# ROBUST SPATIOTEMPORAL ANALYSIS OF ARCHITECTURAL IMAGERY

by

Thommen Korah

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Sciences

Fall 2007

# ROBUST SPATIOTEMPORAL ANALYSIS OF ARCHITECTURAL IMAGERY

by

Thommen Korah

Approved: ______________________________
David Saunders, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: ______________________________
Tom Apple, Ph.D.
Dean of the College of Arts and Sciences

Approved: ______________________________
Carolyn A. Thoroughgood, Ph.D.
Vice Provost for Research and Graduate Studies

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ________________________________________

Christopher Rasmussen, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ________________________________________

Chandra Kambhamettu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ________________________________________

Jingyi Yu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ________________________________________

Xinyang Deng, Ph.D.
Member of dissertation committee

## ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Christopher Rasmussen. His intellect, attitude to research, and clarity of knowledge was a great source of inspiration. Christopher's ability to distill the most important issues from complex problems added a whole new dimension to our brainstorming sessions. While writing papers and preparing talks, I was driven by the challenge of meeting his exacting standards. It is with humble gratitude as well as a sense of pride that I look back on my five years of graduate school as his first PhD student.

I've been fortunate to be part of the Vision group at UD. Chandra Kambhamettu introduced me to the subject in my first semester and has always provided thoughtful advice. Jingyi Yu, who works at the confluence of Graphics and Vision, has been an inspiration for some of the ideas developed in this work. The company of Yenchi Lin, Bill Ulrich, and Mani Thomas ensured that the lab was not only stimulating, but an enjoyable environment as well. I specially acknowledge Mani's desperate efforts in assisting me to submit the thesis.

Graduate school has been a fostering ground for several lasting friendships. Gajju Sasson, my first friend in the US, brought me quickly up to speed with a new culture and way of life. Some of my fondest memories in Delaware were spent on the soccer or frisbee fields. To name all the people that shared those moments would itself comprise a thesis-sized document. I thank Kapil, Sachin, Deepak, Chaitra, Asmita and Trisha for having colored my last few years with unique individuality. I will also miss the delicious pies and grilled burgers made by Conan Weiland.

Two people stand out for whose friendship I am especially grateful – my roommates Anoop Mathew and Ryan Maladen. Rarely does one get to meet people of such high moral character, intellect, genuine sincerity, warmth, and zest for life. In so many indirect but tangible ways, they made this dissertation possible. I will fondly remember the innumerable trips to the airport, Thursday night wings, dinner debates, and their creative manipulations of the English language.

This journey would never have begun without the love and prayers of my family. My parents struggled to make it feasible for me to even dream of graduate school abroad. Their only moment of doubt in seeing me go through a PhD was whether they had laid too much emphasis on the importance of education. In their three children, they have written their own dissertation. I thank my sister Indu for being such an amazing person to whom I could turn to for support and advice at any time. And my brother, Thomas, gives me great pride as he starts his career in investment banking. I thank Appa, Amma, Appachan, Ammachi, Ammamma, Jean, Tommypappen and Lailamama for all their encouragement and words of advice. My heartfelt thanks go to Ivy Aunty for her selfless concern and counsel. I owe a great deal also to my cousins Paul and Tara who made Philadelphia my home away from home.

I specially mention my grandfather, late Dr. Justice T. K. Thommen, who sowed in this impressionable youngster the love of learning and research. As a testament of my gratitude, I dedicate this thesis to him. Watching him pore over cases late into the night taught me the value of hard work and perseverance; those lessons have indeed enabled this thesis to come to fruition.

Last but not least, I could not have finished without my wife Mariam. She provided the inspiration, love, and enthusiasm to keep my engines burning through this arduous undertaking. Whenever things got overwhelming, I could count on her to brighten up my day. Most importantly, I thank her for patience during the years that my research forced us to be apart. She remains my greatest joy and blessing.

# TABLE OF CONTENTS

# LIST OF TABLES

# ABSTRACT

This thesis addresses the issue of understanding and manipulating images of architectural scenes. Automatically modeling the structure and appearance of buildings with a robot is challenging; an end-to-end system would have to tackle a whole spectrum of tasks such as planning, sensor fusion, navigation, image acquisition and matching, structure estimation and texture mapping. Purely bottom-up techniques are inadequate for this task due to ambiguities and missing information inherent in sensor data. My solution is to introduce additional domain-specific models that can capture dependencies such as restricted spatial configurations or geometric patterns in images of buildings. Techniques to encode, discover and exploit these relationships for retrieving semantic information about buildings are illustrated. These interaction models are shown to be powerful for such varied tasks as object recognition and detection, segmentation, inference of missing information, and realistic image synthesis – even without supervised training or other appearance models.

A primary focus of this work is on constructing "clean" texture map mosaics of building facades. Without explicit handling, foreground objects such as trees, signs, and people will appear pasted as artifacts on the model. As a first major contribution, given an image sequence captured around the building, I developed a novel spatiotemporal *timeline-based* inpainting technique to remove non-building pixels from the median mosaic. These polluted regions are a result of the majority of views being occluded, which makes conventional techniques such as the median filter unreliable. Outlier pixels are then automatically identified by a robust measure of spread. A combination of motion cues and an automatically trained appearance-based classifier are used to fill the majority occluded holes with true building background. A second stage of spatial inpainting is applied to the relatively small unanimously occluded regions in which the background was never imaged. Results are shown on a variety of campus buildings.

My second major innovation is a series of methods that enable foreground removal from *single* images of buildings or brick walls without any motion information. The key insight is to use *a priori* knowledge about grid patterns on building facades that can be modeled as Near Regular Textures (NRT). I describe a Markov Random Field (MRF) model for such textures and introduce a Markov Chain Monte Carlo (MCMC) optimization procedure for discovering grid structures on building images. Results are shown on both synthetic NRT as well as building images. This simple spatial rule is then used as a starting point for inference of missing windows, facade segmentation, grammar-based image parsing, outlier identification, and foreground removal.

I also describe related work on how aerial imagery may be exploited for navigating a robot around the building perimeter. A randomized approach to view planning is presented that generates paths to simultaneously address visual coverage and quality. A Monte Carlo Localization framework for vehicle localization and guidance is also described.

# Chapter 1

# INTRODUCTION

The problem addressed in this thesis is that of understanding and manipulating images of architectural scenes captured by an autonomous robot navigating in an urban environment. Realistic ground views of buildings such as those shown in Figure 1.1 may be severely unconstrained unlike aerial views, making it a challenging task. We are primarily motivated by the ability of humans to learn a useful model of a class from a small number of examples, often just a single example. Humans do not have to see thousands of examples to recognize a building and identify its windows, doors, roof and so on. Along with appearance characteristics, we also reason about its functional and spatial context. Whereas a rectangular opening on the facade 10 meters above the ground is most probably a window, an identical opening on the facade with the base reaching the ground is probably a door or a French window.

A grand goal for robotic systems is that of building an agent that can sense its environment, navigate autonomously around it, and interact intelligently with the physical entities in that environment. This interaction encompasses a wide spectrum of tasks such as space exploration (Mars Rover), driving vehicles (DARPA Grand Challenge), cleaning floors or pools (iRobot Roomba), and aerial surveillance (Predator UAV). A fundamental task towards attaining this goal is that of fusing information from various sensors to model and infer the complex nature of the real-world. While it comes so naturally to humans, this task can be frustratingly difficult to teach a computer to perform.

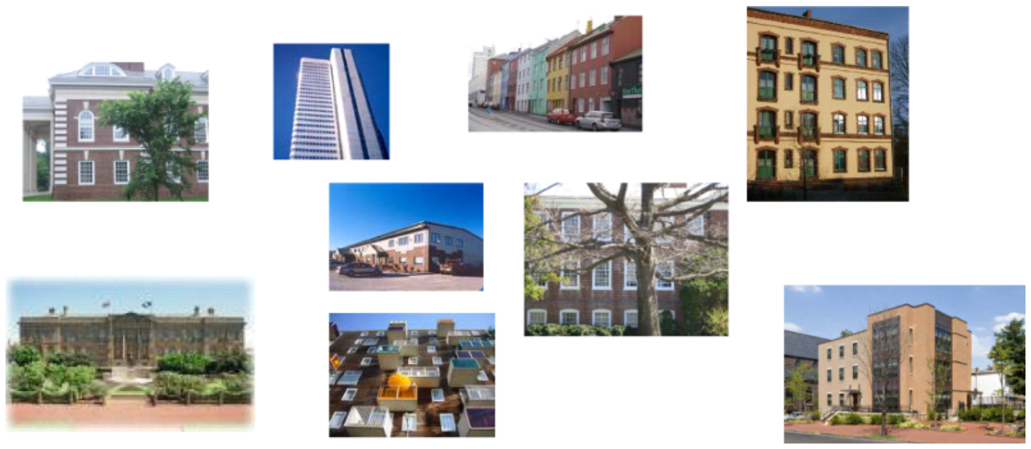


**Figure 1.1:** Ground views of buildings may undergo large variability in appearance, as well as being subject to occlusions and clutter.

With advances in computing and camera technology, *computer vision* techniques have been successfully deployed in various robotics systems. Tracking, stereo, visual odometry, structure-from-motion, and image-based localization and mapping have all reached a level of maturity sufficient to facilitate autonomous navigation over hundreds of miles as in the DARPA Grand Challenge [33]. Most of these algorithms work by detecting local discriminative features and match them across multiple images to infer their temporal or spatial ordering. Features used are often small patches from the image and might have very little "meaning" in the robots domain.

The basic observation behind this thesis is that as robots move out of constrained environments into more unpredictable and cluttered settings, visual sensing must take on the added dimension of *semantic scene interpretation* i.e., reason about the semantic characteristics of the scene as opposed to patch-based processing. A robot only sees a grid of color pixels from a camera with no higher level grouping. What if the robot could instead like humans see and recognize windows, doors, trees, and other *semantic entities*? How can we equip the robot with this ability and what is the best way to act upon this new information?

The past few years have indeed witnessed substantial developments in object recognition or classification [13, 37, 43] and localization [25]. The majority of these algorithms involve identifying candidate patches from a large training set of labeled images to learn independent appearance and spatial models for these parts. This strategy helps to mitigate the effects of occlusion, pose variations, and illumination changes. A classifier would then combine evidence from a new image, possibly using co-occurrence of features as an additional cue, to determine the presence or absence of an object. Without discounting its efficacy for object recognition, this approach fails to capture certain semantic attributes and relationships that are domain specific. For example, windows on a building facade appear similar and are often part of a larger grid, which allow humans to hypothesize both appearance and location of occluded windows. Examples of individual windows or even groups of them will not be able to capture this.

It will be an aim of this dissertation to show that semantics alone can be powerful enough for recognition and subsequent inference. By restricting the robot to work in urban environments, we develop algorithms that exploit established architectural constraints to impose priors on what is being seen i.e., *parse* the image. These priors are specified as topological rules that describe the spatial arrangement of building entities (such as, grid of elements, divisions within a window, and so on). Without a manually labeled training set, any learning has to be bootstrapped with examples extracted automatically using these rules. We contend that the versatile object recognition techniques from the literature, when annotated with the semantics of the domain, may assist robots to resolve ambiguities and make inferences – much as humans do.

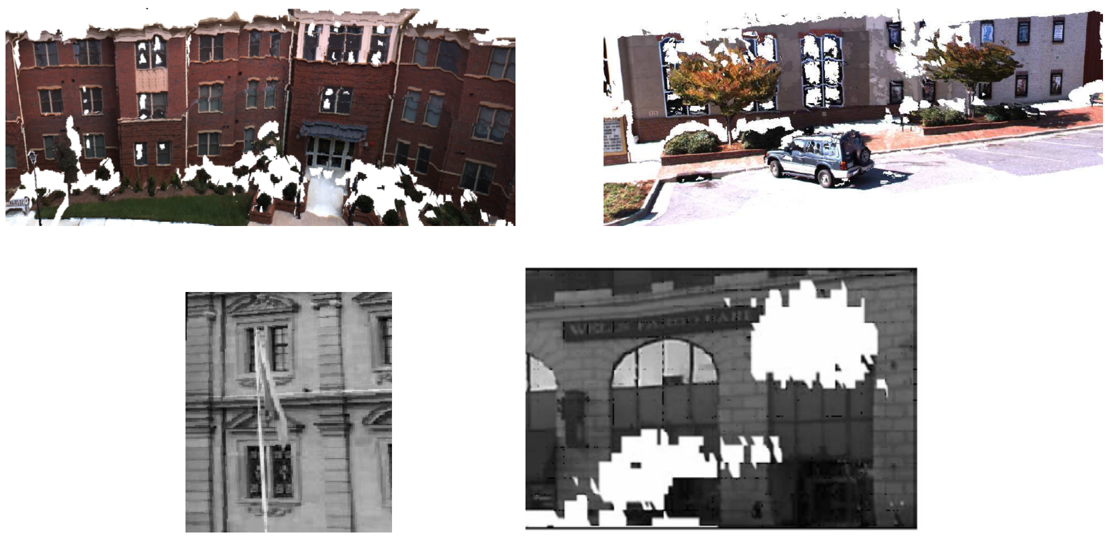


**Figure 1.2:** Some textured models from the UrbanScape Video Modeling project at UNC (top row), F. van den Heuvel's project (bottom-left) at the Delft University of Technology [164] and the 3D City Modeling project at Berkeley (bottom-right) [49]. Foreground objects either cause holes in the model or appear pasted on the texture map.

## 1.1 Problem Statement

The work presented in this thesis contributes towards an end-to-end system whereby a GPS-enabled robot will autonomously navigate around a university campus, capture a set of optimal views around the perimeter of the buildings, and build a photo-realistic 3D model of the campus. The key technical challenges in modeling are view planning and data acquisition, building a 3D model using data from sensors, the geometric alignment of 2D images onto the 3D model, and finally rendering the model given constraints on overlaps, occlusions, etc. This thesis addresses the problem of creating high-fidelity texture maps to render onto the model. A common problem during texture mapping is the presence of foreground objects such as trees, people, and signs causing *holes* in the model or being *pasted* on it. Our accomplishments include how to avoid or minimize these artifacts for a restricted class of scenes. This generally involves the twin challenges of (i) identifying problem areas and (ii) replacing them with the background building pixels.

The images in Fig. 1.2 are reproduced from results of various projects for fully automatic and real-time 3D reconstruction of urban scenes. For a review of these systems, the reader is referred to the next chapter. While very impressive in computing structure, the texture maps overlaid on these models appear corrupted – even tending to diminish the quality of geometry estimation. The holes in the texture map caused by foreground elements like trees and poles stand out. Constraints on parallelism, continuity, and symmetry do not seem to be enforced; facade elements like windows and stairs exhibit jagged edges. Finally, several trees without trunks seem to be hanging in mid-air. This illustrates how the general quality of texture maps have been overlooked in modeling algorithms.

Architectural modeling has been useful for urban planning, historical preservation, military strategy, homeland security, computer gaming, and virtual tourism. Numerous researchers have worked on different aspects of robot-based modeling of buildings; almost all of them focus on recreating the geometry of the scene by establishing correspondences between multiple views. These methods rely on redundancy of data and, although theoretically sound, can be very brittle in the presence of noise and occlusions. Another cause for failure might be the inability to navigate around obstacles or difficult terrain to capture an image from a required viewpoint.

To mitigate effects of clutter and unobserved data while modeling, i.e., make it more robust, we propose that the processing agent should use prior knowledge of building semantics to make hypotheses and inferences about the model. Although architectural styles can vary greatly and be complex, they are generally constructed according to a set of stylistic and practical constraints [36]. An algorithm that has been taught to recognize these constraints can then reason about the scene like humans do. Thus, a high-level AI module that arbitrates between all the lower level systems, such as navigation and model building, is a key ingredient for robots to handle the rigors of the real world.

## 1.2 Motivation

How can semantically enhanced sensing assist a robot to function more effectively in its domain? In particular, considering our scenario of automatic foreground removal from architectural imagery, what would annotated interpretations from the robots visual sensors contribute towards its goal of recreating a virtual scene? In addition to appearance, we wish to exploit spatial context about the domain. Images of buildings and urban environments are dominated by straight lines – often in mutually orthogonal directions. Building walls are typically planar with minor depth variations on the facade. Windows and doors exhibit characteristic shapes depending on the architecture style (such as Victorian or Gothic). Here we illustrate various scenarios in which a robot could exploit these constraints to "think and process information like humans", producing models rich in geometric and visual detail.

Given our goals of understanding and manipulating architectural imagery, we categorize our approaches as follows: (i) Discriminative approaches to bottom-up understanding, (ii) Image manipulation and synthesis, and (iii) High-level top-down reasoning.

**Bottom-up Understanding** Consider the two image patches in 1.3. One of the issues addressed in this thesis is that of automatically learning appearance models to identify foreground and background patches. Spatial and temporal cues from an image sequence are used to generate training examples of characteristic patches. We effectively combine bottom-up techniques such as frame-differencing, feature extraction, image matching, and Gaussian color models for layer discrimination.

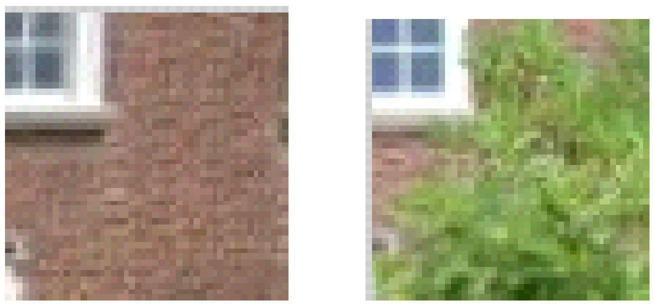


**Figure 1.3:** How do we automatically learn to discriminate between building and foreground patches?

Although not demonstrated in this work, supervised learning and object detection may also come in handy while modeling. For example, columns in front of the building can produce artifacts in motion-based approaches that register images under the planar assumption. A recognition module that could alert the registration process about this might trigger an alternate algorithm to handle these specific cases. Any of the numerous techniques [43, 44] that learn appearance models of building entities from labeled training examples could be used for this task.

**Image Manipulation and Synthesis** A test of how well an algorithm has "understood" the image is to manipulate the image pixels while still adhering to its geometric and appearance context. All processing and synthesis techniques employed in this thesis are strung together by the goal of automatic foreground removal. We remove the artifacts typical of modeling algorithms (Fig. 1.2) by masking out the foreground layer and filling it with the background. A popular approach to hole-filling in images is known as inpainting which propagates pixels from the filled-in regions to the missing portions of the image. We build upon inpainting techniques and better constrain the problem using additional cues such as temporal or high-level semantic information. We show that this is necessary for inpainting architectural scenes that exhibit a lot of structure in the form of strong lines and regular patterns. Other forms of synthesis we use include reconstruction or inference of missing features, photometric alignment, etc.

**Top-down Reasoning for Building Facades** Consider the image of a building facade in Fig. 1.4. Given enough views captured from side to side, one can use parallax information to separate out the foreground tree and recover the background. However, even from a single image, humans are adept at "mentally scrubbing" away the tree and envisioning the appearance of the obliterated regions. What kind of priors do we use to make these inferences? Firstly, we note that windows on a facade resemble each other and often appear as part of a larger grid. As we group these like elements together, disruptions in the pattern suggest some form of foreground occluder behind which windows are likely to be present. The blotches of white amidst the green leaves provide further evidence. Thus the grid prior allows us to hypothesize both the appearance and location of occluded windows on the facade. Other semantic attributes such as brick texture, facade boundaries etc. can also be extracted. Algorithms similarly equipped with prior knowledge of architectural styles and patterns can improve robustness and efficiency. This thesis describes techniques to encode, discover, and exploit these relationships for extracting high-level semantic information about the imaged facade.

One recent form of encoding architectural constructs in computer graphics has been to procedurally model elements [173, 114] by a set of grammar rules specified by the user. Starting from a *basic shape*, the grammar is used to derive a 3D layout of a building consisting of simple shapes with attributes. While there are many formal representations of space and design in architecture, the most popular approach adopted in the graphics community has been that of split grammars. Certain basic shapes with attributes undergo a series of decompositions with each

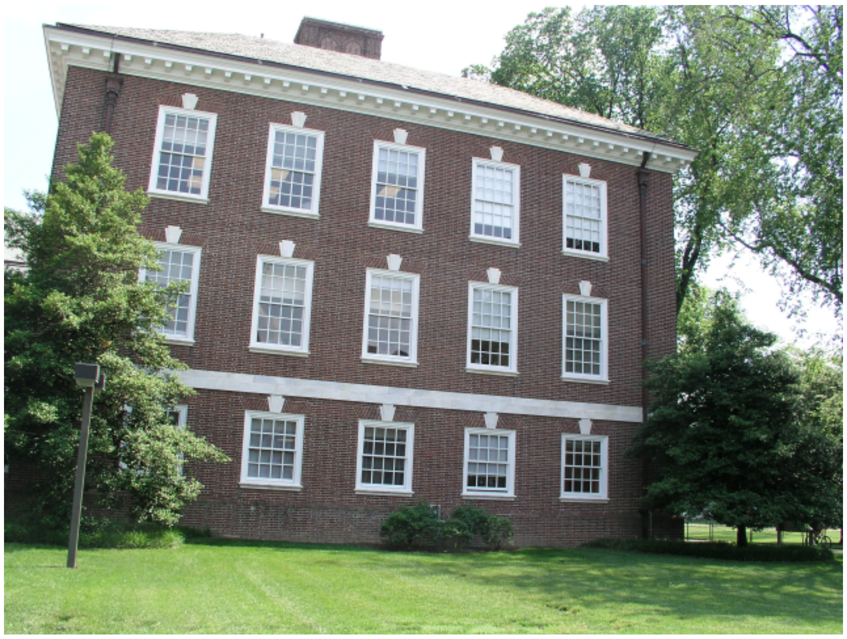


**Figure 1.4:** How do we reason about occluded windows from this image? Prior knowledge that windows often appear as grids allows us to extrapolate. The blotches of white behind the tree provide further evidence.

*split* rule. High quality 3D worlds consisting of buildings have been created from a very manageable number of shapes and rules [114]. While going from a grammar to 3D models or images have shown promising results, the reverse process, i.e. going from an image to the set of grammar rules that derived it, has not received as much attention [6, 115].

In compiler parlance, parsing is the process of analyzing an input to determine its grammatical structure with respect to a given formal grammar [4]. It usually transforms a series of tokens into a data structure that captures the hierarchy of the input. Parsing an image of a building would thus amount to concisely describing the facade by a set of pre-defined rules and estimating its attributes. For example, a building parser ought to be able to describe the window shown in Fig. 1.5 as a square, split horizontally and vertically with some thickness. It should also recognize

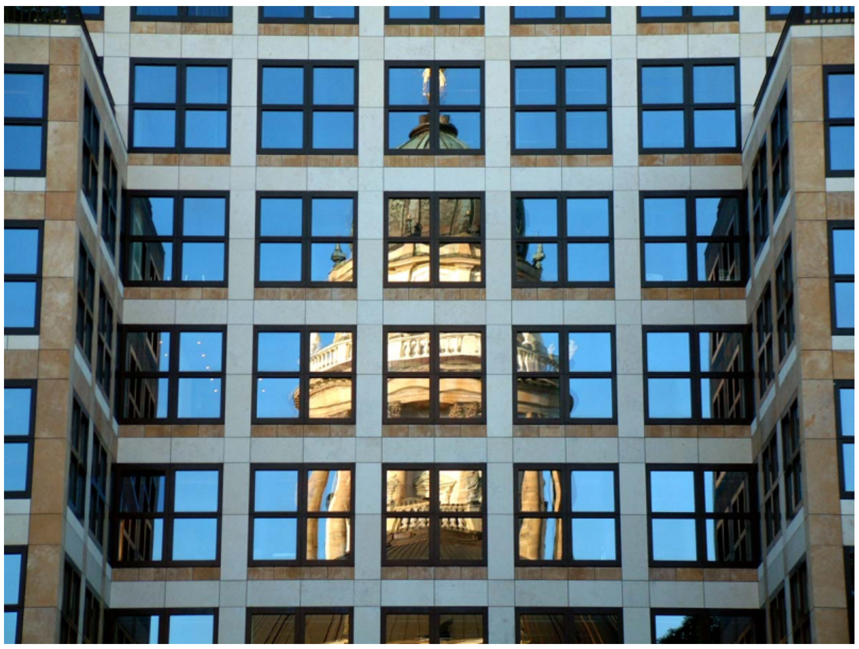


**Figure 1.5:** What is building and what is reflected within the window?

the reflections within the building as being generated by an outlier process.

## 1.3 System Overview

Figure 1.6 gives an overview of the various components and its relationships within the system. The GPS-enabled robot, equipped with an aerial map of the campus layout, takes user-input about the building to model. In outdoor environments, wireless communication via a hand-held PDA could facilitate this interaction between the user and the robot. The Planning Module first plans a set of views that need to be captured from around the building to minimize redundancy and maximize coverage. The aerial image and a vector map of the campus layout can allow the robot to reason about visibility and foreground objects. The view positions are then passed to a path planning algorithm to generate dense way-points for the robot to follow.

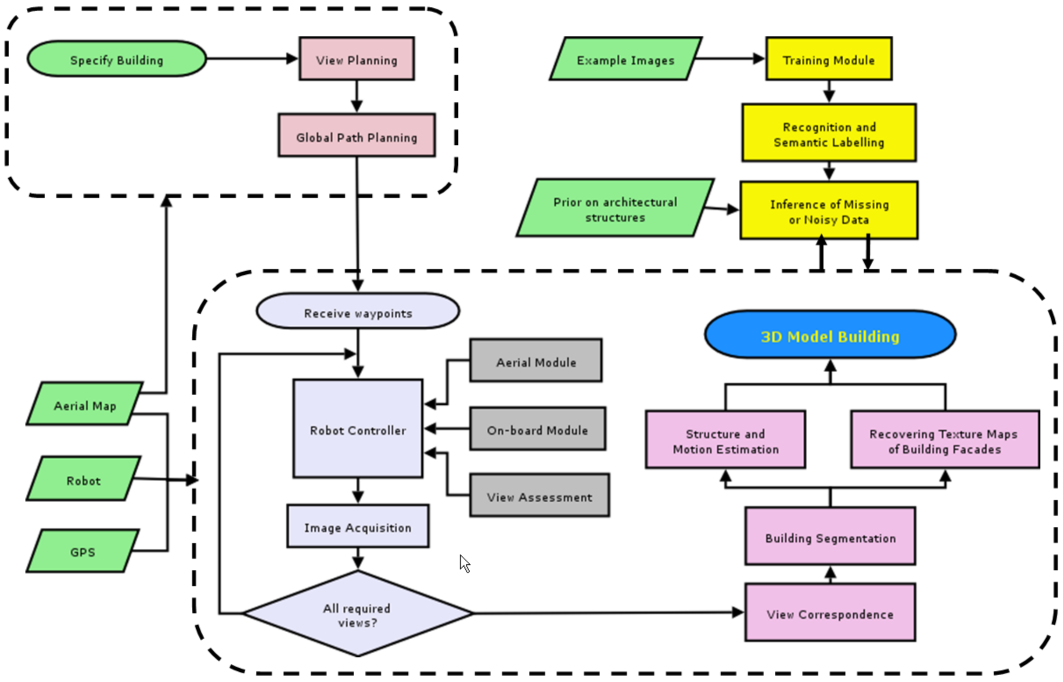


**Figure 1.6:** Various components of the robot-based modeling system.

Both of the planning systems have an in situ component that could be called during operation. While navigating, an Assessment Module must constantly monitor the on-board views on the robot, not only to judge the safety of the terrain in its vicinity (On-board Module) but also evaluate the quality of the acquired images and adapt accordingly. For example, unforeseen occlusions, out-of-focus images, or other optical constraints should trigger appropriate adjustments to the initial plan. During navigation, it is vital for the robot to localize itself on a global map. The Aerial Module keeps track of changes in the robot's position by analyzing satellite imagery and GPS readings. These three components (gray boxes) guide the robot through its controller.

Once all the images are acquired, they are passed to the vision modules for 3D model building. A first step is to establish correspondences between the tens or hundreds of views to restrict processing on a few related pictures at any given time. Pose annotated images should assist in this task. After a preliminary segmentation to separate out non-building pixels, sets of images are processed to recover geometry and illumination details for the 3D digital model.

All urban modeling systems are comprised of the above modules to some degree or other. A key novelty of our proposed system is the Semantic Module that is designed to oversee all other robot and vision tasks. This module contains prior knowledge about the nature of architectural environments. When factors such as noise, missing data (either due to insufficient views or occlusions), and other ambiguities cause the "closed-form" solutions to break, the restricted nature of the robots domain should allow plausible interpretations of the data. Closing the loop with semantic feedback is crucial for recovering from deadlocks and other impasses.

The first issue to resolve is how this knowledge is represented in the semantic database. One obvious choice would be to train the semantic module from exemplar images of buildings and other elements and represent it in some multidimensional space conducive to efficient classification, detection, and recognition. While this approach has been shown to effectively learn appearance models, other topological and functional constraints can be hard, if not impossible, to learn from examples. For spatial configurations, we achieve this by using grammars to specify rules of symmetry, structural coherence, and so on. From this description, a robot must then be able to associate what it sees with semantic keywords from the database.

## 1.4 Thesis Contributions

The previous section describes the end-to-end robot-based modeling problem, comprising of sub-tasks from planning and autonomous navigation to modeling. This dissertation will focus on some of those aspects that have received less attention in previous work. Specifically, we make the following contributions:

- We describe how a "birds eye-view" provided by satellite imagery can be used to assist in view planning and robot navigation. This includes methods to localize the robot on its path given erroneous GPS readings as well as guiding the on-board module in anticipating the nature of the path ahead of the vehicle. This work was developed in the context of the DARPA Grand Challenge.

- We introduce a novel spatio-temporal inpainting technique that recovers a clean texture map of partially occluded building facades from video or images. Images from a sequence are stabilized and stacked together to form a *timeline* of potential pixels that constrain what gets included in the texture map. To overcome inefficiencies of the exhaustive search procedure in classical inpainting, we train a classifier from automatically generated examples to disambiguate between foreground and background.

- Drawing the analogy that building facades are often examples of Near-Regular Textures (NRT) [66], we derive a Markov Chain Monte Carlo (MCMC) approach to discover such patterns from images. A Markov Random Field (MRF) model for NRTs and lattice structures is defined; entities in the image are grouped together based on its adherence to this model.

- The texture discovery is applied to single images (under perspective or rectified) of building facades to identify windows that appear in a grid pattern. Parameterizing this pattern provides a "window" towards semantically understanding the rest of the facade. For example, the location of occluded windows can be predicted or the boundaries of the facade can be segmented out by robustly learning appearance models of the brick texture between the windows.

- To develop the theme of *image parsing*, we show that the inside of windows can be adequately described by split grammars [173]. The MCMC framework is extended to take the detected windows as input and infer the grammar rules that best explain the subdivisions within the window. These semantic descriptions can then be used for outlier removal and image synthesis.

- We develop a virtual scrubbing technique that uses the discovered texture to automatically detect and seamlessly remove unwanted foreground elements from single images taken in urban settings–e.g., trees or people in front of buildings. Without motion, the key assumption is that the background is strongly structured, which allows automatic detection of occluders as outlier pixels in near-regular textures and their replacement via a robust subspace reconstruction process driven by tile appearance statistics.

## 1.5 Thesis Outline

In Chapter 2, we give an overview of previous literature on various aspects of urban modeling that are covered in this thesis. These cover a whole gamut of techniques from computer vision, machine learning, computer graphics, and robotics. We describe some of the successful urban modeling projects, specially noting their attention to the quality of texture maps along with 3D structure. The literature review also includes work that has actively tried to incorporate architectural semantics into modeling. Chapter 3 details how aerial imagery can be utilized as an *a priori* map to assist the robot in view or path planning as well as vehicle localization. This overhead data can be used in preprocessing to generate a global path, or during navigation to adapt to unpredictable situations. Subsequent chapters primarily focus on the computer vision and image processing algorithms to generate texture maps of the building facade. Chapter 4 introduces a spatio-temporal inpainting algorithm that takes a sequence of images captured around a building and automatically produces a "clean" mosaic of the facade with foreground objects like trees, people, etc.

removed. Both motion and appearance constraints are used to detect foreground and inpaint them with the building background.

Chapter 5 begins to address the issue of how one might remove foreground occluders from a single image without any motion or parallax constraints. The underlying assumption used is that building facades are examples of near-regular textures [66] where each window is a texture element. An MCMC algorithm for lattice discovery is presented and results are shown on both synthetic as well as building images. Chapter 6 uses the discovered window grid to infer various semantic properties such as facade extent, occluded windows, and so on. Results are shown of images that have been scrubbed clean of foreground objects or graffiti and replaced with a low-dimensional representation of the facade texture. Finally, we conclude with a summary of contributions and discussion of limitations and future work.

# Chapter 2

# BACKGROUND

Modeling of urban and architectural environments has been studied for several years. Government agencies have traditionally used it for development planning and to study effects of climate, pollution, public safety, and military strategy [69]. Early planners and architects used materials like plastic, foam, wood or paper to create models from elaborate manual measurements. Over the last couple of decades, Computer Aided Design (CAD) tools have allowed them to quickly create realistic digital models for editing, planning, and visualization. The success of recent products like Google Earth, SketchUp, and Microsoft Virtual Earth has enabled urban modeling to be done in a distributed, voluntary, and "wikified" manner. New consumer applications like street maps and automatic GIS tagging of photos are sprouting up, making this an exciting area of research.

Several techniques from photogrammetry, computer vision, and computer graphics have contributed towards automatically creating and visualizing digital models of cities. Some of the relevant methods are Camera Calibration, Structure from Motion, Shape from Silhouette Contours, Stereo Correspondence, Range Scanning, and Image-Based Rendering. These work well with single or a small collection of objects, but often don't scale too well when applied to large-scale urban modeling. Additional constraints about urban scenes are usually incorporated to overcome some of the numerical instabilities. Existing approaches generally involve 3 steps: (i) Build a 3D model using data from the sensor; (ii) Align 2D images onto the 3D model; (iii) Texture the model using the aligned image.

As mentioned in the previous chapter, this thesis describes components of an urban modeling system. The literature review is split into 3 parts based on the contributions and techniques investigated in this thesis: (i) urban modeling systems and texture map acquisition, (ii) facade enhancement based on low-level, mid-level and high-level semantic cues, and finally (iii) robot localization, planning, and navigation using aerial imagery. We specifically describe projects that have attempted to bring in domain semantics about architectural elements. Are these constraints justified? How successful were they? Some of the systems here have the sensors placed on a mobile platform such as a vehicle or cart, which is then driven around by a human. Though not classified as robot-based acquisition, they do have to handle the localization problem. We do not claim that the algorithms we develop in this thesis compete with urban modeling systems reviewed here. Rather, our techniques are designed to be complementary in nature, fusing domain semantics to improve the reconstructed models.

The rest of this chapter proposes to set the background for the thesis. Where appropriate, related papers and techniques are reviewed in the relevant chapters.

## 2.1 Urban Modeling and Texture Map Acquisition

Urban modeling makes the fundamental assumption that man-made structures are likely to appear as smooth surfaces and solid objects rather than disconnected points or lines. Consequently, it is usually represented as "layers" [36], which is a parameterized 3D surface with a boundary, and possibly a texture and depth map as well. Calibration, stereo, and SfM algorithms [64] are used to extract the geometrical structure of these surfaces, while problems such as reflectance modeling, segmentation, object detection/recognition, and grouping primarily focus on the mapping between pixel intensities and the surface. These two paradigms of model-based and view-based representations [112] are coupled together to effectively describe a 3D scene.

A texture can either be a detailed pattern that is repeated many times to tile the plane, or more generally, a multidimensional image that is mapped to a multidimensional space. Texture mapping in graphics is a shading technique for image synthesis in which a texture image is mapped onto a surface in a 3d scene, while avoiding aliasing and other artifacts [67]. It also allows additional details such as surface perturbations, transparency and specularity to be added, with only modest increase in rendering time. In this work, we are primarily concerned with recovering a “clean” mosaic of the facade that can subsequently be used as a texture map for each planar face of the building facade.

### 2.1.1 Modeling from 3D Range Sensors

Active range sensors directly measure the depth of objects by emitting a laser pulse and precisely measuring the *time of flight* to return to the source. Airborne laser scanning in conjunction with GPS and IMU sensors have been used in city modeling for over 4 decades [182, 17, 41, 113, 180, 60, 118, 177]. Satellite imagery facilitates texture mapping of these models for interactive fly-throughs. LiDAR (Light Detection and Ranging) data is in the form of a point cloud which needs to be filtered for noise and registered with imagery. Automatic techniques differ mainly in their segmentation that groups data points into coherent structures. For airborne systems, 2D footprints available from imagery, GIS, or CAD models can assist in this task.

From the ground, registration generally involves matching straight lines in the image to those in the 3D model. However, these can be ambiguous as edges don’t necessarily correspond in the two modalities. The advantage of a ground-based system is in its ability to capture highly-fidelity close-up views that can facilitate walk-throughs [50, 49]. Other vehicle systems that capture 3D data and texture data at ground level can be seen in [7, 150, 69, 181, 102]. One drawback of laser scanners is that they can be cumbersome and expensive. In comparison, cameras decrease size, weight and cost while increasing flexibility — all crucial design issues in robotics.

The AVENUE (Autonomous Vehicle for Exploration and Navigation in Urban Environments) [7, 8] project at Columbia targets complete automation of the urban site modeling problem – from navigation on a mobile platform to geometrically and photometrically accurate models. In terms of scope and goals, their project is most similar to ours. However, they make use of an expensive 3D range sensor (Cyrax laser scanner returning 1K by 1K range samples with a spatial resolution of a few centimeters) to recover dense and regular geometry of the scene. Stamos in his PhD thesis [146], develops algorithms to create 3D solid models of buildings from the point cloud and automatically register range and image data-sets. Matching between range scans requires manual intervention while range-image registration assumes that a sufficient number of common features in the 3D and 2D data can be extracted. Nevertheless, this registration overcomes limitations of fixing the relative positions of the two sensors [50, 181, 128, 139]. Images can be acquired from any vantage point.

In extensions to this work, Liu et al. [102, 103] attempts to exploit all possible relationships between 3D range scans and 2D images by performing 3D-to-3D range registration, 2D-to-3D image-to-range registration, and structure from motion. Two independent pipelines estimate a dense and sparse 3D point cloud from the range data and images (using classical SfM) respectively. A subset of images are then automatically registered to the 3D point cloud, upon which the alignment for the complete set of images can be determined to integrate all data into a single coordinate system. While intuitive and general, no attempt is made to handle missing data in either the images or the range data.

Frueh and Zakhor [50, 49] present an automated method for fast, ground-based acquisition of large-scale 3D city models. Experimental setup consists of a truck equipped with one camera and two fast, inexpensive 2D laser scanners, mounted horizontally and vertically. Monte Carlo Localization utilizing either an aerial image or a Digital Surface Map (DSM) is implemented to determine the global pose of the vehicle. They specifically cite the difficulties of producing detailed textured facades; registering range data to camera images, reflections caused by glass on windows, and holes due to foreground elements are all problematic. After subdividing the data into smaller chunks, points are classified into a foreground and background layer. Missing data in the background layer are filled in by interpolation. Holes in the image texture are also filled by interpolation in homogeneous regions, or by a copy-paste method otherwise. These heuristics are not well-defined, and zoomed in views reveal the shortcomings of this crude hole-filling.

### 2.1.2 Modeling from 2D Images and Video

One of the first urban modeling projects was the Facade system [32, 31] developed by Paul Debevec, which could render highly realistic and visually compelling models. A user works in an interactive environment with a sparse collection of images, manually specifying a model structure and registering its boundaries with actual edges in the image. The model is composed of volumetric primitives (instantiated by the user) such as rectangular and triangular prisms that can be described by only a few parameters. Full reconstruction is carried out in a post-processing step to recover the structure and camera parameters by minimizing the re-projection error between model and image edges. To make the rendering more realistic, they use the recovered camera pose to project different images onto the model depending on the user's viewpoint. This view-dependent texture mapping interpolates between nearby camera positions to render a virtual image that better captures the perspective and shading effects from that viewpoint. Finally, a depth map that models deviations from planarity of each face in the structure is computed using a dense stereo matching algorithm.

The ImageModeler software from Realviz [132] provides a similar interactive system and is very popular in the film industry. Despite this, manually guided reconstruction is not feasible for large urban environments with tens or hundreds of buildings. Domain knowledge allows these systems to model architecture using a few primitive *blocks*; these assumptions could have also been used to handle occluding elements or fine-grained facade geometry. The texture mapping does not prevent foreground objects such as cars and trees from being pasted on the model. Depth maps with constraints that reflect the regularity of building facades might avoid such spurious estimates of geometry.

The City Scanning project at MIT [157, 156] uses several thousand *pose annotated* spherical mosaics to produce a textured CAD model of an urban environment. A coarse model of structure is extracted by edge histogramming across multiple images to identify and localize prominent vertical facades. Their method of texture map recovery and relief estimation is detailed in [153]. Facade extraction consists of first normalizing all projected images to have the same mean luminance. An Environment Mask (EM) that accounts for global occlusion, an Obliqueness Mask (OM) that reflects the perspective distortions, and a Correlation Mask (CM) to identify foreground such as trees are then estimated. These are combined in a weighted averaging scheme to produce a "consensus image" as the facade overlay. To model some of the finer relief details, heuristics on geometric and periodicity constraints are used.

Although they were among the first to tackle foreground removal from texture maps, some of the heuristics used are vague and the resulting texture maps often appear "washed out". The luminance levels of images captured over several hours exhibits too much variability, and an independent per-pixel averaging scheme still results in inconsistencies across the mosaic. The buildings they model are fairly simple box-like structures, and it is not clear how effective their structure and texture recovery would be on more complicated data sets.

A great deal of work has been done by the Visual Geometry Group at Oxford towards fully automatic construction of graphical models of scenes – both from single images [28] as well as a short image sequence [46]. They approach the problem using classical Multiple View Geometry techniques [64] of matching 2D point features in pairs or triples of images to achieve a projective reconstruction. This can then be upgraded to a Euclidean one by means of auto-calibration techniques [127]. For architectural models, a RANSAC estimator then fits planar faces, along with some modeled perturbation, on the recovered set of sparse 3D points and lines. When there is a lack of sufficient texture, these models can appear crude and oversimplified.

Dick [35, 36] introduced the concept of using architectural semantics through high-level recognition of building entities to improve the models. Similar to the interactive systems, a building is described as a set of walls and parameterized primitives such as doors or windows. Prior distributions are defined on the values of these parameters based on architectural norms. A Markov Chain Monte Carlo framework is then used to generate semantically labeled building models and evaluate its likelihood with the image data. The novelty of their method was in combining the classic problems of structure estimation and object recognition to reinforce each other. A drawback is that the models are highly tuned to the respective scene. Somewhere in the middle of this spectrum of automatic methods ranging from specialized architectural models to fitting piecewise planar patches over multiple views [155] is the method of Werner and Zisserman [169, 168]. A coarse planar model of the planes and its delineations in the scene are first computed followed by fitting refined polyhedral models on windows and other indentations. Once again, semantic information is confined to improving the structure with less emphasis on the texture mapping.

The recent UrbanScape project [5, 52] headed by researchers at University of North Carolina-Chapel Hill and University of Kentucky attempts to develop a fully automated system for accurate and real-time 3D reconstruction of urban environments from video streams. Data acquisition is through multiple synchronized video cameras on a vehicle driven in an urban environment, along with GPS and INS sensors to georegister each frame. An extended plane-sweeping stereo algorithm computes an initial depth map on the GPU for each frame. Adjacent depth maps are then fused together while enforcing visibility constraints to produce a more accurate and economical representation of the scene. The final step outputs a triangular mesh modeling planar regions and an image to be used as a texture map. While impressive in terms of scale, efficiency and quality of structure, the texture maps appear "wrinkled" as illustrated in 1.2.

Cornelis et al. [23, 22] describe a system to extract simplified and textured 3D models of cities from video sequences captured by a stereo camera. They present a practical implementation of *cognitive feedback loops* in a city modeling framework, tightly integrating 3D reconstruction and object recognition. The recovered ground plane and camera parameters guide a recognition module searching for cars in the image. The detection results, on the other hand, are used to remove the cars from further processing to avoid artifacts in the texture mapped facade structures. Placeholder models of virtual cars are inserted into the model for heightened realism. This is a very interesting technique that propounds some of the ideas developed in this thesis.

## 2.2 Facade Enhancement

The complicating factor that motivates this aspect of the thesis is the possible presence of other, unknown objects in the scene between the camera and building plane—e.g., trees, people, signs, poles, and other clutter of urban environments. Without explicitly recognizing and removing them, these foreground objects may be erroneously included in the building appearance model. Hence, a "clean" facade mosaic is one without such non-building features; conversely, we call a mosaic with foreground artifacts "polluted." There are two major problems to be addressed here. First, which areas of the scene are problematic, if any? Second, how to actually remove foreground objects to reveal the building structure behind them?

