset also differs in that the taxonomy is chosen to be relevant to a practical application (construction management), rather than based on visual distinctiveness, leading to several groups of highly confusable material types. For example, Varma and Zisserman [180] report accuracy of 96.4% on the 61 CUReT classes using the MR8 representation; the same MR8 representation achieves only 32.5% accuracy on our dataset.

5.2 Dataset

We created the GeoMat dataset (Figs. 5.2, 5.3, 5.4, 5.5, and 5.6) to investigate how local geometric data can be used with image data to recognize materials in real-world environments. The training set consists of "focus scale" 100x100 patches of single materials sampled from high resolution photographs of buildings and grounds. There are two test sets: (i) 100x100 patches sampled from photographs of different physical surfaces, and (ii) "scene scale" photographs of a construction site. Both focus scale and scene scale datasets consist of images and associated 3D points estimated through multiview 3D reconstruction.

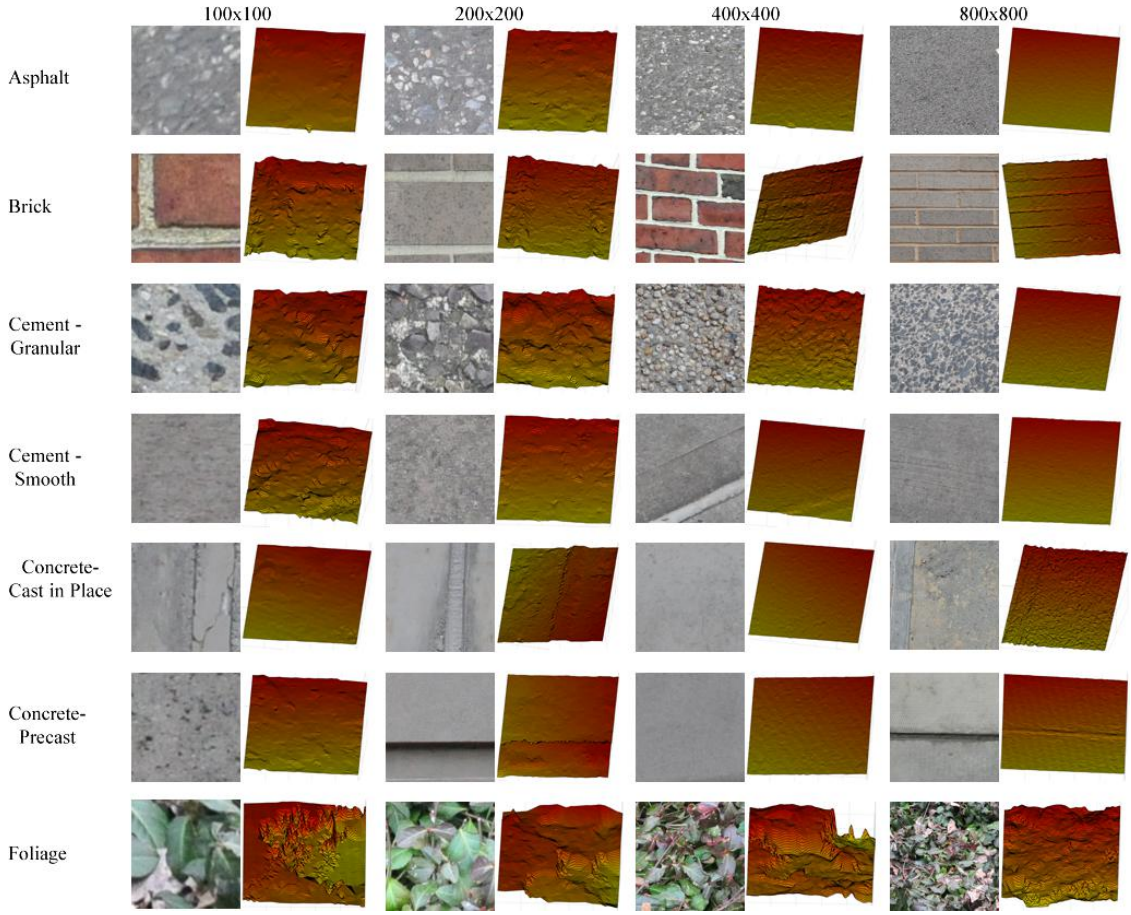


Figure 5.4: Four samples per category (one at each scale) for Asphalt, Brick, Cement - Granular, Cement - Smooth, Concrete - Cast In Place, Concrete - Precast, and Foliage.

5.2.1 Focus Scale Training and Testing Sets

The focus scale data is sampled from high-resolution (4288x2848 pixels) images that predominantly depict a single material, such as a "brick" wall or "soil - compact" ground. The dataset consists of 19 material categories as shown in Fig. 5.3. There are between 3 and 26 different physical surfaces (i.e. different walls or ground areas) for each category; each surface is photographed from 8 to 12 viewpoints (Fig. 5.2). A marker of known scale is present in each image. Structure from motion [167] and multi-view stereo [69] are used to generate a point cloud, normal vectors, and camera intrinsic and extrinsic parameters. The points are manually labeled into regions of interest to facilitate sampling patches that consist purely of one material.

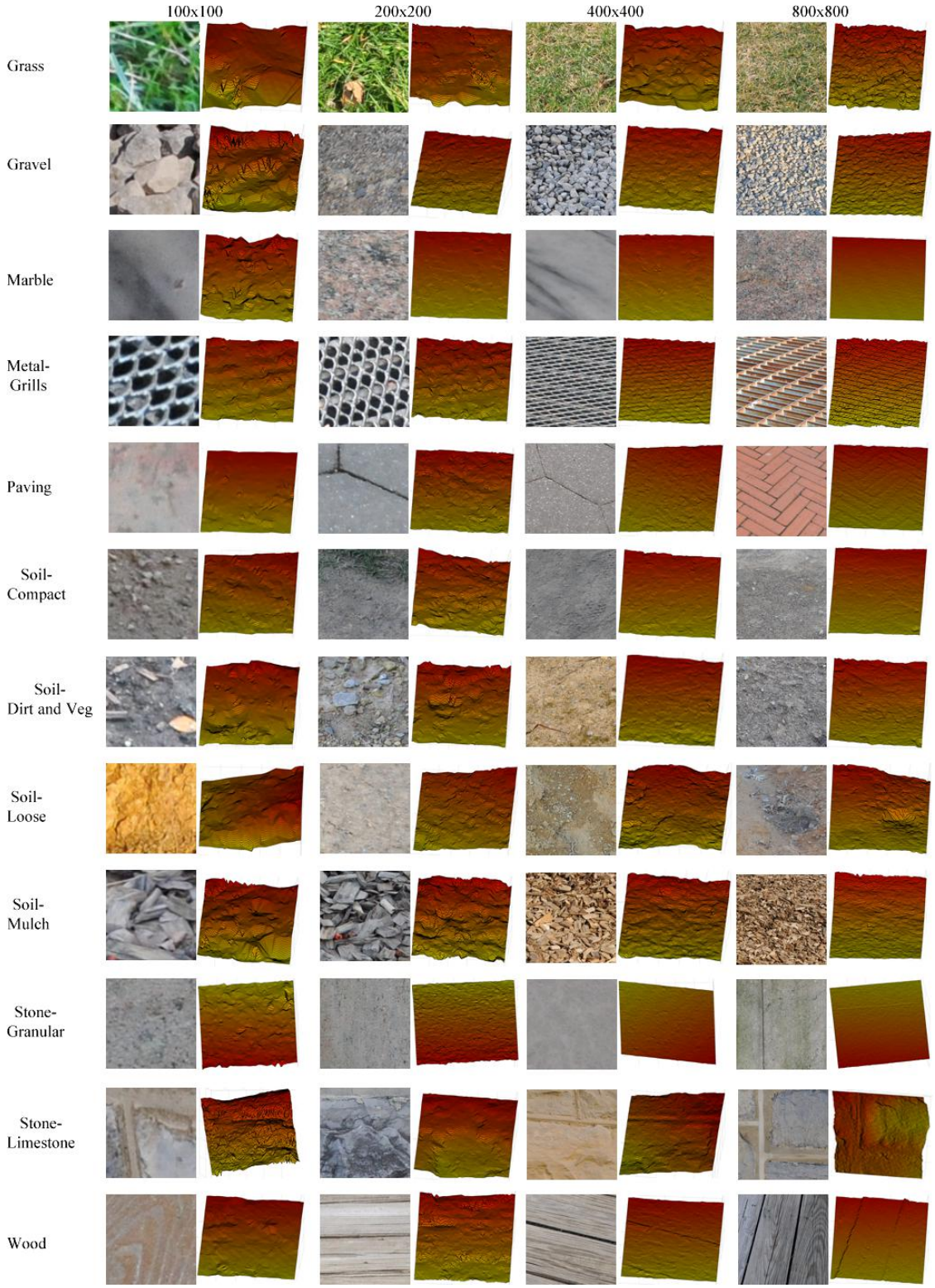


Figure 5.5: Four samples per category (one at each scale) for Grass, Gravel, Marble, Metal - Grills, Paving, Soil - Compact, Soil - Dirt and Veg, Soil - Loose, Soil - Mulch, Stone - Granular, Stone - Limestone, and Wood.

