The Geometry and Photometry of Outdoor Webcams

by

Austin Daniel Abrams

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2014

St. Louis, Missouri
©2014, Austin Daniel Abrams
Contents

List of Figures vi
Acknowledgements ix
Abstract xi
Dissemination of Work xii

1 Introduction 1
  1.1 Contributions ............................................. 2
    1.1.1 Geometric Calibration of Outdoor Cameras ................. 2
    1.1.2 Outdoor Photometric Stereo .................................. 3
    1.1.3 Structure from Shadows ................................... 4
  1.2 Applications ............................................... 4
    1.2.1 In-depth phenological studies ............................. 5
    1.2.2 Geographic forensics ..................................... 5
    1.2.3 High-resolution outdoor modeling .......................... 5

2 Geographic Integration and Calibration of Webcams 7
  2.1 Related Work .................................................. 9
  2.2 A Scalable System for Participatory Webcam Geo-Integration .... 11
  2.3 Geocalibration Methods ..................................... 13
    2.3.1 Background ............................................... 13
2.3.2 Robust Geocalibration ........................................... 16
2.3.3 Geointegration of Pan-Tilt-Zoom Cameras .................... 19

2.4 Evaluation .......................................................... 24
  2.4.1 Robustness of $\ell_1$ calibration method ....................... 24
  2.4.2 Evaluation of Pan-Tilt-Zoom Calibration ...................... 26
  2.4.3 User Evaluation .................................................. 28

2.5 Integrating Imagery into Geographic Information Systems ........ 30
  2.5.1 Automatic Scene Completion .................................. 31
  2.5.2 Calibration-assisted 3D point locations ....................... 33
  2.5.3 Integrating Historical Imagery ................................ 34
  2.5.4 Visualizing PTZ geocalibrations ............................. 35
  2.5.5 Geographic camera control .................................... 36

2.6 Conclusions ....................................................... 39
  2.6.1 Follow-up Forensic Application ............................... 39

3 Outdoor Photometric Stereo ........................................... 41
  3.1 Background and Related Work ................................... 44
  3.2 Method ............................................................. 46
    3.2.1 Model ......................................................... 47
    3.2.2 Optimization .................................................. 49
    3.2.3 Shadow Estimation ............................................ 51
    3.2.4 Implementation ................................................. 53
  3.3 Results ............................................................ 54
    3.3.1 Evaluation ..................................................... 54
    3.3.2 Depth from Normal Field Integration ......................... 56
    3.3.3 Failure Cases .................................................. 58
  3.4 Summary .......................................................... 60
4 The Episolar Constraint

4.1 Related Work .................................................. 64
4.2 Episolar Geometry ............................................. 65
  4.2.1 The Episolar Line ....................................... 67
  4.2.2 Episolar Calibration .................................... 68
  4.2.3 Episolar Integration .................................... 70
4.3 Correspondence Generation .................................. 71
4.4 Results ...................................................... 74
4.5 Summary ..................................................... 77

5 Structure from Shadow Motion ................................. 81

5.1 Related Work .................................................. 83
5.2 Structure from Shadow Motion ............................... 84
  5.2.1 From Tracks to Structure ................................ 86
  5.2.2 Geometric Consistency .................................. 88
  5.2.3 Challenges and Limitations .............................. 89
5.3 Shadow Tracking ............................................... 90
  5.3.1 Shadow Detection and Description ...................... 91
  5.3.2 Frame-to-frame matching ................................. 91
  5.3.3 Linking together matches ............................... 94
  5.3.4 Track expansion .......................................... 95
  5.3.5 Incremental Reconstruction ............................. 97
  5.3.6 Implementation Details ................................. 97
5.4 Results ...................................................... 98
5.5 Conclusions .................................................. 99

6 Conclusions ..................................................... 103

6.1 Degeneracy in photometric stereo .......................... 104
6.2 Degeneracy in shadow-based reconstruction .......................... 106
6.3 Comparison of degeneracies ............................................. 106
6.4 Next Steps ................................................................. 108

References ................................................................. 109
# List of Figures

1.1 Organization of thesis ............................................. 3  
2.1 Data flow diagram of web-based calibration tool ................. 12  
2.2 Qualitative results for camera calibration ......................... 17  
2.3 Experiments to validate the robustness of camera calibration ................ 25  
2.4 Qualitative results for robust camera calibration ................ 27  
2.5 Quantitative evaluation of pan-tilt-zoom camera calibration ........... 28  
2.6 Statistics of novice users using the interface ..................... 30  
2.7 Calibration results from novice users ............................. 30  
2.8 Using known geometric calibration to automatically texture to Google Earth .......... 32  
2.9 Using a calibration-assisted tool to assist a user with 3D correspondence labeling .......... 33  
2.10 Incorporating historical imagery in a geographic interface ............ 35  
2.11 Visualization of pan-tilt-zoom camera calibration ................ 37  
2.12 Controlling a pan-tilt-zoom camera with a virtual 3D environment ........... 38  
2.13 A forensic application of image calibration ....................... 39  
3.1 Results gathered from outdoor photometric stereo for an example camera ........ 42  
3.2 Normal colormap and explanation of outdoor lighting direction diversity ........ 44  
3.3 Shadow detection for outdoor photometric stereo ................... 52  
3.4 Results of outdoor photometric stereo for a variety of cameras ............... 55  
3.5 Evaluation of surface normal estimates to Google Earth ............... 56  
3.6 3D reconstructions of outdoor scenes ............................. 57
3.7 Degeneracy in the presence of diverse data .................................. 58
3.8 Failure cases of outdoor photometric stereo .................................. 59

4.1 Overall pipeline for episolar constraint ........................................... 63
4.2 Example episolar lines ................................................................. 65
4.3 Visualizing the episolar constraint .................................................. 66
4.4 Calibrating a camera through episolar constraints .............................. 70
4.5 Visualizing a small portion of the shadow constraint graph .................. 72
4.6 Generating shadow correspondences .............................................. 73
4.7 Results from episolar constraint integration ..................................... 75
4.8 Experiments on a synthetic dataset ................................................ 76
4.9 Quantitative evaluation of recovered depth ....................................... 78
4.10 Exploring the connectedness of the constraint graph .......................... 79
4.11 Failure case of episolar constraint ................................................ 80

5.1 Results from shadow tracking reconstruction .................................... 82
5.2 Geometry of structure from shadow motion ...................................... 85
5.3 Using track connectivity to guide 3D reconstruction ............................ 87
5.4 Shadow tracking geometric consistency check ................................... 88
5.5 The aperture problem in shadow tracking ........................................ 89
5.6 Shadow detection in time-lapse sequences ........................................ 91
5.7 A visualization of the matching algorithm ....................................... 93
5.8 An evaluation of track linking strategies ......................................... 94
5.9 The effect of track expansion on reconstruction quality ........................ 96
5.10 Shadow tracking on a synthetic sequence ....................................... 98
5.11 Robustness of shadow tracking method with respect to unknown radiometric calibration ................................................................. 99
5.12 Shadow tracking results ............................................................... 100
5.13 Failure case in shadow tracking ........................................... 101

6.1 The condition number of the lighting matrix as a function of day-of-year and latitude. ............................................................. 105

6.2 The uncertainty in surface normal estimation from a single day ....... 107

6.3 Combinations of photometric and shadow-based cues cannot uniquely reconstruct surfaces .................................................... 108
Acknowledgements

First, I thank my advisor Robert Pless for his unending enthusiasm, imagination and guidance through my graduate studies. Robert’s discussions and brainstorms were always engaging, helpful, and fun, and I will always treasure my experience with Robert as a mentor and friend.

Throughout my stay at Washington University, I have deeply valued the advice from other faculty including Tao Ju, Kilian Weinberger, Yasutaka Furukawa, Jody O’Sullivan, and Viktor Gruev. I also had the pleasure of discussing my work with faculty from outside the university, including Kalyan Sunkavalli, Jean-Francois Lalonde, Hanspeter Pfister, Michael Goesele, and Noah Snavely; their valuable suggestions and advice were incredibly insightful.

The staff in the Computer Science and Engineering department were always helpful and understanding, and I know that this process would not be nearly as smooth without the continued assistance from Kelli Eckman, Myrna Harbison, Lauren Huffman, Sharon Matlock, Jayme Moehle, and Madeline Hawkins.

I also express gratitude toward Nathan Jacobs and Tom Erez, for being valuable mentors and role models in tackling problems and adjusting to the research lifestyle. I also thank my officemates Michael Dixon and Jim Tucek for their invaluable lunch discussions and clever hacks to actually get code to work. Thanks to Matt Kusner, for always being willing to talk through ideas and problems during our runs through Forest Park. Thanks especially to the members of the Media and Machines lab, including Stephen Schuh, Kylia Miskell, Agata Kargol, Adina Stoica, Abby Stylianou, Ian Schillebeeckx, Joshua Little, Dan Lazewatsky, and David Lu.
Thanks to my mother Susan, my brother Barry, my sister Anna, and my cat Riley for being a continued source of inspiration through the past five years. Last but not least, I am forever in debt to my wife Emily for loving me unconditionally through the stress and deadlines, and never losing hope.

-Austin Abrams

Washington University in St. Louis

May 2014

This work was supported through funding from NSF IIS1111398 “Analyzing Images Through Time”, NSF IIS0546383 “CAREER: Passive Vision, What Can Be Learned by a Stationary Observer”, and IARPA FA865012C7212 “Advanced Algorithms for Geo-Locating Found Imagery”.

x
ABSTRACT OF THE DISSERTATION

The Geometry and Photometry of Outdoor Webcams

by

Austin Daniel Abrams

Doctor of Philosophy in Computer Science
Washington University in St. Louis, 2014

Professor Robert Pless, Chair

Internet imagery has grown in size dramatically over the last decade. These images are ubiquitous, diverse, and useful for a variety of practical vision algorithms. Unfortunately, due to their unstructured nature, these images are largely uncalibrated. In order to use these images for applications in environmental monitoring and photo-forensics, the images should ideally be calibrated, to know where an image was taken, what nonlinear transformations the camera applies to the raw sensor readings, and the underlying 3D shape of the scene. We present methods to perform this camera calibration and 3D reconstruction for outdoor scenes from a single view over time. First, we give a web-based calibration tool to calibrate for the geometric and geographic context of a camera. Second, as the sun passes over an outdoor scene, surfaces will become brighter or darker depending on the local variation of the surface with respect to the sun. Therefore, intensity-based temporal cues help to uncover the subtle 3D surface variation of a scene. Finally, cast shadows from nearby buildings and trees provide projective-distorted cues for the 3D structures of both the shadow-casting object and the object under shadow. This cast shadow analysis is extended to analyze the motion patterns of shadows, which can be used to reconstruct objects the camera never directly saw.
Dissemination of Work

This thesis consists mostly of previously-published materials of co-authored works, each of which I was the first and primary contributor. The following lists the publications used. All code, experiments, and figures are my work, with the exception of the synthetic scene in Chapter 5, which was generated by Ian Schillebeeckx, and the experiment in the conclusion relating to the condition number for one-day photometric stereo, which was initially conceived by Kalyan Sunkavalli. In each work, co-authors helped with text.

- Chapter 2 follows “Web-accessible geographic integration and calibration of webcams”, an article published in the ACM Transactions on Multimedia Computing, Communications and Applications, and co-authored with Robert Pless [6]. That journal article itself is largely based on two conference publications. The first is “Participatory Integration of Live Webcams into GIS”, presented at COM.Geo and co-authored with Nick Fridrich, Nathan Jacobs, and Robert Pless [1]. The second is “Webcams in Context: Web Interfaces to Create Live 3D Environments”, presented at ACM Multimedia and co-authored with Robert Pless [5].

- Chapter 3 follows “Heliometric stereo: shape from sun position”, at the European Conference on Computer Vision 2012, and co-authored with Christopher Hawley and Robert Pless [3].

- Chapter 4 follows “The episolar constraint: monocular shape from shadow correspondence”, presented at the IEEE Conference on Computer Vision and Pattern Recognition 2013, and co-authored with Kylia Miskell and Robert Pless [4].
• Chapter 5 follows “Structure from shadow motion”, presented at the International Conference on Computational Photography 2014, and co-authored with Ian Schillebeeckx and Robert Pless [7].

Copyright notices

• Chapter 2: © ACM, 2013. This is the author’s version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ACM TOMCCAP, {9, 1, February 2013} http://doi.acm.org/10.1145/2422956.2422964.


• Chapter 4: © IEEE, 2013. The published version is available at http://dx.doi.org/10.1109/CVPR.2013.185.

• Chapter 5: © IEEE, 2014. The published version will soon be available on IEEE Xplore.
Chapter 1

Introduction

This is an exciting time to work in computer vision, in part due to the recent explosion of Internet imagery. Research groups across the world have studied this large source of imagery to develop fast object detection systems [47], build large-scale 3D models of public places [70], and to fuel image completion algorithms [30]. What makes these examples particularly exciting is that these methods are widely applied in practice, including face detection in cameras, image-based 3D models in Google Earth, and Adobe’s image completion algorithms in Photoshop.

This thesis largely uses unstructured Internet imagery to build models of the world by capturing how it changes. Change occurs temporally, as scenes change when the sun passes overhead and illuminates the world from a variety of angles, or as seasons pass and foliage changes color. Change also occurs spatially, from subtle surface variation, up to change across a wide network of distributed cameras. In all domains, it is important to calibrate this imagery. Then vision algorithms may interpret this change to understand the world’s true shape and color.

A significant portion of this work focuses on collecting and analyzing imagery from long-term time-lapse sequences. We have worked to build the largest dataset of webcam imagery, the Archive of Many Outdoor Scenes (AMOS), that actively logs imagery from more than 18,000 cameras every half hour (over half a billion images to date). This is compelling data.
set to build and share as a public resource because webcams are:

- **diverse** There are tens of thousands of webcams distributed across the globe, in urban and rural environments, by a variety of people for a range of purposes.

- **temporally coherent** Because photographers rarely set their cameras internal clock, the timestamps from arbitrary Internet images (e.g. from Flickr) can often be up to 24 hours off. In contrast, webcam imagery can be accurately timestamped at its time of capture. This timestamped information carries with it valuable metadata, such as current lighting and weather conditions.

- **free** These cameras are already pushing new imagery every second without any effort on our part.

### 1.1 Contributions

To support the research aim of calibrating unstructured webcam imagery, we have developed algorithms and tools to put the camera in its correct geographic context, solve for the local shape of a scene through its surface normals, and solve for the global structure of a scene through its shadows. Here, we give a brief overview of the problems and solutions addressed in this thesis, summarized in Figure 1.1.

#### 1.1.1 Geometric Calibration of Outdoor Cameras

The first step in understanding webcam imagery is to place cameras in a geographic context. We developed a user-in-the-loop web application to geometrically calibrate cameras (i.e. determine where a webcam is and where it is looking) from a small set of user inputs. From a handful of 3D-to-2D correspondences, the application recovers the location and orientation of the camera, which opens the door for a variety of new multimedia applications. For example, live webcam streams can be “pasted” into otherwise static texture so that motions of pixels
in the image become measurable in terms of meters when placed in its geometric context.

Also, we can direct dynamic pan-tilt-zoom cameras to rotate and focus on geographic areas of interest by dragging a marker around a virtual 3D scene.

1.1.2 Outdoor Photometric Stereo

Solving for the 3D shape of a scene is a core part of many vision problems. When the scene has a moving light source in an otherwise dark room, the classical photometric stereo problem can recover per-pixel surface variation of the objects under varying illumination. These surface normals provide a high-definition 3D reconstruction.

As the sun passes over an outdoor scene, different surfaces become brighter for different
sun positions, and we can perform photometric stereo on this dataset. However, creating
the first end-to-end approach for 3D scene reconstruction requires many forms of calibration
for this unstructured data. We need: accurate to-the-minute timestamps to recover the
sun position for each image, estimates of ambient lighting and complex shadowing, and the
camera’s exposure or color adjustments (which are usually not reported).

We developed an approach to automatically solve for these parameters consistent with an
outdoor lighting model, turning an image sequence into a 3D model with rich surface detail.
As a byproduct, this scene decomposition also offers measurements of the scenes unknown
color calibration, diffuse color, and exposure variation.

1.1.3 Structure from Shadows

Cast shadows are another powerful geometric cue for scene structure. By finding correspon-
dences between shadows and the objects that cast them, we show that we can treat the sun
as a second moving camera and recover 3D scene structure from a single view over time.
Additionally, we show that this method can even recover the structure of objects not directly
visible to the camera, from shadows cast from objects hidden behind other objects in the
scene.

This approach takes as input a webcam sequence, and returns a sparse scene-scale 3D
reconstruction. Therefore, this method is complementary to the methods in the surface
normal cues from previous section, which give a dense but local reconstruction. This is
compelling because shadows are typically a problem for photometric stereo, but carry useful
information about scene structure.

1.2 Applications

Relative to scientific and commercial applications, the work presented in this thesis is an
important intermediate step. The research we have done to calibrate and reconstruct scenes
from a single viewpoint would be useful for a variety of applications.

1.2.1 In-depth phenological studies

In the past, our lab has aided wide-scale phenological studies to show that the global network of webcams can be repurposed as an environmental monitoring tool [32], by taking measurements of trees as they grow and shed their leaves. However, these reports were initially restricted to simple measurements of the average “greenness” of a tree over time, a low-level cue for the onset of spring.

More in-depth analysis of outdoor scenes over time could give more valuable signals for wide-scale phenology, but require more reasoned outdoor analysis. Some tasks, like counting the number of trees in a scene, are made trivial with rich 3D information. This 3D information might give better insight into the size of a tree’s trunk, its branching structure, or species. Chapter 5 gives some example 3D reconstructions of trees that would support more advanced analysis than low-level image statistics.

1.2.2 Geographic forensics

In the next chapter, we present a user-in-the-loop tool to calibrate for the geographic context in which an image was taken. Making this tool easy to use is especially powerful, because it allows anyone to calibrate images, sometimes for unexpected purposes. In a recent forensic application[71], our lab used this interface to help the St. Louis Police Department locate a burial site for an unsolved case. This application will be addressed in more detail later.

1.2.3 High-resolution outdoor modeling

Recently, commercial mapping products like Google Earth, Bing Maps, and Apple Maps have taken advantage of large-scale aerial 3D reconstruction to provide a rough 3D estimate for outdoor geometry. These reconstructions are intentionally coarse, so that they can be represented with low-polygon models and render quickly on a desktop application.
If the aerial imagery comes from multiple passes of the environment at different times of year, or if the imagery were supplemented with street-level photography, there would be enough diverse lighting conditions to create a model of high-resolution surface detail, as in Chapter 3. Furthermore, since GPUs can “texture” coarse models with these fine-scale surface normal maps, these coarse models can be improved with little computational overhead. Augmenting coarse 3D environments with automatically-generated surface normal maps would provide a much richer 3D experience.
Chapter 2

Geographic Integration and Calibration of Webcams

“It’s useful to go out of this world and see it from the perspective of another one.”
— Terry Pratchett

Currently, the global network of webcams offers recent, unique viewpoints from tens of thousands of locations all over the globe. In the past, viewing webcam imagery with respect to its surrounding geographic context has yielded promising results, including large-scale environmental monitoring and estimating local weather conditions. Due to their widespread use and up-to-date imagery, webcams are emerging as useful resources for large-scale ecological studies and surveillance, especially within the context of geographic information systems (GIS). In this chapter, we describe our efforts to attach geospatial labels to these images in order to support further studies in using webcam images as worldwide sensors.