With the exception of the City Scanning Project [156] at MIT and the 3D City Model Generation work at Berkeley [50], none of the systems reviewed in section 2.1 try to address issues of missing data, occlusions, perspective or other factors that degrade the visual quality of the texture maps. Even these two systems employ very crude methods of blending and interpolation with no guarantees about correctness. Debevec's view-dependent texture mapping [31] gets around this issue without explicitly handling the occluders. In this section, we review various techniques that might be used to automatically "touch-up" the acquired imagery. Some of these fall under the bracket of image editing and enhancement while other algorithms that assume a priori knowledge of the architectural domain can be classified under Image-Based Modeling. The latter class typically incorporates semantic constraints of a building to reduce its degrees of freedom while manipulating image content.

### 2.2.1 Identifying Foreground from Sequences

Given a sequence of images scanning the building, the most obvious cue to identify problem areas is parallax due to the different depths of the facade and the unknown foreground objects as the camera translates. Under the assumption that the building plane accounts for the majority of pixels in the sequence, with robust methods we can estimate the dominant motion of the building and stabilize it against the camera motion. This makes foreground objects virtual moving objects in an otherwise motionless scene, suggesting existing techniques like *foreground subtraction* [42, 119, 135] or *layer extraction* [166, 76, 174, 170]. However, many of these approaches either assume that the moving objects are relatively small or quickly moving, facilitating temporal median filtering [166, 119, 170, 42], or that the objects to be removed are manually identified once in order to segment them later [81, 135]. Since the nature of potential foreground elements between the sensor and the building is unknown *a priori* and may cover large regions, such assumptions are disqualifying.

Several promising papers on foreground/background layer segmentation [26, 148] were recently published. Motion, color and contrast cues are probabilistically fused together with spatial and temporal priors to classify each pixel into one of two layers. However, both take a temporal estimation approach to full-motion video (and assume a fixed camera, though after camera stabilization our problem is similar to theirs), whereas we present a technique that also works with sparser data—i.e., images taken several seconds apart.

### 2.2.2 Inpainting for Object Removal

The second problem is what to put in the building texture map in the "gaps" left where foreground objects are masked out. In order to gauge possible approaches, it is useful to distinguish between situations in which the pollution of a particular mosaic region is due to a foreground object being present in a minority, majority, or unanimity of views over the sequence. We call the set of views of a particular point or patch in the mosaic its *timeline* (this is defined more precisely later). Timelines in which the foreground object pollutes a minority of views are amenable to cleaning via simple outlier removal through median filtering as mentioned above. In the other two cases, some kind of *inpainting* would seem to be required.

There is a very large literature on inpainting for image restoration or object removal using techniques such as PDEs and wavelets [14, 21], exemplar-based matching [40, 27], and space-time or video completion [74, 87]. These methods offer a principled way to remove large foreground elements using spatial contextual information from the rest of a single image or temporal context from preceding and succeeding images in a sequence. PDE-based methods are effective for thin structures and text overlays, but can generate blurring artifacts on larger missing regions. Patch-based exemplar techniques [27, 74, 38, 88] that augment synthesis with a prioritized fill ordering have been more successful than the pixel-based synthesis of [40, 99, 10]. All the above approaches proceed to fill in missing pixels in a greedy manner. Recently Sun et al. [149] and Komodakis [84] proposed inpainting as a global optimization problem that can be solved using belief propagation. Some form of interactive guidance [38, 149, 124] has also been accepted in the inpainting community.

Here we review the inpainting algorithm due to Criminisi, Pérez, and Toyama [27]— henceforth referred to as *CPT inpainting*. In addition to serving as a good example technique for the class of non-parametric exemplar inpainting algorithms, we use it as a baseline method and build upon their framework to clean up occluded building facades. This will be detailed in subsequent chapters.

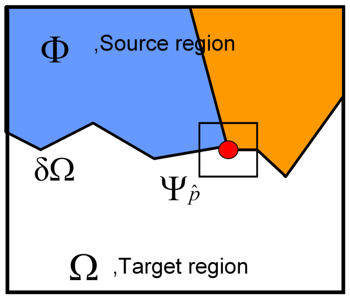


**Figure 2.1:** Source region $\Phi$, target region $\Omega$, target boundary $d\Omega$, target patch $\Psi_{\hat{\mathbf{p}}}$ (from Criminisi *et al.* [27])

As diagrammed in Figure 2.1, an empty target region $\Omega$'s pixels are filled from its border $d\Omega$ inward by copying square image patches from a source region $\Phi$ to target patches $\Psi_{\mathbf{p}}$ centered on $\mathbf{p} = (x, y) \in d\Omega$. Given the next target patch $\Psi_{\hat{\mathbf{p}}}$, an *exemplar* patch $\Psi_{\hat{\mathbf{q}}}$ is selected from $\Phi$ and pixels are copied to the unfilled portion of the target patch $\Psi_{\hat{\mathbf{p}}} \cap \Omega$ from the corresponding part of $\Psi_{\hat{\mathbf{q}}}$. $\Psi_{\hat{\mathbf{q}}}$ is chosen as the source patch with the minimum distance $d$ (e.g., SSD) between it and the already-filled part of the target patch $\Psi_{\hat{\mathbf{p}}} \cap (\mathcal{I} - \Omega)$. As inpainting proceeds $\Omega$ shrinks while $\Phi$ remains constant, leaving a band of filled pixels $\Omega_0 - \Omega_t$ at step $t$. Note that $\Phi$ can be smaller than $\mathcal{I} - \Omega_0$.

A priority function $P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p})$ sets the order in which patches along $d\Omega$ are filled. $C(\mathbf{p})$ is a *confidence* term that measures the amount of reliable information around $\mathbf{p}$ with the formula $\sum_{\mathbf{q} \in \Psi_{\mathbf{p}} \cap (\mathcal{I} - \Omega)} \frac{C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}$. Initially, $C(\mathbf{p}) = 0 \ \forall \mathbf{p} \in \Omega_0$ and $C(\mathbf{p}) = 1 \ \forall \mathbf{p} \in \mathcal{I} - \Omega_0$. When pixels in $\Psi_{\hat{\mathbf{p}}} \cap \Omega$ are filled in, their confidence values are updated from 0 to $C(\hat{\mathbf{p}})$, having the effect of preferring sections of $d\Omega$ that were filled earlier vs. later. $D(\mathbf{p})$ is a *data* term proportional to the dot product of the tangent vector to $d\Omega$ at $\mathbf{p}$ and the gradient vector $\nabla_{\mathbf{p}}$ with the maximum magnitude in $\Psi_{\mathbf{p}} \cap (\mathcal{I} - \Omega)$. This encourages the extension of linear structures (commonly found on man-made objects) by boosting the priorities of patches with a strong edge "flowing into" them—as, for example, in Figure 2.1.

### 2.2.3 Spatio-temporal Synthesis

Pure spatial inpainting is necessary in the unanimously-occluded case where the background is not in the timeline: only context from the rest of the image can guide the filling process. However, an image sequence facilitates temporal search as well to identify which view, if any, is of the background. Combining spatial and temporal search has been done in video inpainting and space-time texture synthesis [83, 82, 89, 171, 3]. The key technical issue, especially with video sequences, then becomes making this search as efficient as possible without sacrificing accuracy. While these methods try to remove objects from each frame of a sequence, generating texture maps requires that the background in each frame be (automatically) identified and pieced together into a panorama.

Two papers that also attempt to fill in occluded areas in image sequences through a modified version of CPT inpainting can be found in [176] and [121]. With regard to [176], they manually identify foreground pixels in all frames. This can be nontrivial in outdoor scenes for objects such as tree limbs and leaves. Furthermore, their running times are quite large (not even including the manual interactions required), taking from hours to days due to the 4-D search problem their lightfield technique engenders. The work in [121] assumes a fixed camera and uses motion alone to detect foreground objects. While their focus is on synthesizing moving but compact foreground elements, we are concerned with recovering large and disconnected portions of the occluded background.

### 2.2.4 Modeling Semantics for Building Facades

Given just a single static image, how can we separate the foreground and background layers to reveal more of the facade? Without parallax, we need another cue to differentiate between the two layers. A common simplification encountered in urban modeling research is that the background is strongly structured enough that it exhibits characteristics of a regular or near-regular texture [91] and dominates the image. This is frequently the case for close-up images of sections of buildings with brick patterns or window grids. The basic idea is that by discovering these textures automatically and collecting statistics on tile appearance, foreground objects can be automatically segmented as texture outliers allowing either the reconstruction or replacement of the tiles with unoccluded patterns elsewhere in the texture.

Most urban modeling research such as Facade [32] and UrbanScape [126] approach the problem from a purely geometric view. Our approach is motivated mainly by the work of [35] which tried to infer the semantic properties of buildings to assist in modeling. Mayer and Reznik [110] describe a similar framework that makes use of some prior learned appearance models of entities such as windows and so on. However, the non-hierarchical labeling in these algorithms makes high-level analysis and manipulation difficult. In the graphics community, split grammars were introduced by Wonka et al. [173] to formally describe the derivation of architectural shapes for procedural modeling. Each production of the grammar corresponds to the decomposition of a basic shape into another shape with derived attributes. This was later used to great effect in [114] for synthesizing realistic 3D models of buildings in various architectural styles and eras.

The dual Computer Vision problem of going from an image to the grammar that derived it is hard, unless some simplifying assumptions are made. A probabilistic sampling approach to infer the attributes (such as split ratios, color, etc.) given a user-specified grammar was detailed in Alegre and Dellaert [6]. They assume that the whole building facade can be represented as a tree of partitions with leaf nodes corresponding to building structures. While the general framework shows promise, it requires the user to come up with a custom grammar for each new building. Moreover, enforcing the grammar rules to the entire facade limits the applicability of the technique to extremely regular structures. Inconsistencies and occlusion cause the algorithm to break.

A few researchers have tried to combine the grammar-based procedural modeling with the concept of *parsing* images of buildings [12, 17]. Recently, Mueller et al. [115] presented an impressive system that takes a single rectified image of a building as input and computes a 3D geometric and semantic model with much greater visual quality and resolution. Mutual Information is used to detect repetitions on the facade structure and subdivide it into tiles. Each tile is matched with a library of 3D architectural elements to identify the region type upon which a 3D model and shape grammar rules can be generated. The model can also be edited by the user.

Most of the above methods use very specialized models and show examples on a restricted set of images. They require that the facades contain repetitive elements exhibiting high regularity. The case of detecting occluding elements or seeing through them is seldom handled, primarily because of stringent assumptions on the nature of symmetric patterns. A more general grouping algorithm to detect near-regular grid structures on building facades could increase robustness. Several papers have examined the problem of finding regular, planar patterns, with notable approaches based on RANSAC [137] and the cascaded Hough transform [162]. An algorithm for near-regular texture (NRT) [91] discovery by lattice growth was recently described by Hays et al. in [66]. Enhanced with architecture-specific assumptions, this class of algorithms could increase the flexibility for facade interpretation.

## 2.3 Aerial Imagery for Robot Navigation

Autonomous navigation in unstructured and unknown environments is a daunting task due to the difficulty in analyzing sensor data from on-board units, such as stereo or laser rangefinders. Recent developments make it possible and economical to acquire high-resolution aerial data of an area prior to robot traversal. Although the resolution of conventional Digital Elevation Maps (DEMs) is too limited to be used effectively for local robot navigation, low-resolution imagery (around 1 meter) is publicly available for most areas and higher resolution data – comparable to that of on-board sensors – can be obtained commercially.

In this work, we investigate the use of aerial images as prior maps for autonomous robot navigation. Overhead data is used to enhance system performance in two areas. First, a polyhedral model of a building's structure is used to compute an optimal path around the building that achieves complete visual coverage. This is called *View Planning*. Second, satellite imagery is processed to localize a robot on the road/path given noisy GPS estimates. *Localization* is especially important in urban environments where GPS readings are error-prone due to triangulation errors. The same processing of the overhead data can be shown to alleviate some of the difficulties associated with myopic navigation using on-board sensors alone. For example, dangerous turns or sharp corners not visible in the robot sensors may be anticipated in advance from the aerial view.

### 2.3.1 Planning

Most robot navigation research aims to achieve the twin goals of safety and efficiency when traversing from one location to another. With only on-board sensors, the vehicle can achieve safe navigation by maneuvering itself around obstacles. However, path efficiency will suffer due to localized planning. Only if prior knowledge about the environment is provided can the system achieve both safety and efficiency. For navigation, aerial data is a natural candidate to complement the local perception systems on the robot with a "birds eye-view".

The DARPA Grand Challenge brought the challenge of long distance navigation to the forefront. Possible solutions are determined by various factors such as waypoint spacing, extent of prior knowledge about environment, and other mission constraints [160]. According to Silver et al. [141], there are three possible categories for vehicle operation:

- Path Tracking: a fixed path through the environment is pre-computed or manually specified for the robot to traverse. Slight deviations may be allowed if on-board sensors detect obstacles [134].
- Full Exploration: no prior knowledge of the environment is given to the robot. Purely on-board sensing must be used to navigate from waypoint to waypoint [142, 70].
- Aided Exploration: rather than a single, pre-planned path, the vehicle uses a combination of prior data and its own perception system to navigate.

The boundaries are not clearly defined and are usually determined by the nature of prior information and density of waypoints.

Global planning usually involves finding a path that best achieves certain priorities like shortest traversable path, lowest risk path, or other constraints on energy utilization. A common approach based on the prior data available, is to generate a *cost map* for the environment. This is represented by a 2D grid where the value of each cell encodes the risk involved in traversing through that cell. The lowest cost path can then be computed. In the full path planning scenario, global planning occurs only once before the robot mission. In exploration, the path is continually replanned based on the perception history of the robot. In aided exploration, a combination of prior data and perception history is used. The cost maps are usually generated by annotating semantics with the environment map.

### 2.3.2 Localization

A primary task involved in equipping intelligent vehicles with autonomous capabilities is that of robot localization, which is the problem of estimating a robot's position relative to a map of its environment. Localization includes both the ability to home in on the position without any prior information of initial state, as well as keeping track of the position as the robot moves. The need for a highly accurate localization process is crucial for tasks such as map-building, path-planning and autonomous navigation [1, 34, 47].

Sensors like Global Positioning Systems (GPS) and odometry [15] have been widely used for this purpose. In practice however, GPS accuracy is heavily dependent on several factors such as the satellite configuration and multi-path errors. Line-of-sight (LOS) issues make GPS less effective in urban canyons and densely forested regions. GPS errors can routinely range from 2- to 15 meters depending on the sophistication of the unit [15].

To correct such noise in the GPS position, several map-matching approaches [116, 151] have been employed. These techniques use a digital road network and a combination of geometric and topological constraints to "snap" onto the correct road. Digital road-maps can be problematic in dense urban environments as there may be several candidate roads close to a particular location. Localization in off-road and desert terrain is particularly challenging as these maps may not be available.

Vision has recently been investigated as an effective tool to correct for such erroneous sensor data. Much research has been done in robot localization [47, 53, 138] to complement GPS or sonar readings with another on-board sensor such as a camera or laser range finder. This usually entails searching for artificial/natural landmarks in the vicinity of the GPS-estimated position for increased accuracy. Information from on-board sensors are compared to a world model to determine the absolute pose of the robot. The models might be built by hand [53] or simultaneously estimated as in SLAM (Simultaneous Localization and Mapping) [94]. These state estimation problems are effectively solved by probabilistic approaches like Bayesian inference, which recursively estimates the posterior probability density over the state space, conditioned on the data collected so far. Implementations of the Bayes filter differ in the manner by which this density is represented. Kalman filters [77] are the most widely used variant due to their efficiency, but have restrictive assumptions such as unimodal Gaussian uncertainty and linear system dynamics.

A powerful means of representing the belief state is *particle filtering* [158], also known as the CONDENSATION [72] algorithm and Monte Carlo Localization (MCL) [34]. Particle filters fall under the general class of Monte Carlo methods which are based on representing a probability distribution function by a set of random weighted samples. The 'particles' represent the distribution of the state vector in state space, and are iteratively updated after an observation. The observation model describes the likelihood of an observation given the current state. Advantages of particle filters include the ability to represent arbitrary probability densities, and applicability to converge in non-Gaussian, non-linear dynamic systems.

In the context of urban modeling that spans multiple city blocks, sensor data tagged with global position estimates are key to minimizing the combinatorics of the problem. Localization via GPS alone [181] is failure-prone for the reasons mentioned above, and methods that can reliably handle GPS-outages are necessary.

Frueh and Zakhor [50] describe a method that eliminates GPS altogether. Relative position changes are computed to high accuracy by matching successive horizontal laser scans, but small errors can cause considerable drift over long distances. Therefore, they integrate this into a Monte Carlo Localization (MCL) framework, combining additional evidence from high resolution aerial images or DSMs to ensure that the vehicle travels on roads, without cutting through corners or buildings. Georgiev and Atanas [53], as part of the AVENUE project, describe their localization system that uses GPS and odometry in open spaces, but resorts to vision-based localization when the robot is close to other tall structures that obstruct a clear view to the satellites. Visual pose estimation consists of matching linear structures in an image of a building taken by the on-board camera with a model. The model is a manually constructed small-scale database of various facades and planar regions in the environment. When the environment is complicated, these models can be hard to construct, not to mention defeating the purpose of automatic modeling in the first place.

## 2.4 Summary

We have reviewed different urban modeling systems that integrate techniques from photogrammetry, computer vision and computer graphics to create digital models of urban environments and buildings. The algorithms chosen are dictated by the type of sensors, automatic- or manually-guided acquisition, mounted platform, human or mobile agent, and intended audience. We note that while much attention has been given to recovering the geometry of the scene, the issue of producing high-fidelity texture maps has often been sidelined. Inpainting is one possible approach to cleaning up mosaics of building facades, and we reviewed the hole-filling literature in static images as well as video sequences. Since these systems work at the mid-level of using context from the neighborhood of a pixel, gross errors can result due to violations of the building structure. We reviewed papers that have tried to extract higher-level semantic knowledge from images of buildings. Our goal is to “marry” inpainting with semantics for automatic foreground detection and removal in urban scenes – even from a single image.

Since our system is robot-based, the mobile-platform must first determine an optimal path around the building and keep track of its position as it navigates through the environment along this path. The challenges of planning and localization for navigation are described, followed by a review of Bayesian filtering techniques that have been popular in robotics. Finally, we briefly discussed localization techniques used by two successful urban modeling projects.

# Chapter 3

# ANALYSIS OF AERIAL IMAGERY FOR ROBOT NAVIGATION

This chapter investigates the use of aerial images for enhancing the performance of a robot in both planning and localization. Firstly, polyhedral maps of a university campus with building outlines are used to compute a visibility map around the structure. Locations for image acquisition are then planned around the building to ensure maximum visual coverage for mosaicing. Secondly, we develop an algorithm to trace traversable paths in the robots vicinity. Aerial images at 1 meter resolution were used for both urban and desert environments. There are two scenarios under which our technique may be useful. If GPS estimates are unreliable but the robot is known to stay on the path, our road tracer can effectively *localize* the robot on the aerial map. Conversely, if GPS estimates are reliable while the robot is prone to small excursions away from the intended path, our algorithm can use the aerial image to *guide* the vehicle back onto the road.

The work presented here acts as a bridge between the larger framework of urban modeling and the more specific image processing on building facades described in subsequent chapters. Both planning and localization are central to the task of moving the robot around from one waypoint to another for image acquisition. To achieve the goals of safety and efficiency in navigation, we advocate Aided Exploration where the vehicle uses a combination of on-board perception modules as well as prior data in the form of aerial maps or images. Not only do aerial images provide evidence of safe and unsafe regions, but certain aerial landmarks such as vegetation or roads also visible from the ground view could be used for absolute positioning of the robot; this could prevent various hazardous situations when GPS estimates are unreliable or unavailable.

We first describe a randomized approach to view planning for a single ground robot scanning a building perimeter to recover a series of texture map mosaics. This algorithm generates paths that simultaneously address coverage and quality (i.e., real-valued distance and foreshortening factors) – very important to produce high fidelity texture maps. This is part of the global path planning which is run prior to the robot embarking on its mission. We then address the mid-level problem of tracing the road from satellite imagery for curve and corner anticipation, or to localize the robot under erroneous GPS estimates. This was developed in the context of the DARPA Grand Challenge, and many of the experiments reflect this bias. However, the underlying principles of planning and localization using aerial images apply equally well to navigating around buildings. We do not describe the details of our on-board road following algorithm in this thesis. Interested readers are referred to [129, 130].

## 3.1 View Planning

Our algorithm for facade texture map acquisition (see Chapter 4) operates on a discrete set of overlapping views. Planning a robot path around the outside of a building that maximizes visual coverage (an aerial photo of an example building and its polygonal outline is shown in Fig. 3.1) is related to the "art gallery" problem from computational geometry [120]. Specifically, the task is to find a set of "guard" positions $\mathcal{G}$ in a polygon $P$ that collectively "see" the entire polygon. The traditional criterion for *visibility* between two points $\mathbf{p}$ and $\mathbf{q}$ is *line of sight*: the line segment joining them does not intersect $P$. Paths along which the entire polygon is seen at least once are called *watchman routes*.

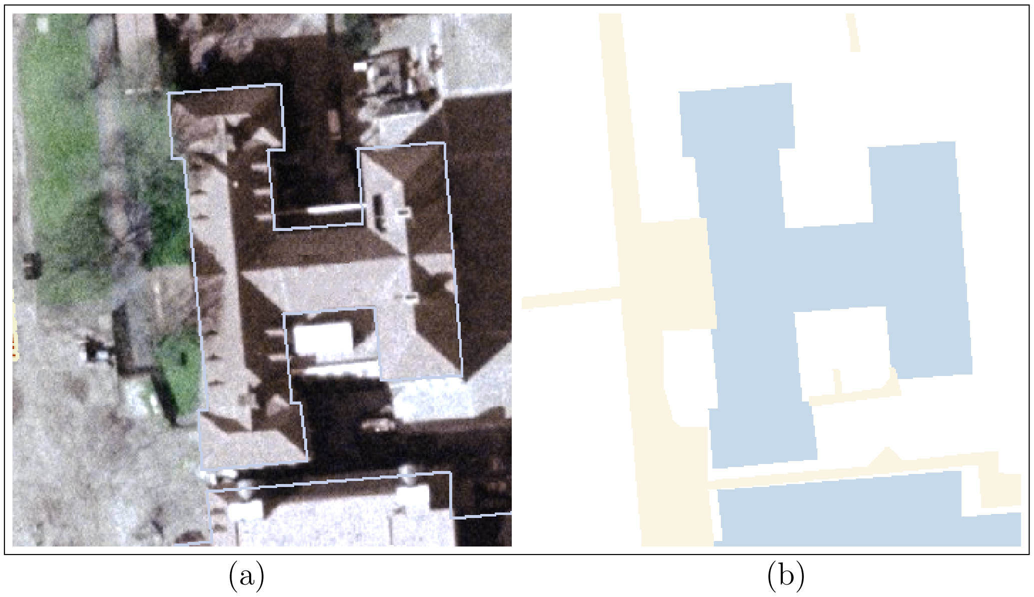


**Figure 3.1:** (a) Aerial view of example building to be mosaiced and its surroundings; (b) Outline of building and neighbor, plus additional map features (free space here is of course the *outside* of the building polygons)

Robotics researchers interested in view planning with real sensors such as laser range-finders have recently extended the notion of visibility in an art gallery framework to include a *range constraint* and an *incidence constraint* [29, 55]. The range constraint models a sensor's inability to work when too far from or too close to an object. Points $\mathbf{p}$ on $P$ whose distance from the sensor position $\mathbf{q}$ falls inside a specific range $d_{\min} \leq d(\mathbf{p}, \mathbf{q}) \leq d_{\max}$ are "range visible". Similarly, the incidence constraint enforces an angular range to model a limited sensor field of view (or exclusion of poor quality range returns at near-grazing angles). Letting $\mathbf{v} = \mathbf{q} - \mathbf{p}$ and $\mathbf{n}$ be the surface normal at $\mathbf{p}$, points for which the angle $\angle(\mathbf{n}, \mathbf{v}) \leq \tau_{\angle}$ are considered "incidence visible." Only polygon points which are visible in all three of the above senses are considered visible from a position $\mathbf{q}$.

Binary line-of-sight and field of view constraints obviously apply to camera view planning, but even within a single image different pixels "see" more or less of the building and with different levels of goodness. Horizontal and vertical foreshortening and finite camera resolution influence the overall goodness of a hypothetical view, and we believe that a real-valued visibility function—i.e., how well, not just whether, a particular point is seen—can better represent this. One contribution of our work is a formulation for such a goodness function, which we demonstrate in an existing point sampling framework. Moreover, overlap between adjacent frames is important for mosaicing. To this end, we introduce an online method derived from particle filtering for finding and linking together guard points that allows dynamics and may be suitable for situations in which *a priori* building maps are not available (i.e., exploration). Finally, we offer some heuristics for improving the roadmap approach to extracting a path from the set of guard points described in [29] that are particular to our task.

### 3.1.1 Problem Setting

Suppose we wish to mosaic or *cover* the exterior of a building $B$ with camera views. $B$ has a known perimeter and facade height $h_B$, and is one of a set of neighboring buildings $\mathcal{B} = \{B_1, \ldots, B_n\}$. Let the robot be equipped with a panning camera that has a vertical field of view of $\phi$, a horizontal field of view of $\theta$, and an image resolution of $w \times h$. We assume that the camera is mounted near ground level with its optical axis fixed parallel to the ground plane (i.e., it does not tilt). Image pixels discretize viewing directions, so for a cylindrical projection $B$'s visibility is sampled by $w$ equally-spaced viewing rays.

### 3.1.2 Goodness Function

We replace the binary visibility criterion above with a real-valued "goodness" function. Consider a candidate viewing position $\mathbf{q}$ in the plane. A particular viewing ray is defined by $\mathbf{q}$ and a unit direction vector $\mathbf{r}$. The goodness of the ray $\Gamma(\mathbf{q}, \mathbf{r})$ is defined as the product of the $\{0, 1\}$-valued line-of-sight, range, and incidence constraints ($\tau_\angle = \theta/2$) defined above and two real-valued terms (only calculated if the first three are all non-zero). These are:

- **Foreshortening** If the normal at the wall point $\mathbf{p}$ on $B$ which the ray strikes is $\mathbf{n}$, the dot product $-\mathbf{r} \cdot \mathbf{n}$ is 1 when the ray hits the building orthogonally and 0 at the extreme grazing angle of 90 degrees. This measures the effective resolution of the pixel.

- **Vertical Framing** An additional measure of pixel utilization is how well the building fills the image vertically. We penalize for being too far away, resulting in sky visible above the building, as well as being too close, cutting off the top of the building. If $\mathbf{p}$ is on the ground, another point $\mathbf{p}'$ that is $h_{\mathbf{p}'} = d(\mathbf{p}, \mathbf{q}) \tan \phi$ meters above the ground would be in the top row of the image. Thus, $\min(h_{\mathbf{p}'}, h_B)/\max(h_{\mathbf{p}'}, h_B)$ measures the vertical fraction of the image that is either sky or cut-off building.

The goodness of a viewing position $\Gamma(\mathbf{q})$ is evaluated by casting one ray per sensor pixel and taking their average goodness. In this work we ignore vertical foreshortening by only casting rays in the plane—i.e., horizontally. A sample synthetic omnidirectional image (only 270 degrees are shown) is given in Fig. 3.2(a). It represents the information available to the planner regarding visibility, foreshortening, and vertical framing via the intensity of each image pixel. Fig. 3.2(b) shows a building surrounded by sampled viewing positions (more on this in the next subsection) and a particular position at which $\Gamma$ is being calculated. The intensities of the rays are proportional to their individual goodnesses. A high-resolution depiction of the components of the goodness objective function is shown in Fig. 3.2(c-e).

### 3.1.3 Next Best View

Using the binary visibility criteria of distance and angle defined above, the first algorithm in [55] based on the Greedy algorithm for finding near-optimal set

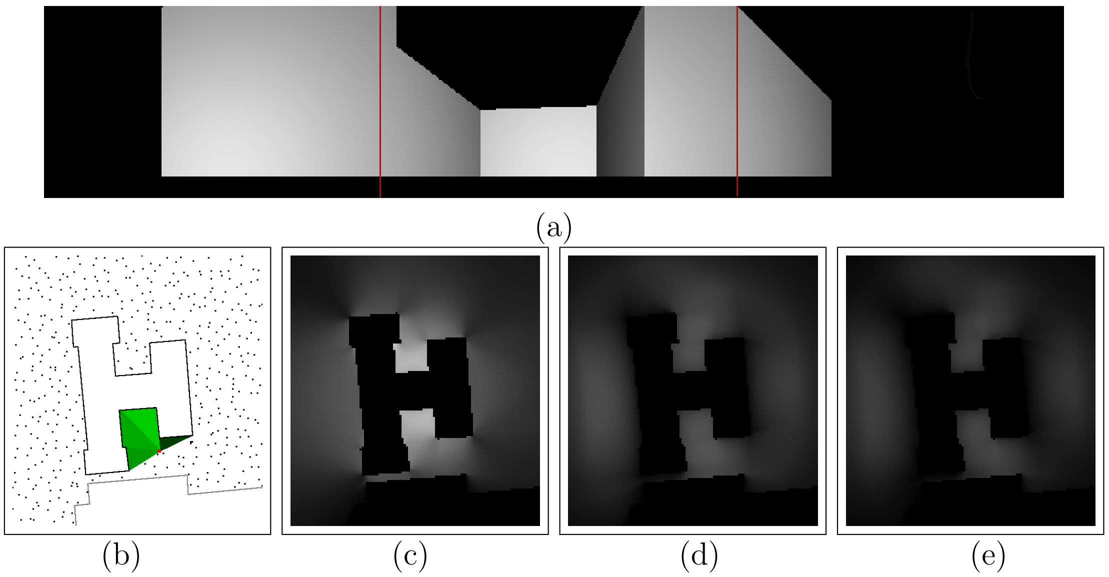


**Figure 3.2:** (a) Sample synthetic omnidirectional image (partial) of building with diffuse shading showing foreshortening (red lines are 90 degree intervals); (b) Building polygon, uniform samples, and viewing position with cast rays (ray saturation proportional to goodness); (c) Goodness function with foreshortening term only discretized at 1 m resolution; (d) Goodness function with vertical framing term only; (e) Combined (via product) goodness function

covers could be directly used to obtain a set of guards $G_1, G_2, \ldots$ that covers the building polygon $B$ with views. Briefly, one would randomly and uniformly sample viewing positions outside any building polygon and within $d_{\max}$ of $B$, compute what sections of $B$'s perimeter are visible from each sample, and pick the guard position that sees the longest overall section of $B$ (summed over visible fragments). These "already seen" sections are marked as invisible to subsequent viewing position candidates, and the process is repeated until all of $B$ (up to some threshold) has been seen once.

The real-valued goodness function above necessitates modifications to this approach since each section of wall is not just "viewed" or "not viewed", but rather "viewed with a certain goodness." Therefore, we set a threshold on the *total goodness* with which every segment of wall must be viewed. The bookkeeping for such a requirement makes the exact ray-sweeping methods used in Greedy inapplicable. Therefore, we discretize the perimeter of the building into initially empty buckets and every new guard chosen deposits its ray goodnesses into the buckets until their thresholds are exceeded, after which subsequent rays hitting such areas have goodness 0. This mechanism ensures as before that new guard positions are chosen rather than the same one repeatedly, and updating this data structure is extremely simple and fast. We call this variant GoodnessGreedy .

### 3.1.4 Path Building

The set of guards obtained with the GoodnessGreedy method above is unordered and not immediately usable as a path for robot motion. We have developed two approaches to making this step.

#### Uniform Sampling

Previous work [29] demonstrated how post-processing could generate an ordering of the guard points using an approximate approach to the traveling salesman problem. A graph $R$ called a "roadmap" is created which contains all guard points and building vertices, with edges joining mutually visible nodes. The shortest path between every pair of guards in $R$ is then computed to yield a fully connected graph $R^*$. A traversal of $R^*$'s minimum spanning tree $R^*_{MST}$ yields an "inspection tour" with length less than or equal to twice that of the shortest possible tour.

We have implemented a variant of this approach using several heuristics in the traversal of $R^*_{MST}$ to shorten the extracted path. Rather than [29]'s method of simply performing a pre-order traversal, we initially choose a path start point that maximizes tree height, and then at each node with multiple children we visit the children in order from shortest to tallest subtree height. These heuristics tend to

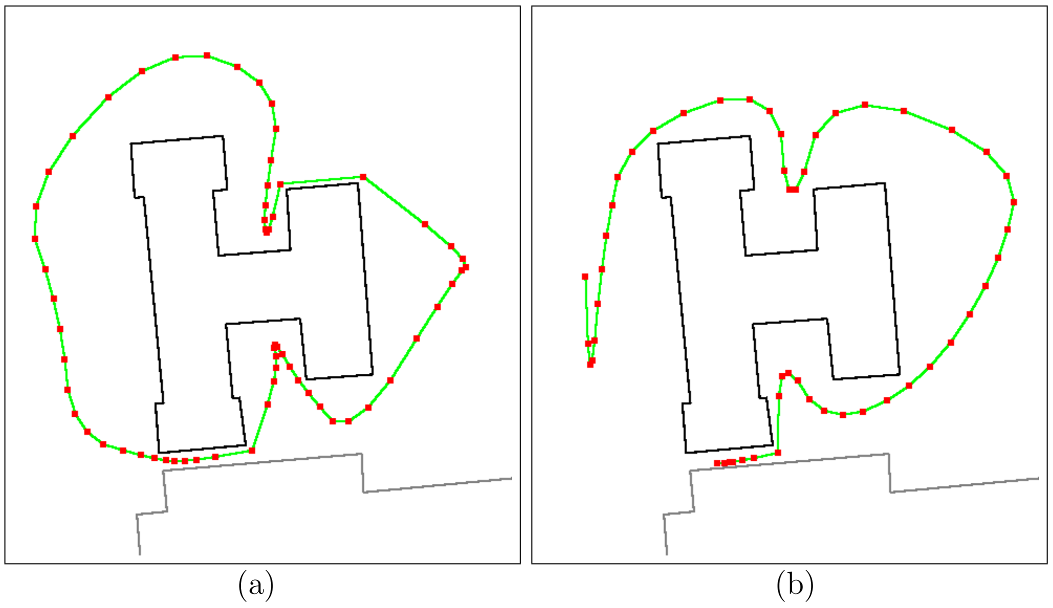


**Figure 3.3:** Sample planned coverage paths (subdivided). (a) Using global, GoodnessGreedy method for guard point selection followed by a distinct guard point linking phase; (b) Using local, particle filter-like algorithm to incrementally add guard points from a random start point

work well for circumnavigating a building with concavities because exploring such indentations are just small detours off of the mainline circuit around the building. Typically, we smooth the final path to eliminate sharp corners for better suitability to robot motor control using an interpolating spline such as Catmull-Rom [68].

### Particle Filter

An alternative, online approach to building a robot inspection path is derived from the idea of reactive path planning via a "tracking quality" function in [145]. Here the idea is to define a "map quality" function on robot positions that is just the real-valued goodness function above.

Given an initial robot position in the area near a building, we sample positions in its neighborhood with a Gaussian in a manner similar to a particle filter [71]. The local sample with the best positional goodness is chosen as the next guard, and the area around it is sampled. Through segments of the building bucket data structure filling up, the path planner is "forced" to move in order to find novel views. This effectively builds a path online for the robot through randomized gradient ascent that constitutes coverage behavior without explicitly building and searching a roadmap graph. Because the sampling is local, of course, the planner can become temporarily stuck in local minima until it random-walks to a new view. In practice we have found that this approach works quite well even on buildings or sets of buildings with many concavities.

### 3.1.5 Experiments

We have performed extensive experiments on large, concave building polygons, both individually and in sets, and observed excellent performance for both the particle filter variant described above and the modified version of the uniform sampling algorithm from [55]. We show some generated paths for a building using each method in Fig. 3.3. 360 rays (1 per degree) were cast per sample position, the goodness bucket width along the building outline was 2 m (the building perimeter is about 325 m), and the vertical FOV $\phi$ was 30 degrees. For uniform sampling, one sample was generated per $5 \times 5$ m square ($d_{\min} = 1$ m, $d_{\max} = 50$ m), while for the particle filter-like approach 300 particles were used, generated from a normal distribution with variance 150.

## 3.2 Probabilistic Contour Extraction for Aerial Road Following

We described the planning module that computes a set of camera view locations and generates a path passing through all of them. This section describes methods to localize a vehicle traveling along this road/path, given noisy GPS waypoints, and an aerial photograph of the surrounding region. The objective is to post-process a noisy GPS track under the assumption that the vehicle was traveling on roads the whole way (such as a manually guided robot). The same road tracing algorithm can also be used for *vehicle guidance* by searching for nearby roads or anticipating sharp turns not visible from on-board sensors. We frame localization and guidance as a Bayesian inference problem to integrate and arbitrate between ground-based and aerial data.

Our road extraction method is most similar to JetStream [123], a particle filtering approach to spatially track edge contours including roads. Tracking generally refers to following the state of a target over time, but for still aerial images "time" is associated only with the progressive extension of the estimated contour as in [123, 178]. In our case, GPS information should bias the tracking and eliminate the high level of user interaction required by JetStream. Monte Carlo methods have been used to localize robots in constrained environments [158], combining measurements from multiple sensors such as GPS, dead reckoning systems, and cameras [54]. Frueh and Zakhor [48] have used particle filters to register laser scan data with Digital Surface Maps, to build 3D textured models of cities. We propose to localize the vehicle using only 2D aerial photographs, which provide higher spatial resolution and important color information.

All the above methods employ only a single cue to measure the strength of their belief. Common cues employed in tracking such as color, edges, or feature templates generally do not work well alone in a wide range of environments. A good measurement likelihood function must be able to determine the most appropriate model at any given time and adaptively switch to the dominant one. Isard and Blake [73] have described a mixed-state CONDENSATION tracker that can handle multiple motion models. While we have a well-defined motion model, we adapt their technique to handle variable modes of perception. Once again, we note that the likelihood function here was designed for vehicles traveling on roads in desert or urban environments. In the case of campus modeling, a more suitable function might need to be designed, while the overall framework can remain unchanged.

After describing how particle filters are used to localize a vehicle on an aerial map of the region, we detail different measurement likelihood functions used in the prediction and update phases. A multi-modal particle filter that detects and switches to the most dominant road model while tracking is also presented. We then describe scenarios where the road tracing algorithm may play the role of a wingman to warn an autonomous robot of road curvature or steer a stranded robot towards the nearest road. Finally, results are shown on varied environments.

### 3.2.1 Particle Filtering and Vehicle Localization

Particle filtering has proven to be adept at tracking in the presence of complicated likelihood functions and non-linear dynamics. Tracking here refers to following *the state* of a set of variables $\mathbf{x}$ as they evolve over time. We wish to estimate $\mathbf{x}^t$ at time-step $t$, given knowledge about all the sensor measurements $\mathbf{Z}^t = \{\mathbf{z}^1, \mathbf{z}^2, .., \mathbf{z}^t\}$ up to $t$. If we construct the posterior density $p(\mathbf{x}^t|\mathbf{Z}^t)$ to represent our belief of the current state, we can infer $\mathbf{x}^t$ by taking either the MAP (maximum a posteriori) or mean estimate.

In particle filters, this belief or posterior density is approximated by a set $S^t$ of $N$ particles $\mathbf{s}_i^t = \{< \mathbf{x}_i^t, w_i^t > | i = 1, .., N\}$, where $\mathbf{x}_i$ is a state (position on the map) and $w_i$ is the importance weight. The importance weights give a measure of how reliable the corresponding state estimate is. The set of samples thus define a discrete approximation of the continuous probability density function. In every iteration, $N$ new particles are sampled from $S^t$ with the probability of survival of a particle $\mathbf{s}_i^t$ being proportional to its weight $w_i^t$. Each particle is then modified according to the *dynamics* and the weight is updated according to the *measurement model.*

Initially, a set of equally weighted particles are uniformly distributed around the starting position estimated by the GPS reading $(u_1, v_1)$ in UTM coordinates.

The state vector $\mathbf{x} = [x, y, \theta, m]$ includes a variable for the road width $m$, where $(x, y)$ is the mid-point of the road oriented at an angle of $\theta$ degrees. At every iteration of the particle filter algorithm, a new set of GPS coordinates $(u_t, v_t)$ is obtained and the relative motion $(R_t, \Theta_t)$ is computed. We now describe two schemes to integrate GPS sensor information into the particle dynamics.

### GPS-driven dynamics

The motion $(R_t, \Theta_t)$ is applied to all the particles along with white Gaussian random noise to predict the new position of each particle. The particles are thus subjected to a drift and diffusion process with a relative movement of $(R_t^{'}, \Theta_t^{'}) = (R_t + N(\sigma_r), \Theta_t + N(\sigma_\Theta))$. On applying the motion model, we predict state $\mathbf{x}_i^t = [x', y', \theta', m']$ from $\mathbf{x}_i^{t-1} = [x, y, \theta, m]$ by applying the dynamics:

$$\mathbf{x}_i^t = \begin{pmatrix} x' \\ y' \\ \theta' \\ m' \end{pmatrix} = \begin{pmatrix} x + R_t^{'} \sin(\Theta_t^{'}) \\ y + R_t^{'} \cos(\Theta_t^{'}) \\ \Theta_t^{'} \\ m + N(\sigma_m) \end{pmatrix} \tag{3.1}$$

where $N(\sigma)$ denotes Gaussian noise with variance $\sigma^2$. The standard deviations are a function of the step size with $\sigma_\Theta$ varying from $0.05\sqrt{R}$ to $0.1\sqrt{R}$ and $\sigma_r$ is set to $0.2R$. When the GPS estimates are fairly reliable, this scheme forces particles to follow the GPS point and look for a possible road within the error ellipsoid.

### Texture-driven dynamics

If the GPS positions are not reliable, the above method overly restricts the particles from following the road. Setting a larger variance with the previous method does not eliminate the problem, as a majority of particles would still follow the incorrect GPS track. This is especially true in cluttered urban regions since there might possibly be many road-like features such as house tops or shadows of buildings.

A more flexible approach in such situations is based on the strength of the underlying texture - i.e. allow particles to follow all possible roads and then use GPS to narrow down on the most plausible one in the update phase. Details of the likelihood function are elucidated in the next section.

The formula for $\mathbf{x}_i^t$ is the same as (3.1) except that the orientation $\Theta_t^{'} = (\Theta_{t-1}^{'} + N(\sigma_\Theta))$ is distributed around the direction the particle was traveling in the previous time step. This allows a much wider angular distribution as well as giving particles a certain momentum to get past erroneous GPS measurements. The formula for $R_t^{'}$ remains the same as before, in order to retain information about the velocity of the vehicle.

**Update phase**

A preliminary weight $w_i$ is computed for each new particle based on the measurement model. This weight could be a measure of our confidence that the estimated position of the particle on the aerial photo lies on the road or how well on-board camera images correlate with the aerial view of the corresponding region. The weighting serves to concentrate the histogram over state space of all the particles around the most likely position that the vehicle could be in. Only the "fittest" particles survive from one iteration to the next, resulting in an evolutionary process. Given this representation of the density function, the current estimate for the position of the vehicle is chosen to be the weighted mean of all the particles.

### 3.2.2 Measurement Likelihood

We employ vision-based techniques to assign relative weights for each particle. These weights reflect the strength of our belief that a particle lies on the road based on the aerial photo. Most particle filters, including JetStream, are flexible and have very few restrictions on the measurement model used. However, a single measurement model alone might not be sufficient to characterize the observation density. Multiple observation models are important for robustness and applicability, with automatic switching between them. We first define the different road models used and in the next section describe a technique to arbitrate between them.