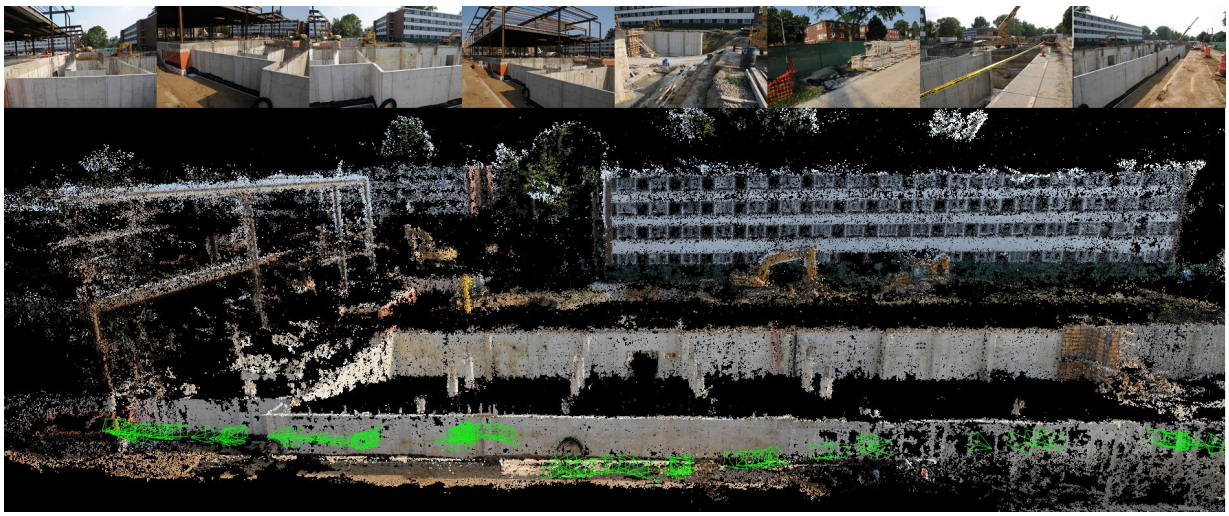


Figure 5.6: The scene scale dataset consists of 160 images of a construction site with an accompanying point cloud, normal vectors, and camera intrinsic and extrinsic parameters. The SFM-registered camera frusta are shown in green. 11 of the 19 material categories are represented: “Brick”, “Cement – Smooth”, “Concrete – Precast”, “Concrete – Cast in Place”, “Foliage”, “Grass”, “Gravel”, “Metal – Grills”, “Soil – Compact”, “Soil – Loose”, and “Wood”.

We make training and testing splits by assigning approximately 70% of the physical surfaces of each category to training and the remainder to testing. For example, given a category with three surfaces, training samples will come from two of the surfaces and testing samples will come from the remaining unused surface. Similarly, for a category with 23 samples, training samples will come from 16 of the surfaces and testing samples will come from the remaining 7 unused surfaces. Since each category consists of at least three different surfaces, this ensures that there are at least two surfaces per category for training, at least one surface per category for testing, and the samples drawn for training are from different surfaces than those drawn for testing.

For each category, we extract 100 training patches and 50 testing patches at 100x100, 200x200, 400x400, and 800x800 resolutions. This results in a total of 400 training patches and 200 testing patches per category. These patches are scaled to 100x100 to simulate viewing the materials at different scales/distances. We extract an equal number of patches from each surface. For example, if we want to extract 200 testing patches from 10 surfaces, then 20 testing patches are extracted from each surface. Since each surface consists of many images, we then divide the intended number of patches evenly among the images of that surface. Continuing with the example, if a

surface has 10 images and we want to extract 20 total patches from that surface, then we extract 2 patches per image. Each patch is then extracted randomly from within a region of the image that was manually annotated as representative of the intended category.

Each patch consists of image data, geometry data, and from which category and surface it was drawn. Example image and normal patches are shown in Figs. 5.4 and 5.5. Image data includes normalized gray-scale and HSV images and the location in the image from which the sample was drawn. Geometry data includes a sparse depth map, sparse normal map, intrinsic and extrinsic camera parameters, gravity vector, and scale.

5.2.2 Scene Scale Testing Set

The scene scale data consists of 160 images (4288x2848 pixels each) of one large construction site. Of the 19 material categories, 11 are represented: "Brick", "Cement – Smooth", "Concrete – Precast", "Concrete – Cast in Place", "Foliage", "Grass", "Gravel", "Metal – Grills", "Soil – Compact", "Soil – Loose", and "Wood". Structure from motion and multi-view stereo were used to generate a point cloud, normal vectors, and camera intrinsic and extrinsic parameters. The point cloud is hand-labeled to match our 19 material categories. Points not matching one of the 19 categories are labeled as unknown. Fig. 5.6 provides a depiction of the scene scale testing set.

The scene scale data is used only for testing. We use the dataset to verify that our conclusions drawn from the simpler focus scale dataset still hold when classifying regions in more typical images. Others could use the data for testing multiview material recognition or transferring patch-based material models to scene-scale images. Labeled 3D points (826,509 total) that are viewable in a given image are back-projected onto pixels, so that a sparse set of pixels (about 21,500 per image on average) has ground truth labels in each image. When testing with the scene scale data, we use the entire focus scale dataset for training.

5.3 Classification with Geometric Features

5.3.1 Features and Modeling

Our main interest is in how to use patch geometry to improve or augment image features. We describe the 2D texture and color features that we use and then describe several cues that leverage the estimated depth and surface normals.

2D Features

RFS/MR8: The intensity pattern of a material is a good cue for recognition [178, 107, 153, 37] because it encodes surface albedo patterns and small-scale shape. Consider brick: we expect to see grainy rectangular blocks separated by layers of mortar. Filter banks have proven useful for capturing these and other intensity patterns for material recognition. We use the RFS filter bank and derived MR8 responses described by Varma and Zisserman [180], which are shown to be effective on the CUReT dataset [178]. The RFS filter set contains first and second derivative filters at 6 orientations and 3 scales (36 filters) and Gaussian and Laplacian of Gaussian (LoG) filters at scale $\sigma = 10$ (2 filters). The MR8 filters are created by keeping only the maximum filter response across each set of orientations for a given scale, along with the two Gaussian/LoG filters. The MR8 filters are intended to provide robustness to surface orientation. In training, filter responses at each pixel are clustered into 10 clusters per category using k-means, following the standard texton approach [107]. The RFS and MR8 features are histograms of these textons (clustered filter responses), normalized to sum to one.

FV/VLAD: SIFT [111] features offer an alternative method of capturing texture patterns and are used by Lui et.al. [109] for material recognition on the Flicker Materials Dataset. We quantize multi-scale dense SIFT features using the Improved Fisher Vector (FV) framework [139] and the Vectors of Locally Aggregated Descriptors (VLAD) framework [78] as described by Cimpoi et.

al. [30]. In training, the dimensionality of the dense SIFT features is reduced to 80 using PCA. For Improved Fisher Vectors, the reduced dense SIFT features are then clustered into 256 modes using a Gaussian Mixture Model. The FV based feature vectors are mean and covariance deviations from the GMM modes. For VLAD, the reduced dense SIFT features are clustered into 512 modes using K-means. The VLAD based feature vectors are the residuals of a feature from the cluster means. The feature vectors are ℓ^2 normalized and sign square-rooted as is standard for Improved Fisher Vectors and Vectors of Locally Aggregated Descriptors.

HSV: Materials can be recognized by their color — grass is often green, bricks are often red and brown, and asphalt is often gray. We incorporate color by converting image patches to the HSV color space. The HSV pixels are then clustered into five clusters per category using k-means, and the resulting histograms are used as features.

CNN: Convolutional Neural Networks offer another approach for capturing texture and color patterns. We follow the approach of Cimpoi et. al. [30, 31], and use the pre-trained VGG-M network of [105]. The features are extracted from the last convolutional layer of the network.

3D Features

We investigate three strategies for including 3D geometric information for material classification: (i) jointly cluster texture features and 3D normal vectors at each pixel (-N); (ii) independently cluster normal vectors, build histograms (N_{3D}), and add them to 2D features; and (iii) frontally rectify the image based on a plane fit before computing texture filter responses.

-N: Image texture is affected by albedo patterns, surface orientation, and small surface shape variations. These factors make classification based on filter responses more difficult. A common solution is to make features robust to surface orientation by learning from many examples or creating rotationally invariant features (as in MR8 and SIFT). We hypothesize that explicitly encoding

geometry jointly with the texture features will be more discriminative.

We interpolate over the sparse 3D normal map to produce a pixel-wise estimate of normals for a given image patch. We then transform the normal vectors according to the camera calibration information so that the normals are in the coordinate frame of the image plane. For MR8 and RFS, we then concatenate the normal vectors onto the filter responses at each pixel and cluster them into 10 clusters per category to create MR8-N and RFS-N textons. The textons are then used to build MR8-N and RFS-N histograms. For FV, we first reduce the dimensionality of the SIFT features to 80 using PCA. Then, we concatenate the 3D normal vectors onto the reduced SIFT descriptors for each pixel and cluster into 256 modes using a Gaussian Mixture Model. The modes include characteristics of both the texture and normal vectors. The Improved Fisher Vector formulation [139] is then used to create FV-N feature vectors.

$\mathbf{N}_{3D}$: It is unclear whether a joint or independent representation of geometry will perform better, and it is also possible that both representations may help with overall discrimination. Thus, we formulate the N_{3D} feature as an independent representation of the sparse normal map.

As described for (-N), we interpolate over the sparse 3D normal map to produce pixel-wise normal estimates for each patch and transform the normal vectors into the coordinate frame of the image plane. Rather than concatenating the normals with the texture features (as was done with (-N)), we independently cluster the normal vectors into five clusters per category using k-means and use the resulting histograms as our N_{3D} features. Note that we also tried clustering the normal vectors using a Gaussian Mixture Model and building Fisher Vectors but saw worse performance using this method.