The world wide web has an increasingly large, diverse, and interesting set of sensors that provide live updates. While many projects have worked to integrate this sensing data within GIS (and to provide frameworks for querying and visualizing this data), they typically represent streaming video sources as “just another sensor”. However, to interpret camera
data automatically, it is often vital to know not just where the camera is, but also its orientation and camera zoom. Thus, this chapter seeks to address this problem within the context of organizing publicly-available webcams so that they can be coherently used as visual sensors.

In this chapter, we introduce novel camera geocalibration methods which uncover the imaging geometry with respect to the camera’s geospatial context. Geocalibration parameters (such as the camera’s latitude, longitude, altitude, orientation, and zoom level) are important for understanding how the camera fits into the environment, and camera calibration is often a necessary first step toward more in-depth computer vision applications; for example, later chapters of this thesis use these calibration measurements as necessary inputs to perform 3D reconstruction from natural lighting variation.

There are several challenges in effectively calibrating all cameras. Fully automated systems for geocalibrating and geo-orienting cameras \([35, 36, 50, 73]\) report geolocation accuracy to the scale of miles, and orientation accuracy to a scale of degrees, which is insufficient to map pixel data onto 3D models. Furthermore, current geo-orientation algorithms assume that the orientation is fixed throughout a long-term time lapse sequence of images, which may be invalid, especially when interpreting popular pan-tilt-zoom cameras. Also, webcams are emplaced by a large collection of different organizations, from private citizens to national parks to businesses of all sizes, and most cameras do not publish accurate calibration information. Even when a camera offers geolocation estimates, they usually reference objects in the scene rather than the camera itself, and to our knowledge, there does not exist a service which offers an estimate of camera altitude. Thus, we believe that in the foreseeable future, human assistance will be necessary to provide reliable geocalibration estimates.

We therefore seek to create tools which make it simple for arbitrary users on the web to calibrate the world’s cameras. We create a web application allowing anyone to calibrate cameras by specifying a few corresponding points, and to create a web based visualization infrastructure that uses the user’s web browser to embed live imagery into GIS. This elimi-
nates the need for any central organization to be responsible for computationally-expensive image manipulation, and makes the system scalable to the large number of cameras by not requiring a central server to touch the live video data.

When using input generated from arbitrary Internet users, we found that a small percentage of their input was arbitrarily wrong (e.g. placing a correspondence many miles away from the ground truth location). As we will show, even a single misplaced correspondence can ruin the prospect of a usable geocalibration. Therefore, we introduce a novel $\ell_1$-based calibration method that is robust to these outliers. We show that even under a large amount of gross errors, our method recovers the correct geocalibration parameters.

Pan-tilt-zoom (PTZ) cameras, where a user can control the camera remotely through a web interface, are emerging as a popular choice for live imagery. PTZ cameras may benefit the most from providing the surrounding 3D geospatial context, because their orientation and zoom can change both dramatically and rapidly. We discuss methods of solving for the dynamic calibration parameters from a small set of points clicked under different viewing directions and show our interface where a user can remotely move the physical camera and watch a virtual, textured camera frustum move to mimic the camera motion. We show that our calibration method works well, recovering the ground truth camera geolocation and geo-orientation, even in the presence of erroneous input.

We conclude by demonstrating several applications of integrating live webcam imagery into GIS, and how to augment these applications once the geocalibration of a camera is known. We discuss methods to embed live and historic imagery into 3D GIS, assisting users with calibration-specific constraints, visualizing live PTZ cameras under varying rotation and zoom levels, and remotely controlling PTZ cameras through a geographic interface.

### 2.1 Related Work

Creating systems to support the organization, querying, and data integration from distributed sensors with known geolocation has been a major focus of work over the last decade.
Classic works include IrisNet [22] and SenseWeb [40, 57], which offer frameworks to organize, collect, filter, and combine sensor feeds, to enable distributed queries with reasonable response times. However, these architectures leave a centralized data store responsible for collecting data, which makes it difficult for them to deal with large sets of video feeds.

There have been several other efforts integrating video into a 3D context. Most recently, Kim et al. [43] describe methods to add dynamic information into virtual earth environments, by displaying multiple video streams in a single context. They provide compelling results by merging several video streams into a view-dependent texture that minimizes artifacts of viewing a back-projected texture from extreme angles.

They also augment the virtual environment through object recognition results (e.g., animating a virtual pedestrian whose position is determined by a pedestrian detector) and in-depth image processing (e.g., estimating cloud cover and updating the virtual cloud models). As a first step to these high-level algorithms, they first calibrate their camera with respect to the surrounding geographic environment. In this chapter, we solve similar geometric computer vision problems, but with the goal of making the calibration process accessible to novice users through a scalable web interface.

Video Flashlights [65] provided the first instance of projecting textures onto a georeferenced space, which places several cameras from a network in a single 3D context, which Sebe et al. [66] extended by implementing a multi-camera tracking system for surveillance purposes, integrated within a geospatial context. Gliet et al. [23] create a GIS-mashup that maps weather data onto the sky part of a webcam image (but ignores the part of the scene below the horizon).

Sankaranarayanan and Davis [63] demonstrate a registration framework for a network of PTZ cameras. They use a spherical ‘fisheye’ panorama to register a PTZ camera to a common geographic space through an affine transformation, and use this geographic space to coordinate the network of cameras and track objects across the different cameras in the network. Although we use a different calibration approach, the motivating ideas and prob-
lem statements are similar. In this chapter, we show a PTZ registration and calibration framework for a single camera that relates the parameters of a movable camera in terms of familiar camera models from computer vision. Furthermore, we extend the work of Sankaranarayanan and Davis by incorporating geographic altitude and the effects of zooming the camera on the resulting image.

Kopf et al. [45] use an image annotated with the geolocation and orientation of the camera, as well as the 3D geographic coordinates of every pixel in the scene to dehaze, relight, and annotate images. In this chapter, we simplify the interface necessary to complete this type of registration and consider applications to live imagery.

2.2 A Scalable System for Participatory Webcam GeoinTEGRATION

While the geometric transformations involved in the mapping of imagery onto 3D geometry are largely well-understood [27], there remain important system architecture issues in building a system which scales to a large number of simultaneous users and accepts untrained volunteer input.

In this section, we describe our developments in implementing a web application for users to embed live imagery taken from webcam images into 3D virtual environments. Here, we discuss the constraints and difficulties of building this system as a web application, and how we resolved these issues to create a publicly-accessible scalable web service.

Creating a publicly-accessible system to map image data onto 3D geometry is not scalable because the central server would require both significant computational and network resources. The streaming image data must come to the central server, then each image must be warped, and the resulting warped imagery must be sent on to the client.

Our system architecture supports 2 modes of interaction: when a user is calibrating a system and when a user is viewing a system.
The first mode of interaction is the registration stage, where a user marks corresponding points between the 3D world and the 2D image. Towards the integration of webcams into a 3D geographic space, we require a user to find a webcam and click on a small set of correspondences between the image and the 3D GIS data (in our work, we use the Google Earth browser Plug-in). When the user is done marking up a scene, we are left with a set of correspondences between the geographic world (i.e., a latitude, longitude, and altitude) and the image. Figure 2.1(b)-(c) describes this interface.

The second mode is more challenging, as users may view a camera many times and many users may be viewing the scene simultaneously. If all data were to go through a central
server, that would become both a bandwidth and computation bottleneck. In our system, during the view of a live stream, the image data only goes to the client computer and all image manipulation is performed on the client side. Furthermore, this is implemented with JavaScript so that the client computers only needs a web browser. This allows a single central server to provide this service to a very large number of potential clients. Figure 2.1 shows the data flow diagram illustrating interactions between the user and the system for both the registration process and how a user views a scene.

2.3 Geocalibration Methods

These correspondences between a 3D scene and 2D image form constraints on the imaging geometry that took place when the image was captured. These constraints form the basis for camera calibration, a well-known geometric vision problem which aims to discover how the 3D scene projects onto the image. By solving for the camera calibration, we discover the camera’s position, orientation, focal length, and more. In this section, we discuss our methods for camera calibration, particularly with the goal of applying geographic reference to the calibration parameters.

We first describe the classical method to solve the calibration problem, and then extend it in a few ways. We provide a robust variant of camera calibration that can handle gross corruptions in the data, which may come from inexperienced or malicious users, and then describe a novel PTZ calibration model that accounts for variations in orientation and zoom.

2.3.1 Background

In solving for the 3D camera calibration we find the $3 \times 4$ projection matrix $M$ that defines a virtual linear camera such that if a world point $(X,Y,Z)$ in 3D projects to some image coordinates...
coordinate \((x, y)\), then

\[
\begin{bmatrix}
xw \\
yw \\
w
\end{bmatrix}
= M
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix},
\quad M =
\begin{bmatrix}
m_1 & m_2 & m_3 & m_4 \\
m_5 & m_6 & m_7 & m_8 \\
m_9 & m_{10} & m_{11} & 1
\end{bmatrix}
\] (2.1)

where \(w\) is a homogeneous scaling factor. Although only 6 correspondences are required to solve for \(M\), we require users to submit at least 12 for numerical stability. Given these correspondences, we then find the optimal \(M\) that minimizes the following least-squares problem:

\[
M^* = \arg\min_M \sum_i \left\| \begin{bmatrix} x_iw_i \\ y_iw_i \\ w_i \end{bmatrix} - M \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \right\|^2.
\] (2.2)

The above equation can be solved by the following equivalent system of linear equations:

\[
\begin{bmatrix}
X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -Z_1x_1 \\
0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_1 & -Y_1y_1 & -Z_1y_1 \\
& \vdots & & & & & & & & & \\
X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -Z_nx_n \\
0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -X_ny_n & -Y_ny_n & -Z_ny_n
\end{bmatrix}
\begin{bmatrix}
m_1 \\
m_2 \\
\vdots \\
m_{11}
\end{bmatrix}
= \begin{bmatrix}
x_1 \\
y_1 \\
\vdots \\
x_n \\
y_n
\end{bmatrix}
\] (2.3)

\[
M^* = \arg\min_M ||A\vec{M} - b||^2.
\] (2.4)

We then solve for Equation 2.4 using standard linear least squares.
Then, $M$ can be further decomposed into:

\[
M = K[R|T] = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} . & . & . & T_x \\ . & R & . & T_y \\ . & . & . & T_z \end{bmatrix}
\]  

(2.5)

where $K$ contains the intrinsic parameters of the camera, $R$ is the $3 \times 3$ camera rotation matrix, and $T$ is the 3-element camera translation vector. Here, $f_x$ and $f_y$ refer to the camera’s focal lengths, $x_0$ and $y_0$ give the image coordinates of the optical axis (usually in the center of the image), and $s$ is the skew of the camera. Once $M$ is known, we solve for $K$ and $R$ by decomposing the left-most $3 \times 3$ submatrix of $M$ using the $QR$ decomposition, which decomposes a matrix into an upper-triangular matrix and a rotation matrix. Finally, we solve for $T$ as:

\[
T = K^{-1} \begin{bmatrix} m_4 \\ m_8 \\ 1 \end{bmatrix}
\]  

(2.6)

Given this calibration we solve for the position and orientation of the camera in geographic coordinates. Then we find the center of the camera in world coordinates, given by $C = -R^\top T$. We then convert $C$ back to geographic coordinates (latitude, longitude, and altitude), and express the rotation matrix $R$ in terms of heading, tilt, and roll.

Once a scene is calibrated, we allow the user to move the virtual camera to match the parameters of the actual camera. Some example calibrated scenes are shown in Figure 2.2. This linear camera model is a commonly-used approximation of how many real-world cameras take images. However, it does not model barrel or pincushion distortion, which is present in some cameras. This simple model could therefore lead to inaccurate correspondences in some cases. We opted to keep the simpler camera model because it provides an effective approximation to a real camera, while keeping the required number of user-submitted correspondences to a minimum. This has not been a significant source of error.
As an implementation detail, we first convert the user’s latitude, longitude, and altitude into a Cartesian coordinate system by representing each point as meters east, north, and up of some arbitrarily-chosen origin point, and this new coordinate is passed into the calibration procedure as $(X, Y, Z)$. This assumes that the ground is locally linear around the origin point, and since the altitudes of the cameras we work with are well below the altitudes where we might experience distortions due to the curvature of the Earth, this is a safe assumption in practice.

### 2.3.2 Robust Geocalibration

In informal observation, we notice that novice users are typically experienced at dragging 2D points across an image, but generating high-quality 3D corresponding points remains a challenge (see Section 2.4.3 for our observations and attempts to alleviate these difficulties). Furthermore, the building geometry provided within Google Earth is itself provided by arbitrary Internet users. Although submissions to Google Earth are monitored, there are still a large quantity of buildings in Google Earth that do not accurately portray their real-life counterparts. Therefore, the set of correspondences generated with the web interface may be of relatively poor quality.

In response to this issue, we propose a calibration procedure that is robust in the presence of these kinds of outliers. The system of linear equations shown in Equation 2.1 can be solved using linear least squares, which is done in previous work [1, 5]. Although least squares is a popular tool for many data-fitting problems, it is notoriously fickle in the presence of outliers. A small percentage of outlying points can corrupt the entire calibration routine, even if the rest of the correspondences are accurate. This brittleness emerges from the squared error; any outlier will accrue a large error that dominates the rest of the optimization, and least squares solutions typically fit solutions that accrue smaller errors everywhere, rather than

---

1 At the time of this writing, the only 3D models in Google Earth were supplied by users. Although many of these models have since been replaced with aerial-based 3D reconstruction from images, the models are coarse and often incomplete.
Figure 2.2: A selection of webcams and the polygons users have added to scenes (left column), the view from the positioned and oriented camera in Google Earth (center column), and the calculated location and orientation of the camera in the context of the scene (right column). The views in the center column have not been altered from the default calibrated camera view.
large errors at a few sparse locations.

A common solution is to apply RANSAC, which attempts to isolate inliers and outliers during the fitting step. However, these methods work best when the number of data points far exceed the number of points required for any fitting step (in our case, six). An ideal solution would take advantage of all (potentially noisy) data points to minimize the constraints on how many points a user must submit.

The $\ell_1$ norm, or the sum of absolute values, is a robust estimator when we expect some portion of the data to be corrupt. Because the absolute error measures the magnitude of the error rather than the magnitude of the squared error, the resulting errors from outliers are on the same order of magnitude as the inliers. Therefore, $\ell_1$ methods support solutions that give large errors to a few data points, so long as the errors themselves are sparse. Furthermore, in contrast to RANSAC, we can incorporate all data points, yet maintain robustness properties.

We incorporate the $\ell_1$ penalty into the optimization in Equation 2.4:

$$M^* = \arg\min_M |A \text{vec}(M) - b|_{\ell_1}.$$  \hspace{1cm} (2.7)

By denoting $A_i$ and $b_i$ the $i$th row of $A$ and $b$, we define $t_i = |A_i \text{vec}(M) - b_i|$. Then, the optimization Equation 2.7 is equivalent to the following constrained optimization:

$$\arg\min_{M,t} \sum_{i=1}^{n} t_i \hspace{1cm} (2.8)$$

s. t. \hfill \begin{align*}
-t_i & \leq A_i \text{vec}(M) - b_i \\
A_i \text{vec}(M) - b_i & \leq t_i
\end{align*}

Finally, we express Equation 2.8 as a canonical linear program:

$$\arg\min_{M,t} \begin{bmatrix} 0_{11} \end{bmatrix}^\top \begin{bmatrix} \text{vec}(M) \\
t \end{bmatrix} \hspace{1cm} (2.9)$$

s. t. \hfill \begin{bmatrix}
A & -I_{2n} \\
-A & -I_{2n}
\end{bmatrix} \begin{bmatrix} \text{vec}(M) \\
t \end{bmatrix} \leq \begin{bmatrix} b \\
-b
\end{bmatrix}$$
Where $0_k$, $1_k$, and $I_k$ define the $k$-element 0 vector, 1 vector, and $k \times k$ identity matrix, respectively. This formulation is then solved using standard linear programming algorithms.

### 2.3.3 Geointegration of Pan-Tilt-Zoom Cameras

Here, we extend the results in the previous sections to handle calibration of popular pan-tilt-zoom (PTZ) cameras. These cameras are useful for surveillance applications, because they can view a large portion of its surrounding geographic region, but zoom in to focus on regions of interest.

With dynamic cameras, geographic context is especially important. Even if a PTZ camera has been placed near easily-recognizable landmarks, arbitrary users can easily rotate and zoom the camera into geographically uninteresting locations. Therefore, understanding the geographic position and orientation of any view in the camera’s range is critical in attempting to understand the geometric foundations of the scene.

Each camera used in this work advertises its internal pan, tilt, and zoom values through a web interface in terms of three integers, with upper and lower bounds\(^2\). The goal of the calibration step is therefore to apply geographic meaning to these numbers, by inferring the geographic orientation and focal lengths from an arbitrary (pan, tilt, zoom) triplet.

Here, we describe a camera model that responds to variations in rotation and focal length, and show how we calibrate such a camera given a set of correspondences. As before in Equation 2.1, we assume that some 3D point $(X_i, Y_i, Z_i)$ projects onto an image point $(x_i, y_i)$. However, since this matrix is dependent on the camera’s

---

\(^2\)In particular, we used the LiveApplet series of cameras, although our approach is appropriate to any camera for which the internal state is readily available.
rotation and zoom, we now represent this relationship as:

\[
\begin{bmatrix}
  x_i w \\
  y_i w \\
  w
\end{bmatrix}
= M(p_i, t_i, z_i)
\begin{bmatrix}
  X_i \\
  Y_i \\
  Z_i \\
  1
\end{bmatrix},
\]

(2.10)

where \(M(p_i, t_i, z_i)\) is a function of the pan, tilt, and zoom of the camera at the time the image was taken. Here, we assume that \(p_i\) and \(t_i\) are normalized between 0 and 1, and that \(z_i\) is between \(z_{min}\) and \(z_{max}\), the given bounds. In other words, \(z_{min}\) and \(z_{max}\) are the zoom values at its largest and smallest focal lengths (fully zoomed in and fully zoomed out), respectively. To be clear, we refer to the “zoom” \(z\) as the arbitrarily-scaled integer that the camera advertises, and the “focal length” \(f\) as the unit (in pixels) used in the camera’s calibration geometry.