**Measurement Models**

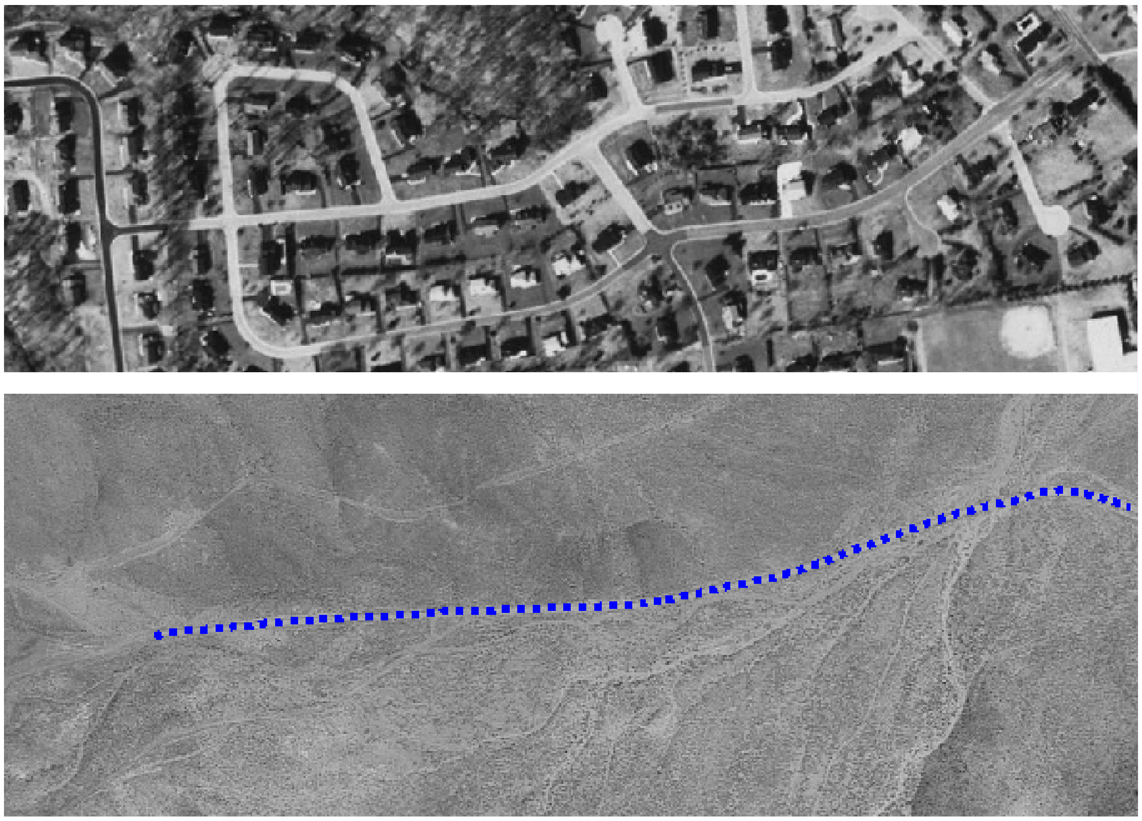


**Figure 3.4:** Example aerial images of urban and desert regions from which we wish to spatially track the road traveled by the vehicle. Roads might appear bright or dark and may have very little contrast in off-road environments.

Figure 3.4 shows example aerial images of off-road and urban environments on which we run our technique. JetStream [123] uses the norm of the luminance gradient as a cue to track high contrast contours. The aerial images that we employ are low contrast, noisy and characterized by excessive clutter in urban regions, making simple edge- or color-based methods impractical. We have therefore chosen

Gabor filters [93, 154], widely used in texture analysis, to give an initial confidence estimate for each pixel being road or non-road. The general functional for the two-dimensional Gabor filter family can be represented as a Gaussian function modulated by an oriented complex sinusoidal signal. In polar form it is written as

$$G_n(x, y, \lambda) = e^{-\pi[x^2/a^2+y^2/b^2]}e^{j2\pi[r\cos(\theta-\phi)/\lambda]}. \tag{3.2}$$

The real part of the Gabor filter ($RG_n$) has even symmetry and is a proven blob detector while the imaginary part ($IG_n$) can be used to detect step edges. Since roads appear as banded segments oriented at some angle, we attenuate $RG_n$ along its width, to give an elliptical pattern that retains only the central 40 percent of the estimated width. While this particular pattern effectively detects bright roads in a darker region, a negated filter ($RG_n^-$) can detect darker roads. The width of the roads in the image dictate the choice of various scales($\lambda$), and for each scale 10 equally separated orientations are selected from $[0..\pi]$. We then use normalized cross-correlation rather than convolution to pre-compute the response for each of these filters. Doing so compensates for intensity changes, while enabling seamless fusion of multiple scale filter responses.

The measurement models are a function of the features detected by one or more filters in the pre-processing stage. Other features such as color or shape can also be used when available. We currently use only $N_o = 2$ different models defined analytically as:

- $M_1=$

$$RG_n(x^+, y^+, \lambda_1) + RG_n(x^-, y^-, \lambda_1) + RG_n^-(x, y, \lambda_2)$$

- $M_2=$

$$RG_n(x^+, y^+, \lambda_1) + RG_n(x^-, y^-, \lambda_1) + RG_n(x, y, \lambda_2)$$

The superscripts over $x$ and $y$ indicate the road edges on both sides of the particle computed from $m$ and $\theta$. Model $M_1$ looks for a dark (with respect to the surrounding region) road with thin parallel lines running along the road. This model is very effective for urban roads as well as certain shadowed or occluded regions. Model $M_2$ is used to detect brighter roads, typical of rural or country roads. A weighting scheme may also be used in the above models to emphasize certain features more than others. For example, the sidewalk is very prominent in suburban areas and so a higher weight factor can be multiplied with the road edge response. Adding more models is straightforward and the method to switch between them is described in the next section. The weight of a particle $\mathbf{X} = [x, y, \theta, m]$ using model $o$ is the output of $M_o$ at orientation closest to $\theta$ for the point $(x, y)$.

For efficiency, image processing is done only within a small floating tile ($61 \times 61$ here) around the particles, allowing convolutions to be cached until a minimum fraction of the particle set leaves its confines. This allows real-time performance on large images (a second-level system of loading and unloading adjacent raw aerial image files has not yet been implemented, as all of our runs have thus far been limited to $2.5 \times 2.5$ km square areas).

### 3.2.3 Integrating Multiple Models

Having defined various models that determine the road likelihood, there is the issue of choosing the appropriate model. We adapt mixed-state tracking techniques [73] to probabilistically detect and switch to the most dominant measurement model at any given time. We define an extended state for each particle to be

$$\mathbf{X}_i = (\mathbf{x}_i, O_i) \tag{3.3}$$

where $\mathbf{x}_i$ is as defined earlier and $O \in \{1..N_o\}$ is a discrete variable labeling one of $N_o$ observation models. Thus $O_i$ determines which observation model to use in the measurement phase for particle $s_i$. We also define a state transition probability matrix $T$ which is an adjacency matrix representation of the possible state transitions i.e. $T_{pq}$ is the probability for a particle to change state from $p$ to $q$. In order to integrate mixed-state models into the particle filtering framework, it is sufficient to split the sampling process of every iteration into two separate phases. In the first phase, the state transition probabilities are sampled from to generate a new observation model density for the particles. The subsequent sampling phase is the same as described previously where particles with higher weights survive while the others are eliminated. In the update phase, the observation model used to measure the reliability of particle $s_i^t$ depends on the value of $O_i^t$.

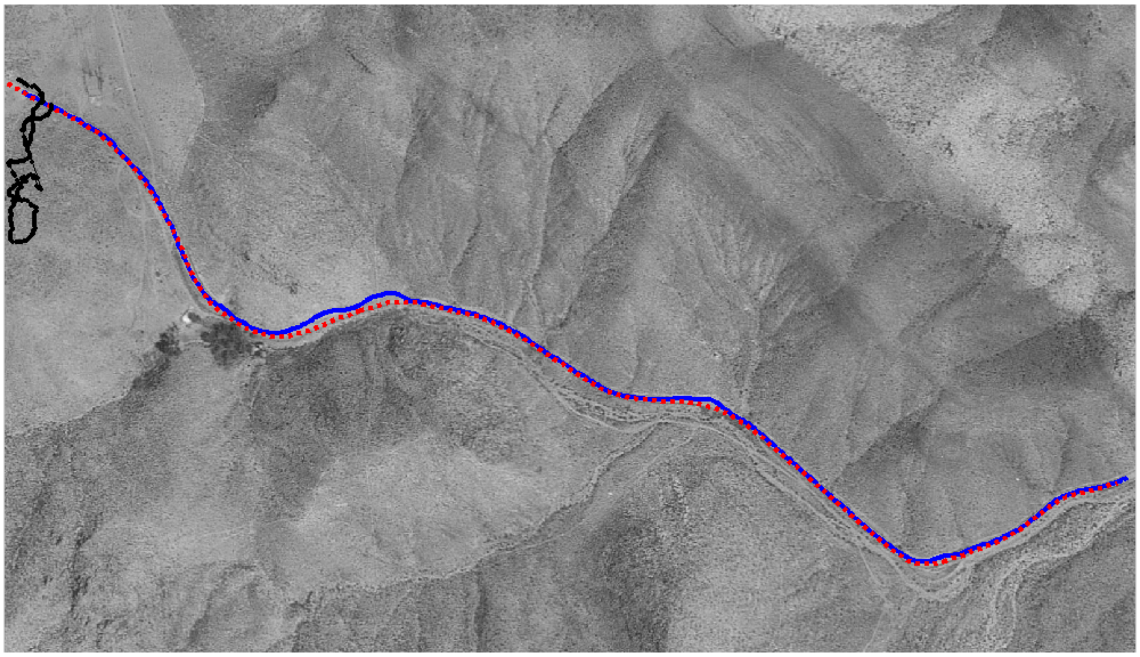


**Figure 3.5:** GPS plot (dotted red) and tracked path (solid blue) through the desert. The darker curve shows the output of conventional JetStream using the luminance gradient.

The formal steps used in the particle filter are described below. We begin with $S^{t-1}$ of $N$ particles $s_i^{t-1} = \{< \mathbf{x}_i^{t-1}, O_i^{t-1}, w_i^{t-1} > | i = 1, ...N\}$ in every iteration followed by:

1. *Sampling:* Construct the $n^{th}$ of $N$ new samples according to the following two steps:

- *Sample transition probabilities:* Sample from $P(O_i^t = q | O_i^{t-1} = p) = T_{pq}$ to find $O_i^t$ for each $s_i$.
- *Sample process density:* Sample from $S^{t-1}$ based on $w^{t-1}$ by generating a random number $j$ with probability proportional to $w_j^{t-1}$ and setting $s_n'^{(t-1)} = s_j^{t-1}$.

2. *Prediction:* We apply our dynamics to each sampled particle as governed by equation (3.1).
3. *Update:* The likelihood for this particle is computed according to the measurement model specified by $O_i$. Weights are updated in terms of the latest image data $Z^t$.

In order to estimate a single most probable position that the vehicle could be in after every time-step, a two-pronged strategy is adopted of first computing the dominant model $\widehat{O}^t$ in force, and then calculating the weighted mean of only those particles in that observation state. $\widehat{O}^t$ is computed according to

$$\begin{aligned} \widehat{O}^t &= \arg\max_j \sum_{i \in \Upsilon_j} w_i^t \text{ where} \\ \Upsilon_j &= \{i | \mathbf{X}_i^t = (\mathbf{x}_i^t, j)\} \end{aligned} \tag{3.4}$$

### 3.2.4 Planning

An autonomous robot traveling along its precomputed path might be confronted with several difficult scenarios, such as those encountered by many of the entries in the DARPA Grand Challenge competition. A brief loss of GPS signal might cause the robot to wander away from its pre-determined path. Purely on-board sensors with its myopic field-of-view may not be able to locate nearby roads for the robot to recover. Another robot traveling at an optimal speed along a straight and level road might not be aware of an approaching hairpin bend at the edge of a cliff, with potentially disastrous consequences if taken too fast. It is in such scenarios that the aerial planner can be used to break free from deadlocked or dangerous situations.

The planner is a straightforward extension to the road tracer. Every frame during navigation, the road tracer is re-initialized by distributing particles uniformly around the vehicle's current GPS position. The tracer proceeds for a fixed number of iterations, generating a trajectory as a dense series of waypoints (up to a few hundred meters ahead). The plan is a series of closely-spaced waypoints leading to and along the nearest road from the vehicle. These are generated by the state of the particle filter as it searches forward in the current and succeeding route corridor segments a fixed distance. A route segment is a corridor of appropriate width between two successive waypoints returned by the path planning module. When the vehicle is outside the corridor, a straight-line plan leading to the nearest point on the mid-line of a segment is created, after which the waypoints come from the road tracer.

### 3.2.5 Results

We show the result of localization on both urban and off-road environments, currently using only aerial images and GPS data. The particle filter was initialized to use 1000 particles - though it is very robust with less than half that number - and first distributed around the starting point. The choice of Gabor filter scales depend on the resolution of the aerial imagery, and for the publicly available 1-meter resolution photos that we used, $\lambda_2 = 10$ was a fair approximation for the width of most roads. To detect the road edges and sidewalks, we used $\lambda_1 = 1$.

To compare our spatial tracker with the pure JetStream approach, we ran the particle filter on a 1.2 mile track obtained during a drive through the southern California desert. After verifying that the Navcom SF-2050G DGPS receiver was accurate to a couple of meters, the particle filter was run similar to JetStream without integrating any of the sensor data. This also enabled us to quantify deviation

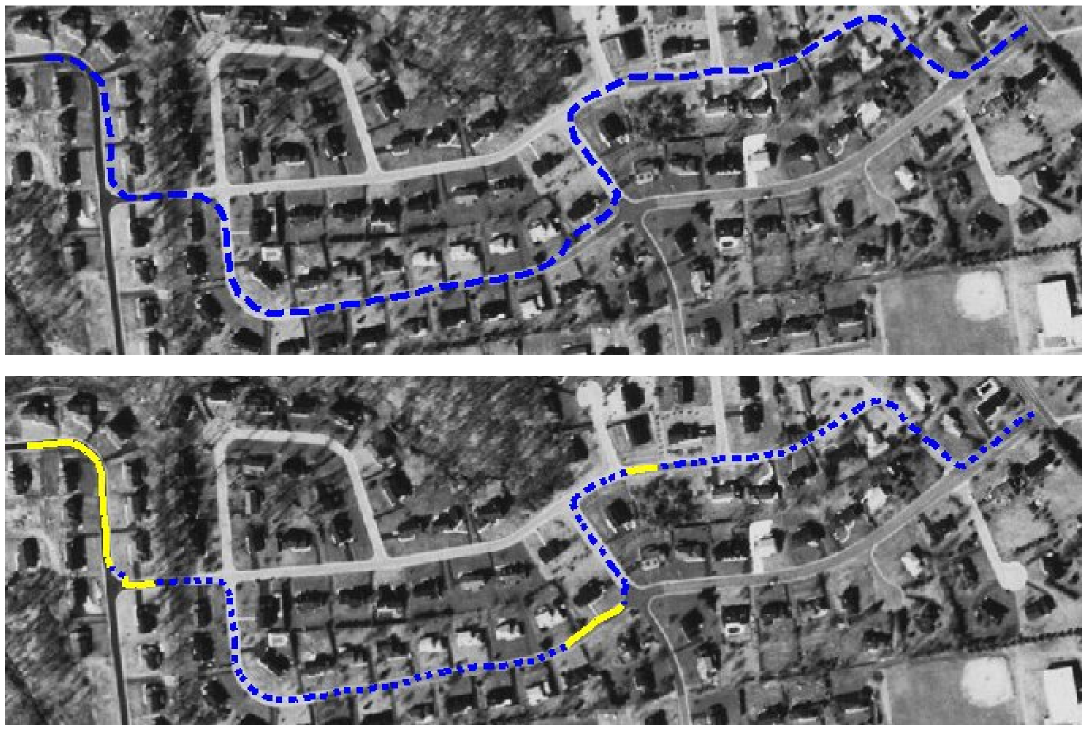


**Figure 3.6:** Comparison of single-mode and mixed-mode tracking using models $M_1$ (dark roads) and $M_2$ (bright roads). Solid yellow indicates that the tracker is in mode $M_1$ and dashed blue denotes $M_2$. Notice how tracking only with $M_2$ causes mistracking on the darker roads.

from ground truth. Shown in figure 3.5 is a plot of the GPS track (dotted red) on one segment and the estimated path (blue) of our tracker. The dark wayward curve shows the output of the conventional edge tracking method as used by JetStream. Due to the absence of strong edge cues, JetStream does not track correctly beyond a few meters. By extracting the local texture information using Gabor filters, our algorithm does significantly better. The mean distance error with the GPS curve on this run was 3.24 meters and standard deviation was 3.1m. On another similarly curvy segment, the mean was 1.4m and standard deviation was 1m. Estimating the road width to be about 10m from the aerial image and the GPS data itself to be accurate only to a couple of meters, we claim that our image processing alone is robust to handle significant drop-outs on the GPS due to LOS issues.

Figure 3.6 shows a suburban neighborhood comparing single- and mixed-mode tracking. The roads are either dark or bright, with abrupt transitions between them. To track all possible roads without GPS is not practical as there is a lot of clutter. The comparatively low cost Garmin GPS 16 used for this example was especially unreliable in such environments with accuracy as bad as 25 meters in some places. In addition to correcting the GPS path, our tracker automatically switches to the appropriate model. The yellow sections indicate when the tracker is in model $M_1$ looking for dark roads, while blue denotes that $M_2$ is dominant at that point. The transition matrix used is $T = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$ with a slightly higher probability enforced for each particle to remain in the current mode. The value of $d_{max}$ was set to 20. Single-mode tracking using only $M_2$ causes mis-tracking to occur on the darker roads.

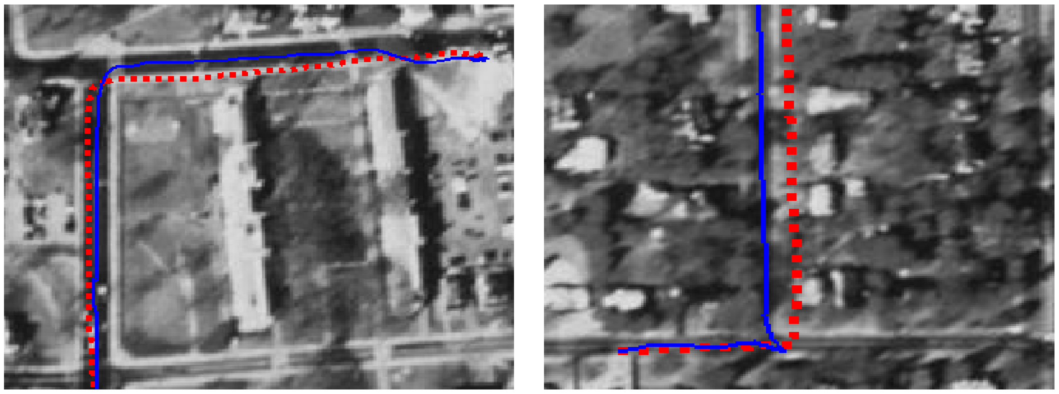


**Figure 3.7:** Corrected GPS positions (solid blue) at intersections. GPS estimates (dotted red) were especially inaccurate at intersections.

Figure 3.7 shows zoomed in regions of corners where the GPS (red) tracks have been corrected by our contour extraction method. Figure 3.8 shows a short GPS run in dotted red. This is difficult both in terms of highly inaccurate GPS readings at the corners, as well as the presence of shadows and trees that mask the road in some places. While GPS-driven dynamics simply do not work correctly in

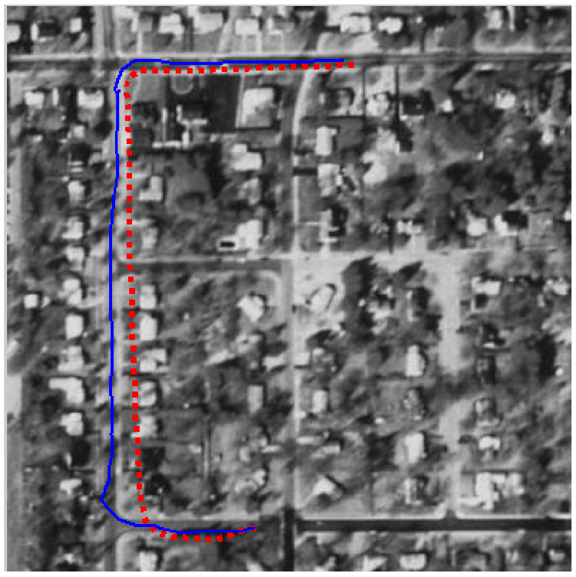


**Figure 3.8:** Using texture-driven dynamics and model-switching (not shown) with highly inaccurate GPS (red) positions. Solid blue curve shows corrected path

this scenario, texture-driven dynamics (blue) is able to trace the road for the entire length of the segment. The particles have enough momentum to keep following the strongest road likelihood without changing direction at every wiggle in the GPS data.

Illustrated in figure 3.9 is a typical scenario of what happens when a GPS receiver loses signal due to over-passes or tunnels. There is a sudden glitch in the GPS outputs, as points veer off to the sides before homing in on the actual position again. Texture-driven dynamics can easily handle such situations as shown in the figure. This situation might also call for some image processing hacks that detect features orthogonal to the road, but that was not required in this case as particles were distributed far enough along the road to overcome negative filter responses under the over-pass.

Fig. 3.10 demonstrates how our algorithm can assist in localized planning. The figures show in green the track of Caltech's 2004 DARPA Grand Challenge

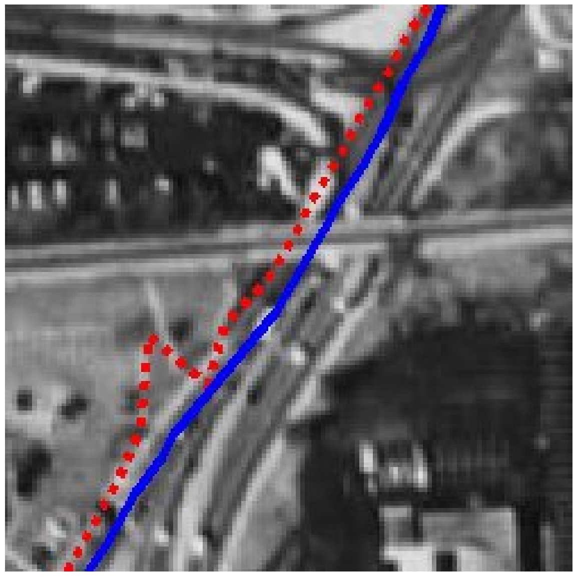


**Figure 3.9:** Using measurement likelihood-driven dynamics to handle characteristic glitches in GPS data caused by over-passes and bridges

vehicle's GPS locations during the 2004 race. When initialized with the vehicle's current GPS position, the particle filter is run for $t$ iterations returning a dense track $p_{0:t}$ of waypoints that are very likely to be on the road. Drawn in yellow, is this prediction of the road immediately ahead or in the vicinity of the vehicle. Using such a plan could have helped their robot avoid and recover from certain off-road situations they encountered during the race.

The bottom two images in Fig. 3.10 are the result of road tracing in later mountainous segments of the race course. While the dynamics of our particle filter can effectively handle various types of roads exhibiting sharp corners as well as high curvature, the likelihood function is able to constrain the particles on the road purely by analyzing the local texture in the aerial image. Even on such low quality images with poor contrast, this is a powerful cue that can assist the navigation modules, for both localization and planning.

## 3.3 Summary

We have presented two parts of an autonomous architectural modeling system that uses aerial imagery for view planning and robot localization. The first is a randomized view planner that efficiently builds incremental paths to maximize coverage and quality. The problem is framed as an "art gallery" problem from computational geometry, and the task is to compute an optimal *watchman route* in which every face of the building polygon is seen at least once. Rather than binary line-of-sight or field-of-view constraints, we formulate a goodness function that takes into account effects of foreshortening and camera resolution. A particle filtering method is then introduced that links together guard points accounting for both visual coverage and robot dynamics.

The second part is a system for road-following on desert and urban roads that relies on road texture analyzed from satellite imagery. Roads near the vehicle are traced, facilitating both vehicle localization as well as generating medium-term plans that avoid local obstacles. We have demonstrated techniques to correct erroneous GPS information for the purpose of vehicle localization. In contrast to map-matching approaches that use digital road-maps, our algorithm works on aerial images of diverse environments. Image processing and probabilistic methods are combined to make an inference about the most likely road that the vehicle is traveling on, based on explicitly defined road models. Detecting and switching to the correct model is done automatically by the mixed-state tracker. Results are shown on a range of images by correcting inaccurate GPS position estimates or tracing the road ahead of the vehicle to guide it in treacherous terrain.

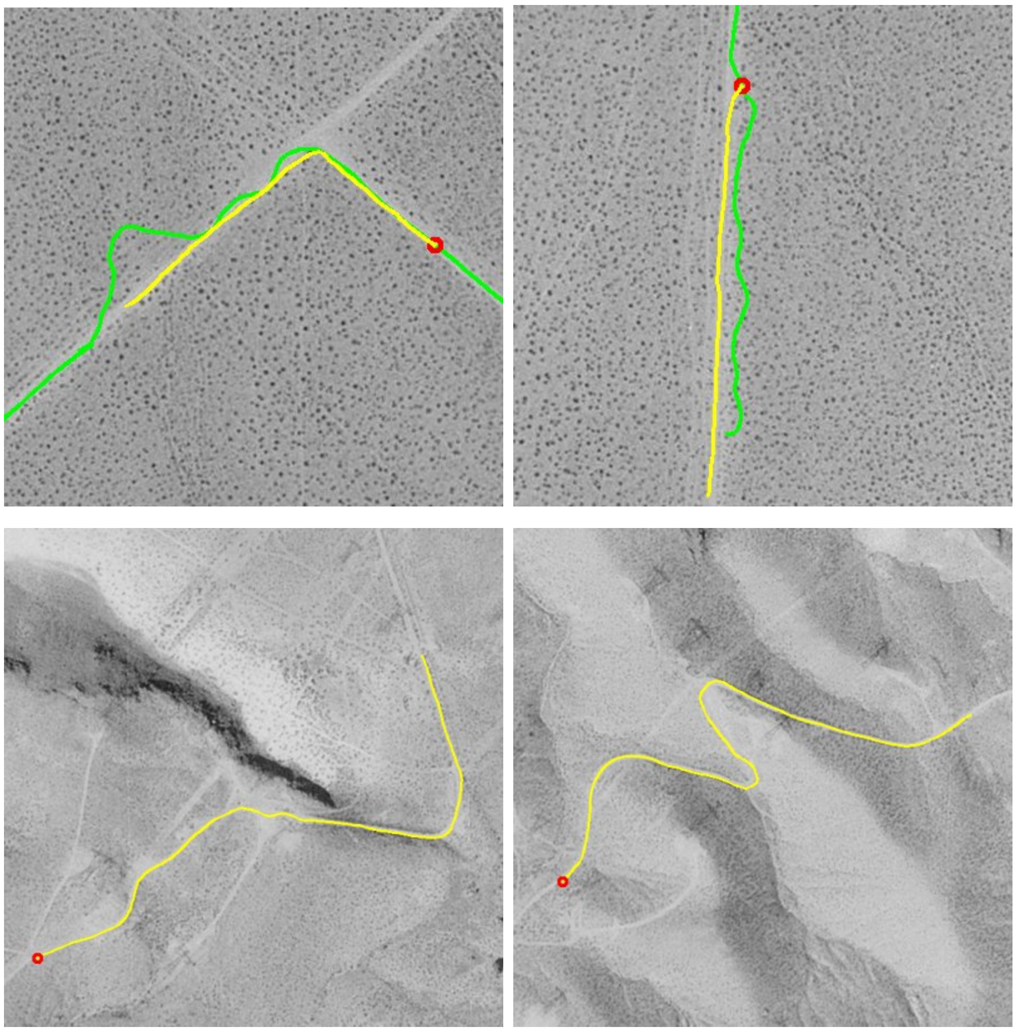


**Figure 3.10:** Top: GPS plot of a 2004 DARPA Grand Challenge vehicle's route (green) on raceday and predicted path (yellow) from a fixed location (red dot) at different sections of the course. The predicted distance is approximately 325 meters with the particle filter run for 75 iterations. The original corridor width for each segment is widened by a factor of 10. Bottom: Road tracing for 175 iterations in a high altitude segment of the 2004 course with the corridor width (not shown) widened by a factor of 10.

# Chapter 4

# RECOVERING CLEAN TEXTURE MAPS FROM SEQUENCES OF PARTIALLY OCCLUDED FACADES

Chapter 3 described an algorithm to plan a series of viewpoints for the robot to capture images from. This chapter focuses on recovering the background building facade from this image sequence, even when a majority or all of the captured images contain views of a foreground object rather than the building. Temporal search over the timeline for at least one such background view should guide the inpainting process rather than a traditional spatial search. The key technical issue becomes identifying which view, if any, is of the background, and this chapter presents several techniques which accomplish this both accurately and efficiently. Building upon and extending the work first presented in [131, 85, 86], we make contributions to both of the large problems set out above:

- We automatically identify mosaic regions polluted by foreground objects through a robust measure of pixel color variance over registered images
- We introduce a novel modification of an exemplar-based inpainting algorithm due to Criminisi, Pérez, and Toyama [27]— *CPT inpainting*—to infer building pixels in polluted regions via combined temporal and spatial search
- We utilize automatically-learned layer appearance models to improve the speed of the temporal inpainting process while maintaining accuracy

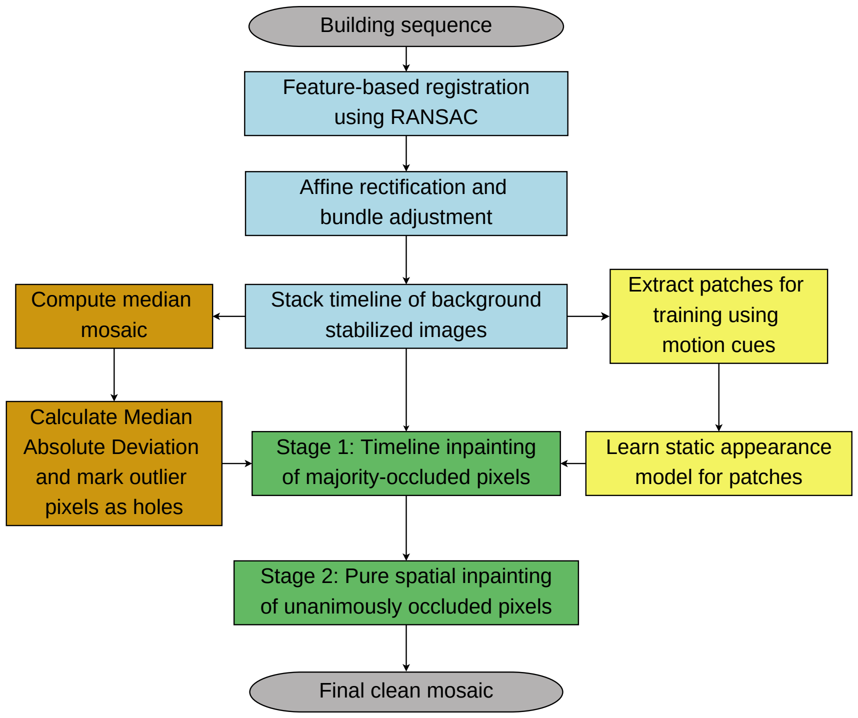

Building sequence
Feature-based registration using RANSAC
Affine rectification and bundle adjustment
Compute median mosaic
Stack timeline of background stabilized images
Extract patches for training using motion cues
Calculate Median Absolute Deviation and mark outlier pixels as holes
Stage 1: Timeline inpainting of majority-occluded pixels
Learn static appearance model for patches
Stage 2: Pure spatial inpainting of unanimously occluded pixels
Final clean mosaic

**Figure 4.1:** System diagram of timeline inpainting for recovering clean texture maps.

In the next three sections we detail our *timeline inpainting* framework for recovering a partially-occluded background layer from image sequences. The major components of the complete system are diagrammed in Figure 4.1. Section 4.1 covers the steps related to image registration and polluted region identification (blue and brown groups of boxes, respectively); Section 4.2 presents our methods for discriminating between the foreground and background layers based on appearance (yellow boxes); and Section 4.3 describes our modifications to CPT inpainting (green boxes). Finally, Section 4.4 shows integrated results for several challenging sequences before we conclude and examine some future directions.

A review of standard CPT inpainting, including the definition of many terms and variables used was presented in the Chapter 2.

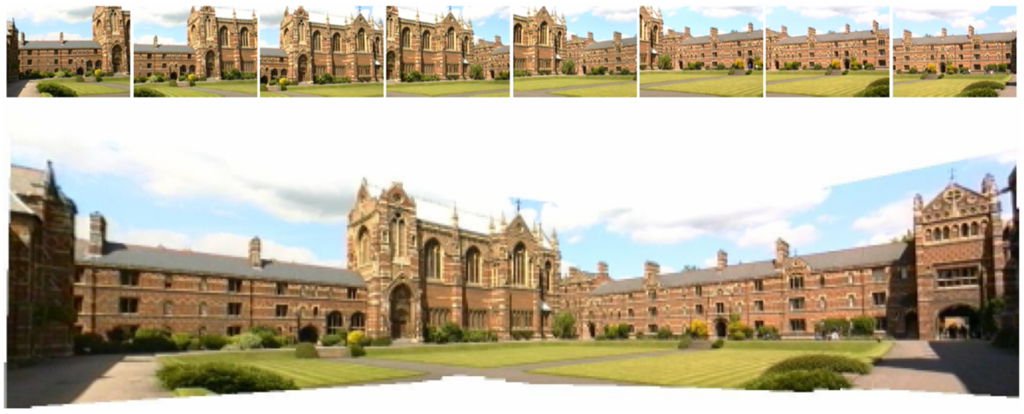


**Figure 4.2:** Panoramic mosaic (bottom) obtained by stitching together individual frames (top) from a sequence. Reproduced from [19].

## 4.1 Image Registration and Pollution Detection

In order to pre-process the image sequence such that corresponding features are registered over multiple frames, we must first stabilize the background. After establishing the mosaic reference frame and its set of implied timelines, we then apply several statistical and appearance tests to conservatively identify potentially polluted regions for subsequent inpainting.

### 4.1.1 Motion Stabilization

Creating a planar mosaic via homography estimation has been thoroughly studied [30, 61, 152]. Figure 4.2 shows a result from the work of [18] where all the frames in a given image sequence are stitched together to create a larger mosaic. The geometric transformation between images is described by a homography, which is valid under one of two conditions: (i) the camera is rotating about its center or (ii) the image surface is a plane. Since we are working with building facades, planarity is a reasonable assumption to make. Estimation of the homography is done through RANdom SAmple Consensus (RANSAC) [45], a robust technique

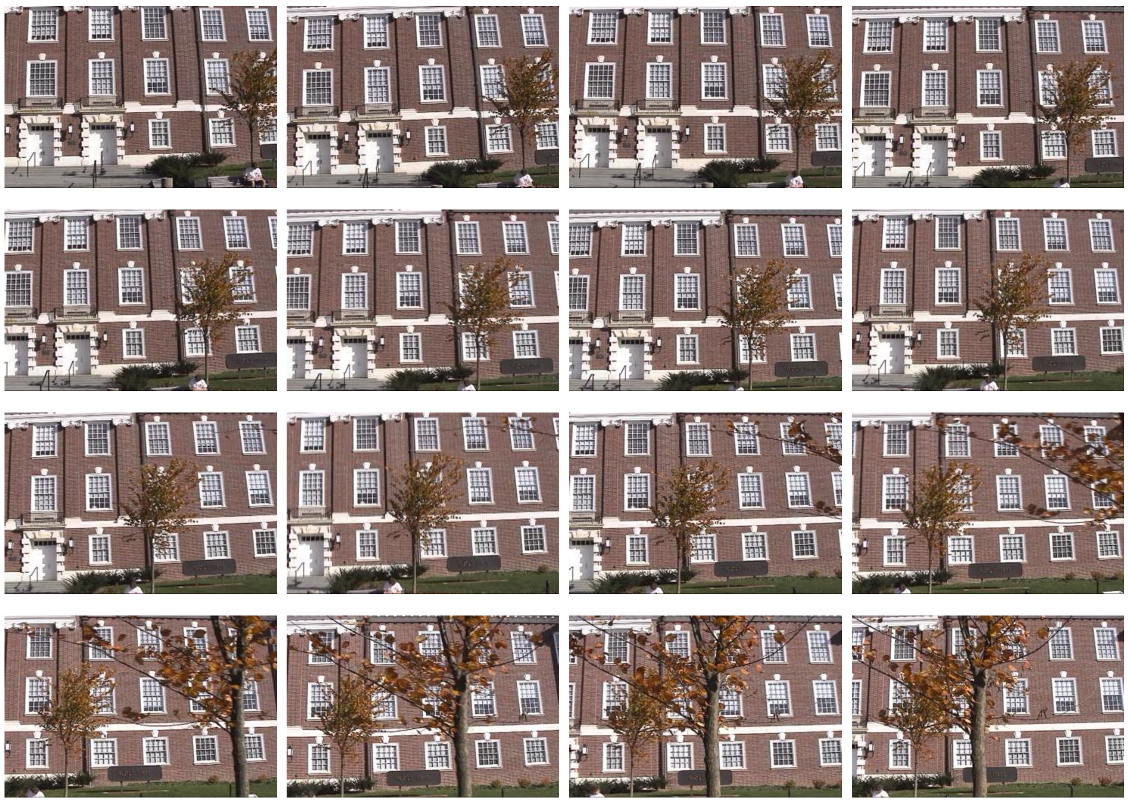


**Figure 4.3:** 16 of 18 frames from the Building A sequence.

to estimate parameters of a mathematical model in the presence of outliers. The method works by considering many random subsets of data, each containing the minimum number of samples required to compute the models parameters, and then selecting the parameter set with the largest number of inliers. In the case of a homography, minimal sets of 4 point correspondences are extracted.

Figures 4.6 and 4.7 (reproduced from [19]) illustrate the image matching process. Features are extracted from overlapping images and a set of putative correspondences established by intensity matching within a small neighborhood. Since each match is independent of the other, a good fraction of these correspondences are outliers inconsistent with the global motion. After RANSAC, the correctly matched points are sieved out from the feature set.

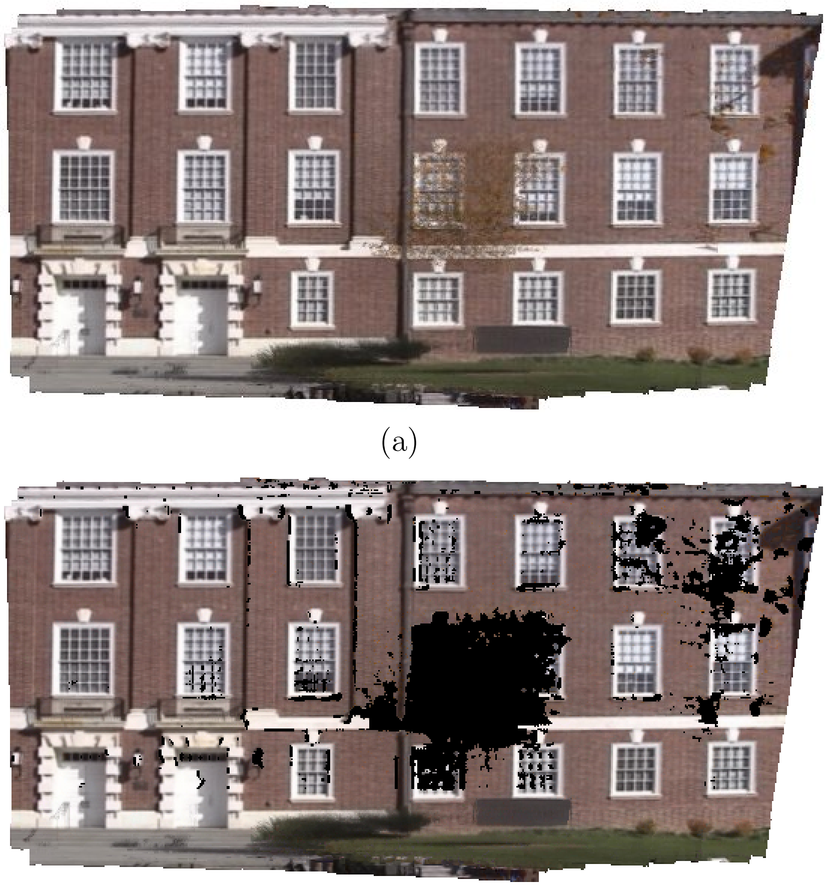


**Figure 4.4:** Result from the first two stages of our algorithm for the Building A sequence (Fig. 4.3). (a) Median mosaic; (b) Polluted region $\Omega$ to be fed to the inpainting algorithm for filling;

We robustly compute the dominant planar motion, assumed to be due to the building, between successive pairs of images $\mathbf{I}_t, \mathbf{I}_{t+1}$ over a sequence of $N$ frames[1] captured with a rough 1-D scanning motion along the facade. The initial frame-to-frame homographies $\mathbf{H}^*_{t,t+1}$ are computed by matching features [140] in both frames followed by RANSAC for outlier rejection [63]. Taking frame number $ref = \lceil \frac{N}{2} \rceil$ of the sequence as the *mosaic reference frame*, the homographies are then concatenated

[1] The frames are automatically warped to remove radial distortion and affinely rectified with a procedure described in [63]

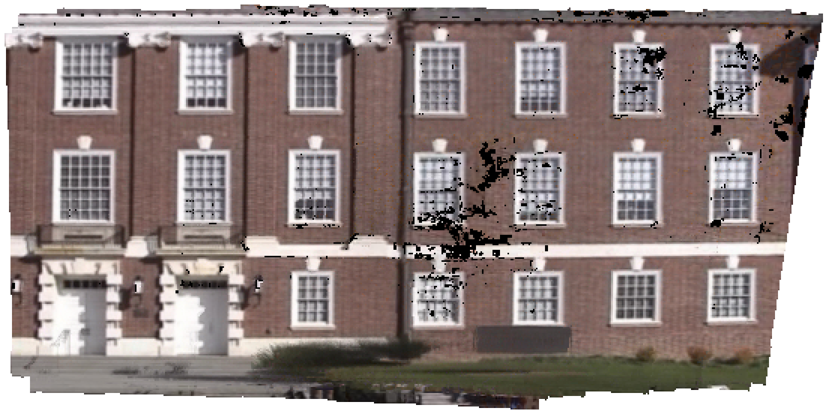
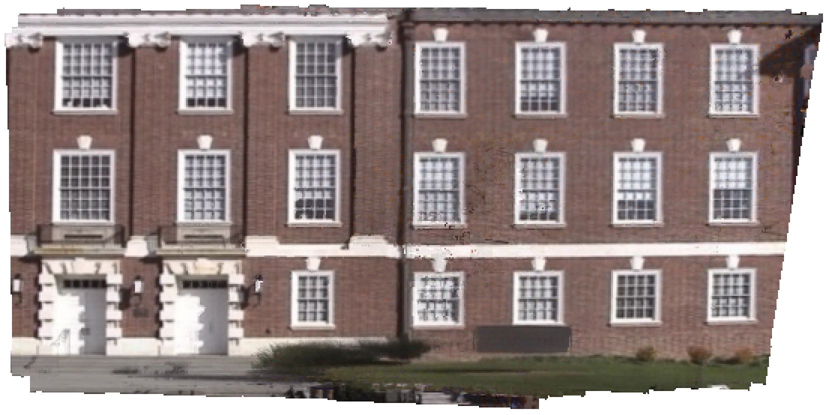

(a)

(b)

**Figure 4.5:** Result from the inpainting stages of our algorithm for the Building A sequence (Fig. 4.3). (a) Result after timeline inpainting Stage 1 ($11 \times 11$ patches); (b) Result after Stage 2 ($21 \times 21$ patches)

together to align each frame with the mosaic—i.e., $\mathbf{H}^*_{ref,ref}$ is the identity; for $t < ref$, $\mathbf{H}^*_{t,ref} = \mathbf{H}^*_{ref-1,ref} \cdots \mathbf{H}^*_{t+1,t+2}\mathbf{H}^*_{t,t+1}$; and similarly for $t > ref$. Warping each frame $\mathbf{I}_t$ by $\mathbf{H}^*_{t,ref}$ with bilinear interpolation results in a mosaic-aligned frame $\mathbf{W}^*_t$.