Rectification: In addition to directly encoding 3D surface geometry, frontally rectifying the image may improve texture features by making filter responses more directly correspond to albedo and microshape changes, removing the confounding factor of overall surface orientation and scale. We perform rectification using a homography defined by making the mean surface normal face the

Features	-	+HSV	+N_{3D}	+HSV+N_{3D}
(RFS [178] / RFS-N)	(33.24 / 37.76)	(45.03 / 47.89)	(49.68 / 49.55)	(51.24 / 52.29)
(MR8 [178] / MR8-N)	(32.47 / 41.34)	(45.32 / 47.84)	(49.74 / 50.63)	(53.03 / 53.37)
(FV [30] / FV-N)	(60.97 / 66.95)	(62.92 / 68.76)	(65.87 / 68.16)	(66.37 / 69.05)
(FV+CNN [30] / FV-N+CNN)	(68.92 / 73.80)	(67.82 / 72.05)	(72.08 / **73.84**)	(70.79 / 72.13)
(VLAD [78] / VLAD-N)	(51.82 / 53.66)	(56.71 / 58.95)	(60.50 / 62.00)	(60.68 / 61.37)

Table 5.1: Including 3D geometry features increases the mean accuracy for all feature sets. Both joint and independent modeling of the 3D geometry improve the mean accuracy. The best mean accuracy is 73.84%.

camera. The rectified patch is scaled to 100x100.

5.3.2 Classification

For this work, we are interested in investigating the utility of different geometric features and establishing a baseline for classification with the GeoMat dataset. We use a one vs. all SVM scheme for classification because SVMs have been shown to achieve exemplary performance on texture classification tasks for all our 2D features [89, 30, 31]. Experiments with only histogram features (RFS/MR8, HSV, RFS-N/MR8-N, N_{3D}) benefit from weighting the histograms before concatenating them. We learn the weights by grid search using leave-one-out cross-validation on the training set with a nearest neighbor classifier (which can be done very efficiently by caching inter-example histogram distances for each feature type). The weighted and concatenated histograms are then classified with a χ^2 SVM. For experiments that include non-histogram feature vectors (FV, FV-N, CNN), the feature vectors and histograms are individually L2 normalized before being concatenated. We use libSVM [25] for training.

5.3.3 Application to Scene Scale

In our scene scale test set, the input is RGB images at original scale with a sparse set of reconstructed points. We use this data to verify that our conclusions on the curated focus scale dataset still hold for typical images of large-scale scenes. To apply our patch-based classifer, we segment each image into 290-300 superpixels (roughly 200x200 pixels each) using SLIC [1]. For each superpixel, we extract the image patch and corresponding sparse normal map for the minimum

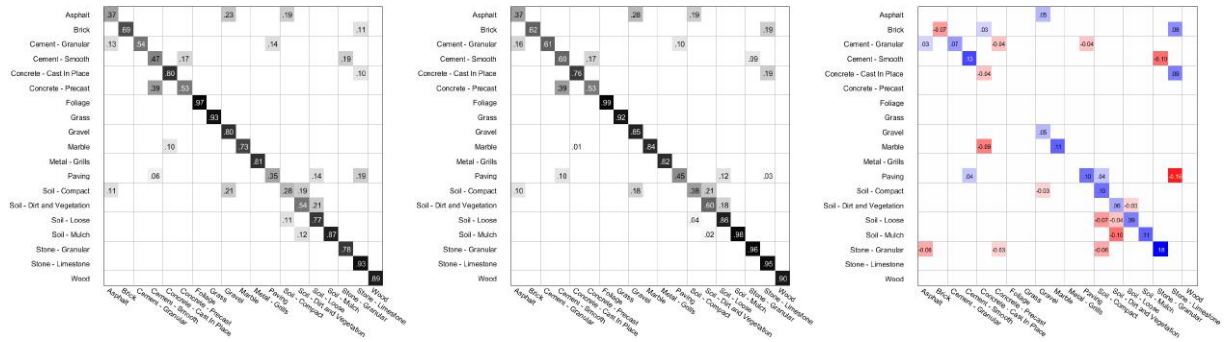
bounding rectangle. The sparse normal map is then interpolated and transformed into the coordinate frame of the image plane. The image patches are resized for the CNN and used as-is for all other features. Classification is done on each patch independently and accuracy is measured as the average accuracy per pixel label.

5.4 Results and Analysis

Table 5.1 provides the mean classification accuracies on the testing data of the focus scale component of the GeoMat dataset. Since jointly clustering texture and 3D geometry (-N) is an alternative representation of the texture features, we display it in conjunction with the texture representation (texture representation / joint texture and normal representation). Then, each extra feature set that is concatenated is shown as another column of the table. We consider all of the original 2D features (RFS, MR8, FV, FV+CNN, VLAD) to be baselines. From this table we see that the highest overall accuracy is 73.84% for FV-N+CNN+N_{3D} which outperforms the best 2D baseline of FV+CNN [30] at 68.92%. Note also that the accuracy of using just N_{3D} features is 32.50%.

We also tried several other baselines. First, we tried the approach of Cimpoi et al. [31]. This approach constructs Improved Fisher Vectors from the output features of the last convolutional layer of the pre-trained ImageNet network [105]. This method achieved 63.79% mean accuracy on our focus scale dataset. We also investigated the texture classification method provided by Sifre et al. [164]. This method learns a joint rotation and translation invariant representation of image patches using a cascade of wavelet modulus operators implemented in a deep convolutional network. We tested this baseline method with the same range of octave options as [164] and achieved a best accuracy of 36.53% with the number of octaves set to 3. Both of these baselines performed worse than FV+CNN at 68.92%.

Both joint and independent representations of geometry improve mean classification accuracy. These two options map to (-N) features and N_{3D} features respectively. From Table 5.1 and comparing column by column, we first see in column two that the (-N) significantly improves the



(a) Confusion: FV+CNN (Best 2D). (b) Confusion: FV-N+CNN+N_{3D} (Best 3D). (c) Difference: (Best 3D - Best 2D).

Figure 5.7: The difference confusion matrix (Best 3D - Best 2D) shows the categories where the best 3D confusion matrix (b) performed better (blue cells) or worse (red cells) than the best 2D confusion matrix (a). The largest improvements are for Soil, Stone, and Cement. These categories often have similar visual appearance, but not necessarily similar 3D geometry. Including 3D geometry alleviates some of the confusion between these categories.

mean classification accuracy compared to the 2D texture features (e.g. FV-N outperforms FV). In column three, we see that HSV provides a boost to the mean accuracies in every case; however, the inclusion of (-N) still improves the mean accuracies by at least 2% and by almost 6% for FV+HSV (e.g. FV-N+HSV outperforms FV+HSV). In column four, we can make two observations. First, we see that the inclusion of independent normal features (N_{3D}) significantly improves the mean accuracy compared to the 2D texture features (e.g. FV+N_{3D} outperforms FV). In addition, we see that in every case except RFS, including both joint and independent geometry features (-N and N_{3D}) improves over using just one (e.g. FV-N+N_{3D} outperforms FV-N and FV+N_{3D}). Note that the improvement for adding either (-N) or N_{3D} (e.g. FV-N or FV+N_{3D}) is larger than the additional improvement gained by adding one to the other (e.g. adding N_{3D} to FV-N). This makes sense because both features are modeling similar information; however, it is interesting that they still both contribute when used together. This trend is maintained with the inclusion of HSV features in column five.

It is not clearly helpful to rectify the images based on surface geometry. Each row of Table 5.2 shows the mean accuracies of the data with and without rectification; denoted as (without

Features	-	+HSV	+N_{3D}	+HSV+N_{3D}
RFS	(33.24 / 34.42)	(45.03 / 44.13)	(49.68 / 50.18)	(51.24 / 53.50)
RFS-N	(37.76 / 39.82)	(47.89 / 49.50)	(49.55 / 50.00)	(52.29 / 53.11)
MR8	(32.47 / 35.03)	(45.32 / 45.29)	(49.74 / 52.08)	(53.03 / 52.24)
MR8-N	(41.34 / 42.05)	(47.84 / 48.42)	(50.63 / 51.63)	(53.37 / 54.32)
FV	(60.97 / 60.26)	(62.92 / 63.32)	(65.87 / 65.70)	(66.37 / 66.29)
FV-N	(66.95 / 66.82)	(68.76 / 68.08)	(68.16 / 67.11)	(69.05 / 68.82)
FV+CNN	(68.92 / 70.13)	(67.82 / 68.50)	(72.08 / 72.47)	(70.79 / 70.74)
FV-N+CNN	(73.80 / 72.97)	(72.05 / 71.79)	(73.84 / 73.68)	(72.13 / 71.87)
FV+**CNN**	(68.92 / 68.95)	(67.82 / 67.55)	(72.08 / 72.05)	(70.79 / 70.58)
FV-N+**CNN**	(73.80 / 73.71)	(72.05 / 72.13)	(73.84 / 73.82)	(72.13 / 72.21)
FV+CNN	(68.92 / 70.13)	(67.82 / 68.47)	(72.08 / 72.53)	(70.79 / 70.92)
FV-N+CNN	(73.80 / 72.92)	(72.05 / 71.84)	(73.84 / 73.50)	(72.13 / 71.95)
VLAD	(51.82 / 50.89)	(56.71 / 56.74)	(60.50 / 60.63)	(60.68 / 61.11)
VLAD-N	(53.66 / 52.58)	(58.95 / 57.82)	(62.00 / 60.82)	(61.37 / 60.84)

Table 5.2: Rectification tends to help for filter features and not help when (-N) is included. Because the better performing features often perform worse with rectification, rectification does not appear to be an effective use of 3D geometry for improving classification. For FV+CNN, we denote which features are using rectification using boldfaced text.

rectification / with rectification). It is possible to apply the rectification to either FV or CNN; thus, we denote which features are using rectification using boldfaced text. From the results, we can see that rectification tends to improve the filter features (RFS, RFS-N, MR8, MR8-N) and some cases where (-N) is not included (FV+CNN). Rectification worsens the results for FV and also for most cases where (-N) is included (FV-N and FV-N+CNN). Because improvements are minimal when they exist and better performing feature combinations are often worse with rectification, we conclude that rectification is not an effective use of 3D geometry for improving classification.