Then, we can represent \(M(p_i, t_i, z_i)\) as a function of the matrices \(K(z_i)\) and \(R(p_i, t_i)\), the intrinsic camera matrix and the camera’s rotation matrix for correspondence \(i\):

\[
M_i(p_i, t_i, z_i) = K(z_i)[R_i(p_i, t_i)|T].
\]

(2.11)

Notice that, because the position of the camera never changes, the translation vector \(T\) is constant across all images.

Let \((x_0, y_0)\) be the principal point of the image (assumed to be half the width and height of the image), and let the focal length of the camera be a function of zoom, \(f(z_i)\). This focal function therefore maps otherwise arbitrary zoom values into real-world focal lengths. Then, we estimate the intrinsic matrix as

\[
K(z_i) = \begin{bmatrix}
  f(z_i) & 0 & x_0 \\
  0 & f(z_i) & y_0 \\
  0 & 0 & 1
\end{bmatrix}.
\]

(2.12)
Previous work [5] assumed that the focal function was linear and bounded by some unknown minimum and maximum focal lengths. Although the linear model works well for many zoom levels, we noted that there is a dramatic nonlinear change in focal length when the camera is almost fully zoomed in. Measuring the apparent size of several objects in the scene with respect to the zoom parameter suggests that, for the cameras we model, the focal function is an unknown scaling and translation of $f(z) = \frac{1}{z}$:

$$f(z_i) = \frac{\alpha}{z_i - \beta} + \gamma,$$  \hspace{1cm} (2.13)

where $\alpha, \beta,$ and $\gamma$ are unknowns.

However, due to the singularity when $z$ is close to $\beta$, a small change in parameters could have a large change in $f(z)$. Therefore, we instead parameterize this function in terms of the unknown focal lengths $f_a$, $f_b$, and $f_c$ at three key zoom levels: $f(z_{\text{min}}) = f_a$, $f(z_{\text{h}}) = f_b$, and $f(z_{\text{max}}) = f_c$, where $z_h = \frac{z_{\text{min}} + z_{\text{max}}}{2}$. Representing $f(z_i)$ in terms of these three values is much less sensitive to small changes in the parameter space, and each variable has a well-defined semantic meaning. Given any $f_a$, $f_b$, and $f_c$ triplet, we find the corresponding $\alpha$, $\beta$, and $\gamma$ parameters through the following closed-form solution:

$$d = z_h (f_c - f_a) + z_{\text{max}} (f_a - f_b) + z_{\text{min}} (f_b - f_c)$$  \hspace{1cm} (2.14)

$$\beta = \frac{z_h z_{\text{max}} (f_b - f_c) + z_h z_{\text{min}} (f_a - f_b) + z_{\text{max}} z_{\text{min}} (f_c - f_a)}{-d}$$  \hspace{1cm} (2.15)

$$\gamma = \frac{z_h (f_b f_c - f_a f_b) + z_{\text{max}} (f_a f_c - f_b f_c) + z_{\text{min}} (f_a f_b - f_c f_a)}{d}$$  \hspace{1cm} (2.16)

$$\alpha = f_b z_h - z_h \gamma - f_b \beta + \beta \gamma$$ \hspace{1cm} (2.17)

These $\alpha$, $\beta$, and $\gamma$ values then define the focal function as in Equation 2.13.

Then, we define the rotation of the camera as a function of the current pan $p_i$ and tilt $t_i$, as well as the rotational offsets $p_0$, $t_0$, $r_0$, which define the default pan, tilt, and roll of the
camera in radians (when \( p_i = t_i = 0 \)). Finally, we introduce two unknown scale parameters \( s_p \) and \( s_t \) which convert the \([0,1]\) scale for pan and tilt into radians (i.e., field of view of the camera over all pan and tilt angles).

\[
R(p_i, t_i) = \text{zxz}(-(s_p p_i + p_0), s_t t_i + t_0, r_0)
\]  

(2.18)

where \( \text{zxz}(u,v,w) \) is the standard \( 3 \times 3 \) rotation matrix that rotates \( u \) radians around the \( Z \) axis, \( v \) radians around the \( X \) axis, and \( w \) radians around the new \( Z \) axis. As \( u \) increases, the rotation moves counterclockwise around the \( Z \) axis. Therefore, we negate the \( u \) term in order to move the camera clockwise as \( p_i \) increases (i.e., as the camera pans from left to right). This rotation scheme is common among GIS applications.

The last calibration unknown is \( T \) as \( \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^\top \), the unknown but constant position of the camera.

For any PTZ camera, the focal function \( f \) (and, in our case, the values \( f_a, f_b, \) and \( f_c \)), the rotational offsets \( p_0, t_0, r_0 \), the rotational scales \( s_p, s_t \), and the position of the camera, \( T_x, T_y, T_z \) define a set of projective matrices. From this definition of the camera, we can find the projection matrix for any pan, tilt, or zoom value in the camera’s range.

Given a geocalibrated camera for which we know the projection matrix \( M \), we solve for the potential points \( Q \) that project onto \((x,y)\):

\[
\begin{bmatrix}
    xw - m_4 \\
    yw - m_8 \\
    w - 1
\end{bmatrix} = \begin{bmatrix}
    m_1 & m_2 & m_3 \\
    m_5 & m_6 & m_7 \\
    m_9 & m_{10} & m_{11}
\end{bmatrix} Q
\]  

(2.19)

where \( w \) is an arbitrary homogenous scaling factor.

Then, to compute the error of some assignment of the unknown values \( f, p_0, t_0, r_0, s_p, s_t, T_x, T_y, T_z \), we first compute the projection matrix \( M(p_i, t_i, z_i) \) as given by Equation 2.11, for any correspondence \( i \). Then, we find some ray \( \vec{Q}_i \) that goes from the camera center
through the ground truth image coordinate \((x_i, y_i)\) as in Equation 2.19. If the camera is placed correctly, the ray \(\vec{Q}_i\) should be exactly the ray \(\vec{R}_i\) that passes through the camera center and \((X_i, Y_i, Z_i)\). We normalize both \(\vec{Q}_i\) and \(\vec{R}_i\) and compute the angular difference error for correspondence \(i\) as follows:

\[
E_i = 1 - \vec{R}_i \cdot \vec{Q}_i.
\]

This error measures the angle between the two vectors, and we combine this error for all corresponding points to create the complete cost function, which we minimize over all our parameters:

\[
\arg\min_{f,p_0,t_0,r_0,s_p,s_t,T_x,T_y,T_z} \sum_{i=1}^{n} E_i w_i
\]

where \(w_i\) is a per-correspondence weight that more strictly enforces the angular error term for larger zooms:

\[
w_i = \frac{z_i - z_{max}}{z_{min} - z_{max}} + 0.1.
\]

Notice that this definition of error depends on \(M(p_i, t_i, z_i)\), and hence, every value that contributes to the position, orientation, and focal length of the camera. The total error for some prospective solution is then the sum of all \(E_i\).

Another way to measure the error of some assignment of the unknowns is to instead compute the reprojection error for each point. In our formulation, we found that the reprojection error results in a much more difficult optimization, because the reprojection error can easily cause unbounded error terms which dominate the optimization; if a point projects outside the image frame, it can accrue an arbitrarily large error. However, because we use an angular error, we effectively bound how much error any one point can contribute (i.e., each error term is between 0 and 2).

Given a set of PTZ correspondences \((x_i, y_i, X_i, Y_i, Z_i, p_i, t_i, z_i)\), we solve for the camera’s position, default orientation, and focal lengths using the Nelder-Mead simplex method [58]
with the angular difference error. We initialize the position of the camera as the mean of all 3D points, the rotational offsets \( p_0 = t_0 = r_0 = 0 \), the rotational scales \( s_p = 320^\circ \), \( s_t = 120^\circ \) (numbers advertised by the camera manufacturers), and the focal parameters as \( (f_a, f_b, f_c) = (12.5, 2.5, 1.25) \times \) times the width of the image.

After calibration, the set of unknowns defines a projective camera matrix \( M \) for any PTZ state \((p_i, t_i, z_i)\). In later sections, we show how this formulation allows seamless integration of live webcam imagery from PTZ cameras into geographic information systems.

### 2.4 Evaluation

In this section, we show quantitative and qualitative measures of accuracy for the proposed methods.

#### 2.4.1 Robustness of \( \ell_1 \) calibration method

Here we report on experiments that describe the sensitivity of the \( \ell_1 \) calibration method to corrupted points. In each of the experiments listed, we corrupt some percentage of the input correspondences and estimate calibration parameters with least squares and the proposed method. For these experiments, we corrupt a correspondence by adding uniform noise to its 3D location.

We first describe a quantitative measure to estimate the sensitivity of calibration methods to larger and more prevalent corruptions. In this section, we explore two different variables of corruption: the corruption magnitude (the amount of uniform noise we apply to the corrupted points’ 3D locations, in meters) and the corruption prevalence (the percentage of points which we randomly sample to corrupt). Furthermore, we use two metrics for calibration quality: the reprojection error and camera position error.

The reprojection error measures how well the computed projection deformation conforms to the set of (uncorrupted) input points; for any point \((X_i, Y_i, Z_i)\), we use Equation 2.1 to
Figure 2.3: Experiments to validate the robustness of the proposed calibration method. In each case, the proposed $\ell_1$ method is more robust to outliers, both in magnitude and quantity.

determine the location of the projected image point $(x'_i, y'_i)$. The reprojection error is thus the Euclidean distance between $(x'_i, y'_i)$ and $(x_i, y_i)$, the ground truth input point. Finally, for fair comparison across cameras of differing resolutions, we normalize the reprojection error by the maximum image dimension.

The camera position error measures how far the estimated camera location is from the hand-labeled ground truth camera location, in meters. Although camera position is not the only product of calibration, measuring error with respect to relative position provides a high-level interpretation for calibration quality.
Finally, for any set of corrupted points, we calibrate the camera using the proposed $\ell_1$ method and the least-squares method. For any particular corruption magnitude, corruption percent, error metric, and calibration method, we report the median error with respect to twelve scenes across five trials.

Figure 2.3 shows the results of this experiment. In all cases, the proposed method is more robust to outliers than the least squares method; even under a small amount of corruption, the least squares method returns erroneous results.

Another method to evaluate the quality of a calibration is through qualitative examination of the resulting camera geometry. Each geocalibration corresponds to a full 3D camera frustum. This representation of the camera shows the camera’s location, orientation, and focal length (i.e., its zoom level). When a camera is mis-calibrated, the resulting camera frustum is malformed or in some impossible configuration (such as inside a building or underneath the ground). Thus, examining the deformations to this frustum gives a rough estimate of how the various methods react in the presence of outliers.

Figure 2.4 shows some example calibration results before and after the addition of outliers. Qualitatively, the least squares calibration technique is fairly fickle in the presence of outliers, while $\ell_1$ methods give more robust results.

### 2.4.2 Evaluation of Pan-Tilt-Zoom Calibration

Here, we describe the quality and robustness of the PTZ calibration model with respect to varying amounts of corruption. Similar to the previous quantitative experiment, we progressively corrupt more and more correspondences by varying magnitudes, and report the average calibration error of the uncorrupted points (in this case, the angular error term used in PTZ optimization) and camera position error. Figure 2.5 shows our results. Although the PTZ model is not built explicitly for robustness, we found that the proposed method is still robust to large corruptions in the data, a proxy for real-world miscalibrations generated by novice or malicious users. For small amounts of corruption, the PTZ calibration method
<table>
<thead>
<tr>
<th>Example image</th>
<th>$l_2$, no outliers</th>
<th>$\ell_1$, no outliers</th>
<th>$l_2$, with outliers</th>
<th>$\ell_1$, with outliers</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="example1.png" alt="Image 1" /></td>
<td><img src="example2.png" alt="Image 2" /></td>
<td><img src="example3.png" alt="Image 3" /></td>
<td><img src="example4.png" alt="Image 4" /></td>
<td><img src="example5.png" alt="Image 5" /></td>
</tr>
<tr>
<td><img src="example6.png" alt="Image 6" /></td>
<td><img src="example7.png" alt="Image 7" /></td>
<td><img src="example8.png" alt="Image 8" /></td>
<td><img src="example9.png" alt="Image 9" /></td>
<td><img src="example10.png" alt="Image 10" /></td>
</tr>
<tr>
<td><img src="example11.png" alt="Image 11" /></td>
<td><img src="example12.png" alt="Image 12" /></td>
<td><img src="example13.png" alt="Image 13" /></td>
<td><img src="example14.png" alt="Image 14" /></td>
<td><img src="example15.png" alt="Image 15" /></td>
</tr>
<tr>
<td><img src="example16.png" alt="Image 16" /></td>
<td><img src="example17.png" alt="Image 17" /></td>
<td><img src="example18.png" alt="Image 18" /></td>
<td><img src="example19.png" alt="Image 19" /></td>
<td><img src="example20.png" alt="Image 20" /></td>
</tr>
</tbody>
</table>

Figure 2.4: Each row shows an example image of the camera, and the results from both the least squares approach ($l_2$) and the robust calibration method ($\ell_1$) in the absence and presence of outliers. The proposed algorithm consistently returns more robust calibrations than traditional methods. In the fourth row, the least squares approach placed the camera kilometers away from the ground truth location and is not shown.
Figure 2.5: Quantitative Evaluation of the proposed PTZ calibration algorithm in the presence of large corruptions.

reliably uncovers the ground truth camera position to within 10 meters, with an average angular error less than 1.3 degrees.

2.4.3 User Evaluation

Here, we briefly discuss our experiences in working with an arbitrary userbase, some common mistakes, and our efforts in improving our interface to avoid these errors.

If a user has a problem annotating a scene, it is usually due to inexperience in manipulating 3D points with a 2D mouse. Initially, we found that users would ignore the part of the interface dealing with 3D interaction, submitting “correspondences” between the image and the default location of the 3D points. We alleviated this problem by validating that a user had at least clicked on some of the points before submitting their correspondences.

In a similar vein, we found that in previous iterations of our interface, users would become confused as to what the next step was. We introduced an interactive tutorial for our system that explicitly tells a user what steps are required in order to successfully calibrate a camera. In these tutorials, certain areas of the interface are highlighted in order to focus the user’s gaze on the elements of interest.

When a user adds points to a brand new camera, the camera position is unknown and so
by default, we place the locations of the 3D points across the globe. In the first iteration of
the interface, we provided a “Center Points” button that would recenter the 3D points on
the current virtual camera view, so that once a user had moved the Google Earth camera
to the real-world camera’s location, he or she could relocate the positions of the 3D points
without having to manually drag them from their default location. However, we noted that
untrained users were unclear about the purpose of the 3D correspondences, and would not
even click the button. We changed the behavior so that the 3D points were always in the
center of the virtual camera’s view as the user geolocates the camera. This follow-the-user
behavior stops when a user moves the first point. After adding this element to the interface,
we found that users were more aware of the 3D aspect and had a stronger idea of the goal.

In practice, the user-provided registration process for is approachable to novice users. As
an informal test of the ability for users to use our interface, we took a convenience sample
of 4 novice users as they registered points to 5 scenes. We gave each user access to the
PTZ registration interface, where the camera had been previously geolocated (but the scene
was otherwise unlabeled). We gave each user a short tutorial on how to use the interface,
and asked each user to submit at least 35 registrations. During this time, we measured
several factors: How long the user spent from the end of the tutorial session until all 35
registrations were submitted (total time), how long the user spent during each registration
step (interface time), the number of times the user clicked on the interface, the final number
of correspondences added to the scene, and final calibration error. Figure 2.6 shows the
results of this informal study.

While monitoring the users, we noticed a few errors. In one test, the user admitted to
accidentally creating misregistrations by incorrectly placing several 3D markers on the wrong
side of a 3D building, which adversely affected the results of the calibration step. Some scenes
have little to no GIS-provided building geometry. In these cases, users sometimes mistakenly
registered the roof of a building in 2D to the footprint of the building in 3D. In the cases
where the user did not make these mistakes, the resulting calibration quality is good enough
<table>
<thead>
<tr>
<th>Scene ID</th>
<th>Total Time (minutes)</th>
<th>Interface Time</th>
<th>Clicks</th>
<th>Correspondences</th>
<th>Final Calibration Error (degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>49</td>
<td>32m 46s</td>
<td>435</td>
<td>36</td>
<td>1.9</td>
</tr>
<tr>
<td>2</td>
<td>47</td>
<td>33m 12s</td>
<td>509</td>
<td>35</td>
<td>8.7</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>28m 17s</td>
<td>432</td>
<td>35</td>
<td>3.2</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>20m 31s</td>
<td>267</td>
<td>37</td>
<td>1.2</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td>24m 06s</td>
<td>342</td>
<td>35</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Figure 2.6: Statistics of novice users using the interface.

Figure 2.7: A series of superimposed camera views and camera frusta generated entirely by correspondences from novice users. The user in (c) had mistakenly selected 3D points on the ground plane, and thus the calibration is incorrect.

to create superimposed images in a geographic setting, as in Figure 2.7.

2.5 Integrating Imagery into Geographic Information Systems

In this section, we describe some potential applications that take advantage of geocalibrated cameras.
2.5.1 Automatic Scene Completion

In order to texture the scene a static camera covers, we require the user to mark off sections of an image and register them to the 3D environment. However, once a camera is accurately calibrated, the transformation that maps the 3D structures and terrain in a scene to an image is known. If the geometry of the scene is also provided (in our case, through the buildings and ground altitude provided by Google Earth), then the mapping for the rest of the scene can be automatically completed.

To automatically generate a scene, we follow these steps.

1. Generate image annotated with 3D geotags for each pixel.
2. Triangulate set of georeferenced 3D points.
3. Remove triangles that cover depth discontinuities.
4. Texture the mesh using camera calibration.

Given the fully geocalibrated camera for which we know the projection matrix $M$, we can solve for the potential points $Q$ that project onto $(x, y)$ by Equation 2.19. From here, we use GIS-provided methods to find the intersection of this ray into the scene’s geometry. We repeat this process for every $(x, y)$ to generate a full depth map.

In most images of outdoor scenes, there can be large depth discontinuities between two adjacent pixels, where the geometry of the scene abruptly changes (such as the edge of a building). Although the pixels themselves are very close, the locations of their corresponding 3D points can be drastically different. A simple triangulation of the scene therefore results in triangles with very long edges, where a triangle crosses a depth discontinuity boundary. However, if the background of an image has a cohesive structure (such as a mountain), then it is expected for two adjacent pixels in an image to be somewhat far away, even if there is no depth discontinuity. In other words, the further away an object is, the 3D distance of two adjacent pixels on that object should inherently increase. To prune only the triangles with high depth discontinuity, we score as the length of its longest edge divided by the mean
Figure 2.8: Some results from automatically-generated scenes. Notice how the depth discontinuities in (a) are preserved, where the mesh is outlined with a violet border. (b) shows the Google Earth view before and after updating the texture with a webcam image from automatically-generated scene. If created by hand, the mesh in (b) would require tens of thousands of perfectly-aligned correspondences.

depth of its vertices, and cull away the triangles with large scores (see [1] for details).