Computing frame-to-mosaic homographies this way worsens misalignment errors for frames distant from the reference. Consequently, we refine the homographies by running the feature detector again on adjacent pairs of warped images $\mathbf{W}^*_i, \mathbf{W}^*_j$ starting from $\mathbf{W}^*_{ref}$ and working outward. At this point misalignments across the

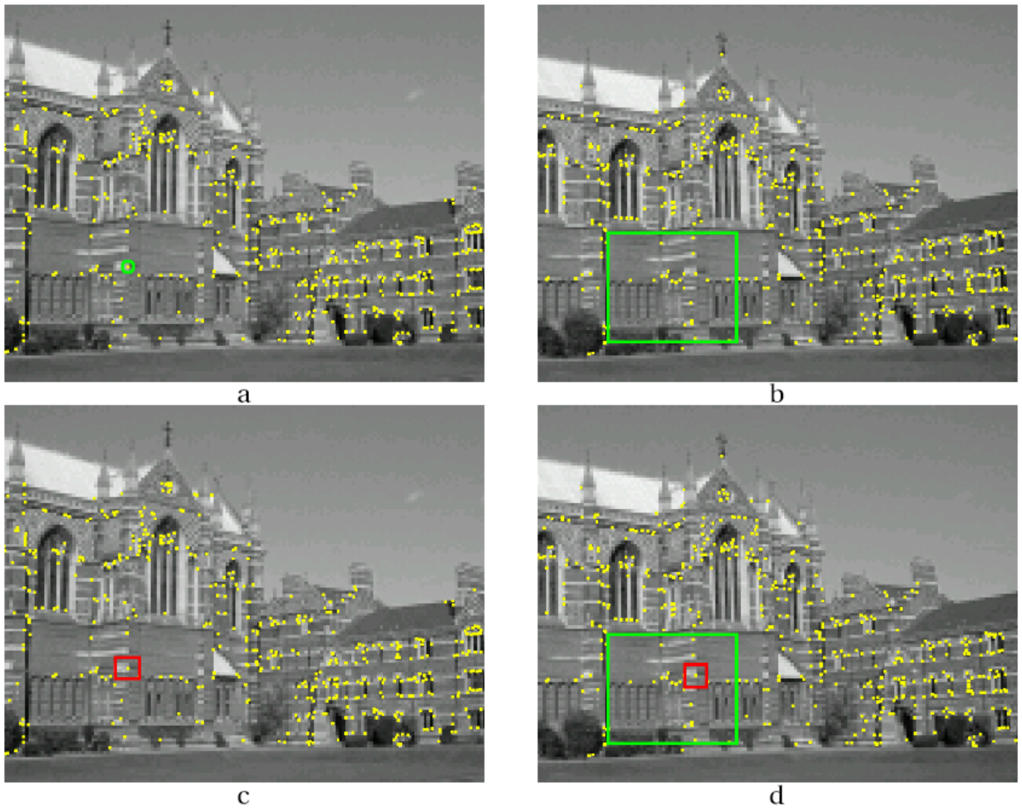


**Figure 4.6:** Obtaining putative correspondences. (a) An interest point on the left image and (b) its corresponding search window shown as a green rectangle. (c) and (d) Possible matches are ranked by computing the Sum-of-Squared Differences or normalized cross-correlation between image patches (in red). Reproduced from [19].

sequence are usually within 2-3 pixels; these are reduced by running a bundle adjustment to minimize reprojection errors across sets of overlapping frames. After cascading each of these transformations, we obtain a final set of refined frame-to-mosaic homographies $\mathbf{H}_{t,ref}$ and stabilized images $\mathbf{W}_t$ that are padded to the mosaic size.

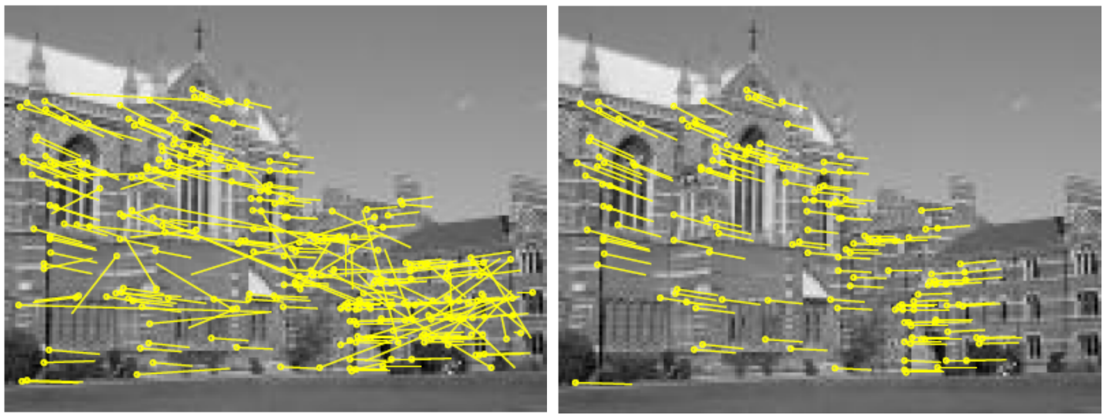

(a) (b)

**Figure 4.7:** (a) Set of all putative correspondences. (b) RANSAC inliers consistent with a single global transformation. Reproduced from [19].

### 4.1.2 Identifying Problem Pixels

Each location $\mathbf{p} = (x, y)$ in the mosaic reference frame has a set of pixels from the background stabilized images $\{\mathbf{W}_t(\mathbf{p})\}$ associated with it which we call its *timeline* $\mathcal{T}(\mathbf{p})$. The size of each timeline $|\mathcal{T}(\mathbf{p})|$ may vary from 0 to $N$ depending whether the pixel at $\mathbf{p}$ was imaged or not in each frame. Intuitively, since all pixels on the building facade exhibit the dominant motion, they should appear stationary in the mosaic whereas foreground objects such as trees and signs move due to parallax. Given that each $\mathcal{T}(\mathbf{p})$ contains an unknown mixture of background and foreground object pixels, our goal is to correctly pick or estimate each background pixel $\mathbf{M}(\mathbf{p})$ where $|\mathcal{T}(\mathbf{p})| > 0$, forming a building mosaic $\mathbf{M}$. In this paper we assume that the lateral and vertical limits of the building associated with corners, the roofline, the ground, etc. are given.

As asserted in the previously, a robust estimator for $\mathbf{M}(\mathbf{p})$ under the assumption that foreground pixels are in the minority (i.e., outliers) in $\mathcal{T}(\mathbf{p})$ is the *temporal median* $\mathbf{M}(\mathbf{p}) = \mathit{median}(\mathcal{T}(\mathbf{p}))$. This is computed separately for each color channel, giving rise to a *median mosaic* $\mathbf{M}_{med}$, an example of which is shown in Figure 4.4(a)

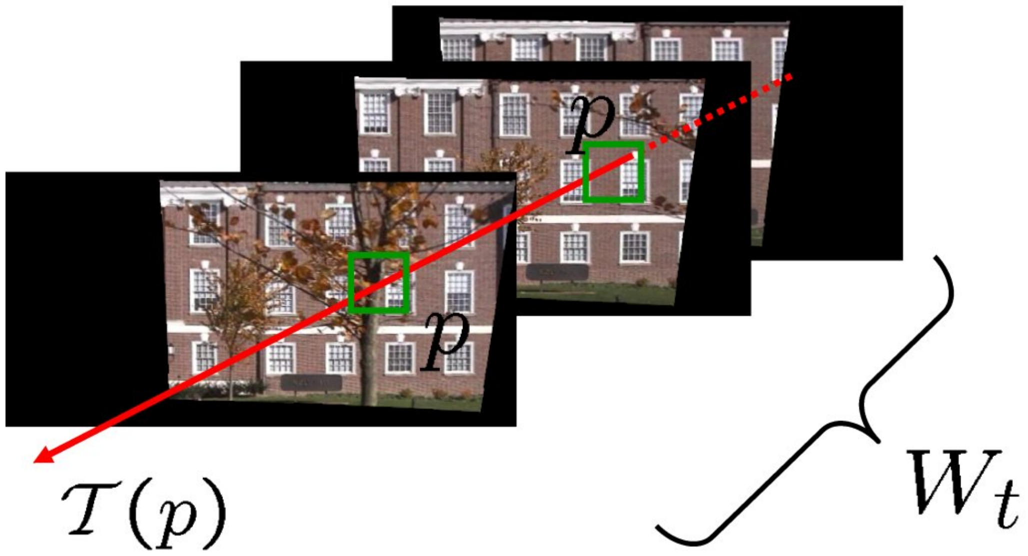


**Figure 4.8:** Every pixel in $M$ has a *timeline* of pixels associated with it in the stabilized sequence

(for the "Building A" sequence in Fig. 4.3). This estimator fails, however, when foreground pixels are in the majority in a particular timeline, and several artifacts can be seen in the figure. Since the background is stabilized, we have found that except for large homogeneous foreground regions or *camouflaged* foreground objects with almost the same color as the background, the likelihood that $\mathcal{T}(\mathbf{p})$ has a majority of foreground pixels is proportional to the variability or "spread" of its color distribution. To robustly measure this variability, we use the median absolute deviation (MAD) [159], defined as $MAD(\mathcal{T}(\mathbf{p})) = median(|\mathbf{W}_t(\mathbf{p}) - median(\mathcal{T}(\mathbf{p}))|)$ over all $t$ in the timeline. A scalar MAD value is obtained at each pixel by computing it separately for each color channel and summing. A high MAD value at $\mathbf{p}$ indicates a higher likelihood that $\mathbf{M}_{med}(\mathbf{p})$ is unreliable, so unreliable median mosaic pixels are filtered out by thresholding their MADs—these are so-called *MAD outlier* pixels. For normal distributions, the X-84 rejection rule [51] has been used to remove outliers using the MAD. Since the distribution of MAD values over all timelines

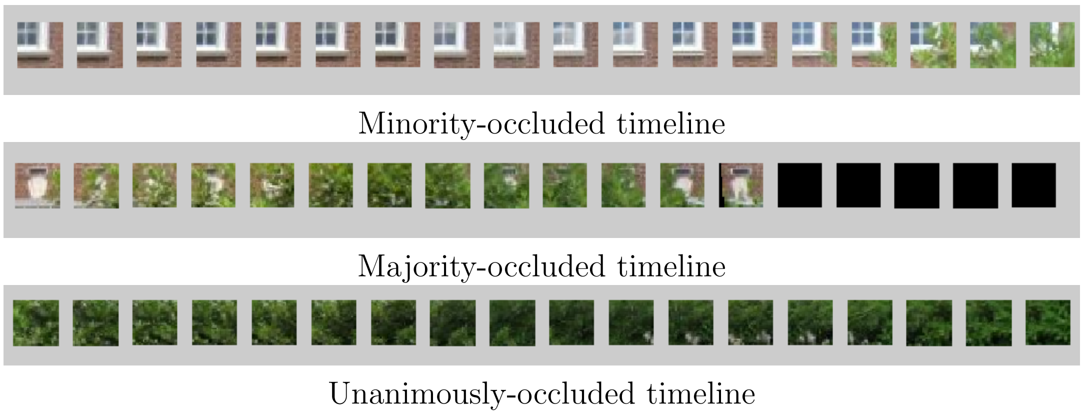

Minority-occluded timeline
Majority-occluded timeline
Unanimously-occluded timeline

**Figure 4.9:** Three categories of timelines. A median filter can recover pixels from minority-occluded timelines, while spatial inpainting is necessary for the unanimously occluded timelines. A primary concern in this work is on how to identify the three categories and recover building pixels from the majority-occluded timelines.

more closely approximated an exponential distribution in our sequences, the mean MAD value was chosen as the threshold. The raw MAD outlier mask is spatially smoothed with a morphological majority operation, forming the polluted region (or set of regions, more generally) to be filled. We call this the *target region* $\Omega$ in the sense of CPT inpainting (see Appendix). An example is highlighted in black in Figure 4.4(b). Note how well it matches the locations of foreground-induced artifacts in the median mosaic of Figure 4.4(a). Another class of pixels put into $\Omega$ is typically due to slight misregistrations of high-frequency features, and this can also be seen in the example around window panes and thin lines. The three categories of timelines for a particular patch are illustrated in Figure 4.9.

## 4.2 Discriminating Building Patches

The major hurdle to overcome in the majority-occluded scenario is determining which patches in a timeline come from the building. Let $\mathcal{T}(\Psi_{\mathbf{p}}) = \{\Psi_{\mathbf{p}}^1, \ldots, \Psi_{\mathbf{p}}^{|\mathcal{T}(\mathbf{p})|}\}$

be the timeline of patches centered on a location $\mathbf{p}$. Given a *building likelihood* function $B(\Psi^t_{\mathbf{p}})$ that intuitively captures the "buildingness" of a particular patch, the most likely patch in the timeline to have come from the building is computed as $\operatorname{argmax}_t B(\Psi^t_{\mathbf{p}})$. Very low values of $B$ should correspond to unanimously-occluded situations where there is no building patch in $\mathcal{T}(\Psi_{\mathbf{p}})$.

The building likelihood function that we use is a product combination of *motion* and *appearance* cues:

$$B(\Psi^t_{\mathbf{p}}) = p_{motion}(\Psi^t_{\mathbf{p}}) p_{app}(\Psi^t_{\mathbf{p}}) \tag{4.1}$$

Intuitively, the motion cue encodes the depth information that we get from parallax; less motion after stabilization means a higher likelihood of having come from the building. The appearance cue incorporates color, texture, or other patch characteristics computable from one image. Such features as the redness of the bricks can help identify a patch as having come from a building rather than a leafy green tree in the foreground. We now describe these in detail.

### 4.2.1 Motion likelihood

The intersection of a pair of successive, thresholded difference images was suggested in [161] as a method for identifying foreground pixels. By converting the warped images to grayscale and scaling their intensity values to $[0, 1]$ to get $\{\mathbf{W}'_t\}$, we can adapt this approach to define a motion energy or *foreground image* at time $t$ as $\mathbf{F}_t = (|\mathbf{W}'_t - \mathbf{W}'_{t-1}|) \otimes (|\mathbf{W}'_{t+1} - \mathbf{W}'_t|)$ where $|\cdot|$ is the absolute value and $\otimes$ is the pixelwise product. Letting $\mu$ be the mean foreground image value over all $t$, we define the motion likelihood for an individual pixel at $\mathbf{p}$ in warped image $t$ as $p^t_{motion}(\mathbf{p}) = e^{-\mathbf{F}_t(\mathbf{p})/\mu}$, and $p_{motion}(\Psi^t_{\mathbf{p}})$ as the fraction of pixels in $\Psi^t_{\mathbf{p}}$ with motion likelihood above $\tau_{motion}$. This threshold was empirically fixed at 0.1 for all the sequences. A conservative estimate was chosen since this is usually used in conjunction with the appearance-based patch classifiers – a more reliable metric for filtering out bad patches.

### 4.2.2 SSD appearance likelihood

The sum-of-squared differences (SSD) $|\Psi_{\mathbf{p}} - \Psi_{\mathbf{q}}|^2$ between the pixel intensities of two patches $\Psi_{\mathbf{p}}$ and $\Psi_{\mathbf{q}}$ is the basic similarity measurement used for matching in CPT inpainting. In [131] we used SSD to define the appearance likelihood for a particular timeline patch $\Psi^t_{\mathbf{p}}$ in the following way:

$$p^{SSD}_{app}(\Psi^t_{\mathbf{p}}) = \exp(-\min_{\mathbf{q}\in\Phi} |\Psi^t_{\mathbf{p}} - \Psi_{\mathbf{q}}|^2) \tag{4.2}$$

where the *source region* $\Phi$ is defined as the complement of the target region $\Omega$—i.e., the portion of the median mosaic that we regard as foreground-free and thus assumed building. In words, the SSD-based building appearance likelihood is proportional to the similarity between a candidate patch and its best match in $\Phi$. The intuition here is that many buildings have repeated patterns such as windows, doors, columns, bricks, etc., so building timeline patches in $\Omega$ should find better matches in $\Phi$ than foreground patches and thus have higher values of $p^{SSD}_{app}$.

### 4.2.3 Learned appearance likelihood

Although the accuracy of the results obtained in [131] using $p^{SSD}_{app}$ were quite good, a drawback of this approach is the expense of computing it and its inadequacy as a binary classifier to prevent foreground patches from polluting the mosaic. To find the best match, an exhaustive search over all candidate locations $\mathbf{q} \in \Phi$, with repeated calculations of the SSD, is required. Since the essence of the problem is deciding whether a given image patch belongs to the building or not, we have also investigated several classifier-based approaches to formulating $p_{app}$. These have proved significantly more efficient allowing more of the background to be recovered from the timeline without appreciable diminution of the quality of the final mosaic.

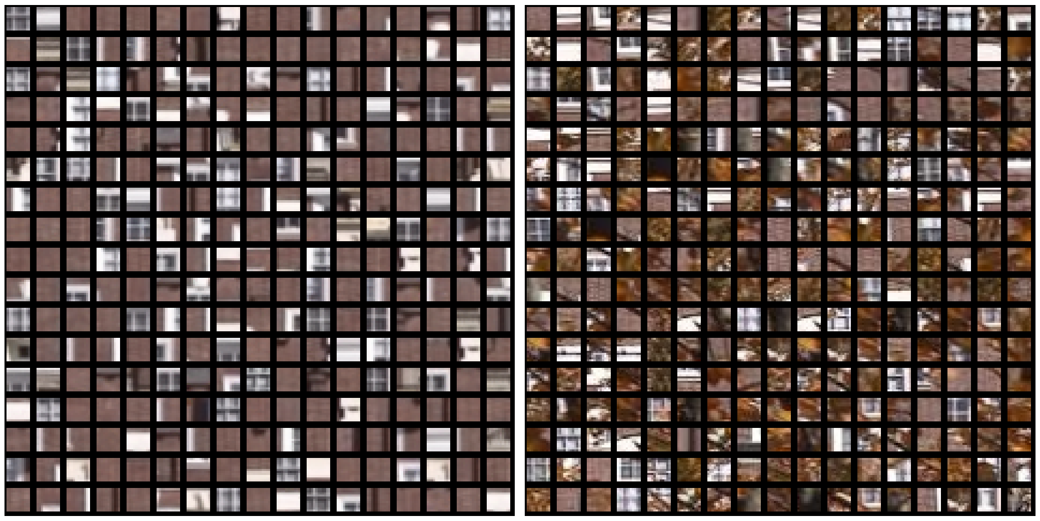

(a) (b)

**Figure 4.10:** Learning a building appearance model from automatically labeled patches: (a) Examples of building patches sampled from $\Phi$ (Building A sequence); (b) Negative examples derived from RANSAC outliers

Many practical classification and semantic recognition algorithms [165, 172, 107] rely on supervised training, with a user manually indicating positive and negative examples. Learning an *a priori* patch-level model of building appearance from a large set of examples is possible, but the value of such models would be highly dependent on architectural style, building materials, and so on. Seeking to allow our algorithm to work on any sequence of building images, we instead attempt to automatically label a training set of patches which are unambiguously foreground or background based on motion cues, permitting a sequence-appropriate classifier to be trained in the standard way. We have experimented with several types of feature vectors $F$ and classifiers $C$; to distinguish the appearance likelihoods derived from them we use the notation $p_{app}^{F+C}$.

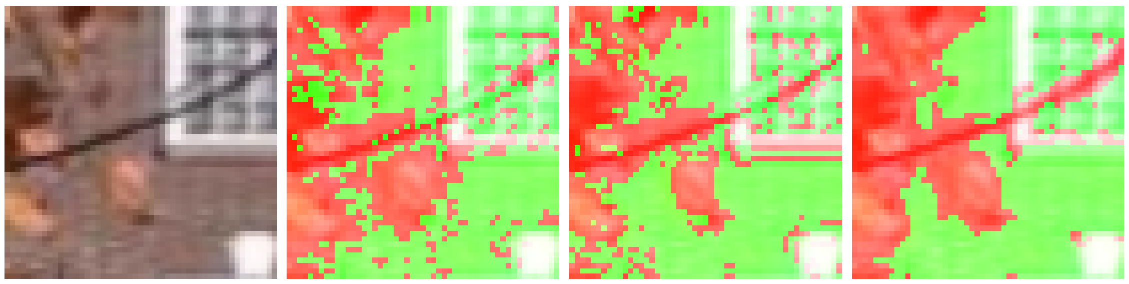


**Figure 4.11:** Comparison of learned appearance models on magnified subimage of tree branch. Green indicates that the patch centered that pixel was classified as building while red indicates foreground

**Automatic labeling**

Nominal positive examples of building patches were selected from the non-polluted mosaic region $\Phi$ by sampling over a regular grid. To try to capture important building features, we augmented this set with patches centered on Harris corners detected in $\Phi$, as well as inliers from the RANSAC process during the image registration stage. Negative examples—that is, foreground patches belonging to trees, foliage and so on—are harder to generate. Simply sampling from part of an image $\mathbf{W}_t$ in the polluted region $\Omega$ may retrieve a building example if the timeline there is not unanimously-occluded. Instead, we use the RANSAC outliers from the image registration stage as the centers of non-building patches, a reasonable assumption since the dominant motion is due to the building. RANSAC features that correspond to occlusion-junctions with both foreground and background are also to be marked as a negative example. Figure 4.10 shows a subset of the patches for the Building A sequence that were used for training. Very few of the building patches are mislabeled, but one can see that some fraction of non-building examples actually are from the building. Since these are in the minority, we rely on the generalization ability of our classification algorithm to be robust to these incorrect labellings.

## Features

We investigated two techniques for reducing the dimensionality of the raw image patches for practical training. First, we used *Principal Component Analysis* (PCA), a standard approach in recognition [163], on each cluster of positively- and negatively-labeled patches. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance after projection lies on the first coordinate, the second greatest variance on the second coordinate, and so on. As a dimensionality reduction tool, PCA enables us to store the most important characteristics of the data by keeping only the lower-order principal components and ignoring the less important higher-order ones. The many applications of PCA in computer vision include face recognition through eigenimages [163], feature matching [78], texture synthesis [143] and texture manipulation [105]. In our case, given the automatically generated patches, roughly the first 30 principal components (eigenpatches) were retained after PCA.

Observing that building patches are likely to contain prominent linear structures in stereotypical horizontal and vertical orientations vs. the more haphazard distribution of tree limb orientations, the second feature extraction technique used was a *filter bank* (FB). Filter banks have been widely used for texture recognition [97, 165] as well as object/scene categorization [172, 107]. We employ the Base Filter Set used by [165] which consists of 34 filters: 8 orientations at 2 scales for 2 types, plus 2 isotropic filters. For gross color information, we append the mean R, G, and B color value of each patch to obtain a 37-dimensional feature vector. Other color spaces could be used as well.

## Classifiers

We compared two classification methods: a distance-weighted $k$-nearest neighbor voting scheme (KNN) [39] and support vector machines (SVM) [75]. A value of

$k = 10$ with a hard threshold of 0.85 as a decision boundary was used for the former classifier, and a radial basis function (RBF) kernel ($\gamma = 1.0$) for the latter. These parameters were chosen by running cross-validation experiments on the training set.

A comparison of the output of the layer classifiers on a subimage from the Building A sequence containing a branch in front of a window is shown in Figure 4.11. Generally, $p_{app}^{FB+SVM}$ exhibited the best performance, and it was used for all inpainting results shown here unless otherwise noted.

## 4.3 Timeline Inpainting

We create a *timeline mosaic* $\mathbf{M}_{time}$ by modifying CPT inpainting (see Appendix for review) in five major ways:

1. There are two stages. Stage 1 consists of searching temporally over timelines, while Stage 2 is traditional spatial inpainting. In Stage 1, each patch-wise pixel copy to the polluted region $\Omega$ comes *from one timeline patch* $\Psi_{\hat{\mathbf{p}}}^* \in \mathcal{T}(\Psi_{\hat{\mathbf{p}}})$ maximally likely to have come from the building. An example result of this stage is shown in Figure 4.5(a).

2. During Stage 1, the updated confidences $C(\mathbf{p})$ of newly-filled pixels are set to the *motion likelihoods* $p_{motion}^*(\mathbf{p})$ of the pixels in $\Psi_{\hat{\mathbf{p}}}^*$

3. To compensate for photometric disparities through the sequence, copied patches are radiometrically aligned with the mosaic

4. If the building likelihood $B(\Psi_{\hat{\mathbf{p}}}^t)$ for every patch in $\mathcal{T}(\Psi_{\hat{\mathbf{p}}})$ is very low (below a threshold), $\Psi_{\hat{\mathbf{p}}}$ is *not filled* at that time. Stage 2 begins when all remaining areas of $\Omega$ meet this *stopping criterion*

5. Stage 2 is pure CPT inpainting with a few heuristics based on architectural patterns and symmetry to minimize mismatching errors. An example result of Stage 2 is shown in Figure 4.5(b).

Each of these modifications is explained below:

**Stage 1 Timeline patch selection** Consider a patch $\Psi_{\hat{\mathbf{p}}}$ in the mosaic $\mathbf{M}_{time}$ that is the next to be inpainted according to the CPT priority scheme. Pixels in its unfilled part $\Psi_{\hat{\mathbf{p}}} \cap \Omega$ will come from the corresponding part of one timeline patch $\Psi^*_{\hat{\mathbf{p}}} \cap \Omega$. We copy pixels from the timeline rather than $\Phi$ to maximize correctness, improve feature alignment, and allow for the retention of unique features not present in $\Phi$. To pick a $\Psi^*_{\hat{\mathbf{p}}}$ that is most likely to contain building pixels rather than foreground pixels, we rely upon the building likelihood function (4.1) defined in the previous section.

For every patch in $\mathcal{T}(\Psi_{\mathbf{p}})$, we jointly measure patch $t$'s motion and building appearance likelihood with the formula $B(\Psi^t_{\hat{\mathbf{p}}}) = p_{motion}(\Psi^t_{\hat{\mathbf{p}}}) p_{app}(\Psi^t_{\hat{\mathbf{p}}})$. Pixels are then copied from $\Psi^*_{\hat{\mathbf{p}}}$ where $* = \text{argmax}_t\, B(\Psi^t_{\hat{\mathbf{p}}})$. As explained in the Appendix, the circumflex notation simply indicates the particular patch that is next in line to be filled.

**Confidence term** The motion likelihoods $p^*_{motion}(\Psi_{\hat{\mathbf{p}}} \cap \Omega)$ are copied as the confidence values of the newly filled-in pixels in $\Psi_{\hat{\mathbf{p}}} \cap \Omega$. This tends to limit the propagation of bad choices in subsequent iterations—i.e., patches bordering areas of lower motion likelihood (i.e., higher chance of being foreground) are bypassed for high motion likelihood areas first (i.e., lower chance). The decaying confidence scheme of CPT inpainting does not apply because timeline patch pixels in the interior of $\Omega$ are no less reliable than those near its edges.

**Photometric alignment** Lighting changes in the scene and automatic camera controls can pronounce the seams between overlapping patches copied into the final mosaic. Graph-cut and gradient-domain algorithms [152, 125] have been used to minimize these effects in mosaicing and texture synthesis. Since most of the photometric variations in our sequence arise due to varying camera parameters, we experimented with exactly recovering the camera response curves for radiometric

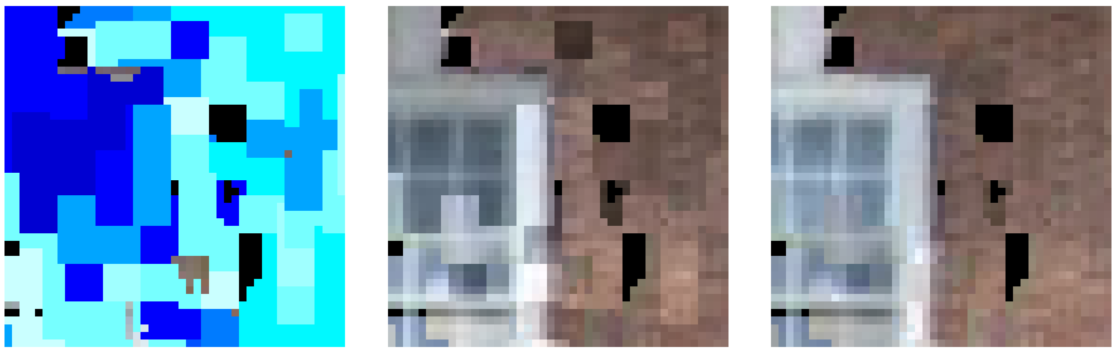


**Figure 4.12:** Results of photometric correction during inpainting on a section of the Building C sequence. Darker shades of blue in the frame index image indicate patches copied from earlier in the timeline

compensation [80]. However this proved very sensitive to geometric misalignments and foreground objects. Noting that an affine transformation across each RGB channel is able to fully account for contrast and brightness changes [18], we simply use a multiplicative term $\lambda_k$ that represents the contrast changes. Since this was able to capture most of the radiometric changes, we chose not to introduce more free parameters into the model. When pixels from the best chosen patch $\Psi^*_{\hat{\mathbf{p}}}$ are to be copied into the timeline mosaic $\mathbf{M}_{time}$, $\lambda_k$ is estimated by least squares minimization over the overlapping pixels. This correction is applied before the missing pixels are filled in.

Figure 4.12 shows the result of inpainting with and without photometric alignment on a section of the Building C sequence. This sequence had significant lighting changes (sun was hidden behind the clouds for a number of frames) and reflections off the windows. The mosaics with photometric correction appear much more consistent and visually pleasing. Our model is applied locally to a small patch, but is still able to propagate across a larger scale.

**Stopping criterion** To prevent copying patches from timelines where the background was never imaged, we set appearance and motion thresholds $T_{app}$ and $T_{motion}$.

$T_{app}$ is essentially the binary decision from the classifier while $T_{motion}$ is set to 0.85 requiring atleast 85% of the pixels to have low motion energy. Specifically, if for every patch in $\mathcal{T}(\Psi_{\hat{\mathbf{p}}})$ either the motion likelihood $p_{motion}(\Psi^t_{\hat{\mathbf{p}}}) < T_{motion}$ *or* appearance likelihood $p_{app}(\Psi^t_{\hat{\mathbf{p}}}) < T_{app}$, $\Psi_{\hat{\mathbf{p}}}$ is not filled. Stage 1 halts when this condition is true at every remaining $\mathbf{p} \in \Omega$. The holes that are left are generally much smaller than $\Omega_0$, with more building structure revealed, and thus Stage 2 can consist of pure inpainting with much better results than if it had been run in place of Stage 1.

Since (4.1) operates on patches, it does not guarantee against blemishes in the mosaic that occur when tiny fragments of foreground pixels are copied over. Thus a per-pixel decision is also made before copying patches from the timeline - once again based on appearance and motion. Pixels with motion likelihood below $\tau_{motion}$ are not copied to $\mathbf{M}_{time}$, preventing thin foreground structures that are usually more sensitive to motion than appearance. Similarly, a threshold on the appearance likelihood $\tau_{app}$ (learned from Gaussian color models) can be defined to avoid copying bits of obvious foreground elements like green leaves into the mosaic.

**Stage 2 spatial inpainting** Mosaic pixels that were never imaged in the timeline are detected in Stage 1 and marked as a hole - to be completed in Stage 2 by a general spatial inpainting algorithm. Given that most of the background has been recovered in Stage 1, only a small fraction of pixels require conventional inpainting. We use the CPT algorithm of [27] with a few heuristics derived from domain knowledge. Firstly, we search within the warped sequence $\mathbf{W}$ rather than the result of Stage 1 to improve the likelihood of finding a good match. Secondly, since building facades exhibit grid-like patterns, we limit the SSD search to lie within a horizontal or vertical band centered on the target patch. This serves to speed up the search through the sequence and reduce the chances of picking a wrong patch to copy into the hole.

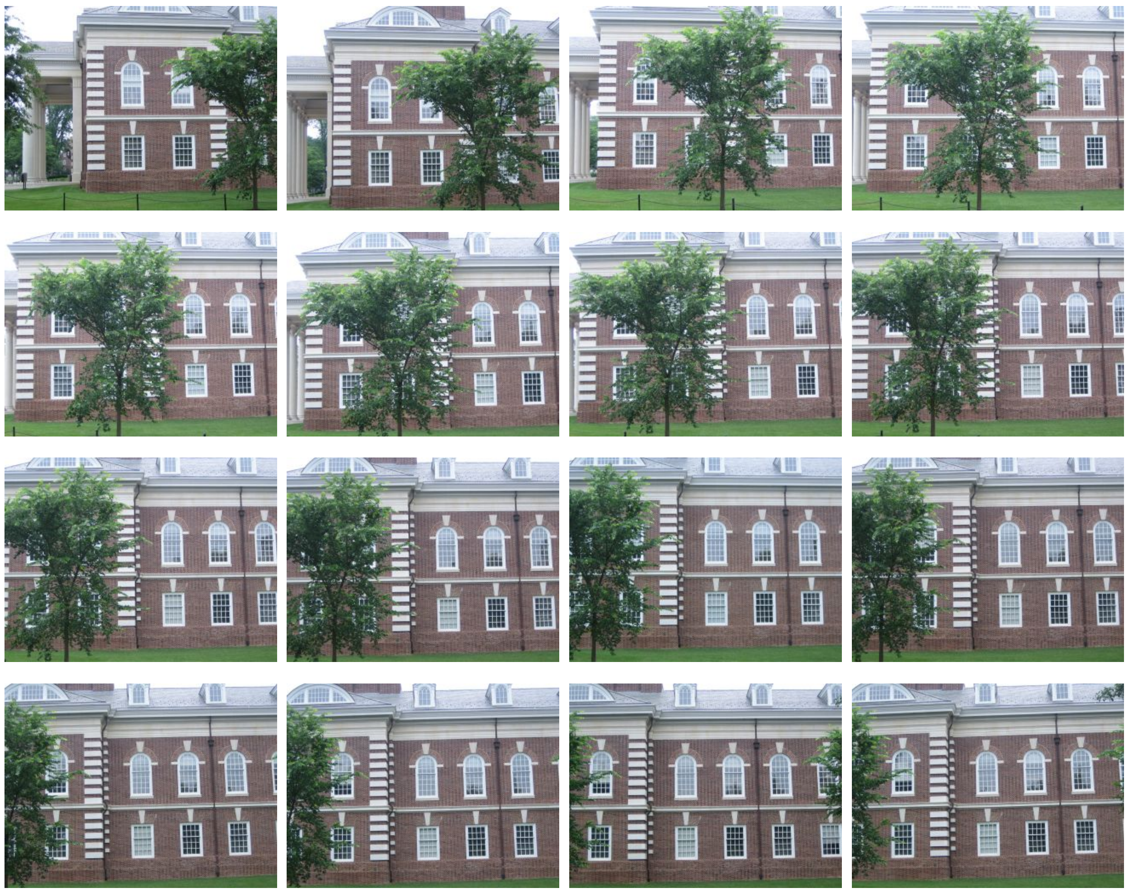
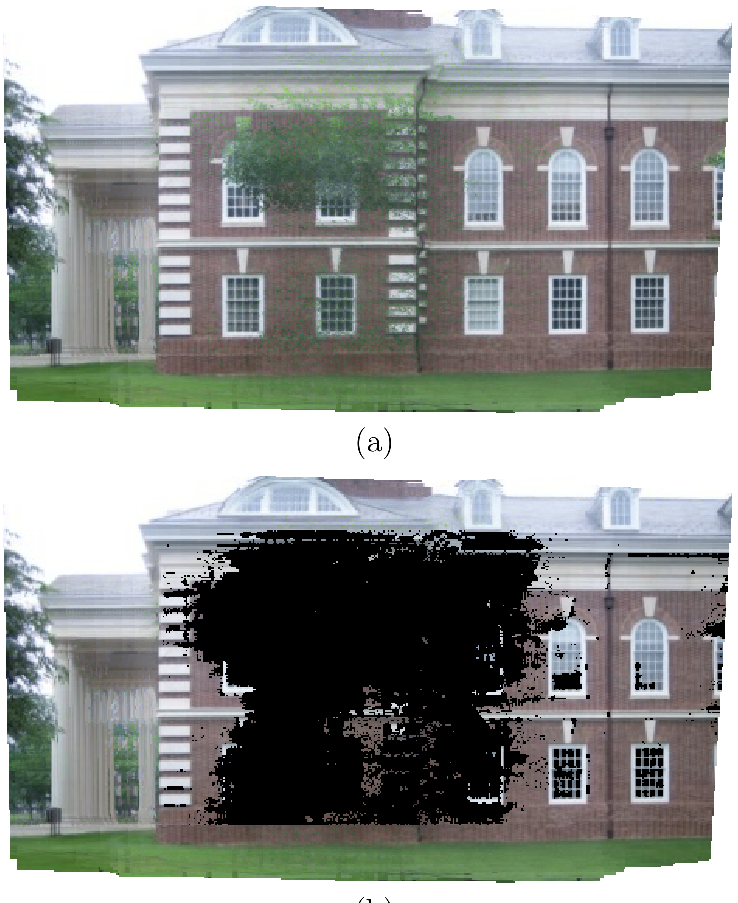


**Figure 4.13:** 16 of 22 frames from the Building B sequence.

## 4.4 Experimental Results

### 4.4.1 Texture Map Recovery

We show the overall quality of our timeline inpainting algorithms both at intermediate stages and in final results in Figures 4.5, 4.15, and 4.16. The Building A sequence (Figure 4.3) consisted of a subset of 18 360 × 240 frames taken at intervals of every 50 frames from a longer sequence. Building B and C sequences were captured as separate images by a digital camera and resized to 320 × 240 resolution. In all cases, the camera was moving roughly parallel to the building facade. Each sequence has several objects, primarily trees, which occlude large

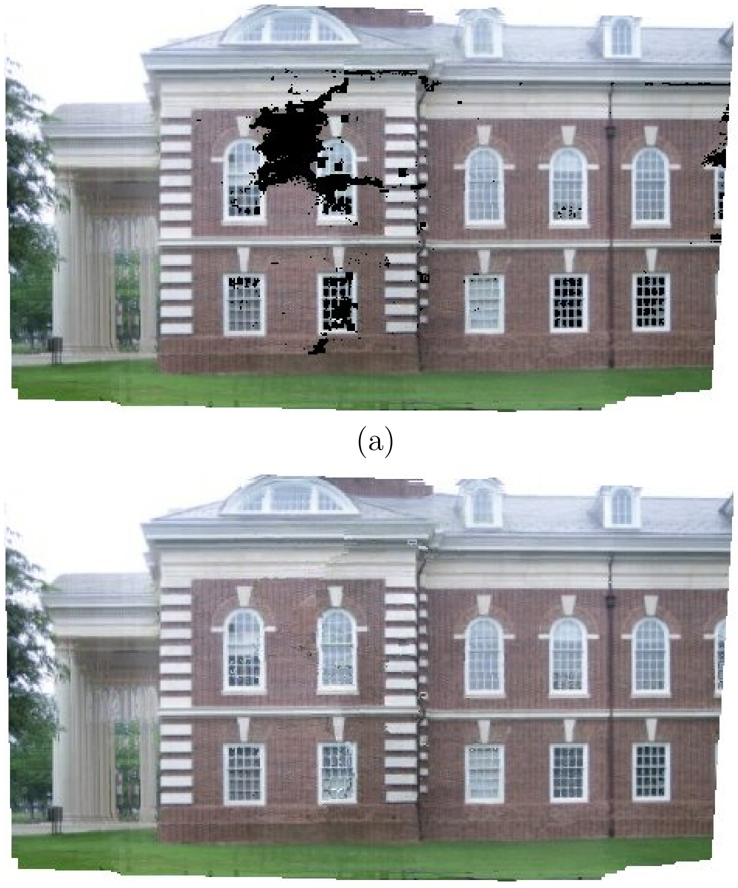

(a)

(b)

**Figure 4.14:** Result from the first two stages of our algorithm for the Building B sequence (Fig. 4.13). (a) Median mosaic; (b) Polluted region $\Omega$ to be fed to the inpainting algorithm for filling;

parts of the building.

The median mosaic $\mathbf{M}_{med}$ for all three sequences is mostly quite good, recovering almost all of the facade cleanly. Foreground objects closer to the camera are almost entirely removed because their large parallax motions cause occlusions to be brief and thus foreground pixels are in the minority in the timeline vs. building pixels. Significant problem areas are created by the more distant trees, which exhibit relatively little parallax motion. These objects occlude many building pixels in a majority of frames, confounding the median filter. Areas where $\mathbf{M}_{med}$ is

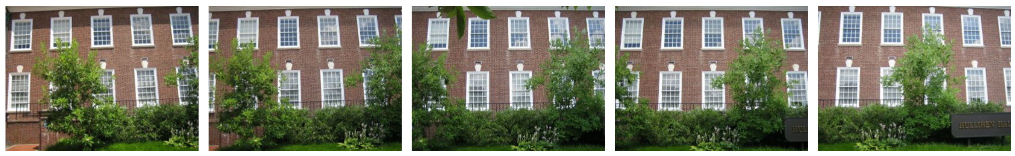

(a)

(b)

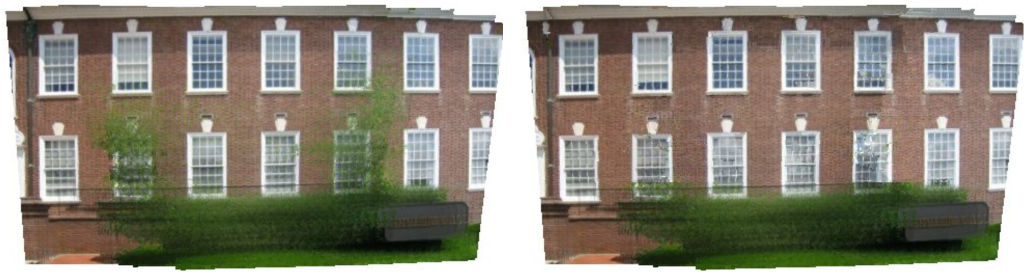
**Figure 4.15:** Result from the inpainting stages of our algorithm for the Building B sequence (Fig. 4.13). (a) Result after timeline inpainting Stage 1 ($11 \times 11$ patches); (b) Result after Stage 2 ($21 \times 21$ patches)

poor correlate well with the MAD outliers. It appears that the unfilled pixels after Stage 1 inpainting also match well with perceived unanimously-occluded areas. These considerably smaller holes are well-filled by Stage 2, though there are some high-frequency artifacts that we attribute to slight misregistrations in the stabilization process. The horizontal and vertical boundaries of the facade were manually specified in this work, though these boundaries could conceivably also be found automatically.

Restricting patch copies to be from the timeline makes the result sensitive to



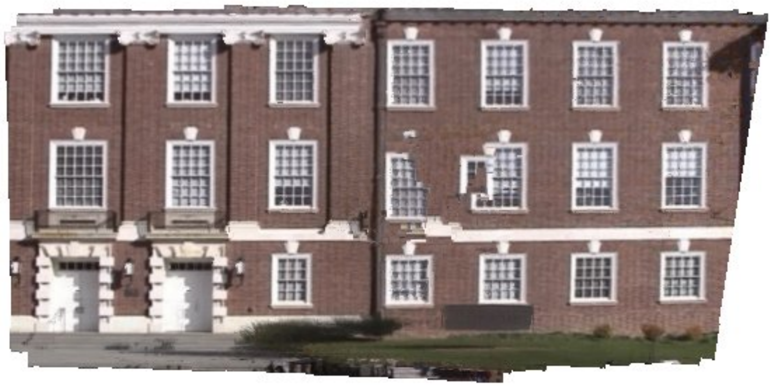
Every alternate frame from 6 to 14 in an 18 frame sequence. Note lighting variations and window reflections.



(a) (b)

**Figure 4.16:** Building C sequence: (a) Median mosaic; (b) Result after timeline inpainting Stage 2 ($11 \times 11$ patches in Stage 1, $21 \times 21$ in Stage 2)

the quality of the initial registration. Frame-to-frame transformations are modeled as homographies which are sensitive to any out-of-plane depth discontinuities. After bundle adjustment, misalignments in the Building A sequence were less than 1 pixel allowing patches from across the sequence to line up correctly in Stage 1 as shown in Figure 4.5(a). In contrast, the Building B sequence contains two planes at different depths, resulting in the dominant plane for image registration shifting from the left side to the right in the middle of the sequence. This effect is reflected in the result of Stage 1 (Figure 4.15(a)) where one can see that edges of the second story window do not line up perfectly. The Building C sequence in Figure 4.16 posed different challenges in the form of large variations in the lighting conditions and reflections off the windows. This hampered registration due to the difficulty of finding correspondences and also compromised the quality of pure spatial inpainting in Stage 2 due to the lack of accurate matches. Nevertheless, photometric correction reduces the seams and the synthesized mosaic is very similar to the true facade, parts


(a)

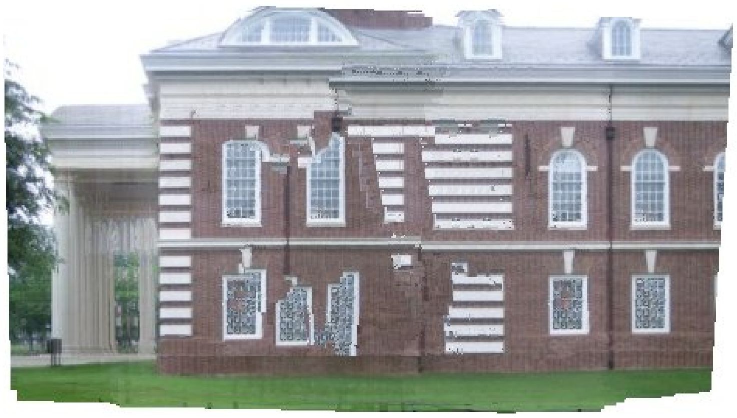

(b)

**Figure 4.17:** Result of CPT inpainting on MAD outliers of (a) Building A mosaic in 4.4b and (b) Building B mosaic in 4.14b.

of which were never imaged.

Figure 4.17 demonstrates the need for timeline inpainting in Stage 1. The results of CPT inpainting on the MAD outliers for both Building A and B sequences are shown. While the smaller holes can be filled in effectively, larger contextual information such as window positioning and edge alignments are not enforced. The result is a completely haphazard hallucination of facade elements, ignoring the temporal cues that timeline inpainting exploits.

### 4.4.2 Classification

The quality of the static classifier in distinguishing between building and non-building pixels based on the local statistics in a patch has a direct impact on the texture map quality. In addition to preventing small foreground pixels from bleeding into the final mosaic, it also selects timeline building patches that the motion likelihood alone would otherwise miss because of its temporal coarseness.