3D geometry helps with categories that look the same visually but have different 3D geometry. Fig. 5.7a and Fig. 5.7b are the confusion matrices of the best performing 2D (FV+CNN) and 3D (FV-N+CNN+N_{3D}) feature sets respectively. For clarity, cells are hidden if they have a value below 0.1 in both confusion matrices. Fig. 5.7c is the subtraction of the best 2D confusion matrix from the best 3D confusion matrix. For clarity, cells are hidden if they have a value below 0.02.

The difference confusion matrix in Fig. 5.7c shows the categories where the best 3D confusion matrix performed better (blue cells) or worse (red cells) than the best 2D confusion matrix. The values along the diagonal (which represent improved classification accuracy for a given category)

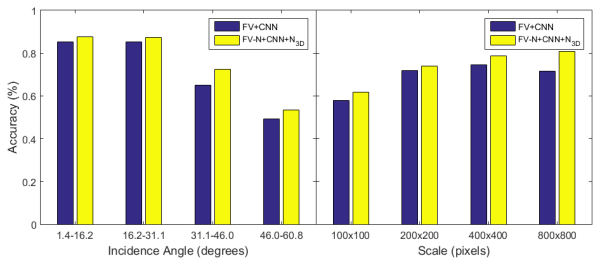


Figure 5.8: Mean accuracy improves as the incidence angle approaches a frontal view. Mean accuracy improves as scale increases. FV-N+CNN+N_{3D} (yellow, Best 3D) outperforms FV+CNN (purple, Best 2D) for all scales and angles.

have improved in most cases. The largest improvements are for Soil (Compact, Dirt and Veg, Loose, and Mulch), Stone (Granular and Limestone), and Cement (Granular and Smooth). The reason we see larger gains in this area is because these materials look similar in terms of color and texture, but not similar in terms of their normal maps. In Fig. 5.1, we show in the two left-most columns of each row an example (image patch and normal map) that was misclassified in 2D but was correctly classified in 3D. The 2D incorrect guess then defines the class for columns three and four, and a hand-selected example is chosen from the training data that illustrates the possible similarity between image patches of the confused classes. It is clear from the examples shown in Fig. 5.1 why confusions are likely and also how the 3D geometry helps to alleviate these confusions. In particular, we see the flat panels and grooves for paving, the large stone outlines and mortar for limestone, the smooth surface of granular stone, and varying degrees of relief for the different types of soil (mulch, dirt and veg, loose, and compact).

Including 3D geometry improves classification accuracy for all scales and viewing directions. Fig. 5.8 shows the accuracy of the mean material classification as it depends on incidence angle and scale. It is interesting to see that there is a general improvement in accuracy for increased scale. We suspect this is because the texture pattern of certain material categories becomes more evident for farther scales (e.g. it is easier to see the layers of brick and mortar). We also see that the smaller incidence angles (closer to being a frontal view) have higher mean classification accuracies; however, the decrease in mean classification accuracy does not occur until we reach angles

Features	Pixel Labeling Accuracy
Hoiem et al. [91]	18.53
FV+CNN [30]	21.01
FV-N+CNN+N_{3D}	35.87

Table 5.3: For our scene scale dataset, the best 3D geometry feature set (FV-N+CNN+N_{3D}) outperforms the best 2D feature set (FV+CNN) and the external baseline, which is consistent with our results on the focus scale dataset.

larger than 31.1 degrees. Lastly, it is worth noting that the best 3D features (FV-N+CNN+N_{3D}) improve over the best 2D features (FV+CNN) for all angles and scales.

Results are consistent for the scene scale data. Finally, we test on the scene scale component of the GeoMat dataset. Results are shown in Table 5.3. We chose to test the approach using the best performing 2D (FV+CNN) and 3D (FV-N+CNN+N_{3D}) feature sets from Table 5.1. The 3D feature set outperforms the 2D feature set considerably (35.87% vs. 21.01%), which is consistent with our results for the focus scale component of the GeoMat dataset.

As an external baseline, we train the superpixel-based classifier from Hoiem et al. [91] that includes region shape, color, and texture cues. The classifier is trained on our focus scale training set and applied to Felzenszwalb and Huttenlocher [60] superpixels generated from the test images, as in their original algorithm. The baseline classifier achieves 18.53% accuracy, which is slightly worse than our 2D features and much worse than our 3D features. Note that our approach and the baseline do not benefit from scene context or image position, which can be valuable cues, because they are trained using focus scale patches. Other constraints and priors could be used to obtain the best possible performance, but our experiments are intended to focus on the impact of geometric features on appearance models.

Chapter 6

Using Geometry and Appearance for Construction Progress Monitoring

In this chapter, we outline our approach for detecting progress on construction sites. Automatically detecting progress is important for keeping construction projects on schedule. Adherence to project schedules and budgets is the most highly valued performance metric by project owners [11]. Despite its significance, more than 53% of typical construction projects are behind schedule and more than 66% do not meet their budget requirements [11]. Some of the major factors that lead to poor performance on jobsites include 1) inconsistency among contractors, subcontractors and owners in terms of how a construction project is faring at any given date, 2) flawed performance management due to lack of frequent reporting of actual performance to project teams, and 3) planners missed connections to most up-to-date construction progress information [26]. In addition, project management teams have to deal with multiple parties (i.e., owners, themselves, and many trades) constantly updating construction documents and schedules.

Our solution is to use 3D point cloud and geometry to reason about progress. Our approach begins with a collection of images of a construction site as input. From these images, we use 3D reconstruction to build a point cloud. Then, we align the point cloud to the building information modeling[103] (BIM) model of the construction site. The BIM model is a 3D model (like a CAD model) showing each individual element of the planned structure (i.e. building, bridge, stadium, etc.). With the point cloud and BIM aligned, we filter the point cloud to detect which elements of the BIM exist in the point cloud. Lastly, we use material recognition to infer the state of progress of each constructed BIM element.

In summary, our **contributions** are: (1) an image-based method to progress monitoring that uses a 3D point cloud and material recognition to reason about progress at an element level; and (2) experiments with data from a real construction site for a large hotel and arena.

6.1 Related Work

Advances in camera, UAV, and 3D computer vision technology has enabled an increase in visual data collection on construction sites. The availability of this visual data has resulted in advances in model-based construction progress detection leveraging as-built modeling techniques. These techniques use image-based point clouds or laser scanned point clouds.

Image-based Point Clouds: Siebert et al. [163] uses a camera-equipped Unmanned Aerial Vehicles (UAVs) to capture images of earthwork projects for creating 3D maps of the terrain. These surveyed point clouds can be used for measuring progress. Similarly, Golparvar-Fard et al. [77, 76] creates point clouds from unordered sets of images. They align these point clouds with BIMs and compare geometries of as-built and as-planned models to reason about progress deviation. To deal with limited visibility and occlusions, Han and Golparvar-Fard [82] propose an appearance-based method that reasons about progress by recognizing textures of materials on construction images that are aligned with BIMs. The images are aligned with BIMs automatically after the image-based point clouds are aligned with BIMs.

Laser Scanned Point Clouds: Turkan et al. [174, 175] uses surface-based recognition to detect building elements from scanned point clouds for automated progress detection and then improves the accuracy of progress tracking using the earned value analysis. Bosché et al. [15] proposes a Scan-vs-BIM object recognition framework for tracking the built status of Mechanical, Electrical, and Plumbing (MEP) works. Similarly, Kim et al. [97] compares 4D BIM with detected building elements from laser scanned point clouds to measure construction progress. These laser scanned methods are based on geometry recognition and generally provide more accurate and denser point clouds of the structures of interest than the image-based methods. However, the image-based methods provide multiple viewpoints and, therefore, wider viewpoints and occlusions.

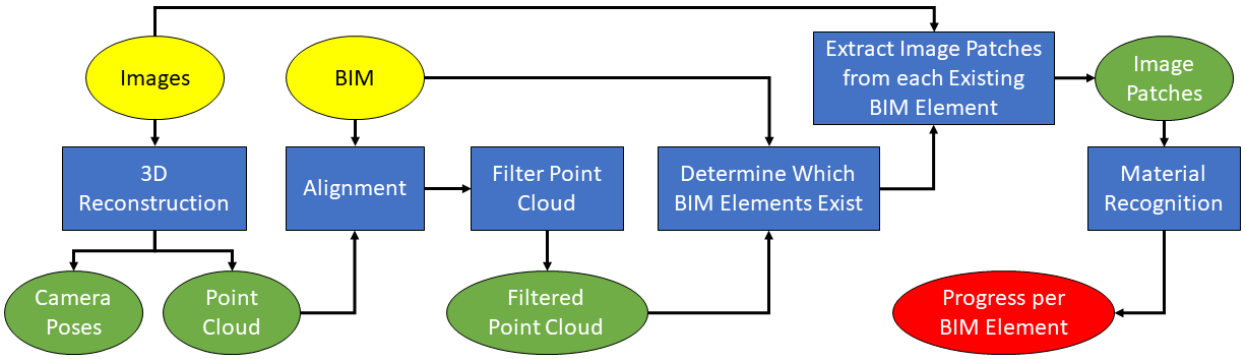


Figure 6.1: Workflow of our approach. The initial inputs are images of a construction site and a BIM model for the site (yellow ovals). Structure from motion followed by multi-view stereo (3D Reconstruction) create a dense point cloud from the images. The BIM model and dense point cloud are then aligned. Then, the BIM model is used to filter erroneous points in the point cloud and the point cloud is used to determine which BIM elements have been built. Image patches are extracted from images for each existing BIM element and their material is classified to estimate the final progress of each element.