Thus, through a web interface, a user can provide a small set of correspondences (at least twelve) between an image and the geographic scene from which the system will automatically perform camera calibration. Given known scene geometry (provided by GIS applications), this calibration provides the transformation required to retexture a large and complicated scene, such as the skyline in Figure 2.8(a). In the previous system, users would have to provide a large number of correspondences related to the complexity of the scene geometry. This is particularly useful for scenes in which the geometry of the scene is not planar. For example, Figure 2.8(b) shows a scene in which the side of a mountain range has been mapped with a live texture. If this scene were to have been created manually, the user would have
2.5.2 Calibration-assisted 3D point locations

As mentioned earlier, during informal observation of users, we have found that registering points in a 3D geographic space is challenging for novice users, who may be inexperienced with manipulating objects in a 3D environment. While a user may be used to moving 2D objects around the screen (as they must do when specifying correspondences on a webcam image), moving 3D points with 2D mouse controls is challenging.

Once the camera is calibrated, we allow users to take advantage of the geometry in the
scene to aid the 3D point manipulation process. As users drag a 2D point around the calibrated webcam image, the corresponding 3D point moves to its most likely 3D location, effectively eliminating user manipulation for the 3D case. This *calibration-assisted* interface solves both the issue of manipulating objects in a 3D environment and creating contextually-significant meshes in the foreground of webcam images.

For any calibrated camera, the relationship between 3D world coordinates and image coordinates is known. Any 2D image point \((x, y)\) in the image has an associated ray in space (defined by Equation 2.19), so we intersect that ray with the 3D scene geometry and place the corresponding 3D geographic point there. However, this assumes that the initial calibration has been done correctly and the underlying geometry of the scene is consistent with the webcam image. We allow the user to adjust the geolocations of the 3D points if either of those assumptions fail. Figure 2.9 illustrates how calibration-assisted registration offers an intuitive, accurate alternative for novice users to create live textured meshes of a scene.

### 2.5.3 Integrating Historical Imagery

In addition to viewing the current webcam stream in a 3D environment, we allow users to navigate historical webcam archives from AMOS to view the scene from weeks, months, or years ago.

To navigate a year’s worth of webcam imagery, we use a summary image visualization, which describes variations across a year’s worth of archived images [33]. Here, each pixel in the summary corresponds to a single image taken during that year, and as a user clicks on various parts of the summary image, the image corresponding to the click point is loaded and embedded into the 3D geometry. In this way, a user can quickly navigate historical webcam imagery. Figure 2.10 describes the historical interface.
Figure 2.10: Each pixel in the summary image (a) is the average color of one webcam image taken sometime during 2009 (or red when the image could not be downloaded at that time). Vertical changes in the image are due to variations over a single day, while horizontal changes are due to seasonal variations over the year. When the user clicks on the pixels circled in blue and red, the interface updates its textures to (b) and (c), respectively.

2.5.4 Visualizing PTZ geocalibrations

To texture the scene in the static case (as in Figure 2.8(b)), it is assumed that the orientation of the camera is known. In the PTZ case, the camera can change its rotation and zoom, which will make any texture coordinates no longer valid. Therefore, any attempt to texture the scene with static texture coordinates will be met with failure when the camera moves, since the texture will not be addressing the same geographic regions across different orientations.

For PTZ scenes, we opted to use a textured rectangular polygon that intersects the camera frustum as the camera rotates and zooms. If the GIS application’s virtual camera is placed in the same geolocation as the physical camera, then this gives the effect of the
live webcam image retexturing a large field of view in the scene. As the physical camera
drives and zooms, the virtual frustum changes accordingly, matching the webcam image
to a geospatial context. A similar approach was demo-ed by [10], which superimposes live
images from a mobile phone onto street-level maps applications (although their demo and
goregistration process has not been released to the public as of this writing). Figure 2.11
shows an example PTZ camera geointegrated into Google Earth, showing live updates of the
estimated view frusta, and the image texture shown in proper geographic context.

2.5.5 Geographic camera control

The proposed PTZ calibration method discussed takes as input a (pan, tilt, zoom) triplet,
advertised by the physical camera, and returns a geocalibration that reflects the camera’s cur-
tent geographic state. However, given a geocalibrated pan-tilt-zoom camera, it is feasible to
solve the inverse problem: given some geographic point of interest, provide a (pan,tilt,zoom)
state that places that point in the center of the resulting image. Such an interface would
be useful for surveillance applications where specifying a geographic point is more intuitive
than specifying the pan, tilt, and zoom parameters by hand.

Given some query point of interest \((X, Y, Z)\) and a desired zoom level \(z\), we solve for the
camera pan and tilt \((p, t)\) that minimizes the reprojection error from the query point to the
center of the resulting image. Defining \(x(p, t), y(p, t)\) as:

\[
\begin{bmatrix}
x(p, t)\\
y(p, t)\\
w
\end{bmatrix} = M(p, t, z)
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix},
\]

we solve for the best pan and tilt as:

\[
p^*, t^* = \arg\min_{p, t} (x(p, t) - x_0)^2 + (y(p, t) - y_0)^2,
\]
<table>
<thead>
<tr>
<th>Webcam Image</th>
<th>Overhead View with Frustum</th>
<th>Superimposed Image</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Webcam Image" /></td>
<td><img src="image2.jpg" alt="Overhead View with Frustum" /></td>
<td><img src="image3.jpg" alt="Superimposed Image" /></td>
</tr>
<tr>
<td><img src="image4.jpg" alt="Webcam Image" /></td>
<td><img src="image5.jpg" alt="Overhead View with Frustum" /></td>
<td><img src="image6.jpg" alt="Superimposed Image" /></td>
</tr>
<tr>
<td><img src="image7.jpg" alt="Webcam Image" /></td>
<td><img src="image8.jpg" alt="Overhead View with Frustum" /></td>
<td><img src="image9.jpg" alt="Superimposed Image" /></td>
</tr>
<tr>
<td><img src="image10.jpg" alt="Webcam Image" /></td>
<td><img src="image11.jpg" alt="Overhead View with Frustum" /></td>
<td><img src="image12.jpg" alt="Superimposed Image" /></td>
</tr>
</tbody>
</table>

Figure 2.11: A series of webcam images taken from a PTZ camera (left column), oriented camera frusta, shown in pink (center column), and the current webcam image superimposed on the virtual camera within Google Earth (right column). Through this interface, the camera frustum and superimposed image change as the physical camera rotates and zooms.
Figure 2.12: Examples of a geographically-aware PTZ interface. (Top row) A user drags the orange query point around the virtual 3D environment. We find the optimal pan and tilt values that orient the camera toward the query point and (bottom row) remotely control the physical camera to match those orientation parameters. In each example, the camera is controlled entirely by the manipulation of the orange point in Google Earth.

where \((x_0, y_0)\) is the center of the image. Since the number of variables is small and the objective function has a closed form, a simple grid search over \(p, t\) is efficient and accurate.

Finally, in the same way that we can query for the camera’s internal state, we can set the camera’s state through a set of web API calls, which remotely controls the physical camera to match the optimal pan and tilt values.

This simple optimization results in an interface where a user can drop a pin that defines an \((X, Y, Z)\) location, and the camera automatically rotates so that the query location is in the frame of the resulting image. So, a user only needs to manipulate the location of a 3D point to change the camera’s position. In our interface, we set the zoom level \(z\) to be proportional to distance from the camera center to the \((X, Y, Z)\) point, so that the camera zooms in on distant query points and zooms out on close points. Figure 2.12 shows an example of the interface.
Figure 2.13: Highlights from a recent forensic application of image calibration [71]. Although the gravesite for a missing Jane Doe has changed dramatically from thirty years ago (a) to today (b), a human-in-the-loop calibration approach is powerful enough to recover the correct geolocation of the grave, which the St. Louis Medical Examiners used to dig out the coffin (c).

2.6 Conclusions

In this chapter, we describe our efforts to create a publicly-accessible tool for arbitrary internet users to provide the correspondences necessary to geocalibrate a webcam. By providing geographic context to cameras we interpret their images more deeply and treat them as more than “just another sensor”, which has immediate applications to surveillance and environmental monitoring. Because we decentralize our server and push image processing tasks to the end user, we are able to maintain a scalable service with minimal computational overhead. We offer a robust geocalibration method that is robust in the presence of large errors caused by naive users, and a dynamic pan-tilt-zoom calibration method that can handle changes in camera orientation and zoom. These methods work well, even in the presence of large corruptions in the data. Finally, this work supports a variety of tools that take advantage of geographic reference, such as embedding live textures into 3D GIS and controlling a dynamic PTZ camera through geographic constraints.

2.6.1 Follow-up Forensic Application

Since this chapter’s initial publication [6], our lab made use of a similar calibration tool to aid in a St. Louis homicide investigation, which we briefly address here. For full details, we
direct the reader to the subsequent publication \[71\].

In 1983, a young girl was found decapitated in St. Louis. After an ineffective investigation, the young Jane Doe was put to rest in a grave which had since been lost; the headstone had been placed incorrectly, and maintenance had been neglected in the thirty years since.

However, during the funeral, a photographer at the St. Louis Globe Democrat took pictures of the gravesite and surrounding area. Although the gravesite has fallen into disrepair, many unique keypoints, such as trees and headstones, appear both in the 1983 imagery and the gravesite today. Therefore, our lab gathered a set of correspondences (this time using readings from a GPS rather than Google Earth geometry, which was not rich enough) and found the calibration most consistent with the older imagery.

Because these images were taken close to the gravesite, the resulting camera location from the calibration revealed the approximate location of the grave, which the St. Louis Medical Examiners used to successfully find the casket.
Chapter 3

Outdoor Photometric Stereo

“Here comes the sun
Here comes the sun, and I say
It’s all right”
— The Beatles, *Here Comes the Sun*

This chapter presents an approach to heliometric stereo — using the sun as a moving light source to recover surface normals of objects in an outdoor scene. This is a classic application of photometric stereo because the position of the sun is known very accurately, but made challenging because of variations in lighting and weather. Additionally, most long term imagery is captured by webcams that may not share geometric or radiometric calibration information. Thus, we explore what it would take to fully automate the solution to the photometric stereo problem for uncalibrated, outdoor cameras.

Our approach is to optimize the similarity between long term time-lapse imagery and an image formation model that includes the surface normal and albedo for each pixel in the scene, color and intensity of the ambient and direct lighting terms in each frame, and a shadow mask in each frame. This optimization results in estimates of all these parameters of the scene structure, albedo, lighting and camera properties. This method is unique in tractably computing explicit, geo-referenced surface normals from outdoor image time-lapses without any user interaction.
Figure 3.1: Given a sequence of geo-located, timestamped imagery taken over the span of several months (a), we recover a shadow mask for each image (b) (here shown as the mask for the leftmost original image), albedo (c), a dense normal map represented in East-North-Up Coordinates (d) (detail image shown in (e), and colormap described in Figure 3.2), a per-pixel contribution of ambient light (f) nonlinear camera response (g), and for each image, measures of light color and intensity (h) and ambient lighting (i). We show these two temporal variables (h) and (i) as a function of the lighting direction, viewed from above. Best viewed in color.
Alternative approaches without user interaction parameterize pixels with respect to time-series basis functions which have an unknown, non-linear relationship to surface normals [34], or cluster pixels groups that have approximately the same, but unknown surface normals [46]. Approaches that compute metric surface normals require user interaction to identify three pixel locations on orthogonal surfaces [73], and all these approaches result in surface normals in the coordinate system of the camera, not a geo-referenced coordinate system.

Thus, our approach is appropriate to deploy on very large webcam archives [34, 49] in order to extract quantitative remote sensing measurements without human input. Despite the allure of using already emplaced webcams for environmental monitoring, the few published uses of uncalibrated webcams are limited to simple color changes [32, 61]. We hope that our approach to automatically estimate scene shape, scene lighting and surface reflectance will support a larger range of environmental measurement uses of these webcam archives.

There are three major contributions of this chapter. First, we adapt the photometric stereo algorithm [74] to work for outdoor scenes by integrating a richer image formation model, and present a gradient descent approach with methods for initialization and regularization of this optimization. Second, we test this across a variety of types and scales of natural scenes and highlight the ability to capture very small scale surface structure. We report surface normals geo-referenced to an “East-North-Up” coordinate system, and we are the first to offer quantitative comparisons between our results and 3D geometry from Google Earth. These highlight both the accuracy of our results and limitations in the completeness and resolution of the ground truth. Third, we characterize performance under which this approach gives good results. This emphasizes the importance of using imagery from many months, because over the course of one day or one week, the solar path does not give sufficient constraints to recover surface normals (see Figure 3.2(b)).
Figure 3.2: (a) The colormap used for all normals in this chapter. Notice that the normals are represented in an absolute, East-North-Up reference frame, rather than with respect to the camera’s optical axis. (b) Input solar lighting directions for one camera, using progressively longer spans of time (1 month, 3 months, and 6 months, from left to right). The longer the input sequence, the richer the set of lighting directions.

3.1 Background and Related Work

The original constraints for photometric stereo [74] assumed multiple images of scenes with known lighting directions captured with a calibrated camera. While there is immense literature on variations of this problem, in this review we consider only those most relevant to our problem domain of long term, outdoor, webcam imagery. Specifically, our imagery is captured with known lighting directions because the primary light source is the sun whose direction can be computed given a timestamp and geolocation of the camera. However, lighting may vary due to atmospheric effects (such as the red hue of sunlight at sunset), and the camera may perform some unknown nonlinear radiometric distortion before publishing the image.
Papers that address uncalibrated photometric stereo have primarily concentrated on indoor scenes lit by unknown lights. One approach clusters pixels into groups with the same albedo and surface normal to provide constraints to solve for the unknown radiometric response and the generalized bas-relief ambiguity [69]. Another works with constant albedo objects and uses non-linear optimization to solve for the radiometric response, lighting directions, and surface normals [56]. Because these focus on indoor images (in a dark room), they have relatively noise-free images and no ambient light term.

There is less work on outdoor, uncalibrated photometric stereo. Shen and Tan [68] explored the solution for photometric stereo from unstructured internet imagery, combining multi-view stereo to register images from different viewpoints. Assuming the surface albedo and surface normals are constant at a scene location between images, an optimization was used in order to estimate the lighting parameters and therefore infer the weather of each image.

A related approach [9] focused on a photometric stereo variant for recovering surface orientation from webcam images. This method also exploited multi-view stereo to solve for scene geometry, which was then used in a photometric stereo by example approach, which transfers normals from locations with known surface normals to the rest of the scene. Our approach does not require or use such an example, and is applicable to time-lapse captured from a single location.

In [72], Sunkavalli et al. factor a time-lapse sequence of images taken from a single day into several meaningful components, including shadows, albedo, and a one-dimensional surface normal projection. We expand on this work by using a more flexible model that accounts for solar variations over the span of a year, which uncovers the full three-dimensional normal field. In addition, we incorporate the effects of nonlinear camera response.

Kim et al. [44] present an approach to take a series of images taken through the span of a day and use changing light to infer radiometric calibration and exposure values for each image. Using a PCA basis for camera response functions introduced by Grossberg and
Nayar [25], Kim et al. recover the response function that is most consistent with a Lambertian assumption. We take a similar approach toward solving the exposure and radiometric curves, but we simultaneously solve for surface normals in the process.

Recently, Ackermann et al. [8] describe an approach to perform photometric stereo on outdoor webcams. While they use a richer reflectance model to explicitly handle specularities, we offer a model which allows a tractable optimization, giving our algorithm a large speedup. We also perform a more rigorous evaluation, by comparing surface normals to Google Earth models and producing 3D shapes, and use a shadow estimation procedure appropriate for months of imagery.

Some works take advantage of fundamentally different depth cues for shape from a single view. Given a sequence of images taken under partly-cloudy conditions and an estimate of wind direction and speed, Jacobs et al. [31] provide an algorithm to extract shape based on cues derived from observing cloud shadows through time. While Jacobs et al. focus on time-lapse sequences on the scale of minutes or hours, we take advantage of the changing position of the sun over the span of many months. This increased timespan relaxes the assumption that the image was taken under specific meteorological conditions, and solves for surface normals rather than depth.

### 3.2 Method

This section presents our outdoor image formation model, and our methods for solving for the parameters of that model. The inputs to this model are timestamped images, and the direction to the sun computed from the timestamp and geolocation. Our algorithm extracts several components of our image formation model, including surface normals and photometric properties of the camera.
3.2.1 Model

The input consists of \( n \) images \( I_1, \ldots, I_t, \ldots, I_n \), each represented as a \( p \)-pixel vector in \([0, 255]^p\). Given the latitude and longitude of the camera, as well as the timestamps from each image, we use \([60]\) to determine the sun direction for each image \( L_1, \ldots, L_t, \ldots, L_n \), represented as a 3-vector in the East-North-Up coordinate frame. To keep consistent notation, temporally-indexed variables are annotated with the subscript \( t \) from 1 to \( n \), and spatially-indexed variables with the subscript \( x \) from 1 to \( p \). This convention is used for the rest of the thesis.