A few key factors can be noted from Fig. 4.11. All methods seem to detect most of the foreground or tree pixels. PCA essentially works on the RGB color values and does not seem to be able to pick up some of the high frequency variations that the filter bank can. Texture-based classification is thus able to do marginally better on some of the tiny leaves or thin branches that occlude the building. SVM seems the cleanest among all three methods for segmentation. It can generalize well over the training set with several examples and the classification is done in fixed time. In contrast, nearest neighbor approaches become very inefficient as we add more patches.

For quantitative results, we ran a leave-one-out test on the 2,908 patches in the training set for Building A. The best result using PCA was 15.1% error with 10 nearest neighbors and a distance threshold of 0.85. The lowest error using filter banks was 11.2% under the same settings. The best accuracy by far was obtained with SVM which misclassified only 3.6% of the training examples. On closer inspection, it was observed that most of these errors could be attributed to incorrect labels in the training examples, demonstrating the generalizing power of SVM.

Figure 4.18 demonstrates the shortcomings of pure SSD-based appearance likelihood [131]. As long as a background patch is present in the timeline, SSD matching with the rest of the mosaic is mostly able to recover it. However, setting an arbitrarily high motion threshold $\tau_{motion}$ is the only way to filter out bad patches.

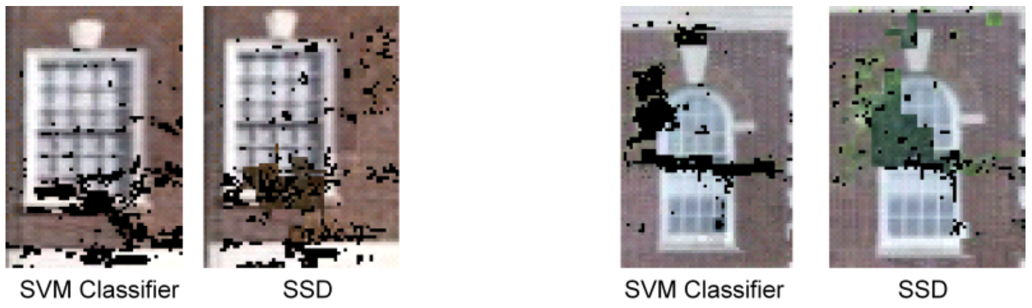


**Figure 4.18:** SSD-based matching relies on motion only to filter out bad patches. This is not very reliable when the foreground is large and homogeneous.

The figure zooms in on a couple of particularly challenging areas where building pixels were seldom, if at all, captured by the camera. Keeping the rest of the system intact while changing the appearance model to use SSD reveals how large homogeneous foreground patches can pollute the mosaic. In contrast, the appearance-based classifiers are better at sifting through the timeline patches.

### 4.4.3 Timings

The timeline inpainting algorithm presented here is fairly efficient. On our test machine,[2] the image registration and polluted region identification steps take less than 2 minutes for each of the three building sequences, and learning appearance likelihoods with SVM from thousands of training patches takes less than a minute. The bundle adjustment step is the only portion of the system written in Matlab; the rest is coded in C/C++. The speed-ups obtained in Stage 1 of the inpainting by using classifiers vs. the exhaustive search of the SSD method are documented in Table 4.1. The slowest part of the algorithm is Stage 2 of inpainting, taking up to 10 minutes per sequence as it must search over the entire mosaic for matches (although the grid constraints discussed above help considerably).

[2] A laptop running Windows XP with a Pentium M 1.7 GHz CPU and 512 MB RAM

| Sequence | $p_{app}^{SSD}$ | $p_{app}^{PCA+KNN}$ | $p_{app}^{FB+KNN}$ | $p_{app}^{FB+SVM}$ |
|---|---|---|---|---|
| A | 1176.9 | 82.0 | 97.7 | 89.5 |
| B | 1983.4 | 130.6 | 261.8 | 121.7 |
| C | 1422.2 | 93.1 | 118.7 | 112.3 |

**Table 4.1:** Timeline inpainting Stage 1 execution times (in seconds) using different appearance models

## 4.5 Summary

We have presented a novel approach to detecting and removing occlusions of building facades in image sequences using a combination of temporal and spatial inpainting. We also described a method of training appearance-based classifiers from a coarse RANSAC-based segmentation to recognize static imagery. Motion is used to bootstrap the learning and generalize over the training examples. We have applied the learned appearance models and motion cues in a spatiotemporal inpainting algorithm to recover texture maps of occluded building facades. Various types of visual features—both intensity-based and texture-based—were proposed for accurate classification of image patches. Good results were shown on building sequences that are quite difficult due to the high amount of structure accentuating slight errors or misalignments. We believe that the other algorithms cited here would not be able to recover as much of the facade behind the trees as well as we do.

# Chapter 5

# DISCOVERING NEAR-REGULAR TEXTURES ON BUILDING FACADES

The previous chapter showed how to recover the building facade using a combination of motion and appearance models from an image sequence. Parallax provides a means to separate out the background and foreground layers, upon which patch-wise appearance models can be learned. The combination of spatial and temporal cues are then used to recover as much of the building pixels in the texture map mosaic, as long as the pixel was imaged in at least one of the views. We now begin to address the issue of texture map recovery from a *single* image knowing that it is of a building. In addition to foreground objects, other challenges include removing graffiti on the walls, window reflections, and shadows.

Section 1.3 describes our system infrastructure for using high-level rules to extract semantic information from images. One semantic rule is that windows often appear as part of a larger grid. This rule can be powerful enough to recognize windows without any prior appearance models. We introduce an algorithm based on Markov Chain Monte Carlo (MCMC) simulation to parse this rule – i.e., discover and group grid structures in an image. Our customized proposal distribution is efficient and guides the simulation to converge quickly. Although the idea of identifying window patterns to glean additional information about the facade has been used by [92, 111, 115], our grouping algorithm is novel and not confined to the building domain alone.

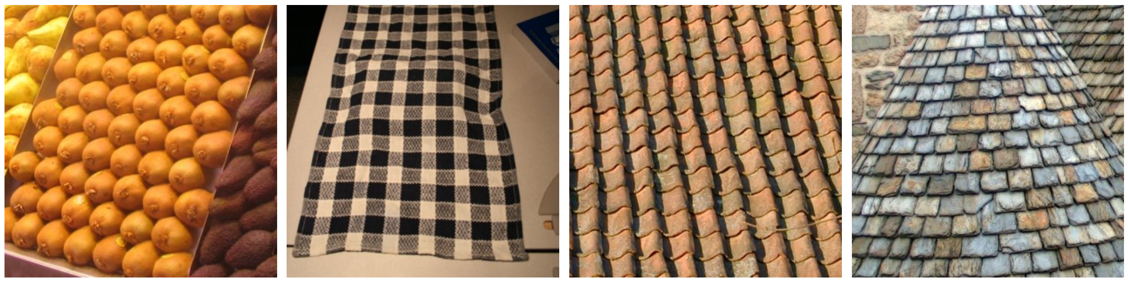


**Figure 5.1:** Examples of Near-Regular Textures from the PSU NRT database [91].

The notion of a grid or lattice pattern corresponds to the definition of a regular texture – a regular tiling of easily identifiable elements organized into strong periodic patterns. These tiles, called *texture elements* or *texels*, form the basic atomic unit of the repetitive pattern. Appearance models, while powerful, only rely on local information. Modeling all appearance and geometric deformations of a texel is hard. Instead, by discovering and grouping repetitive elements, we can exploit higher-order topological constraints characteristic of many man-made structures. Isolated elements that fail to pass a detector/classifier could still be correctly grouped due to global coherence. The key observation developed in this chapter is that building facades are Near Regular Textures (NRTs). Identifying and parameterizing this texture provides a platform for image understanding.

The next section defines near-regular textures and its relation to building facades, reviewing previous work on discovering and parameterizing them from images. We then describe two approaches to texture discovery specifically tailored in one case to brick wall images and in the other to window grids typically seen on large office buildings (both frontoparallel and under perspective). Instead of inferring the symmetry group of the pattern, for now we take user input indicating which type of texture it is. In the first case, there are several hundred bricks visible in a high-frequency running bond pattern, requiring a very efficient global approach. The second algorithm is a more general Bayesian approach to grouping elements

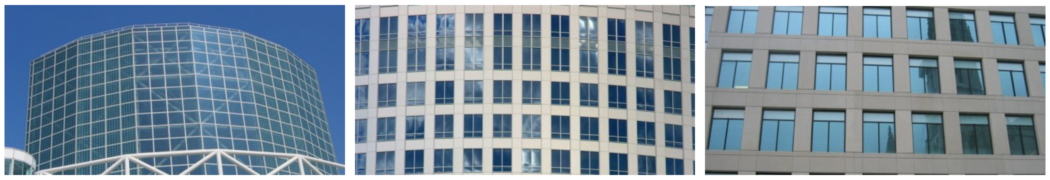


**Figure 5.2:** Buildings and Near Regular Textures.

according to lattice constraints. Finally, results are shown on a variety of synthetic patterns as well as building images.

## 5.1 Near-Regular Textures and Discovery

A near-regular texture (NRT) is a geometric and photometric deformation from its regular origin of a congruent wallpaper pattern formed by 2D translations of a single tile (texel) [105]. They can be encountered in buildings, wallpapers, tiles, windows, and arts. Figure 5.1 shows some examples from the PSU NRT database [91]. It is apparent that despite random noise and stochastic variations, these patterns exhibit a tendency towards regularity and symmetry. Liu et al. [106] states that NRTs may be categorized according to departures from regularity along the dimensions of color, intensity, global or local geometric deformations, and pixel resolution.

The starting point for classifying a pattern as an NRT is the concept of regular tiling [57] which states that all translationally symmetric regular textures can be generated by a pair of shortest vectors $t_1$, $t_2$ applied to a minimum texel. This results in a partitioning of the texture into like elements that simultaneously produce a covering (no gaps) and a packing (no overlaps). According to **wallpaper group theory**, these generating vectors form a 2D *quadrilateral lattice* with 5 possible lattice shapes [24]. While a single pair of of $t_1$, $t_2$ vectors uniquely describe a regular texture, the translation vectors of an NRT is location dependent and described by $t_1(x, y)$, $t_2(x, y)$ [66]. Despite these local and global variations, the lattice topology

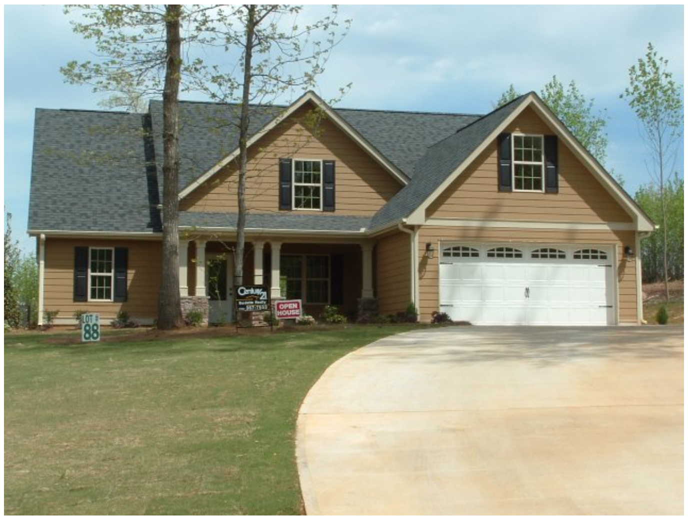


**Figure 5.3:** Suburban homes may not have dominant grid structures. We do not handle these in this work.

in an NRT still remains the same – i.e., quadrilateral.

Our motivation for modeling building facades as NRTs is to exploit the commonly occurring grid topology of windows. If such a pattern exists and can be discovered, it could allow us to identify anomalies such as occlusions and variation within windows. However, none but the most regular urban structures (top row of Fig. 5.2) would be classified as NRTs, as they've been defined above. The conditions of *packing* and *covering* restrict windows to appear right next to each other with no stochastic variations between them. This is not always true on building facades. The urban scene domain allows us to benefit from certain assumptions about the kind of texture we are looking for. First, we assume that the tiles (e.g. windows, bricks) are rectangular. Secondly, neighboring tiles of the lattice structure may be disconnected in the image, as long as they satisfy global appearance and topological constraints. These will be defined later.

It is important to note that the analogy is not valid in all cases. In particular, suburban homes (Fig. 5.3) often do not contain repetitive structures that dominate the building facade. These scenarios call for other context based scene analysis built on object recognition and visual category classification. In Section 1.3, we categorized this function as belonging to the Training and Recognition components of the Semantic Module. Very often, these systems depend on the generalization capability of learning algorithms from a small set of examples. Choosing the wrong training set can have dire consequences. In contrast, a rule such as "Identify Grid Structures" imposes less constraints and is widely applicable in most city and campus environments.

### 5.1.1 Texture Discovery: Related Work

A robust texture segmentation procedure is a necessary first step for background reconstruction. Texture discovery and segmentation are long-studied problems [109, 58]. One method by Leung and Malik [96] relies on looking for distinctive elements in the image and searching in its neighborhood for similar structures (repetitions). Regions with high matches are grouped together greedily without regard to the global structure. This "spatial tracking" approach has also been adopted for NRT lattice initialization by [144, 100]. Here, a user-selected texel is correlated with its neighborhood and the peaks of the correlation surface become candidates for addition to the lattice. The lattice construction proceeds in a greedy manner along the two principal translation directions of the NRT. None of these algorithms enforce global constraints on the structure. Without backtracking, incorrect assignments cannot be altered with new evidence about the pattern.

Rather than placing strong requirements on the appearance of each texel, other approaches assume that the texture is completely regular and can be exactly modeled by a parameterized transformation. Schaffalitzky and Zisserman [137] group repetitions of lines and rectangles to build a high-level feature that could be used for matching or shape recovery. Their RANSAC approach to finding repeated elements related by an elation [64] (translations on a plane) is shown to be effective in man-made scenes. Turina et al. [162] use the cascaded Hough transform to group

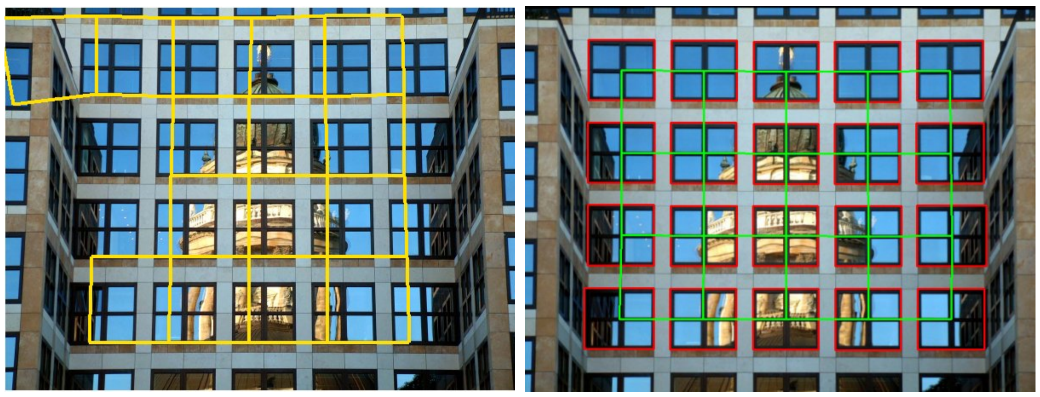

(a) (b)

**Figure 5.4:** Comparison of the resulting lattice discovered by (a) Hays et. al [66] and (b) our algorithm.

regular repetitions under all kinds of homologies. Being geared towards planar patterns under perspective, these approaches [137, 162] seem appropriate for grouping windows on building facades. However, many buildings in our test set do not exhibit such "checkerboard" style consistency. Independent variations in the size or location of windows cannot be modeled well by a single global transformation. A near regular texture with locally smooth geometric and appearance deformations seems more apt.

An algorithm for NRT discovery by lattice growth was described by Hays et al. [66]. They use more relaxed appearance and geometric constraints, but impose a global lattice structure on the texture. Lattice finding is formulated as a higher-order correspondence problem where neighbor relationships are established among a set of potential texels, with constraints on smoothness, self-assignments and so on. A bottom-up process extracts candidate texels which then have optimal neighbors assigned in the two principal directions. This *lattice assignment* is performed by a spectral technique [95] that optimizes affinities between all pairs of texels and edges. Since NRTs may be arbitrarily deformed, the above steps are repeated in an iterative framework to regularize and grow the current lattice.

Such region-growing techniques can occasionally be stymied by the presence of foreground objects in some tiles, preventing proper matching with their unoccluded neighbors. Although the method in [66] worked well on many images we tested, it did not find the entire textured region when blocked by significant occlusions. Besides being computationally expensive (averaging over half an hour per image on an Intel Core2 Extreme X6800), discovered tiles do not correspond exactly to semantically meaningful units. As Figure 5.4 shows, while the scale is often correct, there is an arbitrary shift that requires post-processing to center on the windows. In comparison, our algorithm returns the window boundaries as well as the neighborhood graph.

Bayesian approaches have also been applied for localizing grid structures. Markov Random Fields [98] have been used in microarray analysis [62, 20], where a global grid template is matched to a grid structure undergoing both global and local deviations of the nodes. However, it assumes that prior models of appearance exist and the deformation is purely Gaussian. An initial rough grid is placed on the structure followed by simulated annealing to maximize the posterior distribution of the grid nodes. Since they are targeting applications in genome sequencing, it is unclear how one would determine the initial grid for arbitrary NRTs. The extent of grid deformation is limited by their use of a global template. Lin and Liu [101] also define an MRF model for tracking dynamic deformable NRT lattices. By enforcing the topological invariance property of NRTs, they were able to show superior performance against previous tracking algorithms.

## 5.2 Spectral Analysis for Brick Patterns

Our baseline texture analysis method is a power spectrum approach [109] that although simple works well on a variety of brick images. Absolute horizontal and vertical gradients $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are computed and averaged in the first case down columns to obtain a column edge strength function $C(x)$ and in the second across rows to get a row edge strength function $R(y)$. A 1-D discrete Fourier transform is applied to each function and a mean-square power spectrum or periodogram is then constructed. The dominant vertical and horizontal periods are inferred from the highest-power frequencies; these are the estimated average brick height $h$ and half of the width $w$ (because of the running bond pattern). An origin is found by choosing the horizontal shift $-w/2 \leq \Delta x \leq w/2$ and vertical shift $-h/2 \leq \Delta y \leq h/2$ which best align the predicted grid with $C(x)$ and $R(y)$, respectively. Brick rows often do not line up perfectly, so a last adjustment allows individual rows to shift independently to optimize registration.

As some have noted [58], this approach is prone to failure on scenes containing more than one large region of texture, but it is extremely efficient for images with a large number of tiles.

## 5.3 Grouping Lattice Structures

As a more general method, we define a lattice-based Markov Random Field (MRF) [98] to model the window grid in particular and NRTs in general. The reviewed literature suggests that an effective methodology for grouping should combine the region growing techniques of [96] with the global topological restrictions of [137], while allowing smooth fluctuations from the perfect grid. This is indeed the approach used by Hays [66]. However, we wish to reduce some of the excessive time and space complexity of their algorithms, especially since the grouping is performed on hundreds of candidate tokens from the image.

This section describes our Markov Chain Monte Carlo (MCMC) [9, 117] algorithm to extract grid structures from the image. First, we enumerate various image discretization methods to identify potential texel elements. A Bayesian formulation that captures the likelihood of grid structures in the image is defined. A novel MRF prior that models the lattice topology is then described. This is followed by details of the optimization to arrive at a MAP estimate by dynamically adding and removing edges from the MRF.

### 5.3.1 Image Tokenizing

Many texture processing and segmentation algorithms [135] use the pixel-grid as the underlying image representation. A more natural alternative for grouping high-level features would be to work with the perceptually distinctive entities in the image. Similar to the lexical analysis phase of a compiler, the image is preprocessed to demarcate potential texels. The nature of the texture should dictate the particular method used, as our grouping algorithm works independently. Previous algorithms for texture discovery [96, 137, 162] have used interest point or corner detectors. Hays [66] uses MSER features [108] to propose initial texels, following which peaks of a correlation map are used. While general, these features often do not correspond to visually semantic entities in the image. Since we are interested in rectangular shaped windows, a method to generate independent rectangle proposals from the input image is used.

#### Hypothesizing Rectangles

The first stage consists of finding as many straight lines as possible. Canny edge detection is applied to locate high gradient pixels in the image. Similar to the technique of Zhang and Kosecka [179], the orientation of each edge pixel is quantized into 8 bins. However, instead of computing the gradient direction from the Sobel operator, we convolve the image with filters from the RFS filterbank [165] at 8 orientations and a single scale. Each pixel is assigned to the bin of the filter that had the maximal response. This technique was found to be more robust to noise, especially in the presence of occlusions or weak edges. Adjacent edge pixels in the same orientation bin are grouped together using connected components to obtain a number of straight lines from the image. For efficiency, we retain only lines that had more than 15 pixels (approximately 2% of image dimensions) grouped together.

Each connected component is parameterized as a line by its centroid, slope and magnitude. This is computed using weighted least squares fitting.

If the image has been rectified, rectangular structures may be represented by just 4 parameters – upper-left corner $p_k$, width $w_k$ and height $h_k$ for rectangle $k$. Rectangles under perspective could be parameterized by its vanishing points as used by Han [59], but this would imply that any errors in the initial vanishing point estimation would cascade through to the ensuing stages. Therefore we simply define them as $(p_k^0, p_k^1, p_k^2, p_k^3)$ denoting the 4 corners of a quadrilateral in anti-clockwise order with $p_{k1}$ as the upper-left position. A discriminative bottom-up approach for rectangle finding has been mentioned by [179]. In that work, rectangles are hypothesized from two pairs of lines, each of which is picked from two different vanishing directions. The validity of the hypotheses is verified by looking for corner features in the image. Since ours is a top-down approach that takes high-level topological constraints into account, we can afford to be more conservative about which rectangles pass through to the next stage [59].

To generate possible rectangles from the perspective image, pairs of line segments ($l_i$, $l_j$) are exhaustively drawn and tested. Compared to [179] and [59] which need all 4 lines, the sufficiency of line pairs makes our algorithm less sensitive to broken edges in the edge detection. The endpoints of $l_i$and $l_j$form the corners of a quadrilateral with interior angles $\theta_l : l \in 0, .., 3$. The test for parallelism and magnitude requires that $\max(|\cos(\theta_l)|)$ be less than a threshold $\tau_\theta$. We choose $\tau_\theta = 0.1$ for rectified images and relax the constraint for perspective with a value of 0.3. If the lines qualify, the four corner points may be refined by the presence of actual image edges before being added as a rectangle hypothesis. Except for extremely oblique views of buildings, we have found this simple criterion effective. Additional constraints such as aspect ratio could also guide the hypotheses generation.

This process results in a couple of thousand rectangles for a typical image.

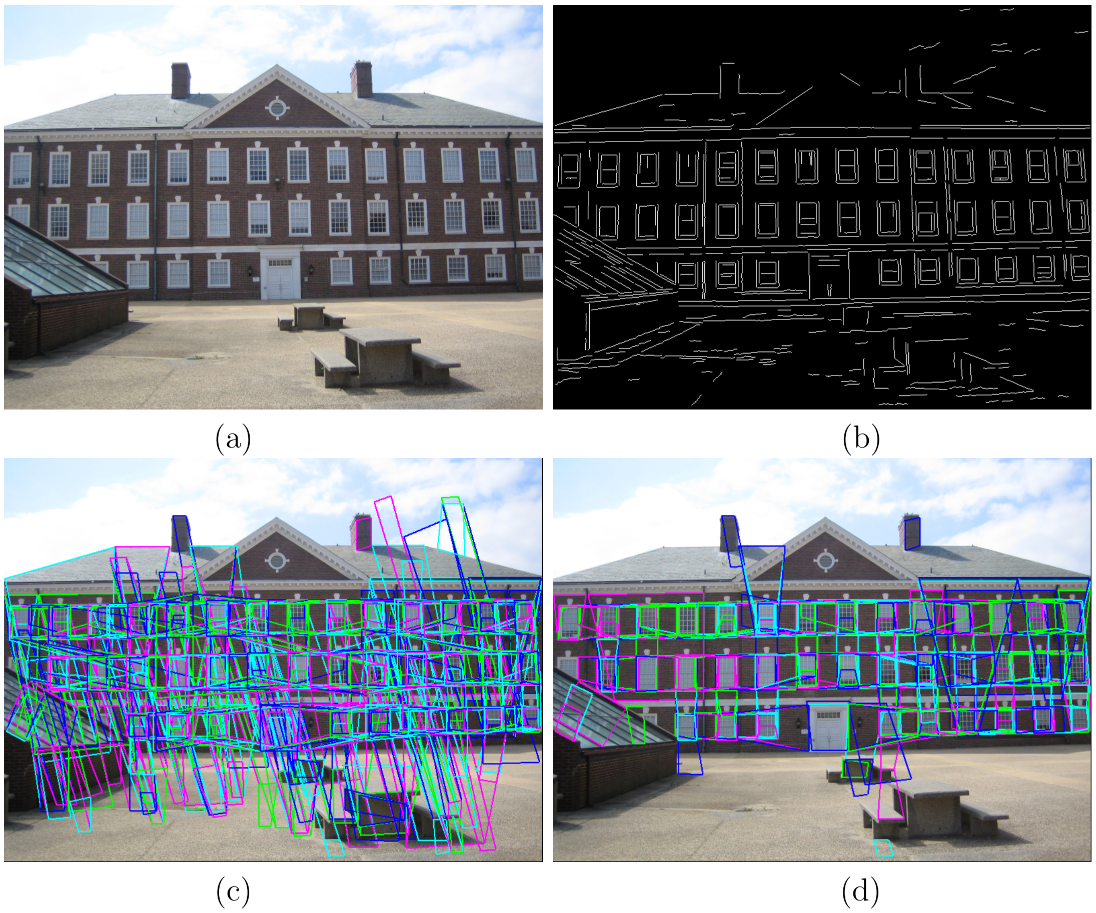

(a) (b) (c) (d)

**Figure 5.5:** Line detection and rectangle hypotheses. (a) Input image; (b) Straight lines obtained by grouping Canny edge pixels; (c) 1291 hypothesized rectangles from all pairs of approximately parallel lines; (d) 700 rectangles after pruning based on mean gradient strength at the borders.

Some amount of pruning can be done by sorting the rectangles based on mean gradient strength along its boundaries and removing ones that are not well aligned with the edges. We conservatively keep the top 700 such rectangles (the average number of windows in our images is 15-20). By overestimating this number, the grouping algorithm is allowed to recover the best possible lattice without using hard thresholds early in the pipeline. Figures 5.5(b) and 5.5(c) show fitted straight lines and hypothesized rectangles for an input image. These rectangles are then passed to our lattice discovery algorithm described below.

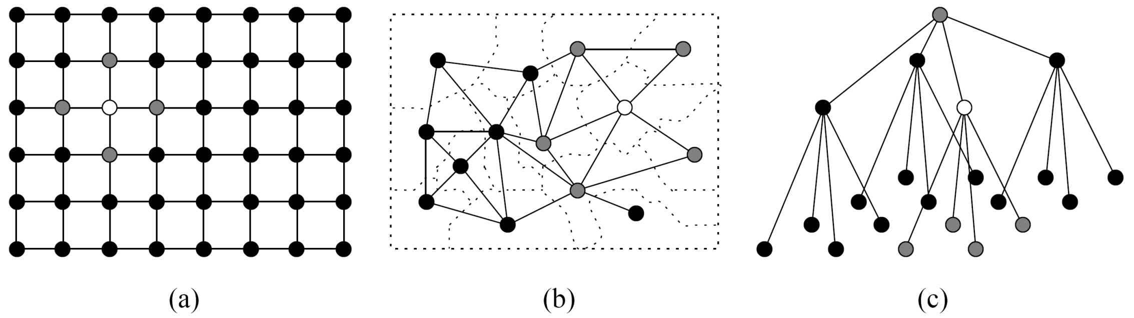
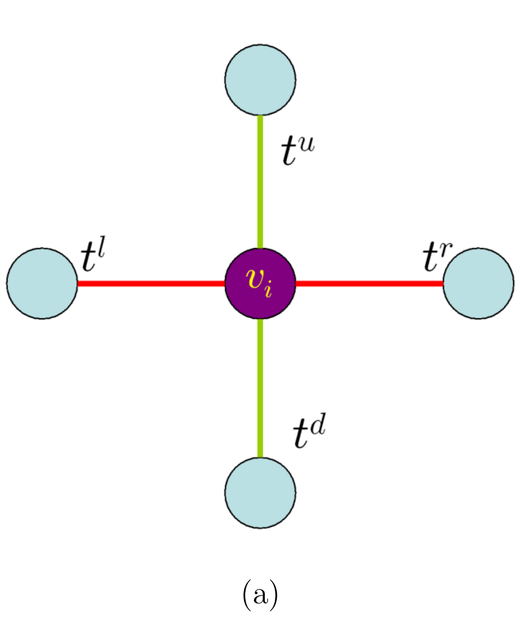


**Figure 5.6:** Three typical graphs supporting MRF-based models for image analysis (from Perez [122]): (a) rectangular lattice with first-order neighborhood system; (b) non-regular planar graph associated to an image partition; (c) quad-tree. For each graph, the grey nodes are the neighbors of the white one.

### 5.3.2 MRF Grid Extraction

A Markov Random Field (MRF) is an undirected graph $(V, E)$, where $V$ and $E$ denote the set of vertexes and edges in the graph, respectively. Each vertex in the graph corresponds to a random variable. The joint probability of all random variables is factored as a product of local potential functions at each node, and the interactions between nodes are defined on neighborhood cliques represented by the connected edges in the graph. The most common form of MRF is a pairwise MRF in which each clique is a pair of connected nodes in the undirected graph. MRFs have been extensively used in pixel labeling problems like segmentation, matting, stereo, super-resolution and so on. Figure 5.6 shows typical MRF neighborhood configurations used in image analysis [122].

Given the set of tokens $v_i \in V$, we construct a pairwise MRF $G = (V, E)$. Each token is a random variable that constitutes a node of the undirected graph $G$, with edges $e_{ij} \in E$ representing the dependency between $v_i$ and $v_j$. Statistically, the probability of the states of a texton in an NRT is only locally dependent – the position and appearance of a texton is influenced more by its neighbors than distant textons. The MRF model can naturally embed the global lattice structure while



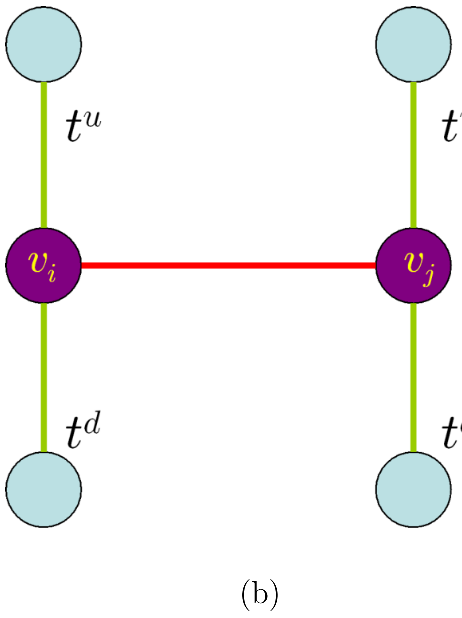


**Figure 5.7:** MRF (a) local node and (b) clique potential "neighbor" vectors

preserving this Markov property. Lin and Liu [101] exploit it for tracking dynamic NRTs even under significant self-occlusion and deformation. Their objective is to preserve an initial lattice topology over time as the NRT moves and deforms. In contrast, our goal is to build up the initial grid by grouping together image tokens that best fit the grid model.

The solution involves gradually evolving the lattice configuration by iteratively adding and removing edges. At the end of the process, links are created along vectors $\mathbf{t}_i^o : o \in \{r, l, u, d\}$ to the most likely right, left, up and down neighbors of $v_i$ (Fig. 5.7) without violating grid constraints. Similarly, $v_i^o : o \in \{r, l, u, d, NULL\}$ denote its neighbors, if any, in each direction. For object segmentation, Barbu et al. [11] and Wang et al. [167] describe a Bayesian approach to grouping adjacent "superpixels" [133] using prior models of appearance and shape. In addition to these local priors, we also need to define a global topology prior while constructing the lattice. Since connected nodes do not have to be adjacent in the image (according

to our relaxed constraints), each node can also potentially be linked to several other nodes, increasing the combinatorics of the problem.
Given image $I$, we wish to obtain the MAP estimate for the graph configuration

$$p(G|I,T,S) \propto p(I|T,S,G)\ p(S|G)p(T|G)\ p(G). \tag{5.1}$$

The image likelihood $p(I|T,S,G)$ is encapsulated in the rectangle detection and is neglected here. Color histograms, proximity of rectangle boundaries to image edges, or learned appearance models are all possible likelihood models. The shape prior $p(S|G)$ can be used to favor known shape models, though here we set it to unity since we are only dealing with rectangles. The graph prior $p(G)$ models any global intuition about the nature of the grid or the degree of connectedness. We set this also to unity for the building images.

The topology prior $P(T|G)$ is represented as a pairwise MRF whose joint can be factored into a product of local node potentials $\Phi$ and clique potentials $\Psi$:

$$P(T|G) \propto \prod_i \Phi(v_i) \prod_{i,j\in E} \Psi(v_i, v_j).$$

To model a grid, we measure the symmetry of direction vectors from a node to its neighbors. Let $\delta(\mathbf{t}_1, \mathbf{t}_2) = e^{-\beta||\mathbf{t}_1-\mathbf{t}_2||}$ be a similarity measure between two neighbor vectors assuming both edges are in $G$. The potentials are now defined as:

$$\begin{aligned} \Phi(v_i) &= e^{-\gamma(4-n_i)} * \delta(\mathbf{t}_i^r, -\mathbf{t}_i^l) * \delta(\mathbf{t}_i^u, -\mathbf{t}_i^d), & (5.2) \\ \Psi(v_i, v_j) &= \delta(\mathbf{t}_i^u, \mathbf{t}_j^u) * \delta(\mathbf{t}_i^d, \mathbf{t}_j^d) * \mathcal{B}(v_i, v_j). & (5.3) \end{aligned}$$

where $n_i$ denotes the degree of node $v_i$. Thus we encourage left/right and up/down edge pairs to be 180 degrees apart with similar magnitudes. The interaction potential between horizontal neighbors forces their vertical edges to be approximately parallel. For missing edges, a small fixed value of 0.2 is assigned to $\delta$. These simple functions effectively model the lattice configuration as will be shown on the synthetic NRT images.

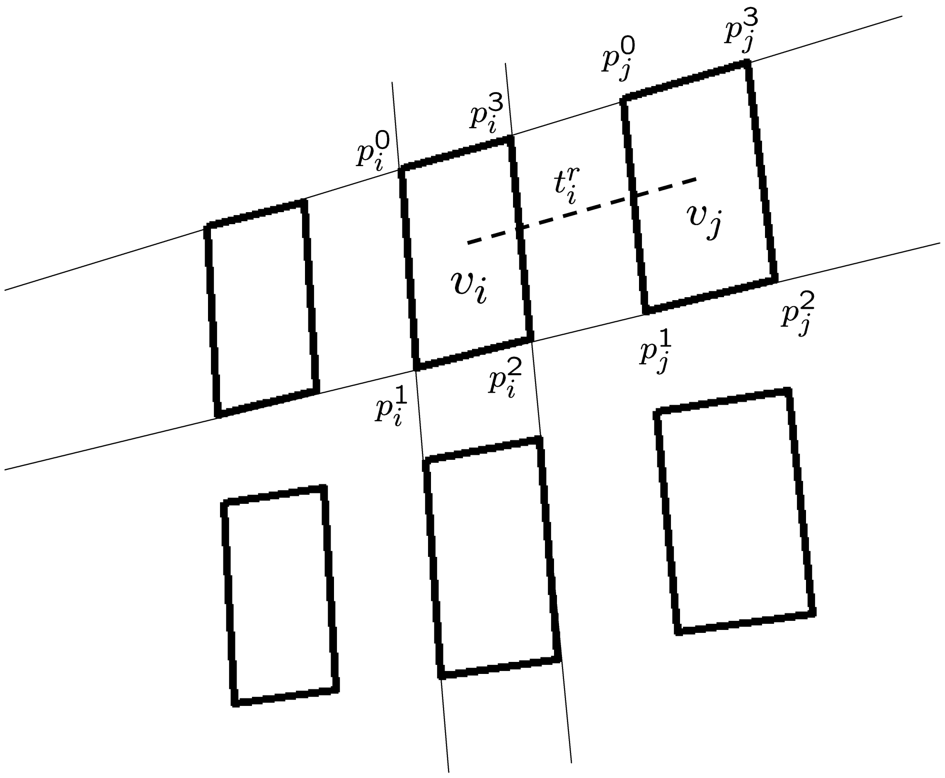


**Figure 5.8:** Illustration of windows under perspective and notation.

The function $\mathcal{B}(v_i, v_j)$ is used to specify any texture-specific pair-wise relationships between the texels. For building images and windows under perspective, we incorporate constraints such as overlap, cross ratio, and appearance similarity. Let $g(x, \sigma) = \exp(\frac{-x^2}{2*\sigma^2})$ be a Gaussian weighting function with standard deviation $\sigma$. Using Fig. 5.8 to illustrate, we list various heuristics that reflect the probability of $v_i$ and $v_j$ being connected by a horizontal edge. The case of vertical neighbors is analogous.

- Windows on the same level have their bottom and top edges aligned with each other, implying that points $(p_i^0, p_i^3, p_j^0, p_j^3)$ and $(p_i^1, p_i^2, p_j^1, p_j^2)$ are collinear. We measure the extent of deviation from this constraint with a commonly used test for collinearity of 3 points. Let $c_i = (x_i, y_i) : i = \{0, 1, 2\}$ be three points. If they are collinear, the area $A(c_0, c_1, c_2)$ of the resulting triangle should be close to 0. This is computed as

$$A(c_0, c_1, c_2) = \begin{vmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix}.$$

We then define a likelihood measure penalizing deviations from collinearity as

$$\mathcal{B}_{col} = g(COL, 4),$$

where

$$COL = A(p_i^0, p_i^3, p_j^0) + A(p_i^0, p_i^3, p_j^3) + A(p_i^1, p_i^2, p_j^1) + A(p_i^1, p_i^2, p_j^2).$$

- One projective invariant is the cross ratio. Given 4 points $x_i$, the *cross ratio* is defined as

$$Cross(x_0, x_1, x_2, x_3) = \frac{|x_0, x_1||x_2, x_3|}{|x_0, x_2||x_1, x_4|}$$

where

$$|x_i, x_j| = \begin{vmatrix} x_{i1} & x_{j1} \\ x_{i2} & x_{j2} \end{vmatrix}.$$

Assuming parallel window sides and negligible noise, the 4 upper and lower points in Fig. 5.8 should have approximately the same cross ratio. We define

$$CR = \left| 1.0 - \frac{Cross(p_i^0, p_i^3, p_j^0, p_j^3)}{Cross(p_i^1, p_i^2, p_j^1, p_j^2)} \right|$$

to quantify this measure, and set $\mathcal{B}_{CR} = g(CR, 0.1)$. For horizontal neighbors, this essentially measures how parallel the vertical edges of the windows are.

- The horizontal dimensions of windows under perspective should also vary smoothly. To penalize deviations in magnitude and orientation among the upper and lower sides, we define

$$\mathcal{B}_{DH} = \min(\delta(v_i^{top}, v_j^{top}), \delta(v_i^{bot}, v_j^{bot})).$$

The term $v_i^{top}$ denotes the line segment $p_i^0 p_i^3$. Similarly, $v_i^{bot} = p_i^1 p_i^2$, $v_j^{top} = p_j^0 p_j^3$, and $v_j^{bot} = p_j^1 p_j^2$.

- Windows and texels in general should not overlap. To test if two connected polygons overlap, we can determine the intersections of its edges with the other. Thus

$$\mathcal{B}_{ovl}(v_i, v_j) = \begin{cases} 10^{-20} & \text{if } v_i \text{ and } v_j \text{ overlap;} \\ 1 & \text{otherwise.} \end{cases}$$

- The corners of each polygon are correlated with each other to ensure appearance similarity. Correlating the entire window would be sensitive to occlusions and pose variations, while the corners are typically more distinctive. Formally,

$$XC(v_i, v_j) = \sum_{k=0}^{3} \frac{NCorr(Patch(p_i^k), Patch(p_j^k))}{4}$$

  where $XC$ is the mean normalized cross correlation $NCorr$ of $11 \times 11$ patches centered at each of the 4 window corners. This is converted into a likelihood $\mathcal{B}_{XC}(v_i, v_j) = g(1.0 - XC(v_i, v_j), 0.4)$.

These heuristics for perspective windows are combined together as

$$\mathcal{B}(v_i, v_j) = \mathcal{B}_{col}(v_i, v_j) \times \mathcal{B}_{CR}(v_i, v_j) \times \mathcal{B}_{DH}(v_i, v_j) \times \mathcal{B}_{ovl}(v_i, v_j) \times \mathcal{B}_{XC}(v_i, v_j)$$

and used in conjunction with the generic lattice potentials defined in (5.2) and (5.3).

### MCMC Optimization

We use a Markov Chain Monte Carlo (MCMC) framework to iteratively maximize the posterior defined in (5.1), probabilistically adding and removing edges from the graph in a fashion similar to the multi-target tracking method of [79]. A Markov chain is defined over the space of configurations $\{G\}$, such that the stationary distribution of the chain approximates the posterior. The target configuration is one where all NRT nodes are connected in a lattice structure, while the other elements remain isolated. We use the Metropolis-Hastings [65] algorithm to generate the chain or sequence of samples. Starting from an initial configuration $G_0$, we repeat for $t = 1..N$:

1. Given current state $G_t$, generate a new proposed state $G'_t$ from the *proposal density* $\mathcal{Q}(G'_t|G_t)$.

2. Compute the *acceptance ratio*

$$u = \frac{p(G'_t|I,T,S)\mathcal{Q}(G_t|G'_t)}{p(G_t|I,T,S)\mathcal{Q}(G'_t|G_t)}).$$

3. Accept this new state with probability $p = \min(1.0, u)$ setting $G_{t+1} = G'_t$, and reject otherwise. In the latter case, set new state $G_{t+1} = G_t$.

The first $b$ iterations constitute the burn-in period and are discarded when selecting the best MAP estimate.

Proposal updates $\mathcal{Q}(G'_t|G_t)$ consist of edge additions or removals applied to a node $v_k$. Modifying edges one component at a time leads to better success rate for transitions, although large state changes require more jumps. The transitions are made only in the up and right directions in order to keep the reverse transition probability simple. To maintain consistency, the edges in the reverse directions are also modified appropriately. Two functions, picked probabilistically, govern how $v_k$ is selected in each MCMC iteration: (i) an unguided scheme $\mathcal{Q}_1$ in which $v_k$ is chosen uniformly from all nodes, and (ii) a guided hypothesis generation $\mathcal{Q}_2$ in which the node and a corresponding neighbor is selected from a dynamic pool $\mathcal{P}_q$ of potentially good edges. As the grid converges to the correct solution, $\mathcal{Q}_2$ facilitates lattice growth and completion by hypothesizing edges close to the good parts of the lattice.