6.2 Method

Figure 6.1 diagrams our method. Blue rectangles indicate the different processing modules of our method. Yellow ovals indicate the initial inputs (i.e. images and the BIM model), green ovals indicate the intermediate inputs and outputs, and the red oval is the final output (i.e. detected progress per BIM element).

6.2.1 3D Reconstruction and Alignment

The process begins by taking input images and generating a 3D point cloud using 3D reconstruction. The 3D reconstruction process is two parts: (1) images are registered to newly created sparse point cloud using structure from motion (VisualSfM [186]), and (2) the registered images and sparse point cloud are used to create a dense point cloud with multi-view stereo (MVE [75]). Then, corresponding features (e.g., corners) between the dense point cloud and BIM are manually picked to solve for a similarity transform (using Horn's method [94]) that aligns the BIM and dense point cloud.

6.2.2 Filtering the Point Cloud

Now that the point cloud and BIM are aligned, we can use the BIM to filter out extraneous points. In particular, we use minimum and maximum coordinates of the entire BIM (min_{BIM} and max_{BIM}) as an initial filtering. This process reduces the size of point clouds substantially and reduces computation times for the subsequent steps; especially when images are captured using a UAV because the surrounding area of the structure is also typically imaged and reconstructed, but not useful for progress detection.

6.2.3 Determining which BIM Elements Exist

We do an occupancy check to test whether or not points are occupied by each BIM element. In particular, for each element we check if points are within the minimum and maximum coordinates (min_{BIM_i} and max_{BIM_i}). During this process, space distribution of a point cloud within each BIM element boundary is computed for filtering out false positives (i.e., there are some points within the boundary but they are not part of any BIM elements). To maximize efficiency and minimize computation time, vectorized computation and minimal computational complexity are critical factors. Therefore, we use a normal distribution with a standard deviation (σpc_i). This approach checks the minimum number of points within each BIM element (θ_{nPc}) and also checks densities to avoid false negatives. The final result of this process is an estimate of which BIM elements have been constructed.

6.2.4 Extracting Image Patches for Existing BIM Elements

The initial step is patch extraction using camera parameters and the list of BIM elements that we estimate to have been constructed (as was done by Han and Golparvar-Fard [82]'s approach). For each image c that is used to create 3D point clouds, N image patches per BIM element $FACE_c^i$ are extracted. These patches are then classified as a material using our material recognition approach.

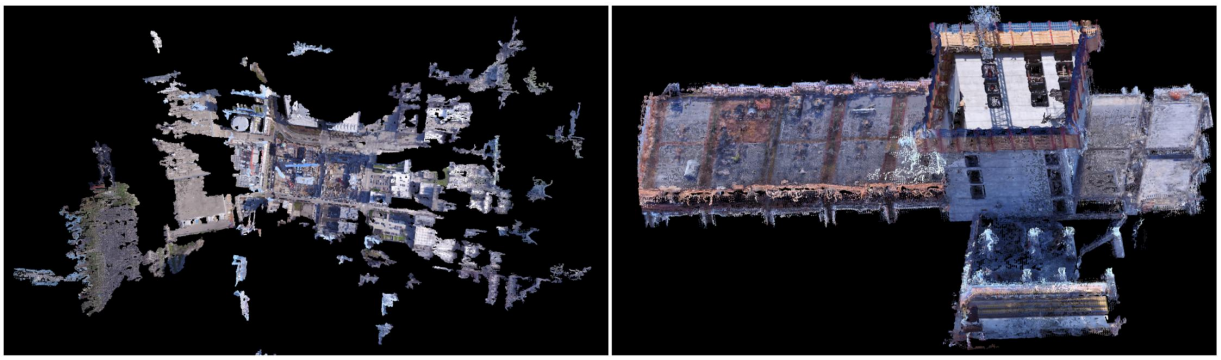


Figure 6.2: The left image is HP before filtering and the right images is HP after filtering. Millions of erroneous points are removed.

6.2.5 Material Recognition on Image Patches

Our material classification follows a similar approach to that of Cimpoi et al. [31] and DeGol et al. [45]. In particular, a combination of Fisher Vectors [139] and Convolutional Neural Network (CNN) [105] features are input to a Support Vector Machine (SVM) for learning. Fisher Vectors are created by first extracting dense SIFT [111] features from each patch. In training, the dense scale-invariant feature transform (SIFT) features are reduced to a dimensionality of 80 by Principal Components Analysis (PCA) before being clustered into 256 modes with a Gaussian Mixture Model (GMM). The Fisher Vectors are then mean and covariance deviations from the GMM modes (ℓ^2 normalized and sign square-rooted). Convolutional Neural Network features are created using the pre-trained VGG-M network of Krizhevsky et al. [105]. The features are extracted from the last convolutional layer of the network rather than the fully connected layers.

Classification is then performed using a one vs. all SVM scheme. This scheme has been shown to achieve exemplary results for 2D texture recognition [89, 30, 31, 45]. A χ^2 kernel is used with the SVM. The Fisher Vector and CNN features are normalized individually before being concatenated for learning. We classify each patch that was extracted for a given BIM element and do weighted scoring to decide the final material class. We assign more weight to the expected material type (indicated by the BIM). This process is done for all existing BIM elements to assign a final class to each.

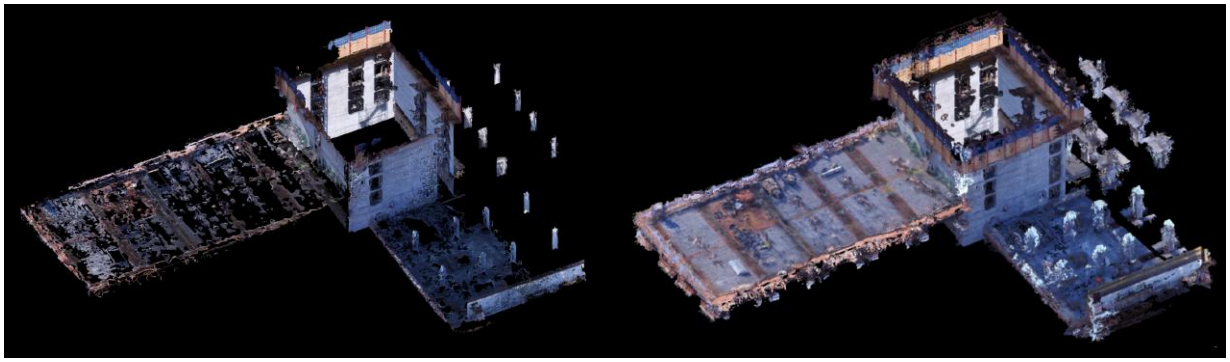


Figure 6.3: The left point cloud is with $\theta_{reg} = 0$ feet and the right point cloud is with $\theta_{reg} = 1.5$ feet. Increasing θ_{reg} beyond 0 results in a more complete detection of BIM elements; however, increasing it too much causes false positives.

6.3 Results and Discussion

We use a collection of images from a real hotel construction project (denoted as HP) for testing. This dataset consists of 532 images. Running 3D reconstruction on these images results in a dense point cloud with $6,390,085$ points. We also use the BIM for this hotel; however, we prune the number of elements down to the relevant ones for the current state of construction. This pruning results in a total of 69 BIM elements. For material recognition, the Construction Material Library (CML) collected by Dimitrov, Han, and Golparvar-Fard [50, 82] was used as the training dataset. CML consists of more than 3,000 images that are categorized into 20 construction material classes. For all experiments, we use a 3.60 GHz CPU with 64 GB of RAM.

Filtering with the BIM significantly decreases the number of points. Before filtering, the HP point cloud consisted of more than 6 million points. Processing these points can be expensive in terms of both disk space and computation. Using our filtering approach, we reduce the total number of points to $1,348,148$, which is a reduction of almost 80%. The computation time to perform this filtering is 0.22 seconds.

Figure 6.2 shows how a point cloud can have a large percentage of unwanted points. The main cause, in the case of HP, was the use of a UAV for data capture. Due to safety concerns related to

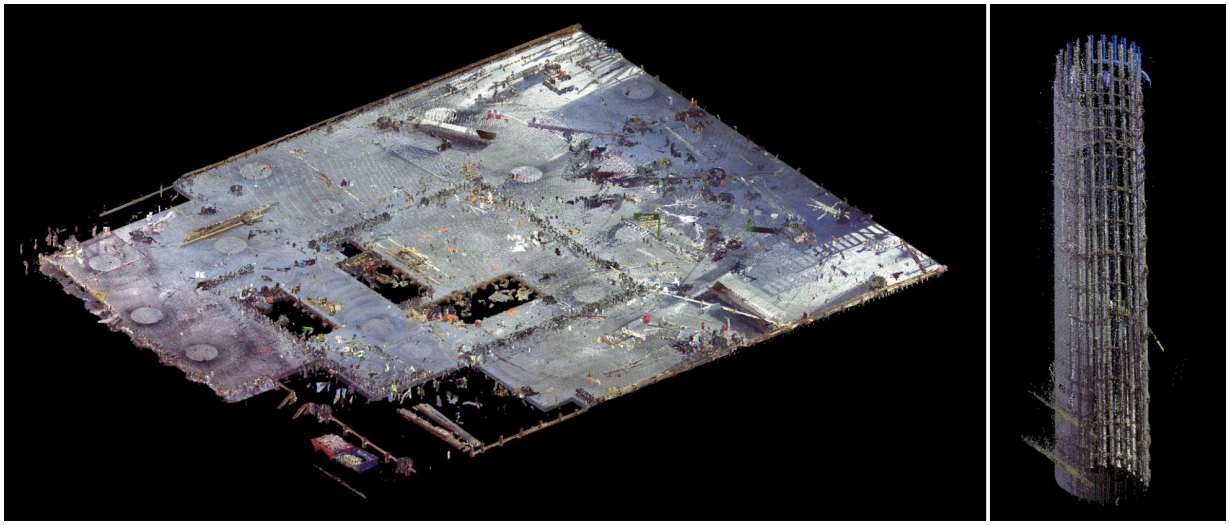


Figure 6.4: Points that represent a concrete slab BIM element (left) and a column BIM element (right).

cranes, a UAV operator had to fly the UAV at high altitudes. Thus, many images had background buildings and roads that surround HP. The initial filtering process removed these objects.