The Lambertian lighting model assumes that the intensity of a pixel \( x \) in image \( t \) is a function of the surface normal \( N_x \in \mathbb{R}^3 \), the albedo \( \rho_x \in \mathbb{R} \), and lighting direction \( L_t \in \mathbb{R}^3 \). More formally, the intensity of image \( t \) at pixel \( x \) is given by the linear model

\[
\rho_x L_t^\top N_x.
\]

Webcam imagery from real scenes is also affected by ambient lighting conditions, which we model in two ways. The first is a per-pixel contribution \( A_x \), sometimes denoted the skylight \([72]\), and related to ambient occlusion \([29]\). Physically, this allows for each pixel to receive a different amount of the surrounding environmental light, which can vary spatially due to local geometry (for example, we expect the skylight term to be large for open, flat rooftops, and small when for narrow crevices and urban canyons where most of the sky dome is occluded). Secondly, we follow \([72]\) and use a temporally-varying term \( a_t \) that modulates the intensity of ambient light over time; since the sky dome changes color and intensity over a long time span, so will the contribution of ambient light:

\[
\rho_x L_t^\top N_x + A_x a_t
\]  

Natural scenes also include both cast shadows, such as a tree shadow on a road, and attached shadows such as the side of a building that is not illuminated. This is modeled with a per-pixel-per-image shadow volume \( S_{t,x} \in [0, 1] \), that models how much light is received from
the sun at each time at each pixel.

\[ \rho_x L_t^\top N_x S_{t,x} + A_x a_t \] (3.2)

The reported pixel intensity of a camera also depends on the radiometric camera calibration. Webcams rarely publish results in RAW format, rarely include meta-data describing their radiometric response, and may not be accessible to allow radiometric calibration using images of known calibration objects. Thus, we must solve for, and include in our model, an unknown, monotonic, nonlinear photometric response function \( f : [0, 255] \to [0, 255] \), which we assume to be fixed through all the images in sequence, and we modulate each image by an exposure value \( e_i \in \mathbb{R} \):

\[ I_{t,x} = f(e_t(\rho_x L_t^\top N_x S_{t,x} + A_x a_t)) \] (3.3)

Here, we take advantage of the invertibility of \( f \) and rewrite the above as

\[ f^{-1}(I_{t,x}) = e_t(\rho_x L_t^\top N_x S_{t,x} + A_x a_t) \] (3.4)

To make the estimation of \( f \) tractable, we use the Empirical Model of Response introduced by Grossberg and Nayar [25]. This is a PCA basis for typical camera response functions, which models the nonlinear response function \( f^{-1} \) as a linear composition of nonlinear bases:

\[ f_v^{-1}(x) = f_0^{-1}(x) + \sum_{j=1}^{b} f_j^{-1}(x)v_j \] (3.5)

where \( f_0^{-1} \) is the mean inverse response curve, \( f_1^{-1}, \ldots, f_b^{-1} \) are the set of basis curves, \( v_j \) are the unknown, camera-specific coefficients that implicitly describe the shape of the nonlinear function, and \( b \) is the number of bases used (in all our experiments, we use \( b = 5 \)). We use \( f_v^{-1} \) as notation to define the radiometric response defined by a vector of coefficients \( v \).
Color

To incorporate color, we largely use the above model independently on each color channel. Allowing different exposures for each color channel can be interpreted as modeling the color and intensity of the sunlight for each image. The only exceptions are that we have a single normal $N_x$ for each pixel, and a single shadow volume $S_{t,x}$. Unless otherwise specified, we do not denote each color channel individually, and in a slight abuse of notation, treat Equation 3.4 as an equality over a three-element vector.

3.2.2 Optimization

Given a set of images and their lighting directions, we wish to extract each component of the lighting model. Our algorithm is a simple gradient-descent procedure that minimizes the following loss function:

$$\arg\min_{v,e,\rho,N,a,A} \frac{1}{pn} \sum_{x=1}^{p} \sum_{i=1}^{n} \left\| f_v^{-1}(I_{t,x}) - e_t(\rho_x L_t^\top N_x S_{t,x} + A_x a_t) \right\|^2.$$ (3.6)

Notice that we do not solve for the shadow volume $S_{t,x}$. Similar to Sunkavalli et al. [72], we first estimate a shadow volume and leave it fixed for the remainder of the optimization. See Section 3.2.3 for our shadow estimation approach.

To make the optimization slightly more linear, we directly optimize over an auxiliary variable $\hat{a}_t = e_t a_t$ instead of $a_t$:

$$\arg\min_{v,e,\rho,N,\hat{a},A} \frac{1}{pn} \sum_{x=1}^{p} \sum_{i=1}^{n} \left\| f_v^{-1}(I_{t,x}) - e_t \rho_x L_t^\top N_x S_{t,x} - A_x \hat{a}_t \right\|^2.$$ (3.7)

After optimization, we can find the original $a_t$ as $\frac{\hat{a}_t}{e_t}$.

Before discussing the details of how we perform the above nonconvex optimization, we first explore some formal ambiguities as presented so far. The first ambiguity is between the albedo $\rho$ and exposure $e_t$. We could replace $\rho$ with $2\rho$ and $e$ with $\frac{1}{2}e$ and achieve the exact
reconstruction. A similar ambiguity exists between $A$ and $\hat{a}$. This is not an ambiguity we can resolve directly from image data without knowing either exposure ratios between images or true measures of albedo $a$ priori. Instead, to ensure that we get a unique solution, we artificially fix $e$ and $\hat{a}$ to each have average values of 1 in each color channel (meaning that, over time, the average sunlight color is pure white).

A more complex ambiguity arises between the exposure and response function. As described by Grossberg and Nayar [24], nearly any method that jointly attempts to recover the camera response function $f$ and exposure $e$ is subject to the exponential ambiguity, where $f(x)^\gamma$ and $e^\gamma$ form an equivalent solution, for all $\gamma$.

Even more upsetting is that the introduction of this nonlinear response function allows for a trivial solution to the above optimization! Consider the response function $f^{-1}(x) = 0$ whenever $0 \leq x < 255$, and $f^{-1}(255) = 255$. This function is monotonic and maps from [0,255] to [0, 255], so it is technically a “valid” response. This response maps almost all input intensities to 0, so additionally setting $e = 0$ and $a = 0$ gives a zero-error solution, which is globally optimal, since the loss is nonnegative everywhere.

To alleviate both the exponential ambiguity and the trivial solution, we employ a smoothness regularization term on the response function that penalizes large changes across intensity bins:

$$R_v = \sum_{i=1}^{255} \left| f_v^{-1}(i) - f_v^{-1}(i-1) \right|^2,$$

which is expressed as a system of linear equations over the curve coefficients $v$.

The overall optimization is therefore

$$\text{argmin}_{v,e,\rho,N,\hat{a},A} \frac{1 - \lambda}{pm} \sum_{x=1}^{p} \sum_{i=1}^{n} \left| f_v^{-1}(I_{t,x}) - e_t \rho_x L^\top_t N_x S_{t,x} - A_x \hat{a}_t \right|^2 + \lambda \frac{255}{255} R_v,$$

Here, $\lambda$ is a regularization constant that defines the weight of satisfying the data term versus

\footnote{Here, assuming that $f$ maps to and from $[0, 1]$ rather than $[0,255]$.}

\footnote{Assuming no intensity is equal to 255, which in our case is true, because we artificially ignore any pixels with over-saturated intensities.}
the smoothness term.

To make this optimization tractable, we take an alternating minimization strategy, which minimizes the normals, albedos, and skylight in one step, then all other variables in another step, and repeats until convergence:

\[
\arg\min_{\rho, N, A} \frac{1}{pm} \sum_{x=1}^{p} \sum_{i=1}^{n} || f^{-1}(I_{t,x}) - e_t \rho_x L_t^T N_x S_{t,x} - A_x \hat{a}_t ||^2
\]

(3.10)

\[
\arg\min_{v, e, \hat{a}} \frac{1}{pm} \sum_{x=1}^{p} \sum_{i=1}^{n} || f^{-1}(I_{t,x}) - e_t \rho_x L_t^T N_x S_{t,x} - A_x \hat{a}_t ||^2 + \frac{\lambda}{255} R_v
\]

(3.11)

This formulation has substantially smaller computational overhead, because Equation 3.10 can be broken into \( p \) independent subproblems, one for each pixel. Using a PCA basis for \( f \) makes Equation 3.11 a linear problem, which is solved using the standard MATLAB solver.

Because \( N \) is a single unit vector over all color channels, we approximate a solution to Equation 3.10 as:

\[
\arg\min_{\text{gray}(\rho), N, \text{gray}(A)} \frac{1}{pm} \sum_{x=1}^{p} \sum_{i=1}^{n} || \text{gray}(f^{-1}(I_{t,x})) - \text{gray}(e_t) \text{gray}(\rho_x) L_t^T N_x S_{t,x} - \text{gray}(A_x) \text{gray}(\hat{a}_t) ||^2
\]

(3.12)

Where \( \text{gray}(x) \) returns the grayscale value of the RGB vector \( x \). Substituting \( \text{gray}(\rho_x) N_x \) by an arbitrary 3-vector \( w \) makes the above objective linear in the unknowns \( w \) and \( \text{gray}(A_x) \).

After solving the above optimization, we find \( N \) as the direction of \( w \). Then, fixing \( N \) gives a linear solution to solve for \( \rho \) and \( A \) for each color channel. Therefore, in practice, the true optimization alternates over three objectives: one to solve for the surface normal for each pixel, one to solve for the albedo and skylight, and another to solve for everything else. After each round of alternation, we scale \( e, \rho, A, \) and \( \hat{a} \) to fix the albedo-exposure ambiguity.

### 3.2.3 Shadow Estimation

To initialize the shadow volume, we initially tried the shadow estimation algorithm from Sunkavalli et al. [72], which uses a per-pixel threshold to define at which times that pixel is
in shadow. This threshold is defined as the bottom 10th percentile of intensity values ever seen by that pixel, multiplied by some constant (Sunkavalli et al. use 1.5).

While this approach works well for one day’s worth of video, over the span of a year, cameras change their exposure more dramatically as the sun moves around the sky; a shadowed pixel when the sun is highest in the sky may have the same intensity as a directly-lit pixel at the opposite time of year, and so there usually is not a single correct threshold.

We modify this thresholding technique by introducing an adaptive, per-pixel threshold which changes over time. For each pixel $x$, we use two centroids $s_x$ and $l_x$ that model the intensity of that pixel in shade and direct sunlight, respectively. We initialize $s_x$ and $l_x$ similarly to Sunkavalli et al., where $l_x$ is the top fifth percentile of image intensities at $x$, and $s_x$ is the bottom fifth percentile. Then, for each image $i$ in chronological order, we compute whether the original image pixel $I_{i,x}$ is closer to $s_x$ or $l_x$, set its corresponding pixel in the shadow volume, and update that threshold as $s_x \leftarrow 0.8s_x + 0.2I_{i,x}$ or $l_x \leftarrow 0.8l_x + 0.2I_{i,x}$. 

Figure 3.3: (a) An example image. The pixel trajectory for the pixel centered in the white box is shown in (b), along with three scalar multiples of the threshold generated by the shadow estimation method of Sunkavalli et al. [72]. The blue line is the threshold suggested in [72]. (c) The centroids used in our adaptive approach. (d)-(g) The resulting shadow images from each approach for the image in (a), where the color of the border indicates which thresholding technique was used.
In this way, as we loop over all times $i$, from 1 to $n$, the values of $l_x$ and $s_x$ rise and fall as needed. Figure 3.3 shows that this method produces better shadow volumes than Sunkavalli et al. when working over the span of many months. To alleviate noise in the shadow labeling, we relabel any “isolated” pixel (any lit pixel whose 8 neighbors are all shadowed is relabeled as shadowed, and vice versa).

### 3.2.4 Implementation

Although we can decompose our optimization into a sequence of convex steps, the overall optimization is nonconvex and is subject to initialization. We perform 10 iterations of alternating minimization. We initialize $e_t$ and $a_t$ as 1 for all $t$ and all color channels, and initialize $v$ so that $f^{-1}_v(x) = x$ (i.e. use the linear response function). The first step in the alternating minimization is to find $\rho, N$ and $A$, so no initialization is required for those variables.

In all of the experiments used in this chapter, we use $n = 100$ images. When selecting images, it is important to select from a wide range of lighting angles, but not to include any times of day when the sun is in view, producing lens flare effects in the image. Furthermore, we aim to select images that are the least overcast or hazy. We use the approach from [8], which assigns a score to each image to model how sunny the image is (based on aggregates on low-level image features like the average blueness of the sky and the average gradient magnitude of the pixels to be optimized), and uses that score to select a set of sunny images with diverse lighting conditions.

As an additional preprocessing step, we use a multi-scale variant of the alignment process described in [33] to remove small forms of camera jitter.

For all experiments, we manually mask off the sky, timestamps, streets, bodies of water, pathways, or other areas prone to transient objects. Optimization is performed only on the remaining pixels. We use $\lambda = 0.9$ in Equation 3.11. In each optimization, we ignore the effects of any saturated pixel, where $I_{t,x} < 5$ or $I_{t,x} > 250$ in any channel, since their
intensities are unreliable.

Runtime depends on the size of the image and number of pixels, but we report timing for the results in Figure 3.1. On a machine with a 2.53GHz Intel Core 2 Duo processor with 8GB of RAM, the entire inference process from loading imagery to completion takes 12 minutes on a sequence with 100 images and 161,162 non-sky pixels using our MATLAB implementation. We perform 10 iterations of alternating descent for each experiment. The main bottleneck is in solving for albedo and normals, which takes about four and a half minutes.

We emphasize that this data comprises 100 images captured over many months, so our algorithm is substantially faster than real-time.

3.3 Results

Figure 3.4 shows the results of our algorithm on real-world cameras. We reliably recover shadows, response functions, and surface normals for a variety of scenes from the AMOS dataset [34].

3.3.1 Evaluation

To evaluate our approach, we compare our surface normals to the normals from Google Earth models. Using the interface from the previous chapter, we geo-calibrated two webcams and generated normals from the surrounding Google Earth geometry.

Figure 3.5 shows a quantitative evaluation of our approach, by measuring the angular difference between our normals and the normals from coarse Google Earth models. These histograms show that many locations in the image have substantial angular error. However, the thresholded images in Figure 3.5(d) help parse this error. When the true geometry is free of windowsills, trees, and other small-scale surface deviations, our recovered normals and the Google Earth models match closely, typically within 5-20 degrees.
Figure 3.4: Results on a variety of cameras. From top to bottom, we show an example image, the recovered shadow mask for that image, nonlinear camera response curves, albedo, skylight, and surface normals. Notice that the shadow mask accurately captures large-scale shadows as well as small-scale details. We reliably recover physically-meaningful, nonlinear response curves, yet our model is flexible enough to allow variety of real-world responses. Because we allow each pixel a unique normal, we produce high-fidelity normals that capture the tiny changes in surface orientation due to windows and the detailed geometry of trees.
Figure 3.5: Comparing our results to the models from Google Earth. (a) Our recovered surface normals. (b) The surface normals from Google Earth. (c) A histogram of the angular difference, in degrees, between our model and the Google model. Although initially these errors appear substantial, the majority of errors come from the coarse nature of the ground truth: (d) shows the pixels with less than 20 degrees of error. The majority of “errors” come from areas where Google Earth models do not capture subtle surface variation like foliage and windowsills.

Even still, there are some especially poor normal estimates where we would expect the ground truth model would match closely in each of these scenes, which we briefly investigate. In the top row of Figure 3.5, many of the rooftops have poorly-estimated normals; these rooftops actually have curved shingles, which our algorithm reconstructs, but are not modeled in the ground truth. In the bottom row, there is a large planar patch in the bottom-left corner that we do not model accurately. In this case, a few images in the sequence came during or just after a storm, so the ground is wet (and therefore no longer Lambertian). Recovering these poor normals is an attempt of our model to fit a diffuse model to a non-diffuse material.

3.3.2 Depth from Normal Field Integration

To integrate together many surface normals into rich 3D models, we first calibrate the camera using either the interface in the previous chapter, or use a calibration-from-shadows method derived in the next chapter. This gives us a pixel-ray vector $r_x$ for each pixel $x$. This
Figure 3.6: 3D reconstructions of objects from webcams in the wild. In each example, we show an original image, and two novel views of the reconstructed object. The 3D reconstructions are shown with a shiny gray material.

Pixel-ray is the direction of the line of 3D points which would project onto the image at \( \mathbf{x} \), expressed in East-North-Up coordinates.

From these calibrated pixel rays, we solve for the depth \( d \) most consistent with the recovered surface normals through a projective Poisson equation:

\[
\arg\min_d \sum_{\mathbf{x}} \sum_{\mathbf{x}' \in \Omega_{\mathbf{x}}} ||N_{\mathbf{x}}^T (r_{\mathbf{x}}d_{\mathbf{x}} - r_{\mathbf{x}'}d_{\mathbf{x}'})||^2
\]

(3.13)

where \( \Omega_{\mathbf{x}} \) is the set of neighboring pixels to \( \mathbf{x} \), and the optimization is performed with the constraint that \( d_{\mathbf{x}} \geq 1 \), to avoid the trivial solution \( d_{\mathbf{x}} = 0 \). Since this integration process is only valid for regions in the image free of depth discontinuities, we manually select integrable regions in the image. Example results are shown in Figure 3.6.
Figure 3.7: The normal maps recovered from our algorithm as the duration of original imagery ranges from 1 week to 2 months. The top row shows the recovered surface normals, and the bottom row shows the set of lighting directions used to generate those results.

3.3.3 Failure Cases

As discussed briefly in the introduction, photometric stereo requires lighting to come from a diverse set of directions. To evaluate its importance, we ran our algorithm on one camera and progressively increased the duration of the input sequence from one week to a few months, starting in August 2011. Figure 3.7 shows that our algorithm requires at least a few months of data before lighting conditions from the passing sun are diverse enough for normal estimates begin to stabilize.

This degeneracy is dependent on both the latitude of the camera and the time of year. For example, a sequence taken in the Northern hemisphere near August has a less-degenerate set of lighting directions than a sequence taken during January, because the sun rises more in the summer than in the winter. In an extreme example during polar summers, the sun circles around the sky, which is in some sense ideal for outdoor photometric stereo. Also, with a shorter sequence in general, there is a greater chance that a pixel will be under shadow the entire day. Therefore, the best results come from months-long sequences. This phenomenon
Figure 3.8: Some failure cases for outdoor photometric stereo. When the scene has non-diffuse materials with non-static geometry (like grass and water) (a), the resulting normals (which should appear as white in the false-color map) are inaccurate (b). When the camera makes wide sweeping rotations that cannot be resolved with jitter correction (two images shown in (c)), the entire algorithm fails, as indicated by the messy recovered surface normals (d).

will be addressed in more detail in the conclusion of the thesis.

Figure 3.8 shows some other common failure cases of our algorithm. The most common error is when the Lambertian image formation model is broken; grass is especially difficult to reconstruct because it is shiny (and therefore non-Lambertian), sways in the wind (and therefore has nonstatic geometry over the span of a day), and slowly grows (and therefore has nonstatic geometry over long periods of time). Similarly, there is no way to recover the geometry of water’s surface, since it changes over time and is far from Lambertian.

Another source of error comes from pixels that are under shadow for the entire image sequence; it is impossible to recover the true surface normal from sunlight-based cues for these pixels. These forever-shadowed pixels will have some amount of intensity variation through time due to exposure changes and noisy sensors, so the shadow estimation procedure will
incorrectly label a few of the brighter intensities as being directly lit. This results in the algorithm attempting to fit a sunlight-driven photometric model to these pixels.

Finally, if the image sequence has been poorly-aligned, any attempts for photometric stereo are flawed from the start. The jitter-correction preprocessing step helps to alleviate small amounts of misalignment, but cannot handle large, sweeping camera rotations.

### 3.4 Summary

We have presented an image formation model rich enough to capture variations in outdoor webcam imagery over long time periods. We acknowledge that there are many additional components that affect outdoor imagery, many of which we could add within our formation model and optimization.

One major step toward practical outdoor photometric stereo is in better modeling of a non-Lambertian world. This is addressed somewhat in Ackermann et al. [8], but only allows mixtures of a small set of shiny basis materials, which are optimized on a scene-by-scene basis. One potential system might include a large but fixed set of common real-world non-Lambertian materials like metal and wet concrete. Fitting data to a model of fixed real-world materials allows more flexibility in recovering complex materials, but regularizes that complexity with real-world constraints.