Let $e^o_{kl} : l \in \{1, \ldots, n_k\}$ be potential edges from $v_k$ to its neighbors in direction $o$. In $\mathcal{Q}_1$ , $o$ is uniformly chosen from the up and right directions. The edge $\mathbf{t}^o_k$ from

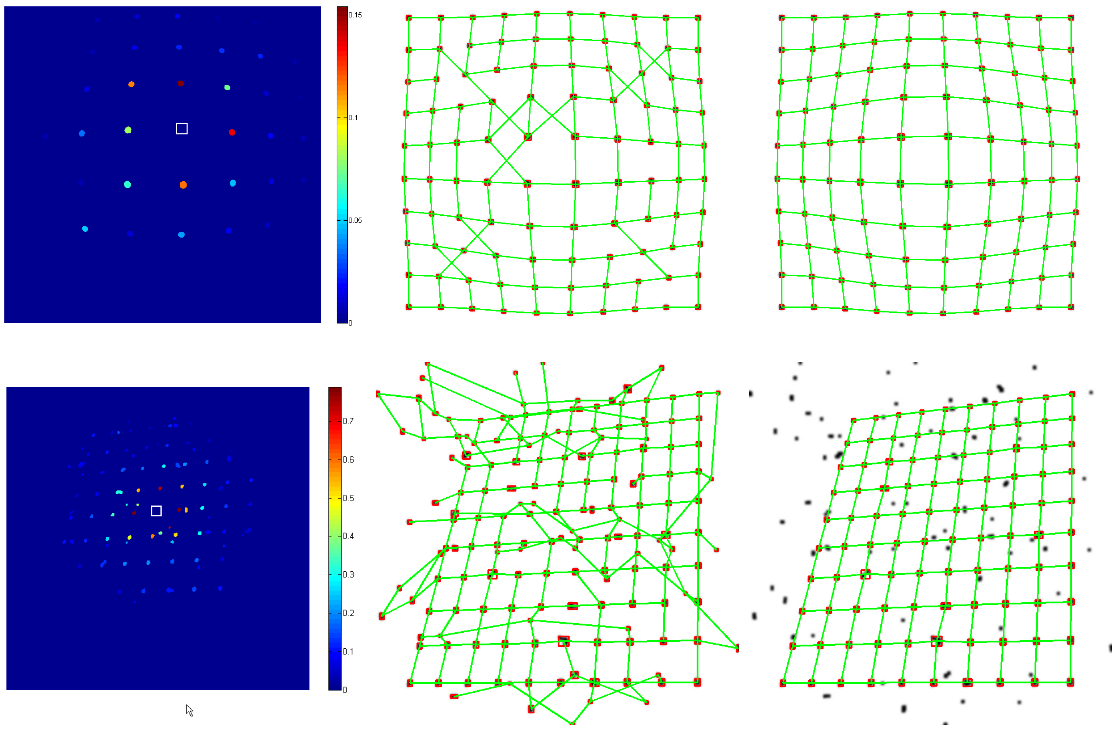

(a) (b) (c)

**Figure 5.9:** Example of our algorithm on two synthetic NRT images from [91]. (a) Column shows a color map of $E_{score}$ for a selected node (white rectangle). This is the sampling distribution for making proposals from this node. (b) Initial lattice $G_0$ obtained by connecting each node to the neighbor with highest $E_{score}$. (c) The best MAP estimate after running 10000 MCMC iterations; lattice nodes are grouped into a single connected component of the graph.

node $v_k$ is turned off with fixed probability $p_{off}$ or assigned a neighbor by sampling from

$$E_{score}(k,\cdot) = \frac{votes(k,\cdot)}{\sqrt{dist(k,\cdot)+\epsilon}} * \mathcal{B}(v_k,\cdot). \tag{5.4}$$

Here $dist(\cdot,\cdot)$ is the point-to-point distance between two nodes, and $votes(\cdot,\cdot)$ is measured as the total number of other nodes within a threshold distance of the line parameterized by the two nodes. Fig. 5.9 illustrates this function for two synthetic images from the NRT database. The first column shows the color coded mapping of

(5.4) for a specific node (denoted by the white rectangle) to all its neighbors. This is the sampling distribution used by $\mathcal{Q}_1$ to hypothesize neighbors in a given MCMC step. Neighbors that seem to conform to the topological and visual constraints are picked more often, leading to faster convergence.

Random selection of a candidate node is effective given enough iterations, but can be inefficient in steering the optimization towards completing the lattice; missing or incorrect edges adjacent to the good regions should receive higher priority for growth. When a node $v_k$ is visited in $\mathcal{Q}_1$ , its unbalanced edges are first identified. Let $m$ be the missing edge direction and $OPP(m)$ be its reverse. Since node $v_k$ is connected to a lattice through $\mathbf{t}_k^{OPP(m)}$, its opposite vector could be a useful cue to propose a new neighbor $v_k^m$ in direction $m$. The algorithm iterates through all neighbors $l$ in $e_{kl}^m$, weighting it by its deviation from $\mathbf{t}_k^{OPP(m)}$ and shape similarity. A new edge $e_{kl'}^m$ is sampled by picking $l'$ according to distribution

$$W_{kl} = \delta(-\mathbf{t}_k^{OPP(m)}, v_k - v_l) * \mathcal{B}(v_k, v_l).$$

The sampled edge $e_{kl'}^m$ is added to priority queue $\mathcal{P}_q$ with ranking function $W_{kl'}$. As the number of iterations increase, this pool of potentially good edges also grows. A new proposal in $\mathcal{Q}_2$ simply involves removing the highest priority edge from $\mathcal{P}_q$ and adding it as an edge in state $G_t'$. Our experiments have also suggested that interleaving proposals $\mathcal{Q}_1$ and $\mathcal{Q}_2$ in this manner is superior to running either one alone. The reverse transition probability is simple in both cases, as the sampling distributions are constant and can be computed beforehand. The chain is *irreducible* because a series of edge additions and deletions can take the graph from one state to any other state. The stochastic elements also guarantee *aperiodicity* by not getting trapped in cycles. Together, they satisfy the conditions of ergodicity to ensure that the chain will converge to the stationary distribution.

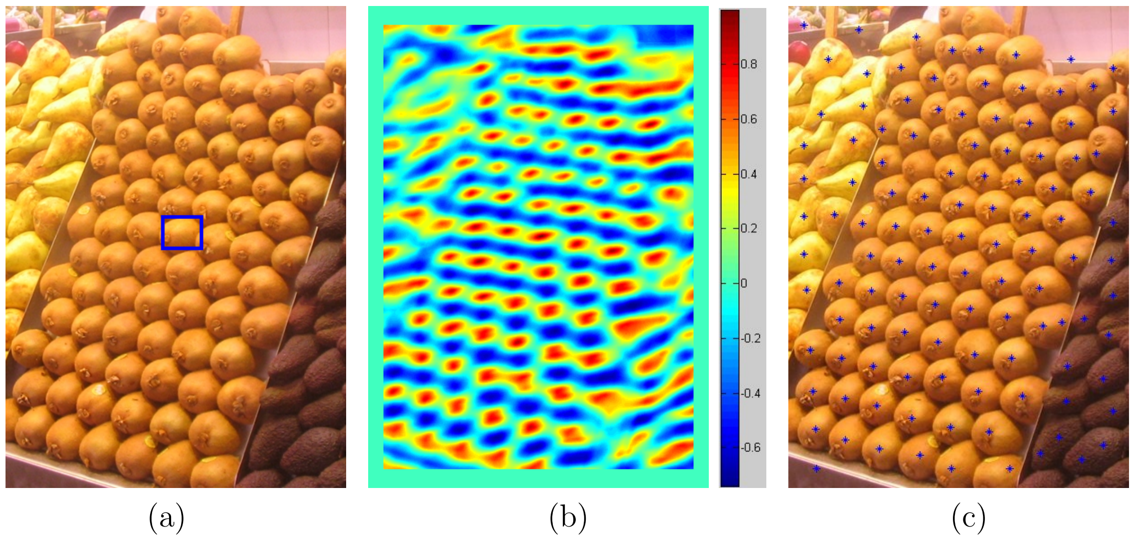


**Figure 5.10:** NRT Discovery Tokenization: (a) User selected patch in blue; (b) Height map after normalized cross correlation of patch; (c) Resulting peaks after regions-of-dominance computation using Matlab code of [66].

### Lattice Initialization

To speed up convergence, the graph $G_0$ is initialized with a maximum likelihood (ML) estimate where each node $v_k$ connects to its most likely 4 neighbors according to $E_{score}(k, \cdot)$ (5.4). Under negligible noise and mostly rigid deformations, this may recover the full lattice structure. In other cases with smooth but non-rigid deformations, the heuristic is only locally valid. Other distractor elements not part of the lattice can also cause inconsistencies in the lattice structure. Nevertheless, it provides a useful starting point for the MCMC simulation. The second column of Fig. 5.9 shows the initialized ML lattice and the third column shows the best lattice after the MCMC optimization.

## 5.4 Discovering pure NRTs

Since this work is primarily about buildings and specifically for window grids, all the above discussion has focused on identifying rectangular NRTs. However, unlike other techniques developed for urban scenes [110, 115], we demonstrate that our grouping algorithm is general enough to work on pure Near-Regular Textures such as those shown in Fig. 5.1 from the NRT database [91]. The key to enabling this is building a new tokenizer that can extract out like elements. Hays [66] uses an iterative procedure where the grouping is first performed on MSER features, failing which random patches are correlated in the image to build a height map and extract regions of dominance [104]. Since their Matlab code was available to us, we used their identical functions to detect correlation peaks for a user-selected patch from the texture image. This interaction only involves selecting a bounding box and eliminates some of the spurious lattices discovered by using randomly selected patches. Figure 5.10 shows the steps of this image discretization procedure. For images where MSER gave a reasonably good initialization, we used those point features only.

MCMC grouping is then performed by considering the dominant peaks (or MSER features) as tokens, and consequently nodes of the graph. Similar to the process of inferring the lattice topology for the synthetic images (see Results section), the goal is to connect these peaks into a 4-neighborhood lattice. Currently, the grouping is done purely by topology as the texture-specific $\mathcal{B}(v_i, v_j)$ function for dots is set to unity. However, even without a well-defined function that considers appearance similarity between neighbors when hypothesizing edges, we show encouraging results for lattice discovery.

## 5.5 Results

This section presents results of the grouping algorithm on both synthetic and real images from the NRT database as well as building images from our test set. 10000 MCMC iterations were used with the first $b = 1000$ iterations being the burn-in period. The best MAP estimate is chosen from the remaining samples. A breadth-first traversal is used to separate out the various connected components of the graph configuration. A component with more than 4 elements is considered a lattice. A user can then iterate through the handful of such lattices to pick the best one, or the selection can be done automatically. In the latter case, heuristics such as alignment with edges or any *a priori* knowledge may be used. For building images, we use edge alignment to rank each connected component.

Despite using very simple potentials, we are able to model many rigid and non-rigid grid configurations—e.g., the test set dots images in the PSU NRT database [91]. While this work is primarily focused on rectangles and structured scenes, we demonstrate that the grouping framework is general enough to handle various types of NRTs with the appropriate bottom-up techniques. The algorithm of [66] repeats over 5 times with random initializations, each run consisting of 16 iterations of lattice refinement and unwarping. The final best lattice is chosen according to some heuristics. We argue that MCMC provides a more natural means of embedding such an iterative framework, without committing to a single best lattice. Instead, we obtain a probability density over the space of all topologies. Our local pairwise MRF potentials are also more space efficient when compared to the $N^4$ complexity of [66], where $N$ is the potential number of texels.

Arguably, more robust than taking the MAP estimate would be to determine the mode of the sample distribution. This would require maintaining a histogram counter for every possible state. With hundreds of elements, each potentially connecting to tens of other elements, the state space is huge. A histogram in this space that needs to be updated every iteration can therefore be very inefficient. The slowest part of our algorithm is in computing $E_{scores}$ and $\mathcal{B}(v_i, v_j)$ before running the MCMC simulation. The current unoptimized implementation is an $O(n^2)$ operation and takes 20.8 seconds for 440 nodes (rectangles) on a 1.7 GHz Pentium M laptop. Once precomputed, running 10000 MCMC iterations takes only a couple of seconds on the same platform.

### 5.5.1 Building Images

Figures 5.11 to 5.18 demonstrate the results of running our algorithm on building images collected from the campus as well as other urban environments. The images are characterized by occlusions, shading effects, reflections, and variation within windows; purely appearance-based systems can be sensitive to these effects. A fixed value of $\beta$ was used for all experiments. Both windows as well as its topological structure in the form of a neighborhood graph has been captured.

Firstly, figures 5.11, 5.12, and 5.13 compare the result of our MCMC-based grid finder with the algorithm of Hays [66]. One of the main disadvantages of [66] is efficiency. Generating the results for each image (at 800x600 resolution) took approximately 10 minutes. In contrast, our method takes less than a minute in total. For example, the timings for fig. 5.11 are:

- Rectangle hypotheses - 20.2 sec (in Matlab)
- Initial graph construction - 15.9 sec (C++)
- MCMC Grouping with 10000 iterations - 1.9 sec (C++).

Rectangle hypotheses can be speeded up by porting to C++ while the main bottleneck in the graph construction is due to exhaustive searching among nodes for nearest neighbors. Approximate matching techniques such as [78] should be able to speed this process significantly. The method of [66] also suffers from the lack of centering on windows or other semantically meaningful aspects of the image. Finally, after repeated iterations, the lattice that it eventually picks is not perceptually the best. For the comparisons, the images were handpicked based on what we thought was the best. The algorithm's best results are shown in 5.14 and seems to favor more tiles and hence more of them.

By only enforcing local smoothness in appearance and geometry, the grouping is robust to changes in window dimensions (such as those on the first floor). Along with the heuristics that constitute $\mathcal{B}$, it effectively handles perspective images shown here. Another advantage is that the algorithm is not dependent on the quality of low-level operations like image rectification or vanishing point detection as a preprocessing step.

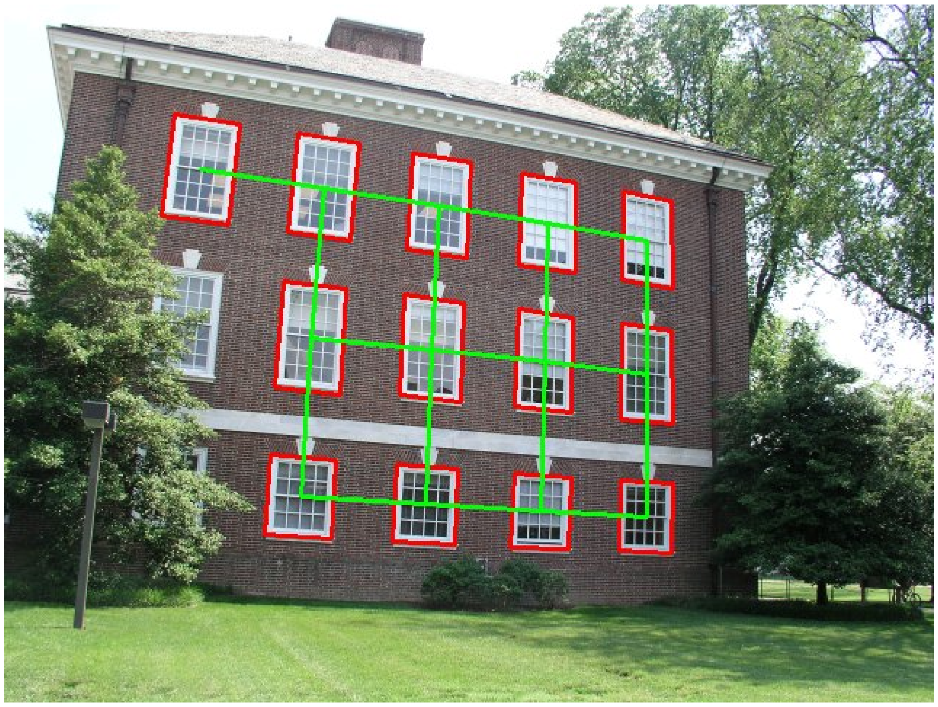
### 5.5.2 NRT Database Images

Figures 5.20 to 5.27 show the inferred lattice configuration for several images from the NRT database [91]. The two figures in 5.20 were tokenized using MSER features as they seem to give a good initialization (shown in Fig. 5.19). For the other images, a template patch was selected by hand and correlated with the rest of the image.

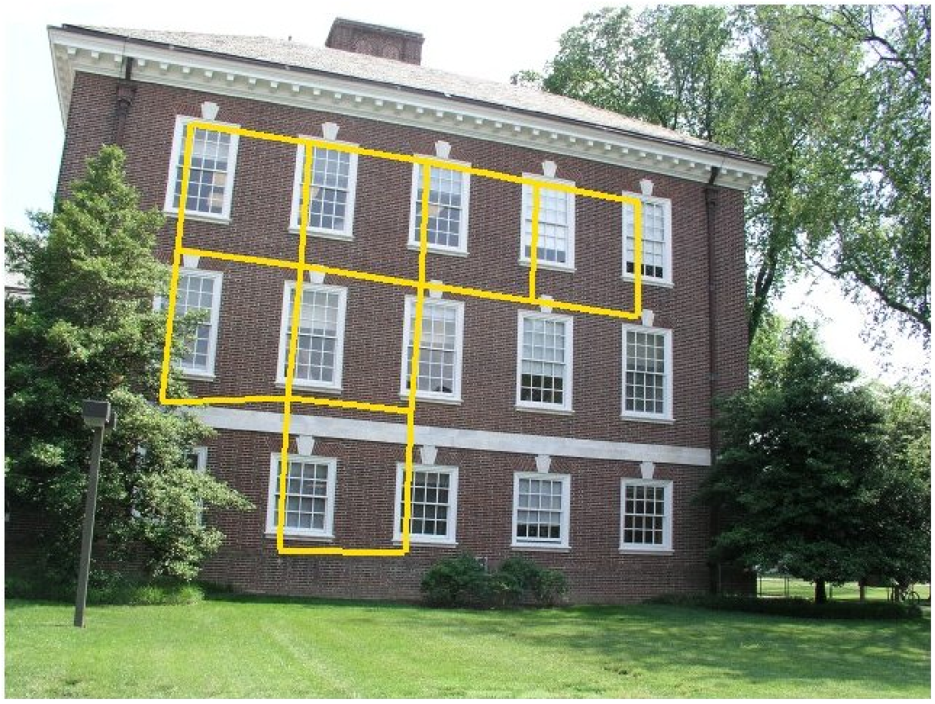
The synthetic dots images in figures 5.28 and 5.29 give perfect initialization. We simply use thresholding followed by connected components to extract out each candidate node. This allows us to decouple the grouping aspect of this work and test it under various scenarios: rigid and non-rigid deformations, distractor elements, noise, and missing nodes. The discovered lattices for various synthetic images are also shown.

The only difference between the window grid extractor and the more general NRT discovery method here is the texture-specific $\mathcal{B}$ function – something that can be plugged into the framework easily without incurring excessive overhead in programmer effort and execution time. The results demonstrate that our MRF formulation of the lattice is powerful. With tokenization using normalized cross-correlation and Regions-of-Dominance, it was able to discover the connectivity well in many of the images from [91]. Using a suitable $\mathcal{B}$ function could for example have prevented some of the boundary pears from connecting to the different colored fruits next to it, or the fabric texel in Fig. 5.22 from connecting to an outlier token.

Discretizing the image using cross-correlation seems general enough to work on the structured building facade of Fig. 5.24 as well as the more cluttered pears

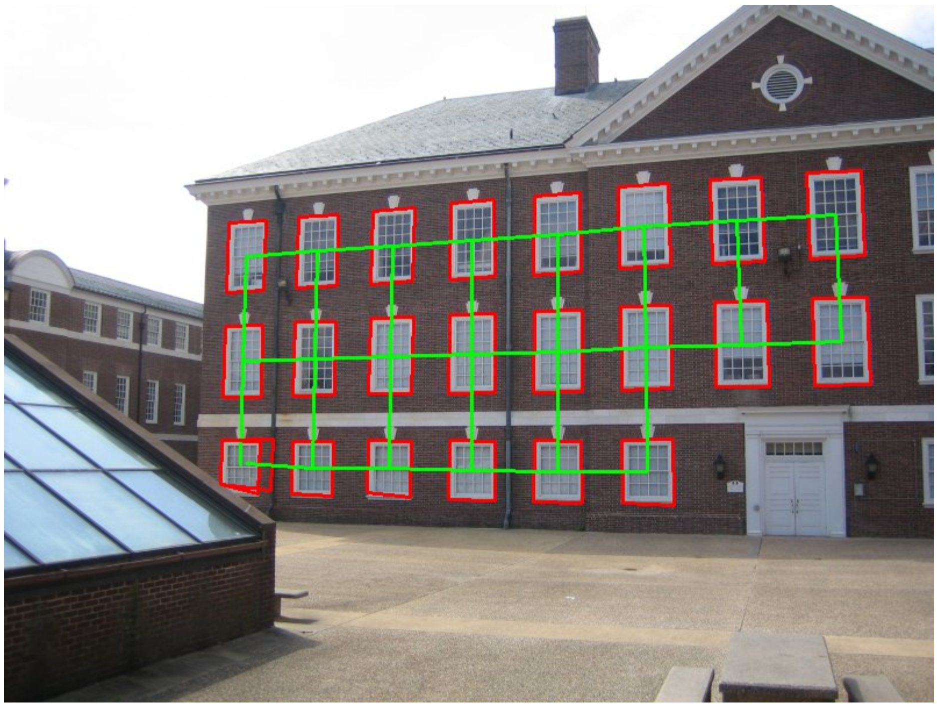


(a)

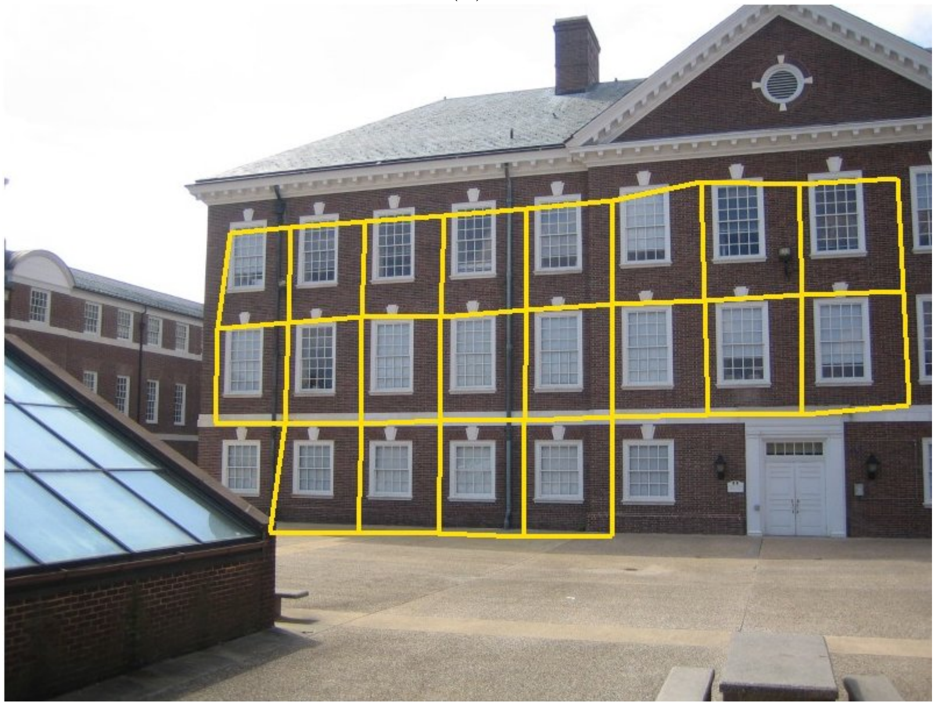


(b)

**Figure 5.11:** Comparison of (a) our window grid discovery method with (b) the lattice discovery of Hays [66]

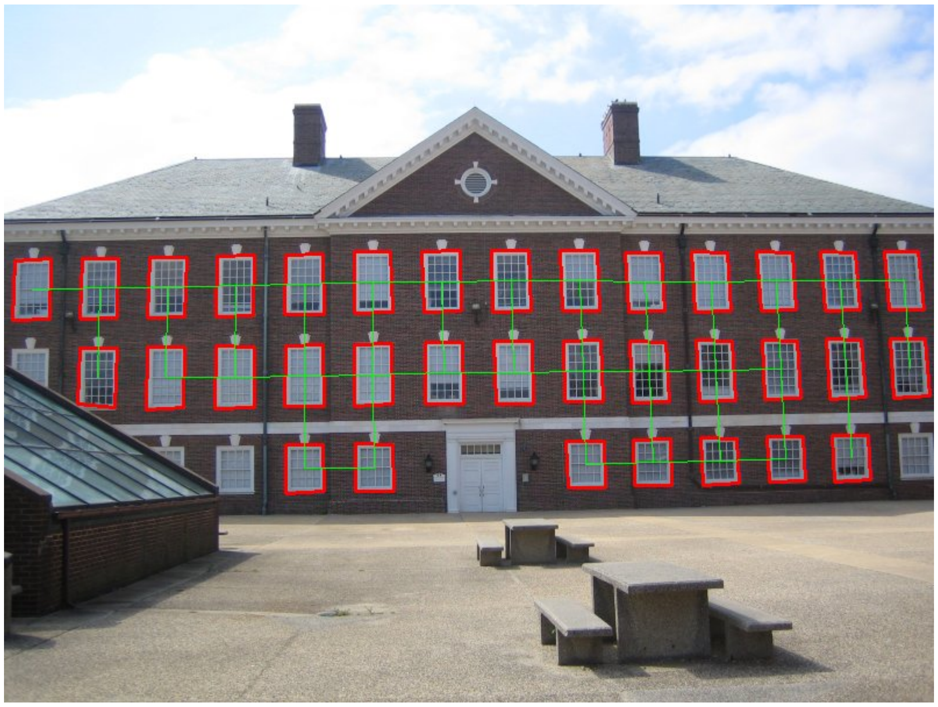


(a)

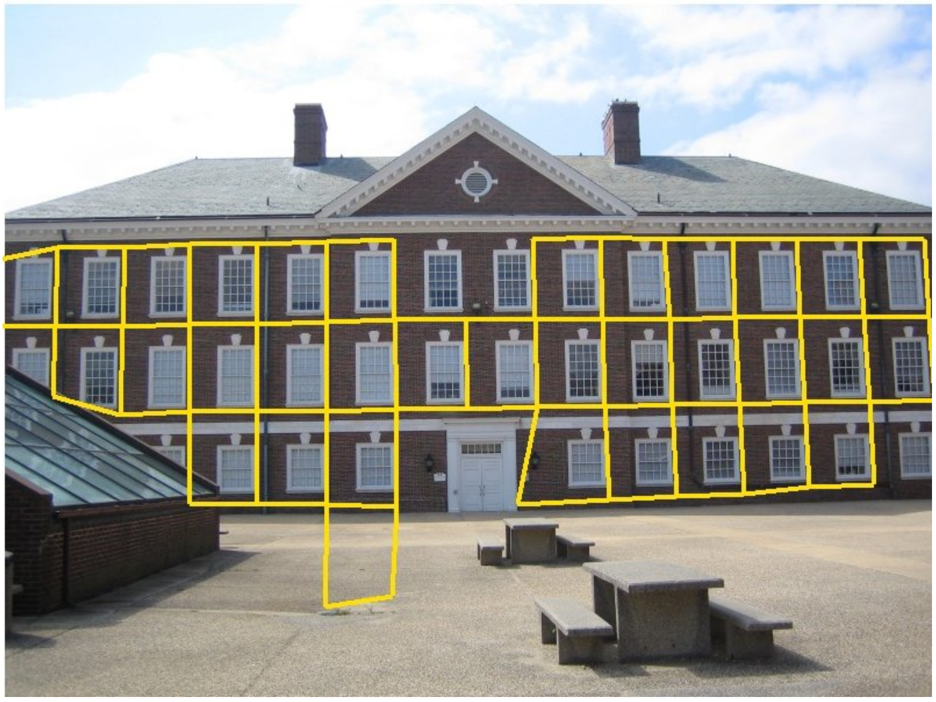


(b)

**Figure 5.12:** Comparison of (a) our window grid discovery method with (b) the lattice discovery of Hays [66]



(a)



(b)

**Figure 5.13:** Comparison of (a) our window grid discovery method with (b) the lattice discovery of Hays [66]

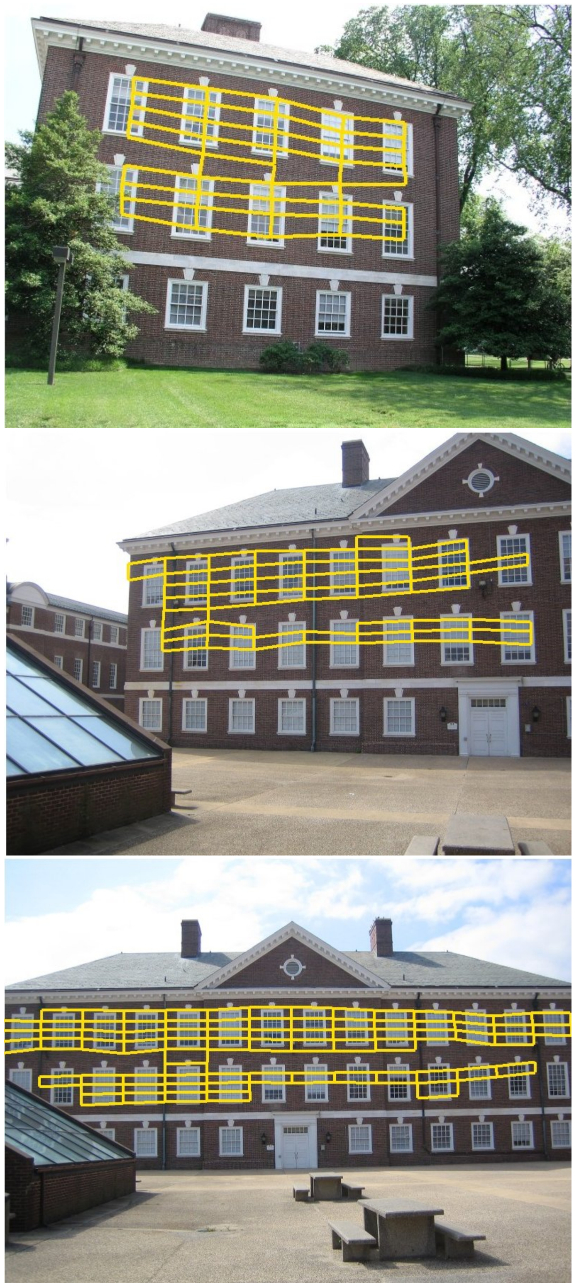


**Figure 5.14:** The best lattices picked by [66] after several iterations. The algorithm seems to favor smaller tiles and more of them in number.

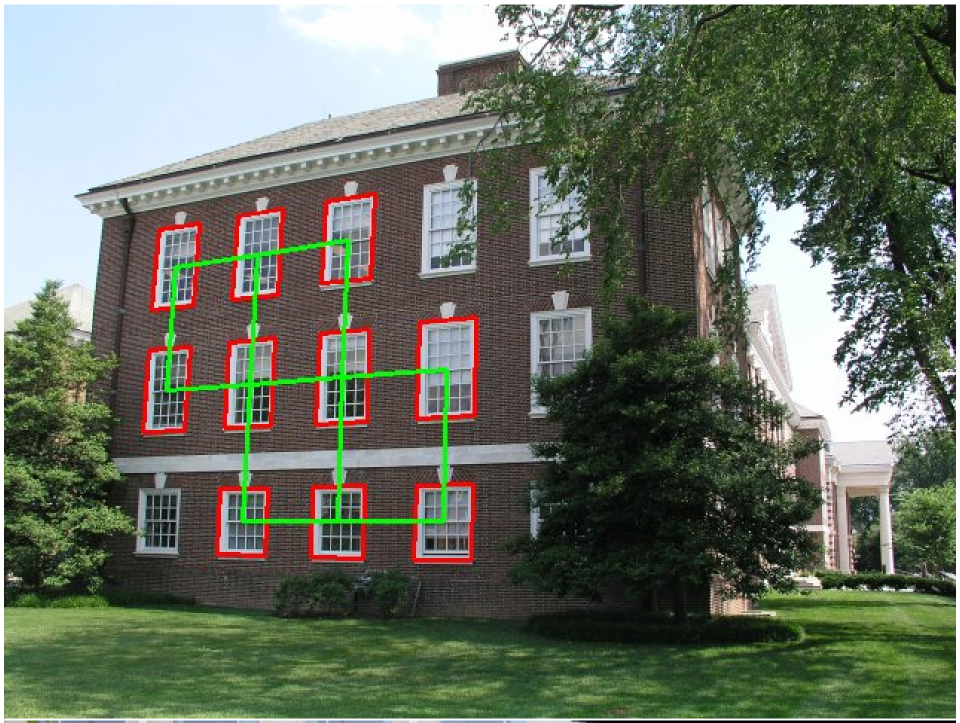
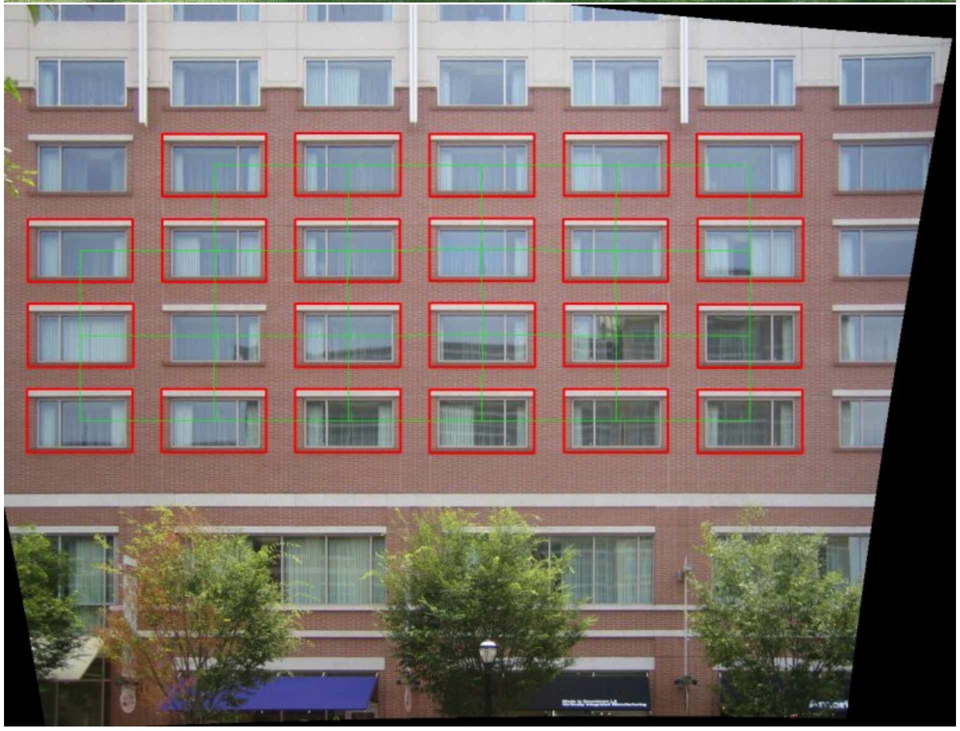



**Figure 5.15:** More results of lattice discovery on various building facades.

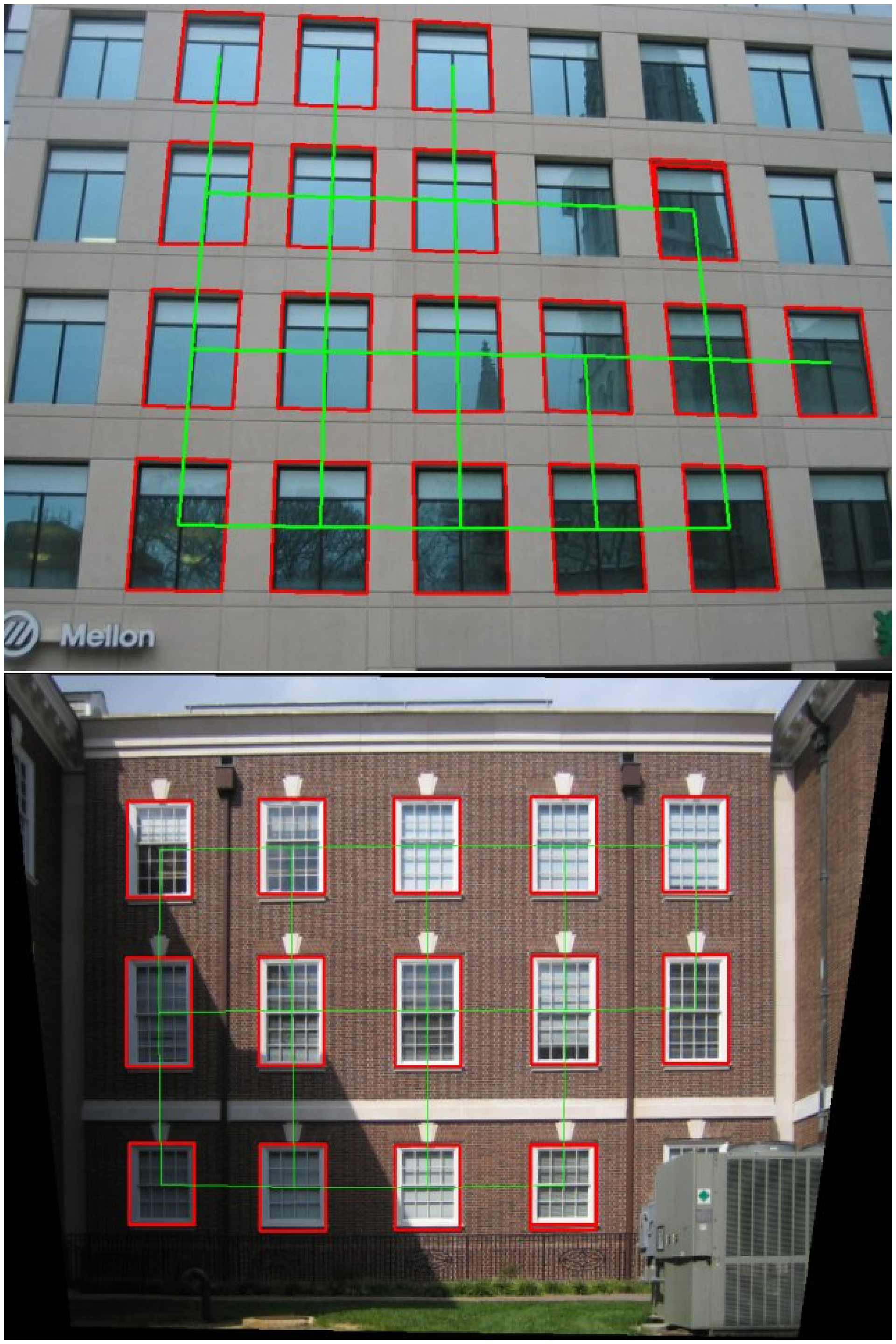


**Figure 5.16:** More results of lattice discovery on various building facades

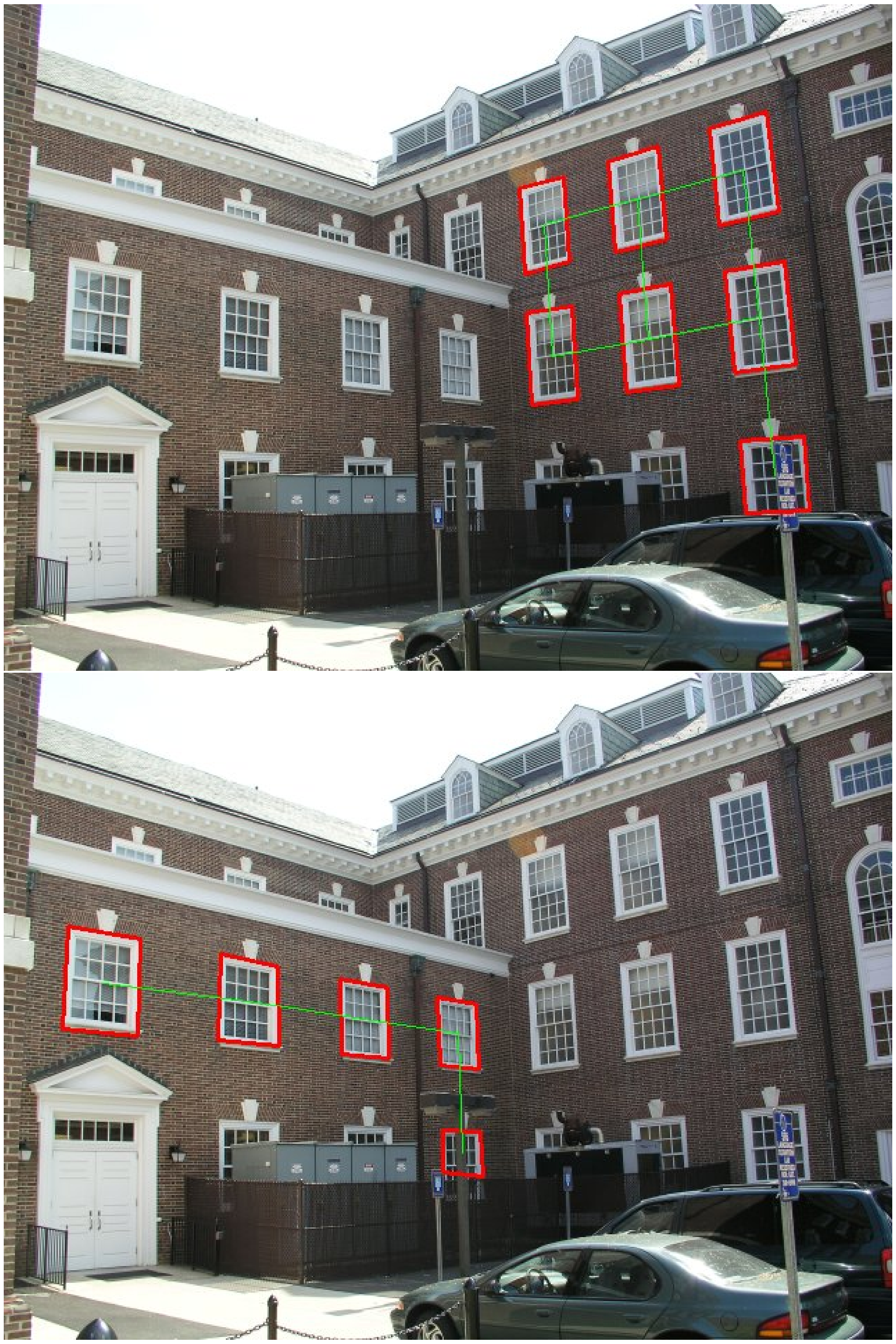


**Figure 5.17:** More results of lattice discovery on various building facades

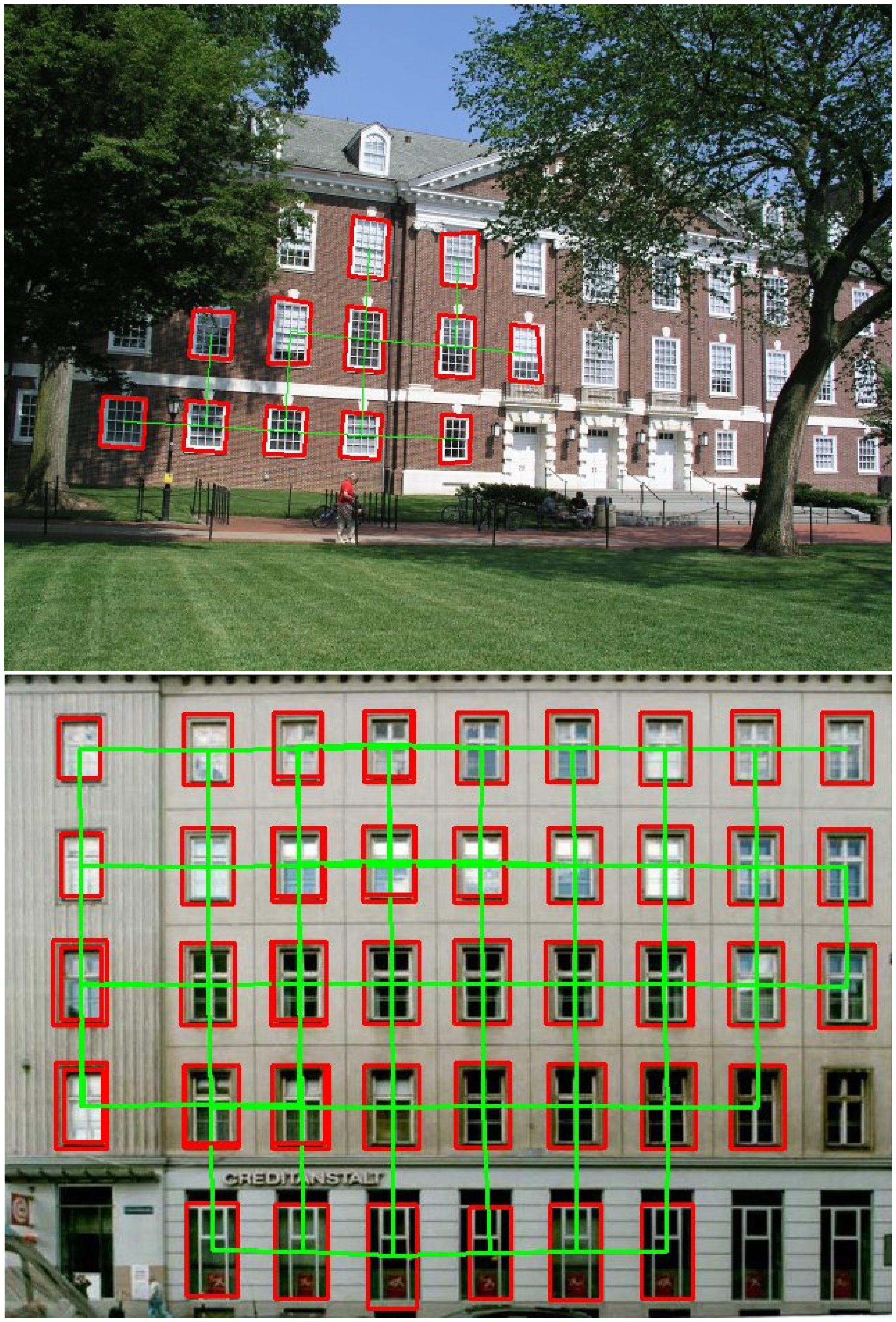


**Figure 5.18:** More results of lattice discovery on various building facades

image in Fig. 5.21. However, there are a couple of disadvantages in the way we use it here. Firstly, correlating patches can localize the texel centroid well, but not the semantic extent of it (unless the image is rectified in which case the user-selected patch can be translated to each of the centroids). When the NRT assumption of *packing* and *coverage* are violated, such as in the building images, it is difficult to reason about window boundaries or occluding elements between them. This was our main motivation for using the rectangle tokenizer for buildings.