Our method successfully detects all existing BIM elements. Figure 6.5 shows that increasing θ_{reg} from 0 to 0.25 and beyond improves the detection accuracy of BIM elements. What this shows is that there are errors in the alignment of the point cloud and BIM. These errors are caused by several things: (1) there are errors in the point cloud where images are not perfectly registered; (2) the real construction is never exactly to the specifications of the BIM, so the volume/area of constructed elements is often different than the BIM elements; and (3) the similarity transformation to align the point cloud and BIM may not be perfect. For these reasons, θ_{reg} is important because it provides some slack in finding points for each BIM element. Figure 6.3 shows the qualitative difference between $\theta_{reg} = 0$ and $\theta_{reg} = 1.5$. We can see that the model is more complete when θ_{reg} is not zero; in particular, the formwork of the core walls is captured better. Figure 6.4 provides a zoomed in view of our method identifying points for a large concrete slab and column. These are two qualitative examples demonstrating the effectiveness of our approach. Note also that increasing θ_{reg} also impacts the number of false positives because incorrect points are now considered

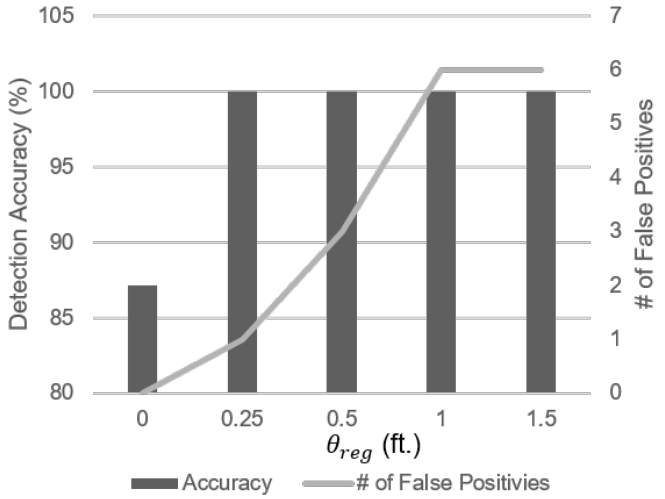


Figure 6.5: When $\theta_{reg} = 0$ feet, detection accuracy is about 87%. When is increased to and beyond $\theta_{reg} = 0.25$ feet, detection accuracy increases to 100%. However, as θ_{reg} increases, the number of false positives also increase.

for BIM element detection. Thus, we want to choose a middle ground that achieves high accuracy with few false positives (e.g. 0.25 feet for HP). The computation time for this process is 1.64 seconds.

Weighting material classes based on the BIM prior improves classification accuracy. Figure 6.6 shows our material classification results for varying values of w_{BIM}. A w_{BIM} value of 1 means that all classes were weighted equally. In this case, the classification accuracy is about 65%. However, as we increase the value of w_{BIM} from 1 to 2.5, we see a large increase in the classification accuracy to 91%. This large increase is because of two factors. First, some of the formwork used for HP has blue meshing on it that is not well represented in the CML dataset. Secondly, HP is a vertical construction project with many occlusions. These occlusions cause numerous image patches to be extracted that do not capture the textures of interest. By using the material indicated by the BIM element as a prior for classification, we were able to overcome these challenges and greatly improve our recognition accuracy.

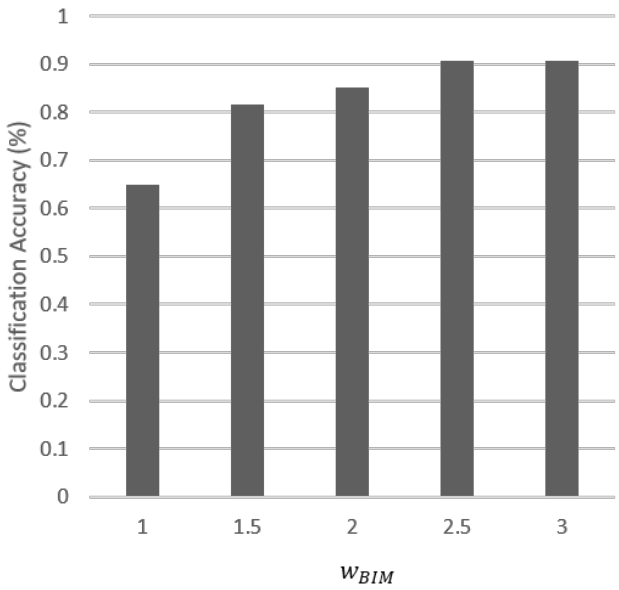


Figure 6.6: When $w_{BIM} = 1$ (i.e. no weighting), the material classification accuracy is 68%. As the value of w_{BIM} increases (i.e. as the weight increases for the material class specified by the BIM element), the classification accuracy also increases to 90% (until $w_{BIM} = 2.5$). This is because the prior information about the material class is useful in classification.

6.4 Conclusion

We present a new approach for progress detection on construction sites that uses both 3D geometry and images. We test our method with a real construction site for a hotel, and show that we are able to detect all of the BIM elements, and recognize over 90% of their materials. We also show that using the material specified by the BIM element as a prior for classification significantly increases material classification accuracy.

Chapter 7

Conclusion

In this dissertation, we make contributions towards autonomous monitoring of built environments using robots with cameras. In particular, we introduce (1) a simulator to improve the path used for image capture, (2) a detection algorithm that can detect our new fiducial marker at a negligible computational cost, enabling SLAM integration without sacrificing real time processing, (3) a new structure from motion approach that significantly outperforms other approaches when markers are detected, and (4) a new dataset of material patches with both image and 3D geometry data and a recognition algorithm that uses both 2D and 3D information for improved recognition.

We hope that our work will inspire others to continue improving data collection, robot navigation, 3D mapping, and recognition technology so that we can one day achieve autonomous monitoring for built environments. Indeed, there are still many problems left to solve. In regards to data collection, we still do not know the best way to collect image and video data to ensure quality 3D maps are created. Our method allows us to predict if a path will succeed, but it does not plan paths. However, our method can be used to generate many different trajectories quickly for testing. In this way, we can identify what types of trajectories cause more error during 3D reconstruction. One way to make exploring the space of trajectories tractable is to build trajectories from a set of motion primitives (like building a roller coaster from a set of predefined track pieces). Reasonable motion primitives could include sideways motion and forward motion with varying degrees of out-of-plane rotation and in-plane rotation. Then, various motion primitive trajectories could be simulated in different 3D scenes to see how camera localization and 3D point error are effected. Lastly, given a new scene, a trajectory planning algorithm could build the trajectory from motion primitives to minimize the camera localization and 3D point errors.

Concerning fiducial markers, we do not know the best way to place markers for improved 3D

reconstructions. Our method takes advantage of markers that have been placed, but it does not provide analysis of engineering best practices for marker placement. This is important because it is time consuming to place markers. Ideally, we want to place the fewest markers while achieving accurate 3D reconstructions. One way to approach this problem is to mask markers in our dataset such that markers are only present in certain parts of the scene. Since we have successful reconstructions using all the tags, we can pinpoint markers by ID and location to mask markers in specific regions of scenes. We know that scenes with few or confusing features are challenging, so we can use this information to choose how scenes are split into regions (e.g. a plain hallway will be challenging because of few features, so we could use markers in this hallway and mask them everywhere else).

Lastly, in regards to material recognition, there is still room for improvement in classifying materials in real world scenes. Challenges preventing perfect material classification in real scenes are that (1) material appearance changes considerably due to light and perspective, (2) materials can look different within a category (e.g. red brick and gray brick, smooth cement and granular cement, etc.), and (3) materials can look similar across categories (e.g. smooth concrete and smooth cement). In our work, we show how sparse geometry can improve material recognition accuracy despite these challenges; however, our classification accuracy on the construction site scene are not perfect. One approach to improve these results is to use the construction site plan and schedule as a prior. We show in Chapter 6 that this approach has promise; however, further exploration could be done by classifying more materials over longer time periods where the construction process goes through additional stages and more materials are used. Moreover, we can expect that future improvements in machine learning will also translate to improvements in material recognition.