If the set of images contains cloudy or rainy images, the algorithm makes mistakes in trying to fit the sunny-day model to non-sunny images. This motivates the use of the preprocessing step from [8] to automatically select good images to use. However, this method can fail if the camera does not see much of the sky. Perhaps a better approach would be to apply a robust estimation technique [15, 76] that handles outliers better than the least-squares error used in this chapter. This would also handle cases where temporary objects, like pedestrians and cars, briefly occlude the reconstructed object.

To facilitate future comparative studies, our data, code, and ground truth are available at research.engineering.wustl.edu/~abramsa/heliometric.
The next two chapters of this thesis explore two complementary ways to reconstruct a scene from a passive, single-view perspective over time, both using shadow based cues. These shadow cues are less sensitive to the effects of nonlinear response and allow a more general form of 3D geometry, but only give a sparse reconstruction.
Chapter 4

The Episolar Constraint

“The shadow can be more important than the substance.”

— Jean Rhys, Quartet

As demonstrated in the previous chapter, a pixel under shadow has a dramatically different intensity than the same pixel under direct lighting. Vision applications often incorporate shadows into their models, either by treating them as noise to be detected and ignored [26, 76], exploiting them as cues for camera calibration [17, 39], or incorporating them into larger image formation models, as in the previous chapter.

In this chapter, we treat shadows as a strong geometric cue: if a pixel is under shadow, then it must be the case that some other object along the lighting direction is casting a shadow onto it. For outdoor imagery, a geolocated camera and accurate timestamps cause this colinearity to have a known georeferenced direction. If the camera also has known geometric calibration, we can express this property as a linear constraint over the depth of each pixel involved. From this geometry, we derive three novel results:

- An image-space constraint between a shadow and its occluder,

- An approach to geometrically calibrate a camera from shadow correspondences, and
Figure 4.1: In this chapter, we exploit the inherent structure of cast shadows to recover shape from a single view. Given a time-lapse sequence from a geographically-calibrated camera (a), we create correspondences (shown as a yellow line) between a shadow (blue) and its occluding object (red) (b). Repeated across the image (c) and across many lighting directions, these tens of thousands of correspondences can be used as a cue to recover a sparse depth map from a single viewpoint (d). Depth increases from blue to red.

- A convex optimization to solve for the unknown depths for a sparse set of pixels from shadow correspondence.

Inferring depth from shadow correspondences has several desirable properties over other monocular cues for depth. First, shadow correspondences capture general shape: we do not require the ground to be planar or even visible, nor do we require the depth surface to be smooth or continuous. Next, since we work directly with binary shadow masks, rather than intensities, we do not need to account for real-world photometric distortions such as variable exposure and radiometric response, so long as the shadow extraction pipeline is sufficiently robust. Finally, we derive constraints which do not suffer from the aperture problem commonly found in other correspondence problems.
4.1 Related Work

A large body of work focuses on recovering shape from shadow-based cues. Early work focused on interpreting shadows from line drawings: Shafer and Kanade [67] introduced a general theory for describing the orientation of surfaces by the shadows they cast onto each other. Lowe and Binford [53] build a reasoning system to infer structure from line drawings, where one cue leverages manually-specified correspondences between a shadow and its caster.

Most shape-from-shadows approaches find depth surfaces consistent with some shadow-or-not labeling across many images. Early work by Hatzitheodorou and Kender [28] introduces an approach to recover the shape of a one-dimensional surface slice from the shadows it casts on itself, extended by Raviv et al. [59] to work with 2D surfaces.

In a recent application to photo forensics, Kee et al. [42] made use of shadow correspondence to expose photos with inconsistent shadows as being manipulated, in an interactive application which highlights small sets of conflicting shadows.

Later, Savarese et al. [64] leveraged epipolar geometry to carve out a surface from shadow labels across multiple views (see [48] for a survey on space carving). Although we work in the single-viewpoint scenario, we borrow concepts from epipolar geometry in a similar way, by treating the light source as a secondary camera.

Shadowgrams [20], shadow graphs [77], and shadow/antishadow constraints [19] all encode a constraint similar to the one presented in this chapter: all pixels on the image-space line between a shadow and its occluder should have a height below the corresponding 3D line. In contrast, we do not place any constraint on the intermediate pixels between a shadow and its occluder, which removes the assumption that the depth surface is terrain-like. Also, these works assume an orthographic camera, whereas we work with pinhole cameras.

Kawasaki and Furukawa [41] treat shape-from-shadows as a kind of structured light, where a wand is waved in front of the light source, and recover depth by constraining that the group of pixels shaded by the wand in any particular frame are coplanar in 3D. In this work, we do not place assumptions on the shape of the object that casts shadows in each
frame.

Recently, Bamber et al. [12] implement a single-view shadow carving algorithm suitable for long-term time-lapses. However, they make the assumption that the ground plane is large and visible. In contrast, we work with scene where the geometry is unknown a priori and the ground plane may not be visible.

4.2 Episolar Geometry

In this section, we derive the geometric constraints between a shaded pixel and the pixel that cast its shadow. The geometry of this constraint is equivalent to considering the sun as an orthographic camera.
Figure 4.3: Visualizing the episolar constraint. Where could the red point in (a) cast a shadow in the scene? This point must lie on the plane spanned by the 3D pixel ray for the red point and the 3D lighting direction, shown in yellow (b). This solar plane intersects the image plane, defining the episolar line (c). Finding the correct shadow correspondence therefore constrains the relative depth of each point.

We denote pixels as boldface vectors \( \mathbf{x}, \mathbf{y} \in \mathbb{R}^2 \). We assume that we know the lighting direction \( \mathbf{L}_t \in \mathbb{R}^3 \) at each time \( t \), which can be recovered using a solar position algorithm, given accurate timestamps and GPS [60]. For this work, we assume the camera is centered at the origin and has been geo-calibrated, giving each pixel’s ray into space \( \mathbf{r}_x \in \mathbb{R}^3 \). For clarity, we assume that the lighting direction and each pixel ray is a unit direction vector \( \|\mathbf{L}_t\| = \|\mathbf{r}_x\| = 1 \).

The goal is to find the per-pixel depth \( d_x \). Throughout the chapter, we treat \( \mathbf{y} \) as the object that casts the shadow, and \( \mathbf{x} \) as the object that receives the shadow. In an abuse of terminology, the phrase “\( \mathbf{y} \) casts a shadow onto \( \mathbf{x} \)” should be interpreted as “the 3D object that projects onto the image at \( \mathbf{y} \) casts a shadow onto the 3D object that projects onto the image at \( \mathbf{x} \)”.

For consistency, in all figures in this chapter, \( \mathbf{y} \) and \( \mathbf{x} \) are represented as a red and blue points, respectively.

Suppose that some pixel \( \mathbf{y} \) casts a shadow onto some other pixel \( \mathbf{x} \) for some lighting direction \( \mathbf{L}_t \); we denote such a correspondence as \( \mathbf{y} \sim_t \mathbf{x} \). Assuming directional lighting, this correspondence emplaces a constraint on the depths \( d \) of pixels at \( \mathbf{x} \) and \( \mathbf{y} \):

\[
\mathbf{r}_x d_x + \mathbf{L}_t \alpha_{xy} = \mathbf{r}_y d_y,
\]

(4.1)
where $\alpha_{xy}$ is the unknown 3D distance between pixels $x$ and $y$. This constraint takes the form of a linear constraint involving the unknown depth of each pixel and the 3D distance between $x$ and $y$. This property holds a close relationship with well-known epipolar geometry, so we denote Equation 4.1 as the *episolar constraint*. See Figure 4.3 for a visualization of the episolar constraint.

Notice that this property is true for all types of geometry. Nowhere do we make the assumption that our scene has a substantial ground plane, or that the depth surface is smooth or continuous.

We take advantage of this linear relationship in three distinct ways. First, this property defines an image-space constraint between an object and its shadow. This reduces the search space to 1D when determining shadow correspondences. Second, we derive a nonlinear optimization to geometrically calibrate a camera from shadow correspondences. In contrast to previous work, this calibration does not place any assumption on the underlying geometry nor require that the camera sees the sky. Finally, given correspondences from a variety of lighting directions, we derive a convex optimization procedure which recovers the depths of all pixels involved.

### 4.2.1 The Episolar Line

Generating correspondences between a shadow $x$ and its occluder $y$ is a challenging problem, but Equation 4.1 sheds some light on the shadow correspondence problem. If $y$ casts a shadow onto some unknown location $x$, then the point $r_x d_x$ must lie in the linear subspace spanned by $r_y$ and $L_t$. This linear subspace corresponds to a plane in 3D\(^1\) which intersects the image as a line passing through $y$. Therefore, if a pixel $y$ casts a shadow, then its corresponding pixel $x$ must lie on this *episolar line*.

Although this constraint alone does not dictate *where* on the episolar line the shadow truly comes from, it dramatically reduces the search space necessary for shadow correspondence.

---

\(^1\)A similar plane forms the basis for much of the work in the shadow carving approach presented in [64].
In Section 4.3 we describe how to complete this correspondence by taking advantage of self-shading priors.

Of practical interest is that because the correspondence search is limited to be along a line, this does not suffer from the common aperture problem seen in other correspondence problems. For example, linking a roofline to its horizontal shadow would be ambiguous without using this constraint; any point on the roof could conceivably produce a shadow anywhere on the shadow edge. However, this horizontal shadow will cross the episolar line at exactly one point, disambiguating the aperture problem. Figure 4.7 has several examples of this behavior.

This is especially useful in generating a dense network of constraints. If we could only generate correspondences on shadow corners, the constraint set might not be dense enough to use reliably. However, since we can create correspondences across shadow edges, our overall correspondence set is much more informative of the underlying geometry. In Section 4.4, we explore the connectedness properties of real scenes and show that, provided there are enough images, the resulting constraints form large connected components across the image.

### 4.2.2 Episolar Calibration

Notice that in order to generate the episolar line, we need estimates of the camera’s calibration to determine pixel rays $r$ in the same coordinate frame of $L$ (in our case, the East-North-Up space). However, estimating the geometric calibration of an outdoor camera is nontrivial. Various approaches exist for calibration from outdoor cues such as sky color [51] or shadow trajectories cast onto the ground plane [17, 39]. Webcams “in the wild” often do not have these features, as the sky might occupy only a small portion of the image, and the ground might not be planar or visible.

However, cast shadows are abundant in most outdoor scenes. Here, we leverage user-supplied shadow correspondences to calibrate a camera. Through the episolar constraint, we find the camera’s geometry calibration parameters $\theta = (\text{pan}, \text{tilt}, \text{roll}, \text{focal length})$
that define a pinhole camera which produces episolar lines most consistent with the given correspondences.

More formally, if a user supplies a set of ground truth shadow correspondences \( G = \{ y_i \sim t_i, x_i \} \), and \( e_\theta(x, t) \in \mathbb{R}^2 \) defines the unit-vector episolar direction for a pixel \( x \) at time \( t \) under camera parameters \( \theta \), we solve the nonlinear optimization

\[
\theta^* = \arg\min_{\theta, \alpha} \sum_{i \in G} ||x_i + \alpha_i e_\theta(x_i, t_i) - y_i||^2, \tag{4.2}
\]

where here, \( \alpha_i \in \mathbb{R} \) is the distance between \( x_i \) and \( y_i \) along the episolar line (i.e. the 2D analog of the \( \alpha \) in Equation 4.1). In practice, we do not optimize over \( \alpha \), but rather substitute the least-squares solution of \( \alpha \) given \( \theta \):

\[
\alpha_i^* = e_\theta(x_i, t_i)^\top (y_i - x_i). \tag{4.3}
\]

After substitution, 4.2 becomes a nonlinear optimization over the camera parameters \( \theta \).

This optimization is nonconvex, so we seed the initialization by trying 1000 random settings of camera parameters, choosing the one that gives the lowest error. From there, we run a Levenberg-Marquardt optimization [54] to simultaneously optimize for the camera’s pan, tilt, roll, and focal length. The correspondences used for calibration are not used for any other step.

In contrast to previous work, our calibration approach does not require any of the sky to be in view, and it does not make assumptions about the underlying geometry of the scene. For many outdoor scenes, such as the camera shown in Figure 4.4, these assumptions would be too restrictive.
Figure 4.4: Calibrating a camera through episolar constraints. Given a few ground truth correspondences (examples shown in (a)), we find the camera position most consistent with those correspondences (b). We compare our results (beige frustum) to the results from the interface in Chapter 2 (green frustum), which uses hand-selected 3D-to-2D correspondences as determined by Google Earth geometry.

4.2.3 Episolar Integration

Given shadow correspondences $C$ across a variety of lighting directions, the episolar constraint yields a depth inference process which can be cast as a constrained convex program:

$$\argmin_{d, \alpha} \sum_{y \sim x \in C} \|r_x d_x + L_t \alpha_{xy} - r_y d_y \|^2 \quad \text{s.t.} \quad d \geq 1. \quad (4.4)$$

Notice that we constrain the solution so that $d \geq 1$. This both sets the scale of the system and prevents the trivial solution $d = 0$. Since our goal is to recover the depths $d$, we can
again express the optimal $\alpha^*_{xy}$ in terms of the following linear system:

\[
L_t \alpha^*_{xy} = r_y d_y - r_x d_x \quad (4.5)
\]
\[
\alpha^*_{xy} = L_t^\top (r_y d_y - r_x d_x) \quad (4.6)
\]

By substitution of $\alpha^*$ into Equation 4.4, we can express the problem only in terms of the unknown depth $d$:

\[
\arg\min_{d \geq 1} \sum_{y \sim x, x \in C} ||(r_x - L_t L_t^\top r_x) d_x - (r_y - L_t L_t^\top r_y) d_y||^2 \quad (4.7)
\]

Although Equations 4.4 and 4.7 are mathematically equivalent, the removal of the $\alpha$ has enormous practical benefits. In the scenes we work with, there are tens of thousands of correspondences, each one with its own $\alpha$. Optimizing only on the depth yields a much smaller optimization problem.

Although this system of equations has a well-defined global solution with only one correspondence, it also extends to a network of linked constraints. That is, if some pixel $y$ casts a shadow onto both $x$ and $x'$ at different times, this formulation constrains the relative depth of $x, x'$, and $y$. Similarly, if $x$ is shaded by two different pixels $y$ and $y'$ at different times, this places constraints onto $x, y$, and $y'$. This network of constraints is demonstrated in Figure 4.5.

Because this process solves for a depth surface consistent with a set of depth differences, we denote the optimization in Equation 4.7 as episolar integration.

### 4.3 Correspondence Generation

Although the episolar line reduces the search space for shadow correspondence to be along a line, it remains an open problem to robustly link a shadow to its caster. In this chapter, we use a fairly simple correspondence generation rule which works well for most cases of
Figure 4.5: Visualizing a small portion of the constraint graph. Although there is never a time when the red point directly casts a shadow onto the blue point (a), there are enough intermediate constraints (b)-(f) to implicitly constrain the relative depths of the two points, and all intermediate points involved (green).

self-shading.

Given an input sequence of imagery from a diverse set of lighting directions, we first apply shadow estimation [2] which returns a shadow-or-not label for all pixels in sequence.

When this method classifies some pixel $y$ on a shadow edge as under direct illumination at time $t$, our goal is to find which pixel—if any—receives the shadow produced by $y$. We employ a greedy strategy by taking incremental steps along the episolar line emerging from $y$. If the first step away from $y$ is under shadow, we walk along the episolar direction until we find a pixel $x$ which is directly lit again. We then create the correspondence $y \rightsquigarrow_t x$. However, if the first step away from $y$ is still directly lit, no correspondence is generated (i.e., contiguous lit regions only generate correspondences on their edges). We repeat this process for all lit pixels $y$ at all times $t$.

Static objects cast shadows starting from the same location over time, but the location of where the cast shadow lands is a function of the lighting direction and local geometry. Therefore, in natural scenes, shadow correspondences tend to start in the same locations.
Figure 4.6: Generating shadow correspondences. From a time-lapse sequence (one example image shown in (a)), we extract a shadow-or-not labeling for each image (b). For all lit pixels on shadow boundaries, we follow their episolar lines until we find another pixel which is directly illuminated (three examples shown). From here, we remove any correspondence that starts or ends in an unlikely place (c), detail crop in (d) (All correspondences marked in cyan are kept, magenta are removed; see text for details). In this case, all correspondences that start on the ground are removed. In (e) and (f), all correspondences that stop at the vertical edge of the building are removed.
in the images (rooflines, convexities in mountain ridges, etc.), but end in many different locations.

From this observation, we use a simple heuristic to remove correspondences which begin or end in unlikely locations. Using the full set of correspondences, we estimate the probability of a correspondence starting or ending at any given pixel \( z \):

\[
P_{\text{start}}(z) = \sum_{z \sim t \in x} \frac{1}{n}, \quad P_{\text{end}}(z) = \sum_{y \sim t \in z} \frac{1}{n},
\]

where \( n \) is the number of images. We remove any correspondence \( y \sim t \in x \) where \( P_{\text{start}}(y) \leq 0.1 \) or \( P_{\text{end}}(x) \geq 0.1 \). The correspondence generation pipeline is described in Figure 4.6.

This heuristic helps to remove two common error modes. First, if a shadow is cast on the ground far away from its occluder, as in Figure 4.6(a), correspondences will be generated from one side of the cast shadow to the other. However, since “correspondences” rarely start in the middle of the ground plane, they will be filtered out. Second, the initial rule will stop many correspondences at geometry edges when the background is lit and the foreground is not, as in Figure 4.6(f). These false correspondences will be filtered out because it is rare for a true correspondence to stop in the same place repeatedly.

### 4.4 Results

We test our approach on several real-world cameras from the AMOS dataset [34] and show qualitative results in Figure 4.7. For each camera, we select 100 images from a diverse set of lighting directions with clear skies, and use a multi-scale alignment procedure adapted from [33] to remove small jitter. To recover lighting directions, we use the solar position algorithm from [60]. When calibrating the camera using the nonlinear optimization in Section 4.2.2, we optimize over 50 manually-chosen correspondences.

Notice that our approach reliably extracts depth from a variety of complicated geometry and that although the resulting depth map is sparse, the network of constraints covers a
Figure 4.7: Results from our depth inference process on cameras from the AMOS dataset. From top to bottom, we show a crop from an example image, its shadow mask, and the extracted shadow correspondences (for two images). The bottom row shows the recovered depth. Correspondences are shown as connections (yellow line) between an occluder (red point) and its shadow (blue). Depth increases from blue to red. Notice that the epipolar line provides enough constraints to overcome the aperture problem, common in other correspondence problems.
large portion of the scene. To give scale, a typical scene has roughly 70,000 constraints across 30,000 pixels.