Correlation is also not as effective on images undergoing significant distortion – either perspective or non-rigid. Thus it misses the border elements in the glass beads image of 5.26(bottom). In Fig. 5.22 also, correlation does not produce a peak of sufficient strength when the fabric is folded, thus missing out some of the interior texels. Missing tokens after initialization are indeed a drawback of our current algorithm as we mention in Future Work. One means of mitigating this is to allow both edges and new tokens to be added during an MCMC proposal. This would require us to use Reversible Jump MCMC methods [56]. Another alternative might be to iteratively run the grouping algorithm, where the current lattice is used to discover new tokens for the next iteration. This is quite feasible because the grouper itself requires very little run-time. Each of the result images took less than a minute to generate (initialization to grouping), compared to [66] which claims to take 30 minutes on average to discover the best lattice for each image in the NRT database.

## 5.6 Summary

We draw the analogy that building facades are often examples of Near-Regular Textures (NRT) [66], and discovering these textures could provide valuable insight into the rest of the facade. After defining NRTs and reviewing texture discovery methods, we elucidate a novel Markov Chain Monte Carlo (MCMC) approach to discover grid patterns from images. A Markov Random Field (MRF) that models spatial interactions between nearby texels for NRTs is defined; entities in the

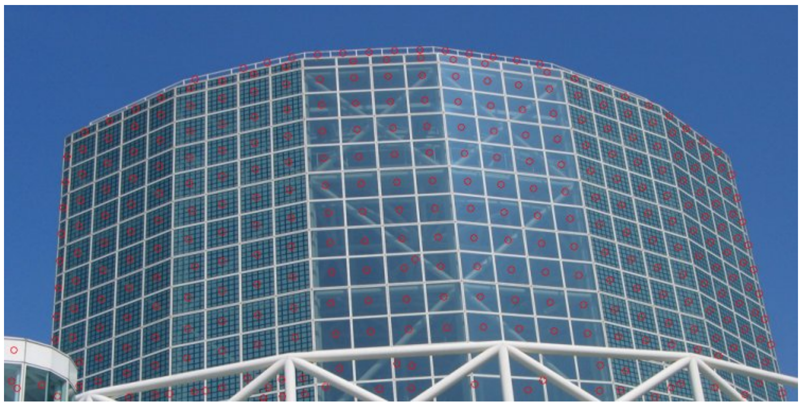
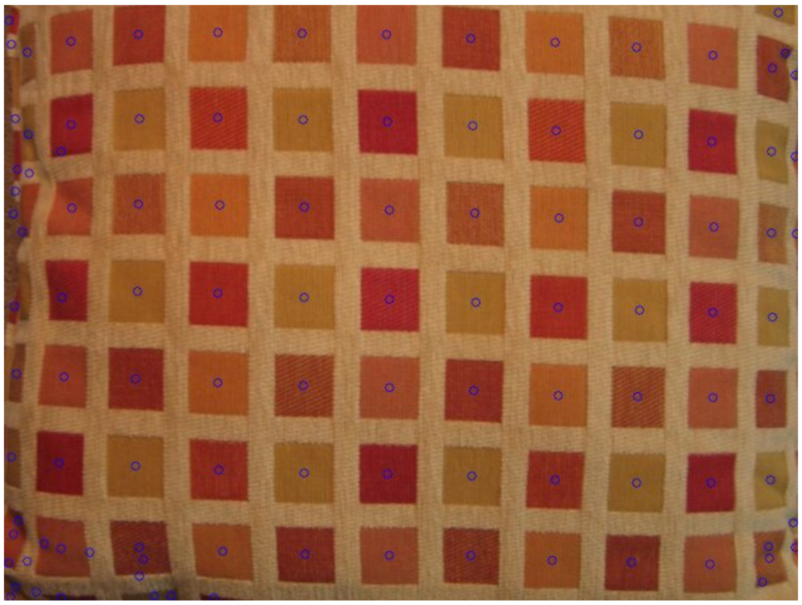



**Figure 5.19:** MSER features (shown as red or blue circles) detected for two images.

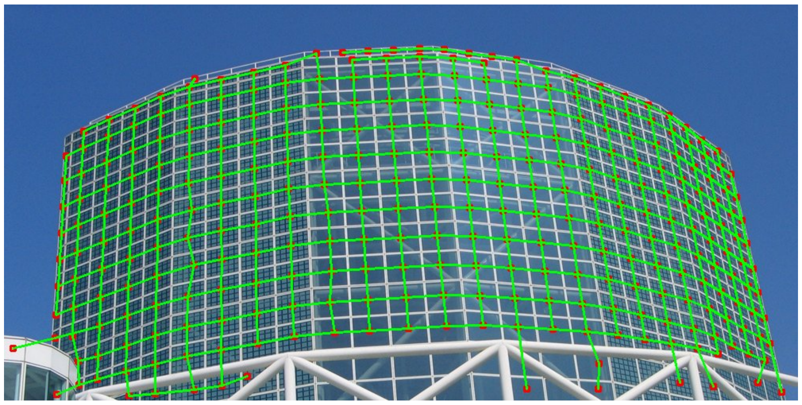
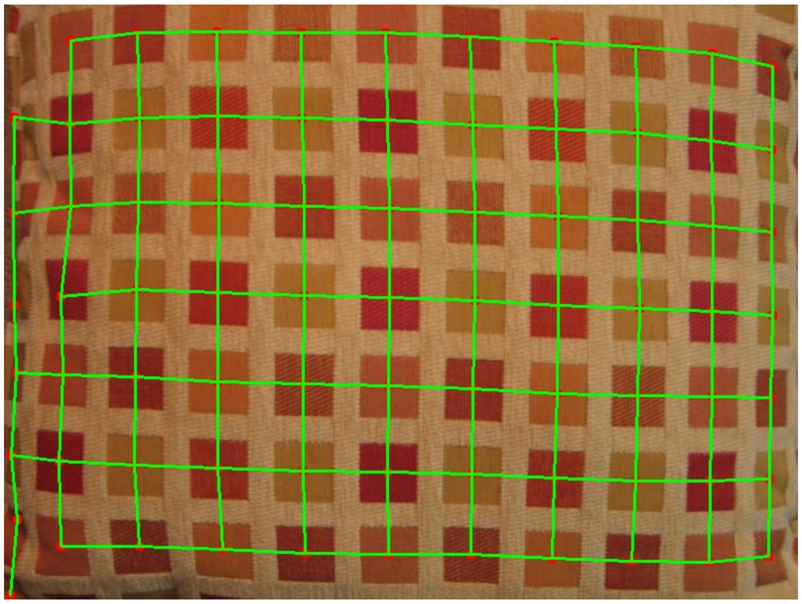



**Figure 5.20:** MCMC grouping on tokens extracted from MSER feature detector.

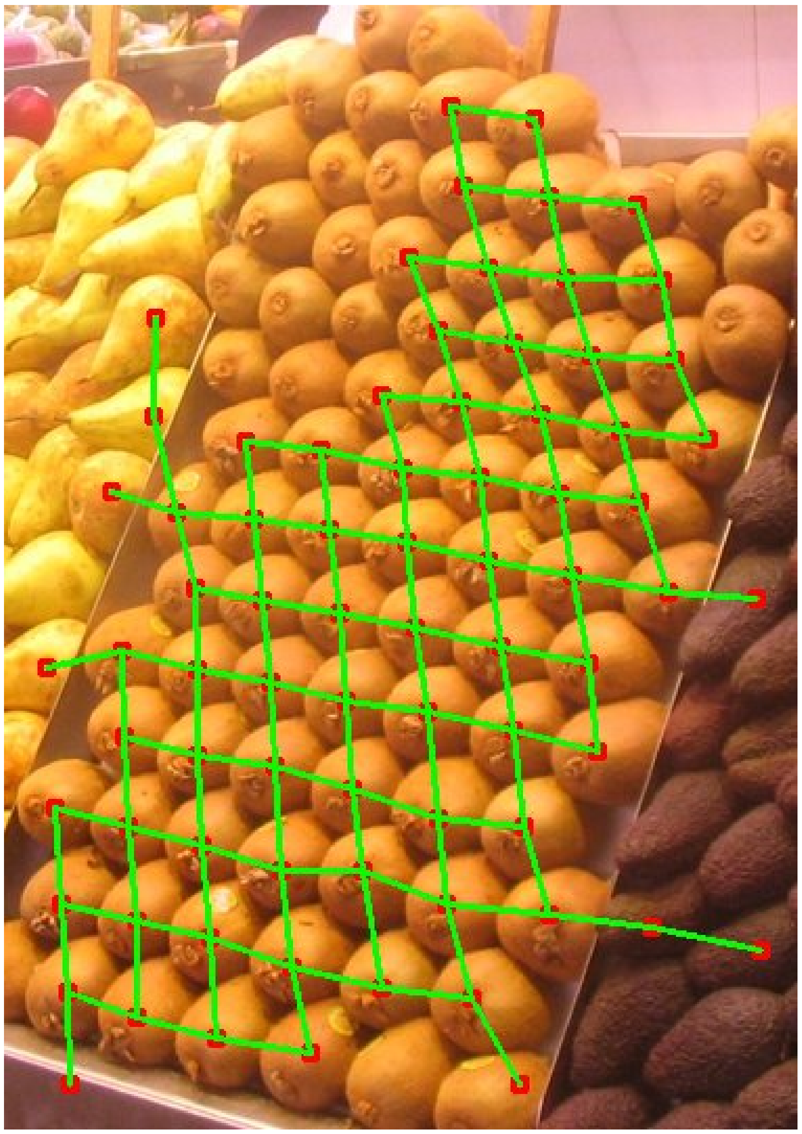


**Figure 5.21:** MCMC grouping on tokens extracted using Regions-of-Dominance. The user selected patch is the same as that shown in Fig. 5.10.

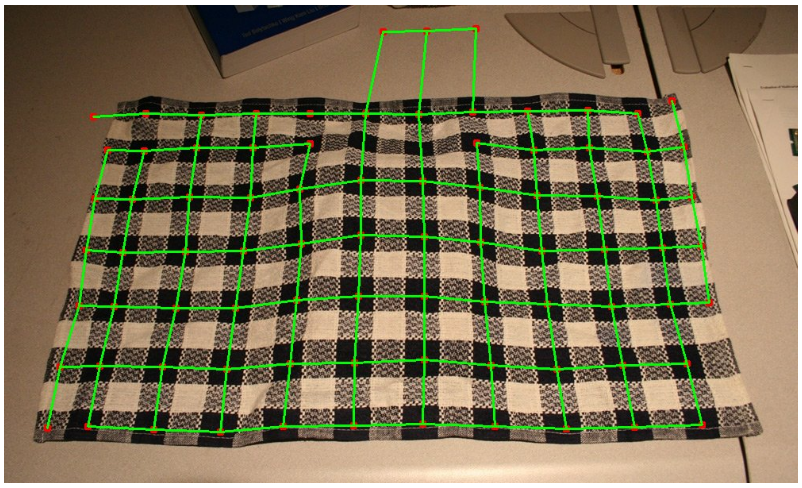


**Figure 5.22:** MCMC grouping on tokens extracted using Regions-of-Dominance.

image are grouped together based on its adherence to this model. These entities are rectangles – very prevalent on building facades – hypothesized from straight lines in the image, though other image discretization functions may be used depending on the nature of the data. Our Bayesian framework then integrates appearance, shape, and topology constraints during grouping. The texture-specific interaction potentials we use for buildings under perspective are then defined. The Metropolis-Hastings algorithm with a customized proposal density is used to generate the chain or sequence of samples that approximate the posterior. Results are shown on both synthetic as well as real building images.

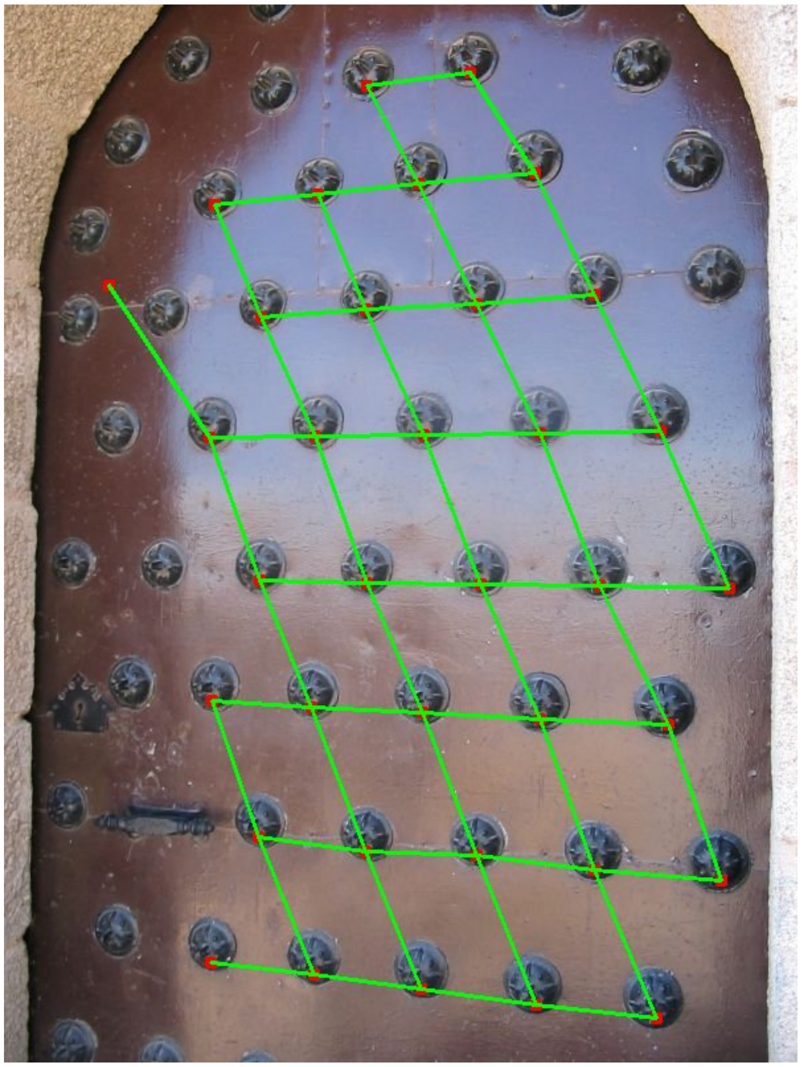


**Figure 5.23:** MCMC grouping on tokens extracted using Regions-of-Dominance.

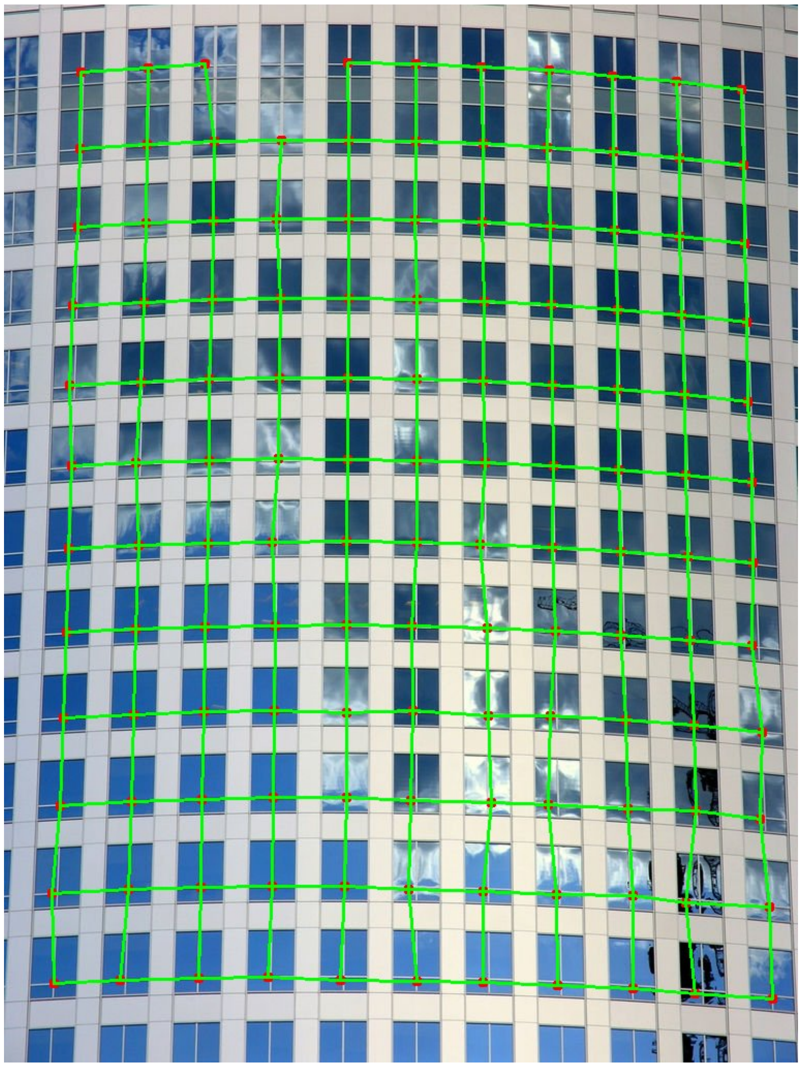


**Figure 5.24:** MCMC grouping on tokens extracted using Regions-of-Dominance.

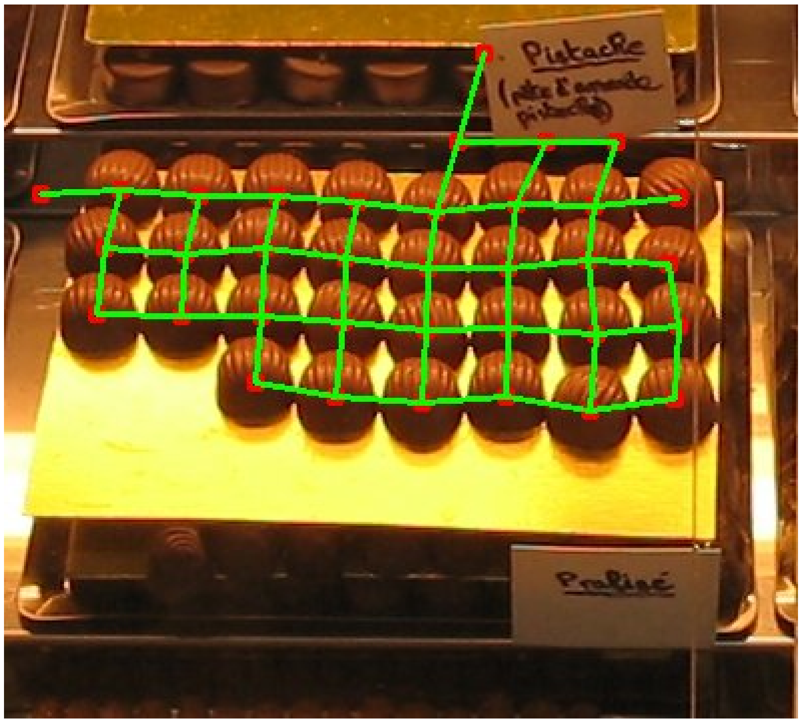


**Figure 5.25:** MCMC grouping on tokens extracted using Regions-of-Dominance.

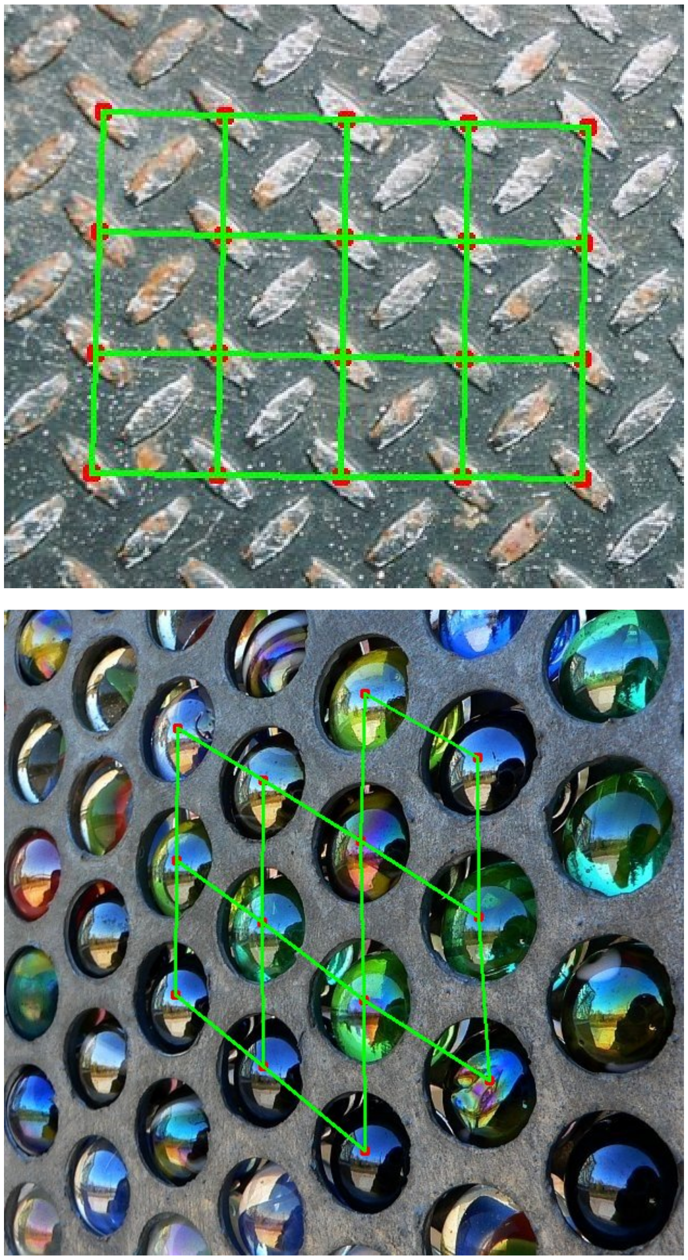


**Figure 5.26:** MCMC grouping on tokens extracted using Regions-of-Dominance.

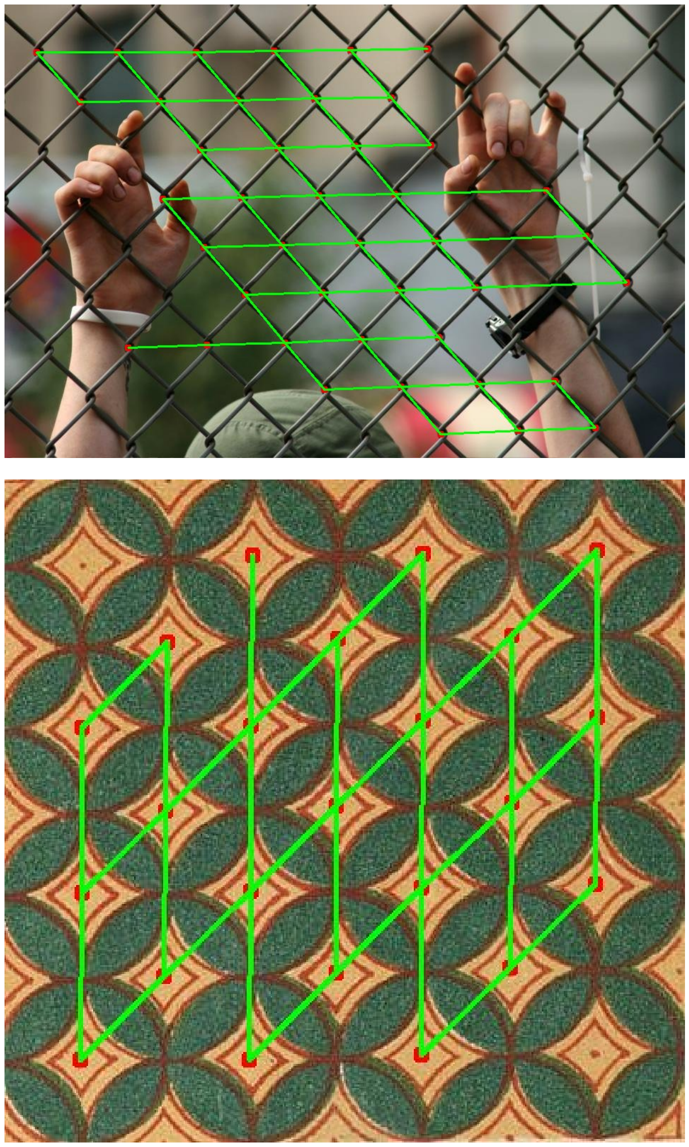


**Figure 5.27:** MCMC grouping on tokens extracted using Regions-of-Dominance.

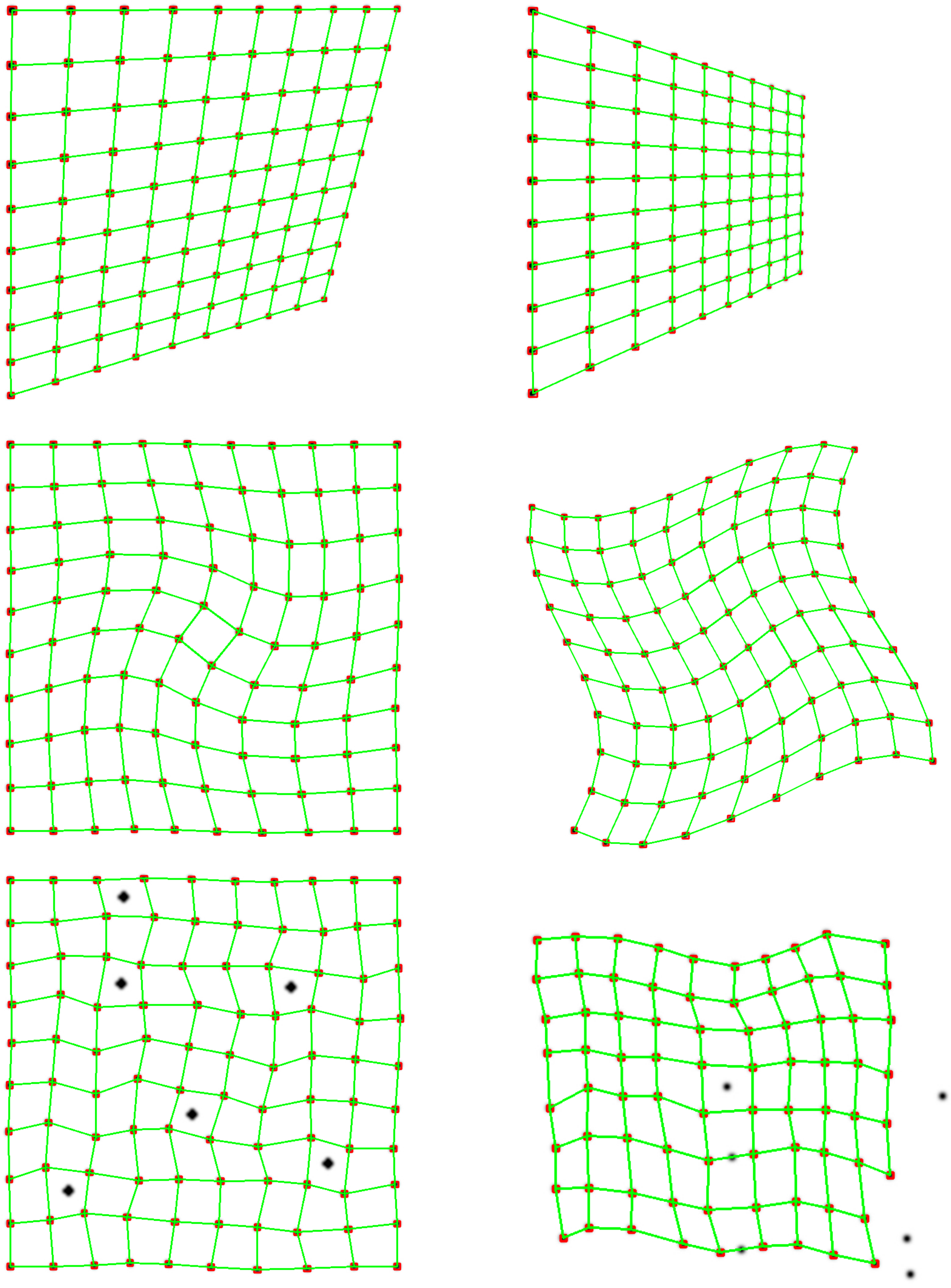


**Figure 5.28:** Discovered lattices from synthetic NRT images.

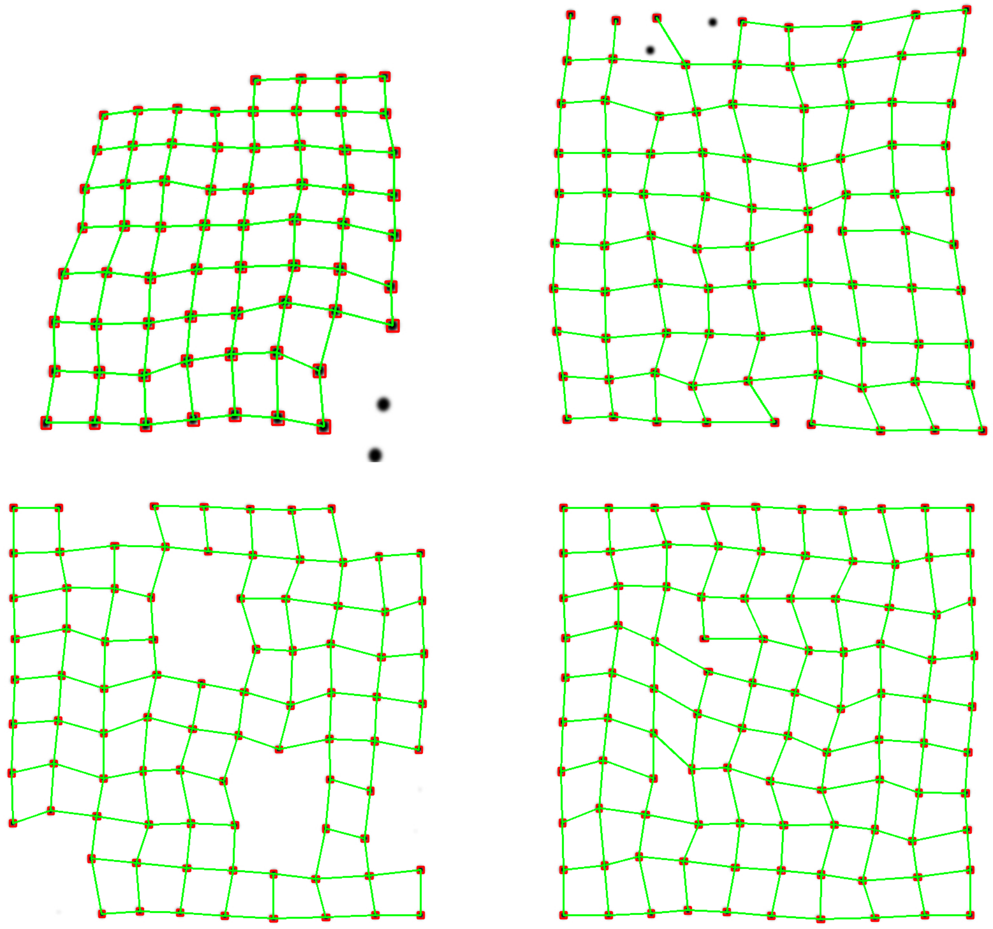


**Figure 5.29:** Discovered lattices from synthetic NRT images.

# Chapter 6

# EXTRACTING SEMANTIC DESCRIPTIONS OF BUILDING IMAGES

The previous chapter introduced an algorithm to discover the window lattice from images of building facades. We now address the task of extracting other semantic properties. In addition to window localization, the discovered grid provides vital cues about positioning, layout, and scale. These parameters may enable facade synthesis or enhancement. In chapter 4, we used motion to extract out candidate patches for various layers and learn appearance models. Could the window lattice be leveraged to extract training patches for generalization? What assumptions and heuristics would be applicable? A common thread that weaves together the techniques described here is the goal of automatically detecting and removing foreground elements from the image.

Building upon the techniques for discovering and parameterizing building window and brick textures, this chapter attempts to semantically explain images of buildings. The components covered in the following sections are:

- Recovering parameters of the grid such as window dimensions and grid spacing to possibly identify occluded windows.
- Semantically describing the inside of window texture elements by inferring the rules of a split grammar that might have generated it.

- Automatically learning Gaussian color models of building and window elements for building segmentation.
- Identifying outliers at a per-pixel level with a robust measure of spread.
- Reconstructing partially and fully occluded tiles to obtain a "scrubbed" image.

To ease the processing, all the above methods assume that the input image is rectified. One approach might be to leverage the discovered window lattice to compute the rectifying transformation. We have simply assumed that building scenes contain several parallel lines in two orthogonal directions, facilitating automatic rectification.

## 6.1 Lattice Completion

Occlusions or errors in the rectangle detection step might cause missing nodes in the lattice. The already grouped elements facilitates parameter inference of the regular grid. The median height, width, and magnitudes of the horizontal and vertical $\mathbf{t}$ vectors are computed first. We then pick the node with the highest likelihood according to (5.1) as an origin. This completely specifies a regular grid that can be overlaid on the image to hypothesize missing or occluded lattice elements.

This approach is not as accurate if the actual windows deviate from the perfect grid assumption. For instance, windows on the ground floor have slightly different dimensions. However, we observed that windows on the same floor are similar and are centered horizontally and vertically with its neighbors. As explained in the previous paragraph, the parameters describing a regular grid (and which best approximates the discovered lattice) are first computed. Then for every window location, we test whether an actual window was detected in the grouping stage or not. If so, that window is drawn according to its parameters. Otherwise, a missing grid element is inferred at the location aligned horizontally and vertically with its neighbors. The size is set to be the same as that of its horizontal neighbor.

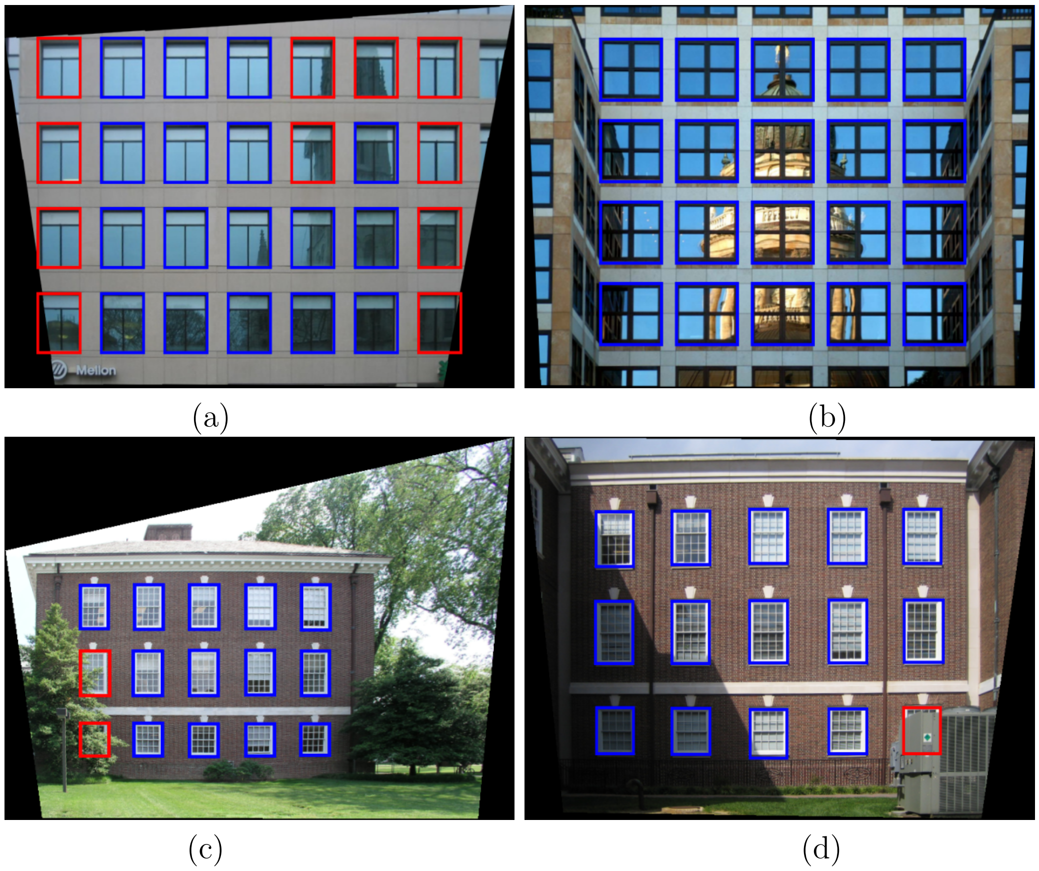


**Figure 6.1:** Discovered grid shown as blue rectangles on a variety of images. Images were automatically rectified as a pre-processing step. Rectangles plotted in red are occluded or missing windows inferred from the result of grouping.

Figure 6.1 shows examples where the discovered grid (drawn in blue) is used to identify missing or occluded grid elements (drawn in red). The images in the top row are highly regular and can be fully described by a single set of global parameters. Predicting the location of occluded windows in figures 6.1 (c) and (d) however required the more adaptive approach that accounts for systematic variations in window dimension. Attempting to describe the grid with just a few parameters obviously works only with very regular structures. Nevertheless, even when the window spacing is not systematic, these grid parameters can provide a good initial

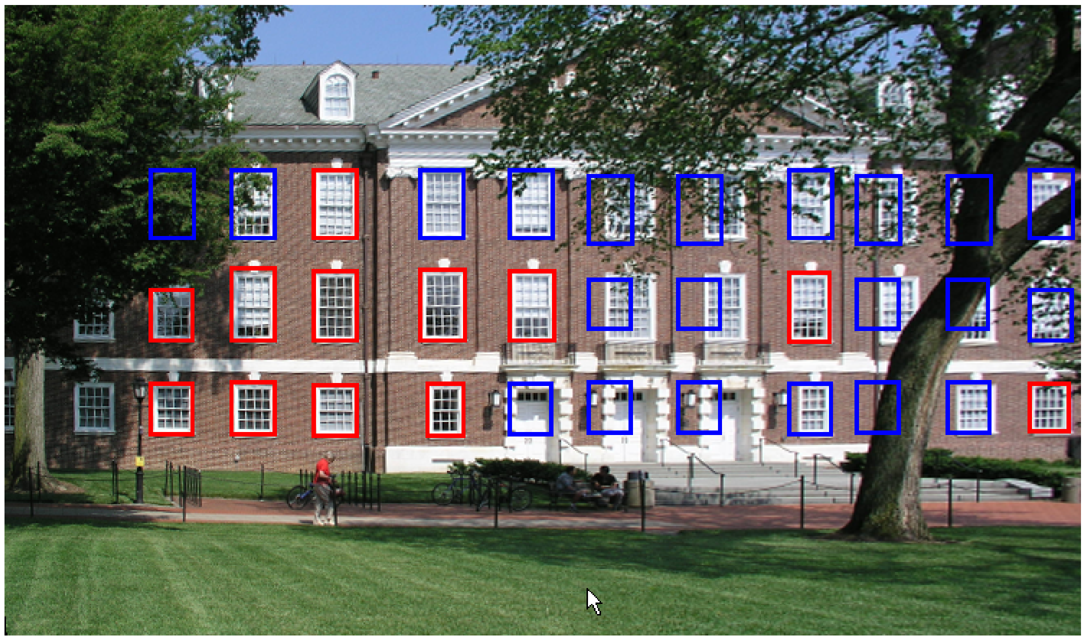


**Figure 6.2:** The parameters can be used for an initial guess about window locations even when the spacing is very irregular.

guess about the location of the windows. The example of inferred windows on the highly occluded and geometrically irregular building of 6.2 demonstrates this.

## 6.2 Split Grammars for Parsing Window Interiors

The windows detected by our algorithm can exhibit large variations across buildings. Inspired by the work of Alegre and Dellaert [6], we model the inside of windows by a set of hierarchical partitions generated by rules of a context free grammar. While [6] tried to semantically partition the entire building facade, we feel that the inside of the windows are more amenable to grammar-based inference. Their approach would take an input grammar specified by the user and find the attributes associated with each production rule. We propose to infer both the rules and the attributes, taking as input only the number of regions to partition the window into.

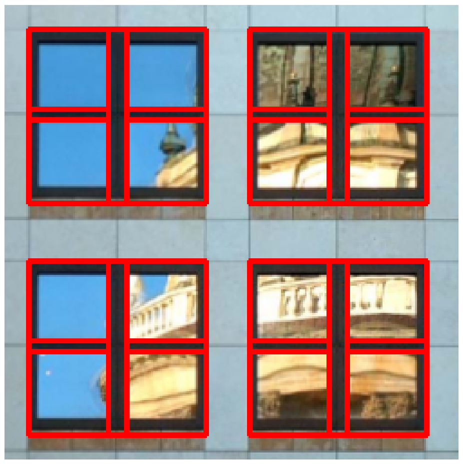
To a user, this is much more intuitive and it circumvents the ambiguity of how fine-grained the resulting parse tree needs to be. We feel that such semantic inference can have various benefits from determining building style to explaining variations among the windows of a building.

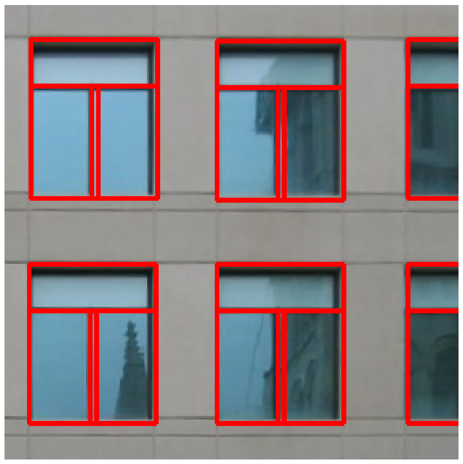
Split grammars were introduced by [173] to describe architectural shapes for procedural modeling. Each production of the grammar corresponds to the decomposition of a basic shape into another shape with derived attributes. Similar to [6], our focus is only on 2D rectangular shapes (corresponding to windows) of a rectified image of the building facade. The nature of structured environments allows us to assume that any split operation is horizontally symmetric and applicable to all detected windows. Parsing in our context is the process of determining the sequence of splits that most likely generated the pixels inside the window. We adapt the general framework of [6] and their MCMC sampling of the posterior to work on our window lattice.

Following regular convention for generative grammars, the grammar is defined as a 4-tuple $G = (V_N, V_T, R, S)$ such that (1) $V_N$ and $V_T$ constitute a finite set of non-terminal and terminal symbols, (2) $R$ is the set of rules or productions and (3) S is the start symbol. $S$ is initialised to each of the bounding rectangles returned by lattice discovery as opposed to the whole facade in [6]. Grammar symbols consist of **operators** and **regions** and each production takes the form below:

$$r : op[attrib_1, attrib_2] \longrightarrow r_1, r_2$$

where $op \in \{hsplit, vsplit\}$ correspond to ways of splitting a region $r$ into two subregions $r_1$ and $r_2$ along the specified coordinate direction. Region attributes $(x, y, w, h)$ correspond to position, width and height in the input image of the rectangular region while attributes associated with a split operation are ratio of total area and thickness of the dividing line. A series of derivation steps that partition the starting window into subregions is naturally encoded by the parse tree.


(a)


(b)

$0 :: hsplit[0.5, 5.0] \longrightarrow 1, 2$

$1 :: vsplit[0.49, 3.0] \longrightarrow 3, 4$

$2 :: vsplit[0.49, 3.0] \longrightarrow 5, 6$

(c)

$0 :: vsplit[0.29, 0] \longrightarrow 1, 2$

$2 :: hsplit[0.5, 2.0] \longrightarrow 3, 4$

(d)

**Figure 6.3:** Parsed window focusing on a section from (a) dome image of Fig. 5.4a and (b) building image of Fig. 6.1a. (c) Derivations for parse tree of (a) and (d) derivations for parse tree of (b).

The semantic rules of a split operation govern how child nodes inherit attributes from the parent. Consider a vertical split parametrized by area ratio $a$ and thickness $th$ on a region with attributes $(x, y, w, h)$. The division occurs at offset $h_{split} = a \times h$ resulting in two subregions with attributes $(x, y, w, h_{split} - th)$ and $(x, y + h_{split} + th, w, h - h_{split} - th)$.

Given a grammar specification customized for the input image, [6] tries to infer the parameters of the attributes that best explain the data. We only require the user to specify the total number of required partitions $p_w$, thus inferring both the rules and the attributes. A particular challenge faced by [6] was due to occlusions. Since occlusions are inevitable when modeling a whole facade, we use grammars to model only the inside of a window. The posterior is sampled using Data-Driven

MCMC to generate proposals that have a high probability of being accepted.