Each of these problems is challenging and will require significant work before we see robots autonomously monitoring built environments. However, the resources and human lives that can be saved by achieving autonomous monitoring easily justifies the future work in this area.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012. 76

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *EEE 12th International Conference on Computer Vision*, pages 72–79, Sept 2009. 4, 8, 46, 51, 53

[3] Artoolkit. http://www.hitl.washington.edu/artoolkit/. 29, 30

[4] Aruco: a minimal library for augmented reality applications based on opencv. http://www.uco.es/investiga/grupos/ava/node/26. 27, 30

[5] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy. Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments. *The International Journal of Robotics Research*, 2012. 2

[6] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013. 9

[7] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *CVPR*, 2015. 5, 68

[8] F. Bergamasco, A. Albarelli, L. Cosmo, E. Rodola, and A. Torsello. An accurate and robust artificial marker based on cyclic codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2016. 3, 27, 28, 29, 30, 37, 45

[9] F. Bergamasco, A. Albarelli, E. Rodolà, and A. Torsello. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 113–120, June 2011. 30

[10] F. Bergamasco, A. Albarelli, and A. Torsello. Pi-tag: a fast image-space marker design based on projective invariants. *Machine Vision and Applications*, 24(6):1295–1310, 2013. 30

[11] M. Bevan and J. Steve. LCI National Webinar: Preliminary findings from national project performance research, 2016. 82

[12] Bigsfm: Reconstructing the world from internet photos. http://www.cs.cornell.edu/projects/bigsfm/, October 2016. 4, 8

[13] T. Birdal, I. Dobryden, and S. Ilic. X-tag: A fiducial tag for flexible and accurate bundle adjustment. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 556–564, Oct 2016. 45

[14] J.-L. Blanco, F.-A. Moreno, and J. González. A collection of outdoor robotic datasets with centimeter-accuracy ground truth. *Autonomous Robots*, 27(4):327–351, November 2009. 10

[15] F. Bosché, A. Guillemet, Y. Turkan, C. Haas, and R. Haas. Tracking the built status of mep works: Assessing the value of a scan-vs-bim system. *Journal of Computing in Civil Engineering*, 2013. 5, 83

[16] I. Brilakis and L. Soibelman. Shape-based retrieval of construction site photographs. *Journal of Computing in Civil Engineering*, 2008. 66

[17] I. Brilakis, L. Soibelman, and Y. Shinagawa. Material-based construction site image retrieval. *Journal of Computing in Civil Engineering*, 2005. 66

[18] U. S. C. Bureau. Value of construction put in place at a glance. Technical report, U.S. Department of Commerce, 2013. 1

[19] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016. 10

[20] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 9

[21] L. Calvet, P. Gurdjos, and V. Charvillat. Camera tracking using concentric circle markers: Paradigms and algorithms. In *2012 19th IEEE International Conference on Image Processing*, pages 1361–1364, Sept 2012. 27, 30

[22] L. Calvet, P. Gurdjos, C. Griwodz, and S. Gasparini. Detection and accurate localization of circular fiducials under highly challenging conditions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3, 27, 28, 29, 30, 37, 45

[23] B. Caputo, E. Hayman, and P. Mallikarjuna. Class-specific material categorisation. In *ICCV*, 2005. 64, 67

[24] C. Celozzi, G. Paravati, A. Sanna, and F. Lamberti. A 6-dof artag-based tracking system. *IEEE Transactions on Consumer Electronics*, 56(1):203–210, February 2010. 27, 30

[25] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2011. 76

[26] S. Changal, A. Mohammad, and M. van Nieuwland. The construction productivity imperative: How to build megaprojects better. *Insights & Publications, McKinsey&Company*, 2015. 82

[27] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao. Wld: A robust local image descriptor. *PAMI*, 2010. 67

[28] Y. Cho, J. Lee, and U. Neumann. A multi-ring color fiducial system and an intensity-invariant detection method for scalable fiducial-tracking augmented reality. In *In IWAR*, 1998. 27, 29

[29] C. I. I. (CII). *A Guide to Activity Analysis*, 2010. 1

[30] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *CVPR*, June 2014. 65, 67, 68, 74, 76, 77, 81, 86

[31] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. 5, 67, 74, 76, 77, 86

[32] D. Claus and A. W. Fitzgibbon. Reliable fiducial detection in natural scenes. In *Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV*, pages 469–480, 2004. 31

[33] D. Claus and A. W. Fitzgibbon. Reliable automatic calibration of a marker-based position tracking system. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2005. 31

[34] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 1988. 19

[35] N. R. Council. *Advancing the Competitiveness and Efficiency of the U.S. Construction Industry*. The National Academies Press, 2009. 1

[36] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. SfM with MRFs: Discrete-continuous optimization for large-scale structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(12):2841–2853, December 2013. 4, 8, 10

[37] O. Cula and K. Dana. Compact representation of bidirectional texture functions. In *CVPR*, 2001. 73

[38] V. F. d. C. Neto, D. B. d. Mesquita, R. F. Garcia, and M. F. M. Campos. On the design and evaluation of a precise scalable fiducial marker framework. In *2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images*, pages 216–223, Aug 2010. 30

[39] K. Dana, B. Van-Ginneken, S. Nayar, and J. Koenderink. Reflectance and Texture of Real World Surfaces. *ACM Transactions on Graphics (TOG)*, 1999. 64, 66, 67

[40] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. 3

[41] J. DeGol, A. Akhtar, B. Manja, and T. Bretl. Automatic grasp selection using a camera in a hand prosthesis. In *EMBC*, 2016. 6

[42] J. DeGol, T. Bretl, and D. Hoiem. Chromatag: A colored marker and fast detection algorithm. In *ICCV*, 2017. 3, 45, 61

[43] J. DeGol, T. Bretl, and D. Hoiem. Improved structure from motion using marker matching. In *In Submission to ECCV*, 2018. 4

[44] J. DeGol, M. Golparvar-Fard, and D. Hoiem. Geometry-informed material recognition. In *CVPR*, 2016. 5

[45] J. Degol, M. Golparvar-Fard, and D. Hoiem. Geometry-informed material recognition. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2016. 86

[46] J. DeGol, D. Hanley, N. Aghasadeghi, and T. Bretl. A passive mechanism for relocating payloads with a quadrotor. In *IROS*, 2015. 6

[47] J. DeGol, J. Y. Lee, R. Kataria, D. Yuan, T. Bretl, and D. Hoiem. Feats: Synthetic feature tracks for structure from motion evaluation. In *In Submission to ECCV*, 2018. 2

[48] J. DeGol and M. Nam. A clustering approach for detecting moving objects captured by a moving aerial camera. In *ICASSP*, 2014. 6

[49] A. Dimitrov and M. Golparvar-Fard. Vision-based material recognition for automated monitoring of construction progress and generating building information modeling from unordered site image collections. *Advanced Engineering Informatics*, 2014. 64, 68

[50] A. Dimitrov and M. Golparvar-Fard. Vision-based material recognition for automated monitoring of construction progress and generating building information modeling from unordered site image collections. *Advanced Engineering Informatics*, 28(1):37 – 49, 2014. 87

[51] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017. 9

[52] Drone deploy. https://www.dronedeploy.com/. 1, 7

[53] E. Dunn, J. v. d. Berg, and J. M. Frahm. Developing visual sensing strategies through next best view planning. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009. 10

[54] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 2006. 3

[55] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley Publishing, 2008. 1

[56] J. Engel and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014. 3

[57] C. Eschmann, C.-M. Kuo, C.-H. Kuo, and C. Boller. Unmanned aircraft systems for remote building inspection and monitoring. *6th European Workshop on Structural Health Monitoring*, 2012. 7

[58] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 44

[59] X. Fan, L. Zhang, B. Brown, and S. Rusinkiewicz. Automated view and path planning for scalable multi-object 3d scanning. *ACM Trans. Graph.*, 2016. 10

[60] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 81

[61] C. Feng, V. Kamat, and C. C. Menassa. Marker-assisted structure from motion for 3d environment modeling and object pose estimation. In *Construction Research Congress*, 2016. 47, 54

[62] J. Fernandez-Galarreta, N. Kerle, and M. Gerke. Uav-based urban structural damage assessment using object-based image analysis and semantic reasoning. *Natural Hazards and Earth System Sciences*, 15:1087–1101, 06 2015. 7

[63] M. Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596 vol. 2, June 2005. 27, 29, 30

[64] M. Fiala. Comparing artag and artoolkit plus fiducial marker systems. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, Oct 2005. 30

[65] M. Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1317–1324, July 2010. 45

[66] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan. Example-based synthesis of 3d object arrangements. In *ACM SIGGRAPH Asia 2012 papers*, SIGGRAPH Asia '12, 2012. 9

[67] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, 2010. 8

[68] S. Fuhrmann, F. Langguth, N. Moehrle, M. Waechter, and M. Goesele. Mve – an image-based reconstruction environment. *Computer and Graphics*, 2015. 11

[69] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 2010. 65, 69

[70] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 9

[71] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014. 30, 45

[72] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481 – 491, 2016. 30

[73] L. B. Gatrell, W. A. Hoff, and C. W. Sklair. Robust image features: concentric contrasting circles and their image extraction. In *Proc. SPIE*, 1992. 29

[74] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361, June 2012. 10

[75] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007. 84

[76] M. Golparvar-Fard, J. Bohn, J. Teizer, S. Savarese, and F. Peña Mora. Evaluation of image-based modeling and laser scanning accuracy for emerging automated performance monitoring techniques. *Automation in Construction*, 20(8):1143–1155, 2011. 5, 83

[77] M. Golparvar-Fard, F. Peña Mora, and S. Savarese. Application of d4ar a 4-dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication. *ITcon*, 14:129–153, 2009. 5, 83

[78] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 73, 76

[79] Y. Guo, G. Zhao, and M. Pietikäinen. Discriminative features for texture description. *Pattern Recognition*, 2012. 67

[80] Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard. Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works. *Visualization in Engineering*, 2016. 7

[81] K. Han, J. Degol, and M. Golparvar-Fard. Geometry- and appearance-based reasoning of construction progress monitoring. *Journal of Construction Engineering and Management*, 2018. 7

[82] K. Han and M. Golparvar-Fard. Appearance-based material classification for monitoring of operation-level construction progress using 4d bim and site photologs. *Automation in Construction*, 53(0):44 – 57, 2015. 5, 83, 85, 87

[83] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. Understanding real world indoor scenes with synthetic data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 9

[84] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla. Scenenet: An annotated model generator for indoor scene understanding. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5737–5743, May 2016. 9

[85] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *ICRA*, 2014. 8, 9

[86] C. Harris and M. Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988. 43

[87] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 54

[88] H. Hattori, V. N. Boddeti, K. Kitani, and T. Kanade. Learning scene-specific pedestrian detectors without real data. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3819–3827, June 2015. 9

[89] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh. On the significance of real-world conditions for material classification. In *ECCV*, 2004. 64, 67, 76, 86