Our runtime is largely dependent on the complexity of the shadow masks and image resolution, but we report timing with respect to a camera with 135,000 pixels on a 2.53 GhZ Intel Core 2 Duo with 8GB of memory. The most time-consuming aspect is in computing the shadow masks, which took 4m40s. Creating and filtering correspondences takes another 42 seconds, and solving for depths took 23 seconds.

We also generated a synthetic sequence with rendered cast shadows on a complex scene, shown in Figure 4.8. Our recovered depth surface almost exactly matches the ground truth.
The largest difference comes from a small patch of pixels near the base of a building, where no shadow correspondence was available to link together the two depth components.

To measure quantitative error on real scenes, we compare our depth maps to Google Earth geometry. Since our depth maps are known only up to an unknown scale, we use the ground truth to resolve the scale and compare relative error. Figure 4.9 demonstrates that our depth maps closely match the ground truth, with most of the error coming from the left face of the building. These errors largely stem from noisy shadow segmentation, which creates false correspondences.

Finally, to emphasize the importance of using many images, in Figure 4.10 we explore how the connectedness of the constraint graph increases with more images. This shows that up to roughly 20-30 images, most correspondences form relatively local clusters constraining few pixels. There is then a transition where groups merge and the size increases quickly. After that, shadows that cast onto new parts of the scene are incorporated into the model and the size of the largest connected component grows linearly. This suggests we have not yet reached diminishing returns, in that we can continue to add more imagery and expect more of the scene to be incorporated. We chose to use 100 images, which offers enough correspondences to recover a wide-spanning model in a reasonable time frame.

## 4.5 Summary

Of course, there are cases in which our admittedly naïve correspondence generation technique will not work correctly. For example, the shadow labeling between the tip of a vertical pole to its shadow on the ground plane will almost certainly not be entirely shaded, thus creating a false correspondence. This behavior is demonstrated in Figure 4.11. Despite this limitation, our simple rule works well for most cases of self-shading. In the next chapter, we provide an alternate formulation which uses shadow motion as a structural cue, which alleviates this issue.

Our approach only gives a sparse representation of the depth, reconstructing the depths of
Figure 4.9: Quantitative evaluation of recovered depth. Given a sequence of images (example in (a)), we recover shadow correspondences (b) and a depth map (c). We compare our results to Google Earth models (d). For this structure roughly 120 meters away from the camera, almost all pixels are less than 4 meters away from their ground truth location (using the ground truth to set the scale). This corresponds to a 3.2% error. The errors on the left face of the building originate from noise in the shadow segmentation (f), which creates false correspondences (g).
pixels which cast a shadow or had shadows cast onto them. While this network of constraints still covers a large portion of the image, an ideal solution would merge this constraint with other depth inference processes such outdoor photometric stereo from Chapter 3 or shape-from-clouds [31] to “fill in the gaps.”

In this chapter, we present an approach for recovering the depth surface of an outdoor scene by treating the sun as a second camera and establishing correspondences between a shadow and its caster. This provides a nonlocal depth integration algorithm, as well as an image-space constraint which dictates which potential correspondences are geometrically feasible. These constraints are particularly useful for shape reconstruction, because the correspondence step does not suffer from the aperture problem, and our derivation makes no assumptions on the shape of the depth surface.

In the next chapter, we extend this approach to handle a temporal variant of this problem: rather than create a correspondence between a shadow and its caster, we show that tracking a shadow’s movement gives a similar constraint, and allows us to reconstruct the shape of objects not directly visible to the camera.
Figure 4.11: A common failure case when generating shadow correspondences. From the example image (a), where does the pixel at the top of the light post (in red) cast a shadow? Our algorithm computes a shadow segmentation of the scene and searches along the epipolar line (b). Since the search stops at the first possible correspondence (c), it chooses a false correspondence between the light post and the shadow of a bench behind the post (in blue), rather than the true correspondence later on (in green).
Chapter 5

Structure from Shadow Motion

“Find beauty not only in the thing itself but in the pattern of the shadows, the light and dark which that thing provides.”

— Junichiro Tanizaki

We consider the problem of inferring outdoor scene structure based on the motion of shadows in long term time-lapse data. As the sun illuminates a scene from different directions during the day and during a year, it casts shadows onto the scene. The pattern of these shadows, and how they change, depends on what the camera directly views and nearby structures that cast shadows.

This provides a cue to solve for 3D scene structure from images captured from a single viewpoint. Because shadows are purely geometric objects, this approach also does not require photometric camera calibration, and permits the sun as a calibrated light source. Furthermore, the constraint does not require the occluding object to be visible, and instead infers its position from the motion of shadows. Thus, anything that casts a shadow into the scene can be modeled, including structures hidden behind others in the scene.

What makes this problem difficult is that shadows are sparse and difficult to track. In any one frame, shadows only give constraints at shadow boundaries, so it is vital to track
Figure 5.1: From a sequence of outdoor images (a), we track shadow movement; three color-coded example correspondences are shown on the two example images. These tracks are used as a cue for recovering sparse depth from a single view over time (b), where blue is closer to the camera and red is farther away. Our approach can even recover the structure of objects not directly visible to the camera, as seen in the 3D point cloud reconstruction (c), where black points are the reconstructed locations of shadow casters in the scene. In this case, shadow movements reveal the 3D structure of two trees, one of which the camera only observes from its shadows. We invite the reader to view the supplemental material, which shows rotating views of this 3D point cloud.
shadows across frames. This tracking is difficult because only the shape of the shadow boundary can be used to track, and only a few boundary points have distinguishable shapes. Also, in general scenes, the trajectory of a shadow point between images depends on the lighting direction, the relative geometry of object casting the shadow, and the surface on which that shadow lands.

Our approach to find shadow tracks and scene structure is to exploit the geometry of outdoor illumination and derive explicit constraints relating the solar illumination direction, the 3D location of shadow casting objects, and a depth map of the scene. Even when the overall structure of the scene is unknown, these constraints give a rule for evaluating whether a possible track is geometrically consistent.

Our primary contributions are three-fold. First, for a geo-calibrated camera, we derive relationships between shadow trajectories, the scene depth, and the shadow casting positions without any assumptions on structure of the scene. Second, we create a fully automatic shadow tracking approach that is effective in tracking shadows because it makes use of these geometric relationships as a strong consistency check. Third, we show how to use these shadow trajectories to reconstruct scenes from a single view over time.

We find that our tracking approach works best when shadows are cast on relatively simple structures like hillsides and ground planes. However, we emphasize that the geometry is general for all forms of depth surfaces and shadow casters, and the shape of the recovered shadow-casting objects exhibit great complexity (such as the trees in Figure 5.1).

This chapter makes reference to a supplemental video, which can be viewed at https://www.youtube.com/watch?v=at2krYdsSSI.

5.1 Related Work

The geometry of shadows, light sources, and scene shapes has been explored in a large number of contexts, many of which were discussed in the previous chapter. Here we briefly mention some work as it pertains to shadow tracking.
In the context of outdoor imagery, shadow constraints have been used for calibration and scene structure estimation in a variety of contexts. Antone and Bosse [11] assume a stationary camera with known internal calibration, timestamps, geo-location, and define the analytic constraint between vertical objects and the shadow they cast onto a flat ground plane. Junejo and Foroosh [38] calibrate the intrinsic parameters of a camera and solve for constraints on camera geo-location from the trajectories of shadows of two points moving across a ground plane, and Wu et. al [75] extend this to complete camera calibration, geolocation, and the relative heights of the two shadow casting points, even if those are not in the field of view. Caspi and Werman [18] focus more on modeling the scene structure and use the set of shadows of cast by two vertical edges in the scene to reconstruct a plane and parallax model of scene structure.

In this chapter, we extend the method in the previous chapter to work with moving shadows, rather than correspondences between a shadow and its shadow caster. The main benefits are that the resulting geometry is more general and able to reconstruct objects the camera cannot directly see, and in many cases, tracking a shadow from frame to frame is easier than finding a correspondence between a shadow and its shadow caster. The previous chapter’s naive shadow correspondence algorithm would not work for any of the examples shown in this chapter.

The current work combines the co-linearity constraints of the previous chapter with constraints from shadow tracking [75] to better constrain the 3D shape of points in the field of view, and to provide constraints on shapes not directly visible to the camera. To our knowledge it is the first to derive constraints relating scene geometry from shadow tracks without making any simplifying assumptions about that geometry.

5.2 Structure from Shadow Motion

The fundamental constraint this chapter considers is the relationship between shadow motion and scene structure. We represent this shadow motion by finding corresponding points on
Figure 5.2: Shadow tracking geometry. By tracking a shadow’s movement through three frames, shown as white circles in (a) (insets shown in top-left), we can recover the 3D geometry (b) of the shadow casting object (red) and the surfaces that received a shadow (blue) by solving Equation 5.1 with respect to the given lighting directions at each time (yellow).

shadows cast by the same objects in different images. We call the set of correspondences from one shadow caster a track. In this section we characterize the geometric constraints a shadow track must obey. We defer the discussion of generation those tracks to Section 5.3 because our tracking algorithm uses these constraints as part of a consistency check.
5.2.1 From Tracks to Structure

A track $T_i$ is defined as a set of location-time pairs $\{(x_1, t_1), \ldots, (x_m, t_m)\}$, which we can use to recover the depth $d_x \in \mathbb{R}$ of all pixels $x$ on the track, and the location of a shadow caster $C_i \in \mathbb{R}^3$. The set of all $(x, t)$ location-time pairs in track $T_i$ must satisfy the following 3D spatial constraint:

$$r_x d_x + L_t \alpha_{ti} = C_i,$$  \hspace{1cm} (5.1)

where $r_x \in \mathbb{R}^3$ is the unit vector which passes through pixel $x$ in the image plane, $L_t \in \mathbb{R}^3$ is the lighting direction at time $t$, and $\alpha_{ti}$ is the 3D distance along the lighting direction between a shadow and its caster. This geometry is visualized in Figure 5.2. Essentially, the above equation says that if a pixel $x$ sees part of a track $i$, the 3D point seen by that pixel $r_x d_x$ must be in line with the lighting direction $L_t$ and shadow caster $C_i$ for that track.

Notice that the above equation is similar to Equation 4.1 from the previous chapter, except that $C_i$ is allowed to be a general point in 3D space, rather than along the shadow-casting ray. By tracking the movement of a shadow, we remove the assumption that we know exactly where the shadow-caster is, at the cost of needing to tracking shadows across at least two frames.

This work assumes that the camera geo-centric calibration (and therefore $r_x$ for all pixels $x$) is known, as well as the per-image lighting directions $L$, which can be recovered with known geolocation and timestamps via a solar lookup [60].

Equation 5.1 considers one track and relates depths in the scene to the positions of a shadow caster only up to an unknown and ambiguous scale factor. For example, if some $d, \alpha, C$ forms a zero-error solution, then so does $2d, 2\alpha$, and $2C$. Moreover, a different scale could exist for every track, if each track were optimized separately.

However, if tracks are reconstructed jointly, co-dependence on certain variables between the tracks can fix the relative depth scale between two tracks. More formally, given a set of tracks in the scene $\mathcal{T}$, the tracks-to-structure optimization matches scale factors between
Figure 5.3: An example of track connectivity using tracks recovered from our algorithm. The top figure shows the color-coded locations of three tracks in a scene. The bottom row of images shows crops from these three tracks. This group of tracks has two “crossover” locations (indicated in white in the top image, and with dotted black outlines below), where two tracks constrain the same pixel. We use this tracking overlap to fix the depth scale across tracks when solving for scene structure.

tracks with a constrained linear system over the depth $d$, 3D distances $\alpha$, and occluders $C$:

$$
\arg\min_{d,\alpha,C} \sum_{T_i \in T} \sum_{(x,t) \in T_i} ||r_x d_x + L_t \alpha_{ti} - C_i||^2,
$$

(5.2)

under the constraints that $d \geq 1$ (to set the scale) and $\alpha \geq 0$.

This fixes the scale factor between tracks that overlap, because if one image location $x$ is part of two shadow tracks (at different times), it must have a single, consistent depth $d_x$ in both. Therefore, it is desirable that a tracker return as dense a set of tracks as possible. Then each pixel is more likely to be included in multiple tracks so that the set of constraints has more crossover, and therefore there is greater support for many tracks sharing the same
Figure 5.4: A visualization of the geometric consistency check. In (a), both the green and blue tracks have similar appearances through time. In (b), we annotate each detection with its episolar line. This additional geometric check reveals that the green track is consistent with a single shadow caster (since all green lines intersect in a common location), while the blue track is inconsistent.

depth scale. Figure 5.3 shows that in real scenes, there is typically a large amount of crossover which connects distant pixels through a network of constraints.

5.2.2 Geometric Consistency

When a track passes through two or more frames, solving for the unknowns defines 3D scene locations for each shadow point, and the location of the shadow casting object $C_i$. Therefore, we can test for the geometric consistency of a track by estimating how well $T_i$ satisfies this linear relationship.

If we believe some pixel $x$ to be under shadow at time $t$, then we can draw the episolar line on the image starting at $x$. If a track is geometrically consistent, then each of these lines will intersect in a common location, the location of the shadow caster in the image. Therefore, this geometric consistency check can be visualized as an intersection test between many image-space lines, as demonstrated in Figure 5.4.

In practice, we test a track $T_i$’s geometric consistency by solving Equation 5.1 for $d_x$, $\alpha_{ti}$, and $C_i$. If a track is perfectly consistent, there will be no error, and the angle between $L_t$ and $(C_i - r_x d_x)$ will be 0 degrees for all $(x, t) \in T_i$. In the following section we reject a track if, for any $(x, t) \in T_i$, this angle is greater than a half degree, or if $d_x$ is negative (corresponding to
Figure 5.5: Shadow tracking under geometric constraints still suffers from the aperture problem. Consider a vertical pole casting shadows across three frames shown as a single composite image in (a). Where does the shadow in the blue box go in the other two frames? The tracks in (b) and (c) are both geometrically consistent, and have exactly the same appearance over time, but only the track in (b) is correct.

observing a shadow behind the camera), or if any $\alpha_{ti}$ is negative (corresponding to a shadow being projected the wrong direction).

5.2.3 Challenges and Limitations

The geometric constraints relating shadows to scene structure have several formal ambiguities. Characterizing the ambiguities helps to define limits of the approach and suggests properties that are important for robust shadow tracking.

The first ambiguity affects the tracking step and is a corollary to the aperture problem in standard optic flow. When tracking a shadow cast by any straight edge (such as a pole), the shape of the shadow will be a strong edge, and the motion of the shadow along this edge is not well constrained. Unfortunately, the additional geometric constraints do not remove the aperture problem; there are still an infinite number of geometrically-consistent tracks that pass through the shadow’s shape. Figure 5.5 illustrates this case.

The second ambiguity affects reconstruction in the presence of degenerate lighting configurations. In a video sequence taken over the span of one day, the set of illumination directions is often degenerate. In the extreme case of images taken over a day during the equinox at the Equator, the sun passes directly overhead, and all shadows are cast exactly
along east-west lines. Thus, while tracks may cross each other, tracks will never “move
north and south” to unify the scale factors of different parts of the scene. While this effect
is mitigated when the lighting configuration is non-degenerate (e.g. when the camera is far
from the Equator, or during the summer and winter solstices), we find the reconstruction
step for images from a single day to be poorly conditioned. This argument also applies to
the shadow-to-caster style of correspondence used in the previous chapter.

To alleviate the aperture problem, we track shadows in a spatially smooth manner so
that ambiguous edges are more likely to follow the movements of more discriminative shapes.
We handle the second ambiguity by working with data spanning over months to get a more
diverse set of lighting conditions. In this case, the set of sun directions is not planar, but
rather lies on a full-rank subset of the unit sphere. This is not a large limitation, because
large archives of outdoor scenes such as AMOS [34] and Webcam Clip-Art [49] have already
been capturing live webcam streams for years.

Notice that we have a similar requirement on the set of input imagery as we did in
Chapter 3. Both when estimating surface normals and when using shadow-based cues, there
is a degeneracy that occurs when using a data set spanning only a day. In the conclusion,
we discuss this theoretical limitation in more detail.

5.3 Shadow Tracking

The previous section assumes that we have shadow tracks: sets of image locations and times
where an object cast a shadow. Now, we discuss how to generate those tracks from image
data.

The shadow tracking approach has four steps. First, we detect where shadows are in each
image, and describe each point with a local binary pattern. Second, frame-to-frame matches
are found between pairs of images with similar lighting directions. Third, these matches
are linked together in an approach that guarantees geometric consistency, and finally, these
tracks are extended to cover a greater temporal extent.
Figure 5.6: To detect shadows in a time-lapse sequence, for each image (a) we run an edge detection algorithm (b). We keep any edge pixel that was not an edge in many other frames, largely removing edges from persistent structure or texture (c).

5.3.1 Shadow Detection and Description

Given a set of input imagery, we begin by computing the Canny edges [16] on each image. An edge could occur due to shadows, as well as depth or texture. Shadows from depth discontinuities or scene texture remain stationary through time, while cast shadows move as the lighting direction varies. Therefore, for each image, we take the pixels on the edge map, and remove any that were on an edge map more than 10% of the time. The result is a set of pixels that mostly come from moving cast shadows; see Figure 5.6 for an example.

To describe each point \((x, t)\), we compute a local binary pattern feature in a circle with a 10-pixel radius centered at the detection of interest. Our feature \(f\) is a bit-vector which encodes if the pixel \(x\) has a greater intensity than each of the sample locations around it at time \(t\). To alleviate image noise, we additionally smooth the image with a 3x3 average filter before extracting features. The result is a set of detections \(D\) with feature descriptors, denoted as \((x, t, f)\) triplets. As shorthand, we denote the detection \(i\) as \((x_i, t_i, f_i)\).

5.3.2 Frame-to-frame matching

In the next step, we create many frame-to-frame shadow matches. We repeat this process for many image pairs from nearby lighting directions to create a rich set of 2-frame correspondences. Specifically, given two images \(t_1\) and \(t_2\), each with their own sets of detections \(D_1, D_2 \subset D\), we want to find a matching function from \(D_1\) to \(D_2\).
Assuming that shadow motion is small across nearby lighting directions, the shadow’s location and appearance should not vary dramatically. For consistency in this section, we denote variables with \( i \) and \( j \) subscripts when they refer to detections in \( D_1 \) and \( D_2 \), respectively. For each detection \( i \), we gather a set of possible matches \( M_i \subseteq D_2 \):

\[
M_i = \{ j \in D_2 \mid s_{ij} > \tau \},
\]

where \( \tau \) is a minimum matching score, and \( s_{ij} \) is a score that attains a large value when detections \( i \) and \( j \) have a similar appearance (in terms of their local binary patterns) in similar locations in the image:

\[
s_{ij} = w_{\text{location}}(i, j) \cdot w_{\text{appearance}}(i, j)
\]

where

\[
w_{\text{location}}(i, j) = \exp \left( -\frac{(x_i - x_j)^2}{\sigma_x^2} \right)
\]

\[
w_{\text{appearance}}(i, j) = \exp \left( -\frac{(f_i - f_j)^2}{\sigma_f^2} \right)
\]

Furthermore, we expect the matching function should be spatially smooth; nearby shadows at \( t_1 \) should match to nearby locations at \( t_2 \). We model the matching function as a nonparametric warp by assigning a warp vector \( u_i \in \mathbb{R}^2 \) to each detection \( i \), and optimize the following:

\[
\argmin_{u} \sum_{i \in D_1} \min_{j \in M_i} ||x_i + u_i - x_j||^2 + \left| \left| u_i - \sum_{i' \in D_1} v_{ii'} u_{i'} \right| \right|^2
\]

The first term encourages the warp to push \( x_i \) toward one of its potential matches, and the second term is a Laplacian smoothness term, where \( v_{ii'} \propto w_{\text{location}}(i, i') \), with \( v_{ii} = 0 \).