[90] A. Herout, I. Szentandrsi, M. Zachari, M. Dubsk, and R. Kajan. Five shades of grey for fast and reliable camera pose estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 30

[91] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007. 81

[92] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 2013. 10

[93] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 1987. 20

[94] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987. 84

[95] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 1981. 67

[96] B. Kaneva, A. Torralba, and W. T. Freeman. Evaluating image feaures using a photorealistic virtual world. In *IEEE International Conference on Computer Vision*, 2011. 9

[97] C. Kim, H. Son, and C. Kim. Automated construction progress measurement using a 4d building information model and 3d data. *Automation in Construction*, 31(0):75 – 82, 2013. 5, 83

[98] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007. 3

[99] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009. 3

[100] M. Klopschitz and D. Schmalstieg. Automatic reconstruction of wide-area fiducial marker models. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007. 47

[101] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 2017. 10

[102] R. Koch, A. Kolb, C. R. salama (eds, B. Atcheson, F. Heide, and W. Heidrich. Caltag: High precision fiducial markers for camera calibration, 2010. 29, 30

[103] B. Koo and M. Fischer. Feasibility study of 4d cad in commercial construction. *Journal of Construction Engineering and Management*, 2000. 1, 82

[104] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011. 67

[105] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012. 74, 77, 86

[106] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, Sept 2006. 31

[107] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 2001. 5, 66, 73

[108] H. Lim and Y. S. Lee. Real-time single camera slam using fiducial markers. In *2009 ICCAS-SICE*, 2009. 47

[109] C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz. Exploring features in a bayesian framework for material recognition. In *CVPR*, 2010. 5, 64, 67, 73

[110] P. E. Love, R. Lopez, and J. T. Kim. Design error management: interaction of people, organisation and the project environment in construction. *Structure and Infrastructure Engineering*, 2014. 1

[111] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 73, 86

[112] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004. 17, 51

[113] J. Mao, J. Zhu, and A. Yuille. An active patch model for real world texture and appearance classification. In *ECCV*. 2014. 67

[114] Mapillary. https://www.mapillary.com/. 56

[115] J. Marín, D. Vázquez, D. Gerónimo, and A. M. López. Learning appearance in virtual scenarios for pedestrian detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 137–144, June 2010. 9

[116] Matrix vision mvbluefox-200wc. https://www.matrix-vision.com/usb2.0-industrial-camera-mvbluefox.html. 17, 37

[117] J. michael Frahm, P. Fite-georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y. hung Jen, E. Dunn, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *European Conference on Computer Vision (ECCV)*, 2010. 4, 46, 51

[118] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. 13, 14

[119] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 2005. 13, 14

[120] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002. 3

[121] C. Mostegel, M. Rumpler, F. Fraundorfer, and H. Bischof. Uav-based autonomous image acquisition with multi-view stereo quality assurance by confidence prediction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016. 10

[122] P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *2013 IEEE International Conference on Computer Vision*, 2013. 55

[123] P. Moulon, P. Monasse, R. Marlet, and Others. Openmvg. https://github.com/openMVG/openMVG. 4

[124] P. Moulon, P. Monasse, R. Marlet, and Others. Openmvg. an open multiple view geometry library. https://github.com/openMVG/openMVG. 55

[125] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 3, 17

[126] R. Muoz-Salinas, M. J. Marn-Jimenez, E. Yeguas-Bolivar, and R. Medina-Carnicer. Mapping and localization from planar markers. *Pattern Recognition*, 2018. 4, 46, 47, 56, 57, 61

[127] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, 2002. 29, 30

[128] M. Neunert, M. Bloesch, and J. Buchli. An open source, fiducial based, visual-inertial motion capture system. In *2016 19th International Conference on Information Fusion (FUSION)*, 2016. 47, 48, 56, 57, 61

[129] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011. 3

[130] N. A. of Sciences. Advancing the competitiveness and efficiency of the u.s. construction industry. Technical report, National Academy of Sciences, 2009. 1

[131] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen, and S. Huovinen. Outex - new framework for empirical evaluation of texture analysis algorithms. In *International Conference on Pattern Recognition*, 2002. 64

[132] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 2002. 67

[133] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407, May 2011. 3

[134] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407, May 2011. 27, 29, 30

[135] Opensfm. https://github.com/mapillary/opensfm. 4, 15, 20, 46, 55, 56, 57, 61

[136] Optitrack - motion capture systems. http://optitrack.com/. 15

[137] Occupational safety and health administration (osha). "commonly used statistics". department of labor, occupational safety, and health administration. https://www.osha.gov/oshstats/commonstats.html. Accessed: 2018-04-03. 1

[138] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 31

[139] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010. 73, 75, 86

[140] Pix4d. https://pix4d.com/. 1, 7

[141] M. G. Prasad, S. Chandran, and M. S. Brown. A motion blur resilient fiducial for quadcopter imaging. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 254–261, Jan 2015. 30

[142] Rawseeds project. http://www.rawseeds.org/home/. 10

[143] Reconstruct. https://www.reconstructinc.com/. 1, 7

[144] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 31

[145] J. Rekimoto and Y. Ayatsuka. Cybercode: Designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing Augmented Reality Environments*, 2000. 29, 30

[146] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 31

[147] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi. Submodular trajectory optimization for aerial 3d scanning. In *International Conference on Computer Vision (ICCV) 2017*, 2017. 10

[148] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 9

[149] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, 2011. 17

[150] J. Sattar, E. Bourque, P. Giguere, and G. Dudek. Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *Computer and Robot Vision, 2007. CRV '07. Fourth Canadian Conference on*, pages 165–174, May 2007. 29, 30

[151] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier. Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics Automation Magazine*, 2014. 2

[152] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *European Conference on Computer Vision (ECCV)*, 2002. 46

[153] C. Schmid. Constructing models for content-based image retrieval. In *CVPR*, 2001. 73

[154] K. Schmid, H. Hirschmüller, A. Dömel, I. Grixa, M. Suppa, and G. Hirzinger. View planning for multi-view stereo 3d reconstruction using an autonomous multicopter. *J. Intell. Robotics Syst.*, 2012. 10

[155] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 15, 20, 22, 25, 46

[156] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 10

[157] F. Schweiger, B. Zeisl, P. Georgel, G. Schroth, E. Steinbach, and N. Navab. Maximum detector response markers for sift and surf, 2009. 29, 30

[158] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. 47

[159] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. 9

[160] G. Sharma, S. ul Hussain, and F. Jurie. Local higher-order statistics (lhs) for texture categorization and facial analysis. In *ECCV*, 2012. 67

[161] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994. 36

[162] J. Shotton, A. Fitzgibbon, , A. Blake, A. Kipman, M. Finocchio, R. Moore, and T. Sharp. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011. 7, 9

[163] S. Siebert and J. Teizer. Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (uav) system. *Automation in Construction*, 41(0):1 – 14, 2014. 5, 83

[164] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *CVPR*, 2013. 67, 77

[165] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 2006. 8

[166] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *In Proc. ACM SIGGRAPH*, 2006. 46

[167] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *IJCV*, 2008. 65, 69

[168] X. Song and A. Myronenko. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. 17

[169] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3d cad data. In *Proc. BMVC*, pages 106.1–11, 2010. 9

[170] C. Strecha, T. Pylvänäinen, and P. Fua. Dynamic and scalable large scale image reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010. 8

[171] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, Oct 2012. 3, 10, 23

[172] G. R. Taylor, A. J. Chosak, and P. C. Brewer. Ovvv: Using virtual worlds to design and evaluate surveillance systems. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 9

[173] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. 3

[174] Y. Turkan, F. Bosché, C. Haas, and R. Haas. Automated progress tracking using 4d schedule and 3d sensing technologies. *Automation in Construction*, 22(0):414–421, 2012. 5, 83

[175] Y. Turkan, F. Bosché, C. Haas, and R. Haas. Toward automated earned value tracking using 3d imaging tools. *Journal of Construction Engineering and Management*, 2013. 5, 83

[176] Unity - game engine. https://unity3d.com/, October 2016. 11

[177] K. Valkealahti and E. Oja. Reduced multidimensional co-occurrence histograms in texture classification. *PAMI*, 1998. 66

[178] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *ECCV*, 2002. 64, 67, 73, 76

[179] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *CVPR*, 2003. 67

[180] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *IJCV*, 2005. 68, 73

[181] M. Varma and A. Zisserman. A statistical approach to material classification using image patch exemplars. *PAMI*, 2009. 5, 67

[182] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn. Differences between stereo and motion behaviour on synthetic and real-world stereo sequences. In *2008 23rd International Conference Image and Vision Computing New Zealand*, pages 1–6, Nov 2008. 7, 9

[183] D. Vázquez, A. M. López, J. Marín, D. Ponsa, and D. Gerónimo. Virtual and real world adaptation for pedestrian detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):797–809, April 2014. 9

[184] J. Wang and E. Olson. AprilTag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016. 3, 28, 30, 37, 41, 45, 55

[185] C. Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 127–134, June 2013. 8, 12, 15, 20, 46

[186] C. Wu. Visualsfm: A visual structure from motion system. http://ccwu.me/vsfm/, 2016. 84

[187] A. Xu and G. Dudek. Fourier tag: A smoothly degradable fiducial marker system with configurable payload capacity. In *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pages 40–47, May 2011. 30

[188] J. Xu, S. Ramos, D. Vázquez, and A. M. López. Hierarchical adaptive structural svm for domain adaptation. *International Journal of Computer Vision*, 2016. 9

[189] T. Yamada, T. Yairi, S. H. Bener, and K. Machida. A study on slam for indoor blimp with visual markers. In *ICCAS-SICE, 2009*, pages 647–652, Aug 2009. 27, 47

[190] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *IJCV*, 1998. 66