This objective has a similar form to other variational matching objectives, such as dynamic time-warping [62], matching shape context features with a thin-plate spline defor-

\(^1\)Although one could define \( w_{\text{location}} \) and \( v \) with different bandwidths, we chose to use the same bandwidth \( \sigma_x \) for both for simplicity.
Figure 5.7: A visualization of the matching algorithm. Suppose we want to match the shadow edges in (a) to (b), shown as an average image in (c) for visualization. We first find a set of candidate matches, shown in (d) for two detections. The pink detection comes from a corner and only has a few candidate matches, but the cyan detection comes from a shadow edge and is less discriminative, so it could match to many more. We optimize for a smooth warp across the image that maps each point to one of its matches, and accept frame-to-frame matches (white) that came close to one of their candidates (e).

We optimize this objective with gradient descent, initializing $u_i$ as $x_j - x_i$, where $j \in M_i$ is the detection where $s_{ij}$ is maximal. If $M_i$ is empty, we initialize $u_i$ to 0. After convergence, we create a match between detections $i \in D_1$ and $j \in D_2$ whenever $||x_i + u_i - x_j||$ is less than 2 pixels.
Figure 5.8: The percent of detections included in a geometrically-consistent track, as a function of the minimum acceptable track length, for three track linking approaches. The naive method incrementally links together matches with a common endpoint, the RANSAC method finds geometrically-consistent subtracks within those, and the constrained linking approach incrementally links together matches with a common endpoint, so long as it maintains the track’s geometric consistency. The third approach performs the best, incorporating more detections into long, geometrically consistent tracks.

5.3.3 Linking together matches

Given a large set of frame-to-frame matches, we now link these matches into long tracks. There are a few obvious baselines for this problem which we found to be insufficient, so we first describe two baseline track linking approaches, and then our approach which enforces geometric consistency through time.

The simplest method to link matches together is to iteratively group together two matches if they share a common endpoint (i.e. they share a detection), unless it creates a track that passes through the same frame twice. This process is repeated until the track cannot be extended any more, and if the track is sufficiently long, we keep it. We found that this method does not give many tracks which satisfy the geometric consistency check, even for shadows with discriminative local binary patterns.
This is alleviated somewhat if each track is filtered through a RANSAC routine: we choose two random detections in the recovered track, find their shadow caster through Equation 5.2, and see which other detections in the track are consistent with that. After many rounds, we keep whichever subtrack has the largest number of inliers.

The best approach we found was to incorporate the geometric consistency check into the track linking procedure. We still iteratively group together matches with common endpoints, but at each step check if the next addition would break the track’s geometric consistency so far. If there are multiple possible extensions, we choose the one that is most consistent (in terms of angular reconstruction error as in Section 5.2.2).

These three approaches are shown in Figure 5.8 using frame-to-frame matches for the scene from the first page. This plot shows how many detections in a scene are incorporated into a track after each track linking strategy (for varying thresholds on the minimum acceptable track length). The more detections that are incorporated into some consistent track, the more complete the model will be. The geometrically-aware linking algorithm outperforms the other two, typically explaining the motion of 5% to 10% more detections. This is because after a few detections are linked together, the consistent linking routine effectively only searches for geometrically-consistent extensions, whereas the other two baselines enforce consistency as a “post-processing” step.

5.3.4 Track expansion

As a final step, we take each track and try to extend it into frames it does not yet pass through. For each track, we find all detections \( j \) in new frames that have a high score \( s_{ij} \) to some detection \( i \) already in the track. If that detection is geometrically consistent with the rest of the track, we append it, and repeat until no such \( j \) is found.

Intuitively, this step is similar to the correspondence-generation approach from the previous chapter; this method effectively searches along the episolar line away from the shadow caster, and tries to find shadows which it likely created in as-yet-untracked frames. The two
Figure 5.9: Track expansion encourages well-connected reconstructions. Given an image sequence (average image shown in (a)), our full algorithm returns a depth map that covers most of the ground plane (b). (c) and (d) show two “connected components” of a reconstruction fueled with un-expanded tracks. Since the tracks that build these 3D models are shorter and cover a smaller area, overlap between tracks is less likely and the reconstructions are disconnected.

main differences are that 1) the location of the shadow caster comes from a reconstruction of the track so far (rather than analyzing a binary shadow mask), and 2) we leverage knowledge of where the shadow appeared before and what it looked like (rather than blindly accepting the first possible correspondence).

Although the tracks generated before this step are already long and consistent, this extra expansion step helps to create crossover between tracks that might not have crossed over before. Figure 5.9 demonstrates this property, where the un-expanded track set is not quite expansive enough to connect together two large depth components in a scene.
5.3.5 Incremental Reconstruction

Rather than perform the full constrained least squares reconstruction as in Equation 5.2, we take advantage of an incremental reconstruction for robustness and speed. We begin by reconstructing a seed track, chosen as the track that overlaps with the most others. One by one, we reconstruct the track that overlaps the most with the reconstruction so far, under the additional constraint that the depth for pixels already reconstructed stay constant. If the track is no longer geometrically consistent under those additional constraints, we remove the (likely erroneous) detections from that track that accrued the most error and pick another track. This process repeats until there are no tracks that overlap with the model. A typical reconstruction takes about a minute, which is fast compared to the full least squares optimization, which (even with commercial sparse linear system packages) often cannot fit the full linear system in memory.

5.3.6 Implementation Details

For webcam data, geometric camera calibration was derived by manually corresponding scene points to Google Earth models (as in the first chapter) or using manually specified shadow to shadow caster correspondences (as in the previous chapter).

The algorithm, starting from a calibrated camera and a set of 200 images requires, on average, three hours to create 3D models. About 55% of the time is spent on finding and optimizing frame-to-frame matches, another 35% spent on linking together matches, 8% spent on expanding tracks, and the remaining time spent loading images, detecting/describing shadows, and incremental reconstruction.

In our implementation, we find matches between each image and its 5 nearest neighbors, where distance is measured in terms of angular difference in sun position. We use the parameters $\sigma_x$ as 5% of the main diagonal length, $\sigma_f$ as 5% of the feature dimensionality, $\tau$ as 0.1, and the minimum track length as 8 frames.
5.4 Results

To evaluate the approach, we explore results based on a synthetic scene and a collection of images taken by webcams over long periods of time. We invite the reader to view the supplemental video to view the reconstructions in 3D.

To test the accuracy of our approach, we rendered a synthetic scene using virtual sun positions over the span of a year. This synthetic scene is challenging because most shadows are projected onto a curved surface, so shadows distort their shape for even small movements. After fixing the scales between the ground truth depth and our reconstruction, our model’s depth has an error of 2%.

Recent research in single-view shape-in-the-wild approaches shows that the camera’s color calibration needs to be known a priori [8, 31], or that the camera’s response cannot change through time [3]. Figure 5.11 shows an evaluation of the robustness of our approach to unknown color calibration. For this experiment, we artificially distorted each image in a sequence with a random exposure and radiometric response chosen from [25] (i.e. a different
Figure 5.11: An evaluation on the robustness to unknown radiometric calibration. The top row shows an example image an image sequence (a), and the same image artificially distorted by an unknown exposure and tone mapping curve (b). Each image is annotated with the set of detections that were successfully tracked through that image. When our algorithm runs on the original (c) and distorted (d) sequences, the results are almost identical.

response profile for each image), and ran our algorithm on both the original and distorted data. Since our features come from Canny edges and local binary patterns, which are both invariant to response and exposure changes, the result is identical before and after distortion. This level of distortion would cause dramatic errors in any of the photometric methods above.

5.5 Conclusions

In this chapter, we introduce a framework for single-view shape in the wild. This approach extracts shadow trajectories using a new geometric consistency measure and response-invariant features. In some cases, this approach allows us to use shadows to recover the shape of objects the camera never directly saw.

We share a similar error mode to traditional structure from motion, in that scenes with relatively little shadow texture are poorly modeled. If there are not many shadows with
discriminative shapes, then the matching routine is more ambiguous, and the whole pipeline suffers.

Our approach only tracks shadows on the boundary between darkness and light, because they are more discriminative than pixels on the shadow’s interior. However, these interior pixels carry useful information, and could be used to determine more complete 3D models of shadow casters.

Our largest assumption is that the scene remains static, which can break down when imagery comes over the span of a few months. Our incremental reconstruction and tracking algorithms are robust enough that if the scene changes geometry briefly, we only capture the most static mode. However, more formally addressing dynamic geometry is an exciting avenue for future work.

In theory, this geometry works even in the case where the object casting a shadow is behind the camera. In practice, we found that these reconstructions were poor. Figure 5.13 gives an example of this behavior, and a geometric intuition for why shadow casters behind the camera are intrinsically difficult to reconstruct. When the object is in front of the camera (black triangles), we can observe its shadows under a variety of lighting directions, and the triangulation process (transparent blue regions) is well-conditioned. When the shadow caster
Figure 5.13: Failure case in shadow tracking due to limited lighting space. The first row shows two example images (a) from a sequence where all images come from lighting directions corresponding to the sunset, shown in (b). The second row (c) (d) shows two views of the reconstruction of this sequence, where points are colored by their depth, and red/blue points correspond to points in front of/behind the camera, respectively. The poor reconstruction is explained in the third row, where (e) shows a “good” reconstruction when the shadow casting object is in front of the camera, and (f) shows a “bad” reconstruction when the object is behind the camera.

is behind the camera, the camera can only observe its shadows for a narrow range of lighting directions, which leads to poor triangulation (transparent red regions). Although this object
may cast shadows in locations which would lead to a more precise reconstruction (like the violet region), the camera will not observe those shadows.

The largest limitation of our shadow tracker comes from our relatively rigid feature representation. Our local binary pattern features do not have any invariance to scale or rotation, so we depend on the shape of the shadow to not distort wildly between images with similar lighting conditions. Therefore, this approach works best when reconstructing planar surfaces or surfaces with small curvature. However, our geometric consistency checks are valid for any kind of depth surface, meaning that any future change to the feature representation is a drop-in replacement.
Chapter 6

Conclusions

This thesis describes approaches to calibrate the geometric and radiometric properties of a camera, and build 3D models of a scene from a single view over time. The methods presented in this thesis attach some measures of real-world calibration and 3D structure to real webcam imagery. In this conclusion, we explore high-level connections between each chapter, and discuss opportunities for future work.

A common thread which appears from Chapter 3 through Chapter 5 is the overall dependence on the diversity in lighting conditions. Each algorithm presented in those chapters requires at least a few months of input imagery to accurately reconstruct the 3D geometry in the scene.

An ideal approach for outdoor scene analysis would work in a single day. Single-day analysis is attractive not only for the reduced set of input assumptions, but also because it avoids problems that multi-month analysis faces. Over the span of a few months, trees will change shape and color, which simple image formation models cannot handle; over a single day, a tree’s structure is mostly constant. Over the span of a year, a typical camera will change viewpoint dramatically, which breaks the assumption that the image sequence is well-aligned. But over the span of a day, a camera will usually only have subtle jitter.

Unfortunately, the algorithms presented do not work well with limited lighting directions (such as those from a single day). Given the importance of single-day analysis, we briefly
investigate why each algorithm has this single-day degeneracy, and explore the limitations of photometric and shadow-based cues in the domain of limited lighting spaces.

6.1 Degeneracy in photometric stereo

A simple variant of photometric stereo can be written as

$$I_t = (\rho N)^\top L_t.$$ 

If $L_t$ is given, then the optimal $x^* = (\rho N) \in \mathbb{R}^3$ can be expressed as the solution to the linear system

$$Lx^* = I,$$

where $L \in \mathbb{R}^{n \times 3}$ and $I \in \mathbb{R}^n$.

So, any dependence on the lighting directions depends entirely on the rank of $L$. In a standard photometric stereo problem, one will take a variety of images spanning a full-rank subset of the unit sphere when collecting the imagery in a lab setting. However, since our lighting conditions are controlled by celestial geometry, we only get a full-rank lighting matrix when the image sequence comes from a long time sequence.

Over the span of a day, the sun’s path through the sky will make $L$ contain a planar slice of the sky. For example, if the sun rises directly in the East, passes straight overhead, and sets exactly in the West, then $L$ will be exactly rank 2. For this reason, we only experiment with data spanning a few months, to avoid this kind of degeneracy.

However, the sun will only create this path on the Equator during an equinox. Are there better-conditioned times and locations to perform outdoor photometric stereo in a day? The exact solar path depends on the day the image was taken, and at which latitude. Consider a day at the North Pole during summer; the sun will never set, but perform a full 360 degree rotation above the horizon. In some sense, this is the ideal dataset to perform photometric
stereo, since the planar slice of the unit sphere in $\mathbf{L}$ is fully rank 3. In contrast, the sun never rises during a polar winter, and so photometric stereo is fundamentally impossible then.

To quantify this behavior, we computed the condition number of the lighting matrix $\mathbf{L}$ for all latitudes and days of the year, shown in Figure 6.1. There are a few interesting patterns. During an equinox (approximately March 20 and September 22), the set of lighting directions is the most deficient, and is in fact rank 2 everywhere. At the poles, photometric stereo is the best-conditioned during summer and impossible during winter. Otherwise, the sun typically passes through a full-rank subset of the unit sphere at all times of year, although summer months are better conditioned than winter months, since the summer days are longer and have more diverse lighting directions.

It should be noted that these are idealized condition numbers, since a pixel might be in shadow for part the day, which would further reduce the conditioning in the set of lighting directions. In the worst case, there might be a pixel under shadow for the entire day, in which case we cannot recover its surface normal.
6.2 Degeneracy in shadow-based reconstruction

Noted briefly in the last chapter, there is a similar degeneracy on lighting conditions for shadow-based reconstruction. For the worst-case scenario of an image sequence taken over one day during an equinox, each object in a scene will only cast a shadow in a planar slice of the scene’s geometry. While this plane would be accurately reconstructed, there would not be any way to establish scale factors for different planes. This kind of degeneracy occurs both in tracking shadow movements (as in Chapter 5), and in using the episolar line to directly make correspondences between a shadow and its caster (as in Chapter 4).

Again, this is only theoretically true for one-day sequences taken during an equinox. On other days, each object will provide cues for a non-planar slice of the geometry (such as a parabolic shape of a tracked shadow across the ground over the span of a day). In practice, we found that the one-day sequences to still be poorly-reconstructed; the minimal amount of crossover gained from these sequences was not sufficient to unify enough scale factors across the scene.

Notice that this degeneracy takes a much different form here than it did for photometric stereo. In photometric stereo, the lighting degeneracy comes from a rank-deficiency constraint when estimating surfaces normals via a linear system. In shadow-based reconstruction, it provides a constraint when trying to link scale factors between neighboring shadows.

6.3 Comparison of degeneracies

Because these degeneracies come from such diverse backgrounds, one might believe that these issues are complementary; using one cue at a time for scene reconstruction from a single day is fundamentally flawed, but perhaps if one used both shadows and photometric stereo in concert to guide reconstruction, these problems would disappear.

Unfortunately, this is not the case.
Figure 6.2: The uncertainty in surface normal estimation from a single day. On these two views of the same 3D plot, we show the set of sun directions for a given day as large solid circles, colored by the intensity of some Lambertian pixel of interest. The true normal and albedo used to create that sequence is shown as a solid red line. The goal of photometric stereo is to recover the normal direction from the lighting directions and per-frame intensities, but a large set of other normal/albedo pairs, visualized as small circles, also reconstructs that image sequence (within a small tolerance).

To explore why these structural cues are not complementary, we first discuss the specific form of photometric stereo’s one-day deficiency. The rank constraint expresses itself as an uncertainty in estimating one axis of the surface normal. Figure 6.2 demonstrates this problem; over one day, photometric stereo can reconstruct the direction of the normal parallel to the lighting plane, but cannot uniquely recover the direction perpendicular to the lighting plane.

Even when combining these two cues together, the ambiguities present in one-day analysis persist. This is visualized in Figure 6.3. The ambiguity in photometric stereo expresses itself as an ambiguity in the slope of a rooftop. Even when a shadow is cast onto that rooftop, shadow-based cues can only reconstruct one planar slice of the rooftop, and in particular, the shadow-based cue recovers the only slice that gives no information about the slope of the rooftop.
Figure 6.3: Photometric stereo and shadow-based reconstruction are both degenerate over a one-day sequence, and combinations of these cues would not help to disambiguate the solution. Over the span of a day, this cartoon scene (a) sees intensity variation on an East-West aligned rooftop, with shadows cast from a chimney. The degeneracy from photometric stereo expresses itself as an uncertainty in the slope of the roof, as indicated by the span of possible surface normals lying on a plane in (b). The shadow-based cues can only reconstruct shadows cast along an East-West line, indicated by the blue line across the rooftop (c). Therefore, any combination of these two cues does not help to disambiguate the slope of the roof.

6.4 Next Steps

Finally, we discuss some future work that we anticipate will be useful for single-day analysis and time-lapse scene analysis in general.

The approach in Chapter 3 performs radiometric calibration during the photometric
stereo optimization. However, there is still a wealth of work to be done for radiometric calibration for real Internet images “in the wild”. An ideal system would calibrate the response of the camera from a single image. Current single-image radiometric calibration approaches exploit subtle phenomena, such as image noise [55, 37] or the statistics of small RGB patches [52], but use statistics which are not robust to even small amounts of JPEG compression. We believe that continued research into the statistics of JPEG compressed imagery will give insights into more widely-applicable radiometric calibration.

This thesis shows a few methods for 3D reconstruction from a single view over time. As we begin to work with single-day analysis, the approaches presented in this thesis are not appropriate. However, there are a wealth of 3D models of urban areas from Google Earth and more standard 3D reconstruction pipelines [70, 21]. Perhaps these coarse 3D models can help to disambiguate the degeneracies from one-day photometric stereo. For example, one could use the coarse models to understand the general shape of the scene and its depth discontinuities, and use photometric stereo to create a fine-scale reconstruction of surface texture in its integrable regions.
Bibliography