Each iteration of MCMC begins by resetting all window elements to $S$. A new window style is proposed by $n$ applications of rules that generate $p_w$ leaf nodes in the parse tree. This becomes the proposed MCMC state. A rule is generated in three phases. A leaf node is first selected at random from the current set of terminal nodes. A subdivision rule is then picked according to prior probabilities for each rule. In order to align splits with edges, we use the row and column sums of the normalized gradient image over all windows as proposal distributions to sample ratio and thickness attributes. The splits are applied such that horizontal symmetry is maintained. The forward and reverse proposal probability is the product of the probabilities of each step. Since we assume uniformity, the rules are applied to all windows simultaneously.

The MCMC acceptance ratio test requires definition of the likelihood for the newly proposed state. Under independence assumptions, the probability of a parse tree is the product of all rules applied to derive it. Our grammar being relatively simple, all rules are given uniform prior probabilities. For interior nodes of the tree, splits should be aligned with a high gradient edge. Given $h_{split}$ and $th$ of a vertical split in region $r_i$, we search within a small threshold distance ($\approx \pm 10$ scan lines in that region) to find the two rows with maximum gradient sum $M_g$, centered at row $R_g$ and separated by $2 \times th$. The split likelihood $P_{split}(r_i) = \frac{M_g}{W+W} \times \Delta(|R_g - h_{split}|, \sigma)$, where the first term is the mean gradient strength over the two rows and $\Delta$ is a Gaussian weighting function. Horizontal splits are handled in an analogous manner. We found this technique to be robust to misalignments caused by rectification or occluding foreground. In the case of terminal leaf nodes, strong edges should be penalised. Let $S_r^j$ and $S_c^j$ be the maximum gradient row and column sum for terminal node $r_j$. The likelihood $P_{leaf}(r_j) = \exp(-3 \times MAX(\frac{S_r^j}{W}, \frac{S_c^j}{H}))$. It is to be noted that during the computation of $S_r^j$ and $S_c^j$, gradients closer to the region boundaries are given less weight.

Figure 6.3 shows the output of the parsed windows and generating productions for the two buildings in figures. 5.4 and 6.1a. Despite the lack of sufficient image information due to reflections within many of the windows, our use of global information over all the windows allows us to accurately localize the offsets for making a split.

## 6.3 Facade Segmentation

So far, our discussion has mostly centered on the window grid and its interior. Another key property to infer from the image is the building extent and occluded regions. Can we segment out building pixels in a single image without any prior knowledge of background or foreground appearance? By making assumptions founded on common architectural trends, we demonstrate how a binary mask can indeed be extracted. Just as motion was used in Chapter 4 to generate training examples, here we use the window lattice to bootstrap the learning.

Given an image of a building, we assume that the pixels or texture immediately around the perimeter of a window belong to the building wall. Unless the building is comprised of glass walls, this is the case in most buildings. Secondly, a color Gaussian Mixture Model (GMM) can describe the majority of pixels in the image. Many segmentation and foreground separation problems [135, 147] use the latter assumption for probabilistic classification. We will describe how these simple assumptions may assist in recovering the building pixels.

As a first step, training examples are required to learn a color model of the building wall from the given image. Once again, rather than learning a universal model for generic building pixels, an image-specific model is built by extracting the RGB values of pixels in a band around each window. These pixels (shown as a binary mask in Fig. 6.4) are assumed to be generated by the building texture and constitute our training set. Learning involves estimating parameters of a Gaussian

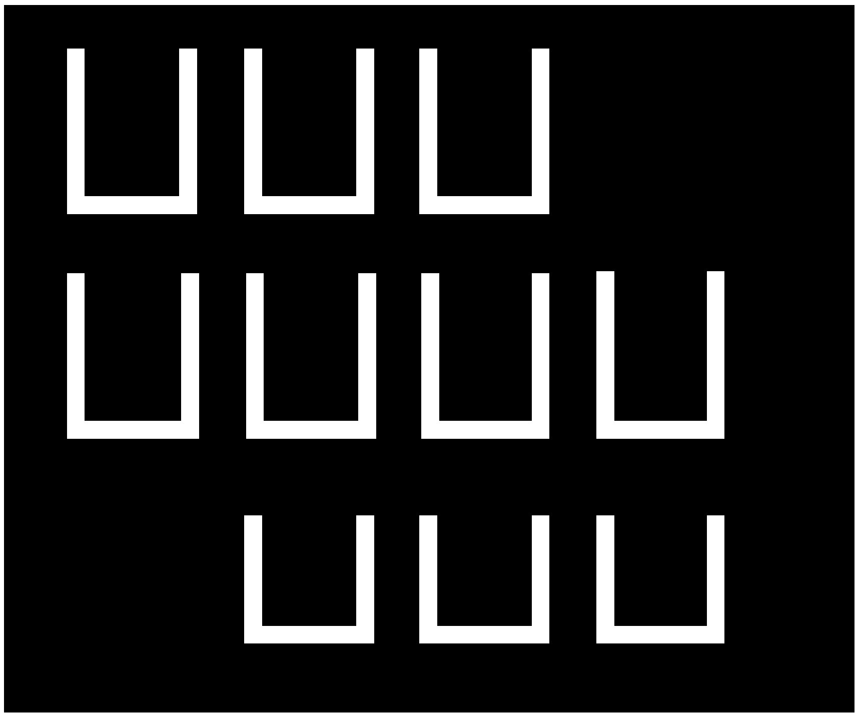


**Figure 6.4:** Mask used to extract training examples for the wall pixels. Pixels around the detected windows are used to learn a color model.

Mixture modeling the RGB values in the example set. We used the publicly available Cluster package [16] to build the Gaussian model. For now, we assume that the generating distribution for the wall is a single Gaussian $W$ represented by mean $W_\mu$ and covariance $W_R$. Building walls with multiple colors or artifacts such as shadows can cause problems. However, most building walls are painted in a uniform color.

Having a model of the wall texture, we proceed to cluster the RGB values in the entire image into separate Gaussian distributions $\mathcal{G}_i : i \in 1..N$ with mean and covariance $\mu_i$ and $R_i$ respectively. The number of subclasses $N$ is typically less than 10 and computed automatically by the Cluster package using the minimum description length (MDL) criterion. Since clustering the whole image can be slow, the image is smoothed with a Gaussian kernel and subsampled at half the original resolution before estimating the parameters. Based on the homogeneous texture assumption, one of the Gaussian components will correspond closely to the wall texture. The maximum likelihood wall cluster $w$ is computed as

$$w = \arg\max_i p(\mathcal{G}_i = W)$$

where

$$p(\mathcal{G}_i = W) = \frac{1}{(2 * \pi)^{3/2} R_i^{1/2}} \exp\{-\frac{1}{2}(W_\mu - \mu_i)^T R_i^{-1}(W_\mu - \mu_i)\}.$$

Figure 6.5 shows the mask generated for the two images, using the window lattice to learn a color model. The mask seems to correspond very well to the building pixels and is especially good at separating out the complicated foreground. Even though there are small holes or islands of incorrectly classified pixels, other neighborhood constraints such as those used by MRF algorithms should be able to rectify them. We can also use the mask to compute the facade boundaries (right column of Fig. 6.5) by thresholding on the row and column sum differences. This technique is valid because of working with rectified images.

It is important to note that the goal of this section is to focus the robot's attention and processing to the most relevant parts of the image. Localizing the building extent immediately provides clues regarding scale, pose, and occlusions. There is also room for contextual reasoning as used by Cornelis et al. [23] where an initial geometry estimate was used to guide searching for objects. The limits of the facade could be used to generate a likelihood map of potential door locations or staircases. At a lower level, the mask can be refined in a manner similar to matting and used for foreground removal. The above techniques serve as a good platform for semantic inference about building images.

## 6.4 Foreground Removal

The output of the texture discovery process outlined in the previous chapter is a set of subimages centered on the tile elements that should be very similar to

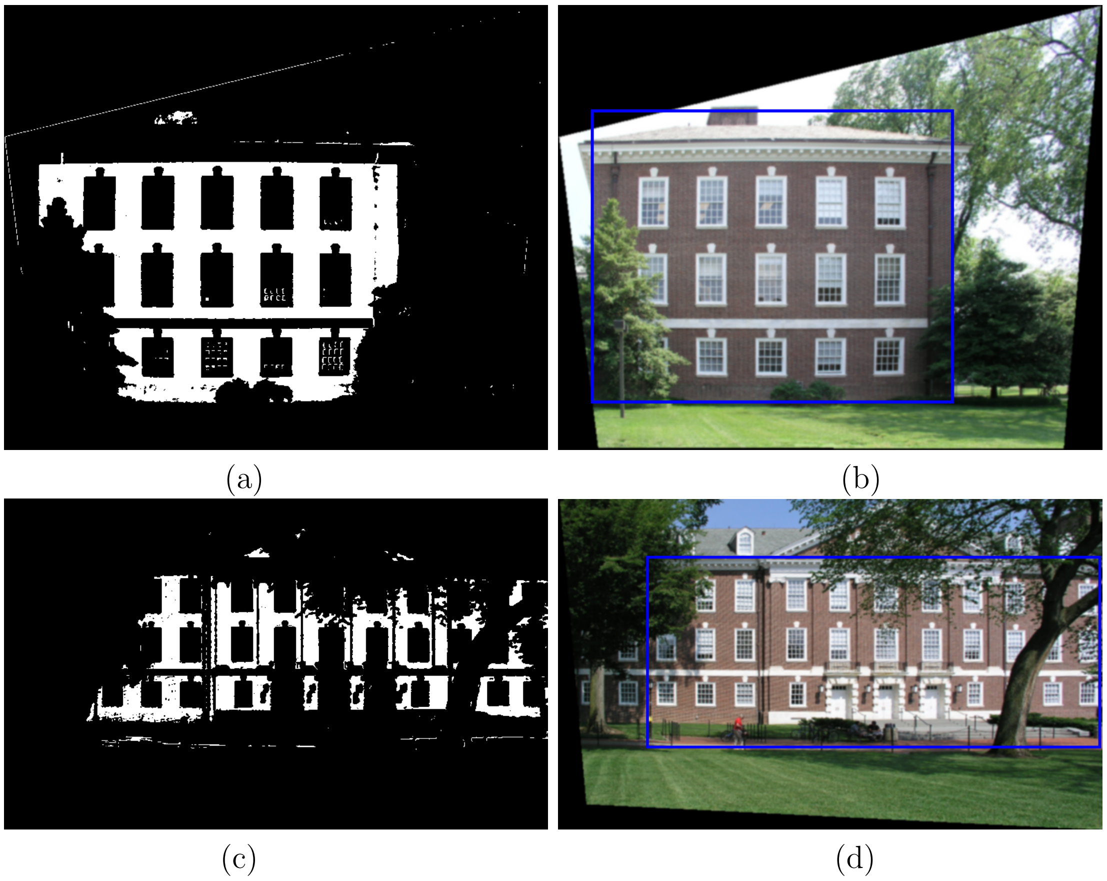


**Figure 6.5:** Mask of the wall pixels after maximum likelihood classification. Thresholding on the row and column sum differences between adjacent locations can be used to approximate the facade extent as illustrated by the blue rectangle.

one another. We say "should be" because although ideally the tiles are identical, there are a number of factors which can cause appearance discrepancies, including natural material variations, illumination changes, spatially-varying resolution due to perspective, non-planar features, and intrinsic errors in the tile alignment process.

Another—and here the most interesting—reason for tile dissimilarity is that some tiles may be occluded by or contain reflections of non-sky foreground elements. The variations enumerated above are mostly describable with low-dimensional models—Gaussians or mixtures of Gaussians for stochastic variation, blurring and shifting

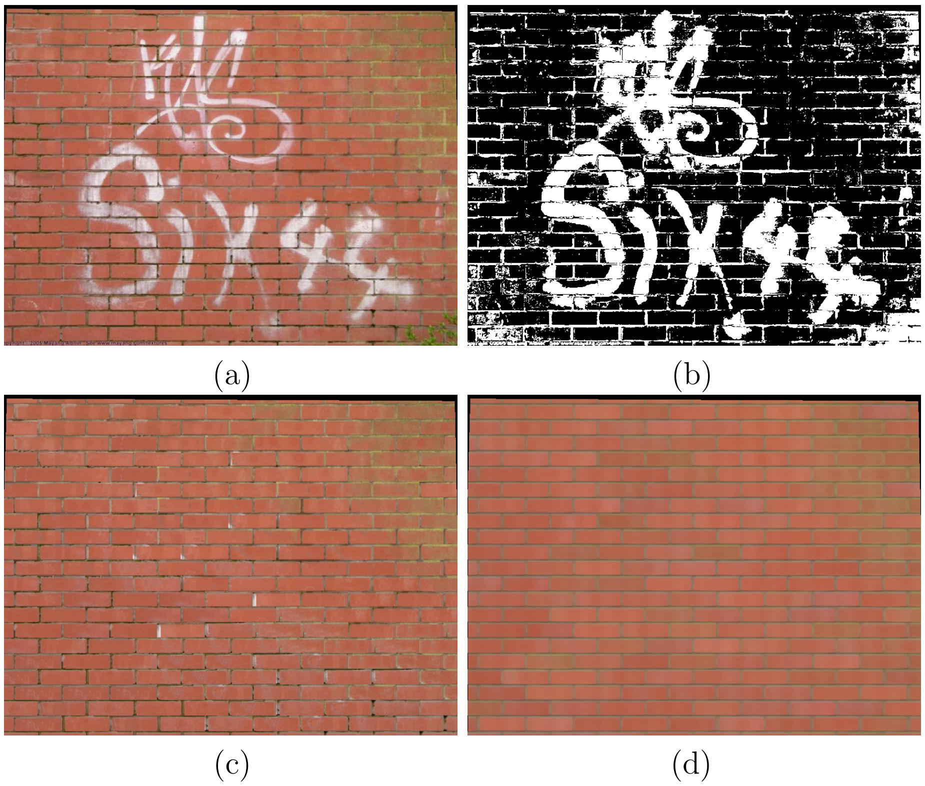


**Figure 6.6:** Virtual graffiti removal. (a) Original image; (b) Foreground pixels as indicated by MAD outliers; (c) Within-brick exemplar-based inpainting followed by full tile sampling; (d) 8-base PPCA reconstruction. Tiles with $> 25\%$ outliers were sampled. While there is some loss of detail in (d), many local characteristics are retained. (Image has been automatically rectified)

for perspective effects and small misalignments, and large variance radial basis functions (for example) for gradual shading changes across a building facade. However, foreground elements can be of any kind, and thus are perhaps best treated as outliers of the building tile pixel model. The two primary questions left are how to explicitly identify such outliers, and what to replace them with in the final rendering.

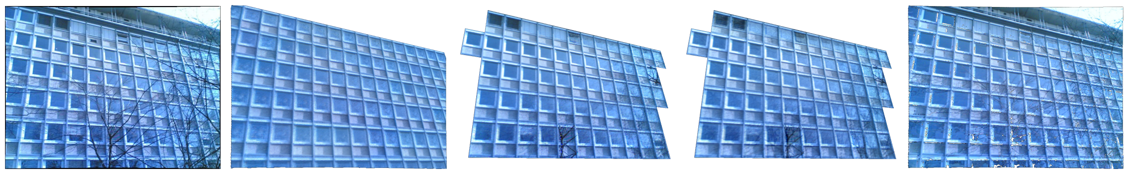
We answer the first question by looking at pixel values in corresponding locations over all tiles under the assumption that the background is visible in a majority of them. A robust measure of spread, the median absolute deviation (MAD) [90], can be used to assess which pixel values vary enough across the tiles to arouse suspicion that a foreground element is present among them.[1] Unreliable pixels are identified by thresholding their MADs—these are so-called *MAD outlier* pixels. To be conservative, the threshold is chosen to treat 50% of pixels as outliers.

An obvious approach to answering the second question is to do spatial inpainting to fill in pixels masked out according to the MAD criterion. A number of texture synthesis algorithms are available for the task, including patch-based techniques [27, 40], near-regular texture synthesis [106], and graphcut-based methods [88]. Figure 6.7(e) shows the result of inpainting MAD-induced holes in Figure 6.7(a) with the method of [27] under the special case that the inpainting source patches are the same sizes as and perfectly aligned with the discovered tiles. This technique removes the thickest parts of the trees but misses fine foreground detail and introduces some photometric artifacts. It must also stop before completion when destination missing pixels only correspond to missing source tile pixels (white areas indicate unfilled pixels).

As bricks can have significant internal texture and color variation across a facade (which Figure 1 shows), for them we favor first copying from within the same tile when possible. This also significantly speeds the process over a whole-image search. In this case, tiles which are non-trivially occluded ($\geq$ 25% outliers) are regarded as not possessing a large enough source area to reliably guide inpainting, and are marked for complete replacement. This occurs in a second stage after "internal" inpainting. Each such tile is replaced by a random nearby "good" tile in a fashion somewhat like image quilting [40].

Another possibility suggested by the alignment of patches is to treat the problem as one of eigenimage reconstruction. Assuming that a Gaussian process

[1] A scalar MAD value is obtained at each pixel by computing it separately for each color channel and summing



(a) (b) (c) (d) (e)

**Figure 6.7:** Comparison of pixel filling methods. (a) Original image; (b) PPCA reconstruction, 8 bases; (c) RPCA reconstruction; (d) PCA construction, 8 bases; (e) Whole-image exemplar-based inpainting [27] with patches scaled to and aligned with tiles (unfilled holes are white)

describes inter-tile appearance variation fairly well, we can use principal components analysis (PCA) to model it (e.g., [163]). The intuition is that we want to take each occluded tile in which some background is visible and "project" it down onto a set of background-only bases in order to remove or lessen the foreground influence. However, with some fraction of the tiles "polluted" by unknown foreground elements, robust PCA (RPCA) techniques [175, 90] are required. [175] throws out whole bad data samples, but as we would like to use any partially visible background, a method which estimates intra-sample outliers such as [90] is more appropriate. This approach uses M-estimators to reduce outlier influence on the reconstruction. It, too, performs outlier detection using MAD as an initial scale estimation step, but in practice we have found that it has some problems with our data. For example, while Figure 6.7(d) shows a standard 8-base PCA reconstruction that as expected still includes many tree branches, Figure 6.7(c), which uses RPCA, is scarcely better. Some tiles are cleaned up compared to their original appearance, but ones with too many outlier pixels are virtually unchanged.

Since the MAD mask with an aggressive threshold seems to identify foreground pixels fairly well, we use another PCA variant called probabilistic PCA (PPCA) [136] which works when missing data is explicitly identified beforehand.

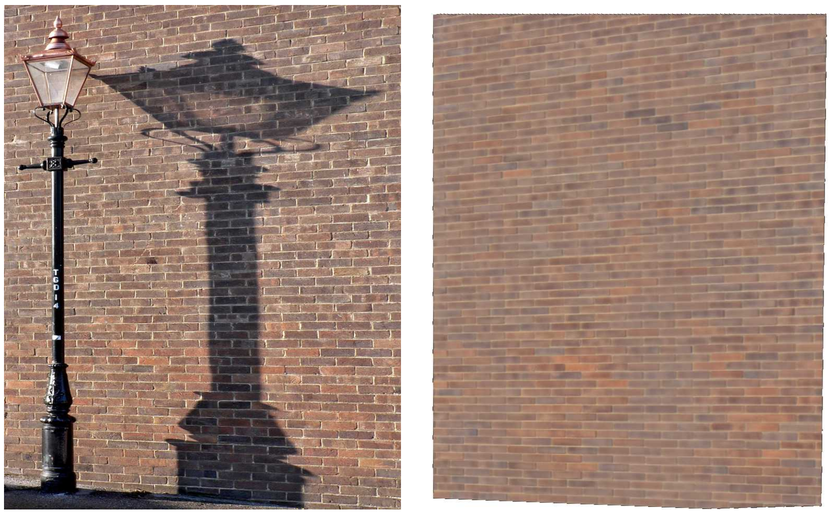


**Figure 6.8:** Removal of lamppost and shadow. Right image is 16-base PPCA reconstruction from MAD outliers. Edge artifacts on bottom image are due to flood-filling black background to white.

Using an expectation-maximization (EM) approach, PPCA generates maximum likelihood estimates for the missing data, which for us serves to fill in the image holes erased by the MAD mask. As the percentage of pixels occluded in a given tile rises, however, the reliability of the reconstruction naturally deteriorates when too many missing values are recovered based on too few real pieces of data. We mitigate this issue by reconstructing only tiles that have $\leq 25\%$ outliers in them according to the MAD mask. All other tiles are treated as fully occluded and simply sampled *de novo* from the learned PPCA basis. Since PPCA can handle missing data, it works on the fractional tiles along the edges of the image as in 6.7(b), whereas RPCA and PCA need full tiles to do a reconstruction.

## 6.5 Summary

The complex nature of the real-world makes purely bottom-up processing insufficient under noisy or uncertain situations. In such cases, we demonstrated how to bring in domain knowledge, especially using topological priors, to reason about the scene. For a task like automatic foreground removal and hole filling from single images of buildings, we introduced several simple techniques that use a partially discovered window grid on the facade to extract some semantic properties of the building. First, the partial grid is completed to identify potential locations of missing or occluded windows. Some insight into the architecture of the facade can be gained by identifying the common subdivisions within each window. These are represented by split grammars and the rules that derived a particular window grid are inferred in the spirit of *image parsing*. Gaussian color models are used to segment out the building wall from the image and localize facade boundaries. Finally, we demonstrated a promising approach to automatic foreground feature removal in static images of building textures.

# Chapter 7

# CONCLUSION

## 7.1 Key Observations

In the last few chapters, we have provided a set of solutions to bring in domain semantics into the larger problem of robot-based modeling of buildings and urban environments. Automatic modeling of architecture has been a popular area of research, and many solutions have been proposed depending on design issues such as type of sensors, automatic- or manually-guided acquisition, mounted platform, human or mobile agent, and intended audience. A vast majority of these methods proceed in a bottom-up fashion by piecing together sensor input from multiple locations and applying efficient but brittle closed form solutions to construct a virtual representation. Developing a completely automatic mobile platform capable of path planning, robot navigation, image acquisition, texture mapping and structure recovery is extremely hard. Researchers have therefore tried to break down the problem and handle each of them in isolation. However, in the context of an end-to-end system, we believe that the whole is greater than the sum of its parts. An integrated solution gives rise to a whole new series of issues and we have tried to focus on those that have been only marginally addressed by previous work.

One such issue is the quality of textures that are mapped to the surface model; this problem is usually sidelined by the seemingly more important task of estimating scene geometry. We also devoted some attention to the task of robot navigation and image capture. Rather than path tracking or pure exploration using on-board sensors alone, we complement these algorithms by extracting higher-order information from satellite imagery. Finally, the very complexity of our proposed system and the nature of the real world guarantee an innumerable number of situations in which our mobile agent will be uncertain about how to proceed. In such a predicament, we believe that the agent should make use of high level domain semantics to reason about the scene. Classical computer vision reasons about appearance, which alone might be inadequate outside lab environments over a long period of time. Like humans, the agent must have the ability to rationalize about functional and topological characteristics as well. We demonstrated how some of these *a priori* spatial relationships may be encoded, discovered, and exploited to extract semantic properties of architectural elements. More than any specific technique, we consider this to be the overarching theme of our work, and a principle that will have a great role to play in mobile robotics.

## 7.2 Contributions

This thesis makes the following contributions towards the goal of automatic robot-based modeling of buildings in an urban environment:

- We introduced the components and their interactions of an end-to-end system whereby a GPS-enabled robot equipped with on-board sensors and an aerial map of the environment can build a photo-realistic 3D representation of an urban environment. In addition to planning, robot navigation, image matching, mosaicing, and structure estimation, we argued the case for a semantic module to combine bottom-up and top-down processes. This scheme of knowledge transfer between processes is referred to as a *cognitive loop* by [22], enabling the strengths of one algorithm to overcome the weaknesses of another.

- We exploited the building outline visible from an aerial image to develop a randomized view planner that efficiently builds incremental paths around the building. The goal is to guarantee that every face of the polygonal facade model will be captured at good quality in a minimum number of views. A quality metric is defined for each cell of the environment map based on foreshortening and camera resolution, and a path is planned keeping robot dynamics into account.

- We described how a "birds eye-view" provided by satellite imagery can be used to assist in vehicle localization in the case of a manually guided robot. A multi-modal Monte Carlo Localization (MCL) framework is introduced that can correct for GPS errors (very common in urban canyons) and localize the vehicle on the road. The underlying algorithm is essentially a spatial tracker that traces the road in the vicinity of the vehicle using either GPS-driven or texture-driven dynamics. To account for multiple types of drivable paths, we defined various road-likelihood metrics with automatic switching to the most appropriate one. Accurate and efficient localization is necessary for any mobile platform, especially when subsequent processes such as image matching rely on GPS-tagged data.

- We adapted our road tracer to work as a localized planner in the case of an autonomous robot. Given the current location of a robot, the road/path ahead of the vehicle is traced in the aerial image to predict the nature of the immediate terrain. Appropriate signals are given to the robot such as warnings about sharp corners or directions to the nearest road in case it is stranded off-road. In the latter case, on-board sensors alone might be too restrictive.

- We developed a novel spatio-temporal inpainting technique that recovers a clean texture map of partially occluded building facades from video or images. Images from a sequence are stabilized and stacked together to form a *timeline* of potential pixels that constrain what gets included in the texture map. Mosaic regions polluted by foreground objects are automatically identified through a robust measure of pixel color variance over the registered images. We then try to infer the building pixels in the polluted regions via combined spatial and temporal search. To overcome inefficiencies of the exhaustive search procedure in classical inpainting, we bootstrap training of a classifier from automatically generated examples to disambiguate between foreground and background. We compared both the accuracy and efficiency of three different classifiers with SSD-based inpainting and demonstrated speed-ups of an order of magnitude, while also improving robustness.

- We draw the analogy that building facades are often examples of Near-Regular Textures (NRT) [66]. We derived a Markov Chain Monte Carlo (MCMC) approach to discover such patterns from images. A Markov Random Field (MRF) model for NRTs and lattice structures is defined; entities in the image are grouped together based on its adherence to this model. The grouping is run on tokens extracted by an image discretization method such as an interest point detector or any other high-level feature detector. Specifically for windows and buildings under perspective, our feature extractor is a simple yet efficient procedure to extract out candidate rectangle structures from the image. For MRF grid extraction, we formulated the problem in a Bayesian sense that takes into account appearance, shape, and topology constraints. Various heuristics are combined in a probabilistic manner to construct the MRF interaction potentials for connected window neighbors. To make the MCMC optimization efficient, we introduced an unguided and guided scheme of proposal updates during each iteration that still guarantee conditions of *irreducibility* and *aperiodicity.*

- We demonstrated very simple methods that illustrate how the parameterized window pattern may be used to glean additional semantic information of the facade such as the position and boundaries of occluded or missing windows, facade extent, and a binary mask of the wall texture. Though our current methods rely on some assumptions, they are built upon common architectural styles and practices and should therefore be widely applicable. The motivation here was more proof-of-concept than hundred percent accuracy.

- To develop the theme of *image parsing*, we showed that the inside of windows can be adequately described by split grammars [173]. The MCMC framework is extended to take the detected windows as input and infer the grammar rules that best explain the subdivisions within the window. A data-driven scheme of proposal updates is described to make state changes that have a high probability of being accepted. By using data from all detected windows, the technique is more robust to occlusions or reflections within windows. The end result is a parse tree and its derivation according to the pre-specified grammar. These semantic descriptions can then be used for outlier removal and image synthesis.

- We compared several methods for replacing or "scrubbing" away missing or occluded tiles (windows/bricks) via robust subspace reconstruction and exemplar-based inpainting. By nature of symmetry, pixel variance allows us to identify outlier pixels. In addition to timeline inpainting, the alignment of patches allows us to treat the problem as one of eigenimage reconstruction. Tiles in which atleast some background is visible are projected down onto a set of background-only bases. Different variants of PCA that can handle outliers or missing data are compared to assess the quality of the background reconstruction.

## 7.3 Limitations and Future Extensions

There are several limitations in the ideas presented in this thesis. A complete solution to the urban modeling problem is beyond the scope of an individual graduate student. We divide the future work into two parts. This section discusses some specific extensions that may be applied to the techniques put forth in this thesis. The next section puts forth some broad open issues that need to be addressed in future research.

### 7.3.1 Robot Navigation

**Particle Distribution and Dynamics** For vehicle localization, one short-coming of the weighted mean estimate of particle locations is that it does not take into account the distribution which could be clustered over several different roads. This happens most noticeably at intersections where GPS data seemed most unreliable. Clustering algorithms could be used to track multiple peaks in such situations. Using GPS-driven or texture-driven dynamics in isolation does not seem robust for long runs. It would be interesting to see the effects of adding the dynamics for each particle as another mode in the mixed-state tracker. This would allow some particles to follow the GPS curve, while other particles would follow the most likely road. Intuitively this seems more robust.

**Combining Multiple Sensors** Future work includes integrating information from on-board sensors such as a camera and laser. With aerial images and GPS alone, our algorithm can identify the possible road that the vehicle is on. The use of additional sensors would allow us to correlate the on-board view with the aerial one, giving information about the position and orientation of the vehicle within the road. This can be useful in multiple lane roads or intersections. In off-road environments shown in figure 3.5, it would be very useful to complement aerial imagery with elevation data sets.

**Vehicle Guidance** Our current system of constructing localized plans from the robots current location was tested on data from the actual Grand Challenge competition with very promising results. Nevertheless, for real-time operation much tighter integration needs to be done between the aerial and other on-board modules that feed into the actual steering decision with the ability to resolve conflicts. The planner currently only finds a single road near the vehicle. We have done some preliminary work using skeletonization and watershed image processing techniques to extract a road network in the vicinity of the vehicle, offering more choices to the vehicle and possibly graph-based path-planning.

### 7.3.2 Timeline Inpainting

**Quality of Synthesis** From the results at various stages of the process, it is obvious that Stage 2 spatial inpainting is the weakest. This is because CPT inpainting relies on greedy matching of patches. A single bad choice can propagate errors without any backtracking procedure. For example, the second window from the left in the upper story of Building B reveals an error in the filling-in process that "snowballs" with subsequent patch copies. The consequences of such mistakes could be reduced with gradient-domain methods for blending patches such as [125]. We believe that incorporating the higher level semantic knowledge (described in Chapters 5 and 6) about building entities like whole windows or doors and their regular arrangements into the inpainting will help the most, and it remains future work.

**Eliminating Depth-induced Artifacts** Our image registration algorithm assumes that the building facade is planar and can be modeled by a homography. Slight deviations from planarity can then appear as artifacts in the mosaic. Other structures such as as columns would need explicit recognition and handling, which we currently do not do. The MAD criterion to detect high energy foreground pixels would also not work for homogeneous regions such as vegetation or billboards. The geometrically correct method to overcome these problems would be to extract a depth map using techniques like plane-sweep stereo and avoid copying pixels from different layers. A more interesting image-based rendering technique would be to apply the principle of Digital Photomontage [2], which combines parts of a set of photographs into a single composite. Currently, we copy small $(11 \times 11)$ patches into the mosaic that are prone to context violations. By running our classifier on each frame of the sequence, we could generate an approximate binary mask of building and foreground pixels; these are usually much larger regions. The building pixels from various images can then be combined into a composite. Like [2], seam objectives can be specified to match colors and gradients at region boundaries where adjacent pixels in the composite come from different source images.

**Removing Foreground from Input Sequence** This work discussed removing foreground elements from the mosaiced image, and we showed visually compelling results by combining appearance and temporal information into an inpainting framework. Armed with a clean mosaic and a depth map computed using dense stereo from the input sequence, we could attempt to remove foreground elements from frames of the input image sequence. The first challenge is to identify the foreground in the frame. Once again, there are two cues that may be applicable. Firstly, a suitable similarity metric on appearance could be defined to compare the input frame and the clean texture map in the same reference frame (input image warped to mosaic or vice-versa). For thin regions that exhibit high parallax, motion cues from stereo could be more powerful. Once we can mask out foreground regions from each frame, pixels may be copied from the timeline mosaic to fill in the holes.

### 7.3.3 Lattice Discovery

Occasionally, there might be a missing edge between two neighboring windows in the grid. However, this can be easily filled in as a post-process; there is enough structure information from the rest of the lattice to facilitate this. A more pressing issue is when the initial tokenizer fails to find a node. Factors such as noise and occlusion do indeed cause this problem. Currently, the MCMC simulation only performs the grouping on the initially extracted nodes. Alternately, it could also add new nodes (in addition to hypothesizing edges between nodes) and accept or reject it within the MCMC framework. This would fall under the class of problems where the dimensionality of the state changes across iterations, and Reversible Jump MCMC (RJMCMC) has been very popular for this. Incorporating this would be a natural extension, and remains future work.

An issue that needs to be tackled is the problem of lattice vectors that loop back to rectangle within another rectangle. This situation can be seen in Fig. 5.18b. Overestimating rectangles sometimes result in similar but separate rectangles being hypothesized around the boundary of a window. While pairwise constraints can prevent neighbors from connecting to such overlapping rectangles, a higher-order constraint needs to be specified to prevent edges from looping back to a window that is already part of the lattice. Possible avenues to explore are heuristics during pruning or including a global prior to prevent this from happening.

### 7.3.4 Building Semantics

The different techniques presented in Chapter 6 primarily show how much information can be extracted from a single building image, starting simply from the grid assumption alone. More a proof-of-concept, they can be extended to extract more qualitative information.

**Irregular Window Spacing** Because of the low dimensional representation of the grid with just an offset and inter-tile spacings, our lattice completion results are not accurate when the window spacing is irregular such as that in Figure 6.2. One approach to identify windows even in these hard situations is to learn the Gaussian color representation of the window interiors and merge rectangular structures that approximate the regular grid alignment as much as possible.

**Subspace Reconstruction** While foreground pixels are erased fairly reliably, the PCA approach results in a loss of background detail. This is primarily because of the low number of bases being used; with a slightly more accurate initial segmentation of outliers and a more sophisticated model of intra-tile color and gradient consistency, more photo-realistic results can be achieved. However, we have also presented the alternative of exemplar-based inpainting, which particularly for bricks can be used to obtain a more realistic final image.

Currently, both our spectral and lattice discovery methods disallow some tile non-regularity in either spacing or size during the foreground removal process. This can result in misalignment artifacts in the scrubbed images. Implementing per-tile optimization of alignment with the image gradient, as well as dynamic programming for ragged tile edges after [40] might improve rendering quality.

## 7.4 Future Directions

**Appearance, Topology and Function** We have demonstrated how topological relationships such as grid structures can be used to reason about man-made structures, when visual evidence alone might be insufficient. Of late, the computer vision community has developed many advanced object recognition and detection algorithms. How do we combine these two paradigms of appearance and topological models to enhance detection rates? Further, can we also add functional semantics to the above two models? For example, the dictionary defines a chair as "a piece of furniture consisting of a seat, legs, back, and often arms, designed to accommodate one person". A functional definition might just be "something a person sits on". By this token, a rock could be classified as a chair. It is also by functional modeling, that a system such as ours should never detect a door anywhere other than on the first floor of a building facade.

**Closing the Loop** This thesis showed how different types of information such as building background, foreground objects, window structures, brick textures, etc. can be discovered without any supervised training procedures. Instead very simple implicit rules about architectural elements have been exploited. The issue of how this information extracted by the semantic module will be passed down to the robot and image processing modules is an open problem. What kind of algorithms and process architecture could allow feedback from the higher module that an image just acquired by the robot was occluded, lacked focus, or did not contain enough features that could be useful for modeling? How can the recognition module abort the structure-from-motion routine from extensively computing the 3D model of a staircase and simply synthesize a 3D staircase with the extracted semantic parameters?

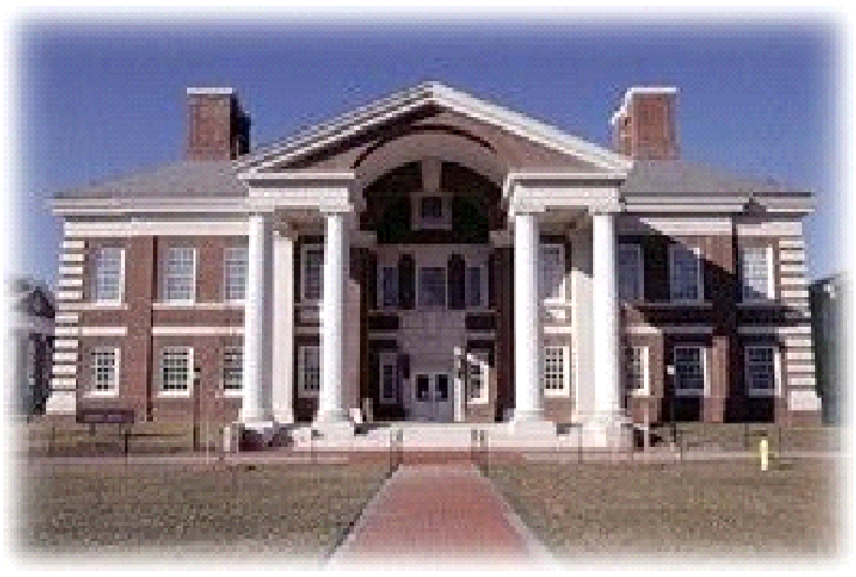


**Figure 7.1:** Can we make use of prior models of specific building elements to estimate structure?

**Urban Synthesis** Supposing we were given a single noisy image of a building such as shown in Fig. 7.1. Based on what we know, humans can conceptualize a rough 3D model of this facade with simplifying assumptions about the appearance of its various components. While one wouldn't want to exert such high demands on an algorithm, there are situations when a similar mode of reasoning could be useful. Suppose the recognition module is able to detect and localize some of the pixels as belonging to a flight of stairs. Would it be possible to compute *structure-from-analogy* as opposed to a full scale structure-from-motion procedure for recovering the shape of staircases? Based on some simple geometric measurements and appearance attributes recovered from the image, the algorithm could just as well synthesize a flight of stairs in the three-dimensional model, potentially eliminating many of the spurious jagged artifacts of Fig. 1.2. This technique could thus make use of prior models of specific building entities to ease the demands on structure estimation.

In contrast to urban modeling, can we design algorithms that can generalize about architectural characteristics from just a few images. Consider a few tens of images captured haphazardly around a city or campus. Would it be possible for an algorithm to recognize the architectural styles, the building pattern, the general textures, and overall design philosophy of the cityscape? If so, can a whole city that follows the same general traits captured by the images be synthesized, without explicit structure recovery? One may not be able to avoid structure completely, as various ratio and area constraints might need to be gleaned from the images. Nevertheless, such an idea would be very popular in gaming or virtual world scenarios.

Another possibility for synthesis would be to change the style of a building (say from Victorian to Gothic) without altering the structure. One can manipulate the pixels to introduce such realistic artifacts. Window shutters could be at different heights, curtains may be altered, or the brick texture could be replaced with stone. By extracting as much semantic information as possible, we can makes this a reasonably achievable target.

**Drawing the Line** One can also ask at what point the robot throws its hands up in the air and gives up trying to understand an image. An ivy-covered building is shown in Figure 7.2. Any trained appearance model would classify most of the facade as plant texture. The windows by themselves have very little to suggest that they are indeed part of a building. Though seemingly wild, humans are adept at detecting traces of man-made structures that allow us to surmise that this is a building in an urban environment and not wild vegetation. More than any single trait, it is the combination of glass-like grid structures, the arched gate, relaxed people at ground level, lampshade etc. that abets this. Can we teach a robot to perform such contextual reasoning from multiple cues even when each cue in isolation has very little evidence for support?

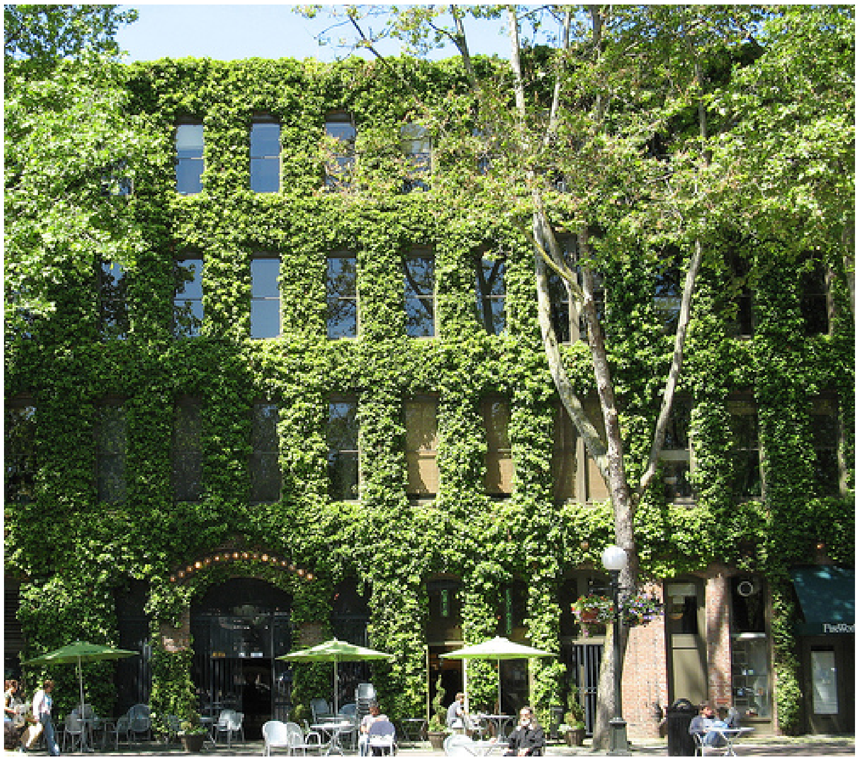


**Figure 7.2:** A challenging ivy covered building for semantic inference.

# BIBLIOGRAPHY

[1] A. Aboshosha and A. Zell. Robust mapping and path planning for indoor robots based on sensor integration of sonar and a 2d laser range finder. In *IEEE Int. Conf. on Intelligent Engineering Systems*, 2003.

[2] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Transactions on Graphics*, 23(3):294–302, 2004.

[3] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David H. Salesin, and Richard Szeliski. Panoramic video textures. *ACM Transactions on Graphics*, 24(3):821–827, August 2005.

[4] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers, Principles, Techniques, and Tools.* Addison-Wesley, 1986.

[5] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 1–8, Washington, DC, USA, 2006. IEEE Computer Society.

[6] Fernando Alegre and Frank Dellaert. A probabilistic approach to the semantic interpretation of building facades. In *International Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*, 2004.

[7] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. Avenue: Automated site modeling in urban environments. In *Third Intl. Conf. on 3-D Digital Imaging and Modeling (3DIM '01)*, volume 00, page 357, Los Alamitos, CA, USA, 2001.

[8] P. Allen, I. Stamos, A. Troccoli, B. Smith, M. Leordeanu, and Y. Hsu. 3-D modeling of historic sites using range and image data. In *Proc. Int. Conf. Robotics and Automation*, 2003.

[9] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, 2003.

[10] Michael Ashikhmin. Synthesizing natural textures. In *ACM Symposium on Interactive 3D Graphics*, 2001.

[11] Adrian Barbu and Song-Chun Zhu. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27:1239–1253, 2005.

[12] Daniel Bekins and Daniel G. Aliaga. Build-by-number: Rearranging the real world to visualize novel architectural spaces. *IEEE Visualization*, 00:19, 2005.

[13] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 26–33, 2005.

[14] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH*, pages 417–424, 2000.

[15] J. Borenstein, H.R. Everett, and L. Feng. Where am i? sensors and methods for mobile robot positioning. Technical report, The University of Michigan UM-MEAM-94-21, 1994.

[16] C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. Available from http://www.ece.purdue.edu/˜bouman, April 1997.

[17] C. Brenner, N Haala, and D. Fritsch. Towards fully automated 3d city model generation. In *Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images III*, 2001.

[18] D. Capel and A. Zisserman. Computer vision applied to super resolution. *IEEE Signal Processing Magazine*, 20:75–86, 2003.

[19] D. P. Capel. *Image Mosaicing and Super-resolution.* Ph.D. dissertation, University of Oxford, 2001.

[20] M. Ceccarelli and G. Antoniol. A deformable grid matching approach for microarray images. *IEEE Trans. Image Processing*, 15(10):3178 – 3188, 2006.

[21] Tony Chan and Jianhong Shen. *Image Processing And Analysis: Variational, Pde, Wavelet, And Stochastic Methods.* Society for Industrial and Applied Mathematics, 2005.

[22] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool. 3d city modeling using cognitive loops. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 2006.

[23] Nico Cornelis, Kurt Cornelis, and Luc Van Gool. Fast compact city modeling for navigation pre-visualization. In *CVPR '06: Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[24] H. S. M. Coxeter. *Introduction to Geometry.* Wiley; 2 edition, 1989.

[25] D. Crandall and D. Huttenlocher. Composite models of objects and scenes for category recognition. In *IEEE Computer Vision and Pattern Recognition*, 2007.

[26] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2006.

[27] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing*, 13(9), 2004.

[28] Antonio Criminisi. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images.* Ph.D. dissertation, University of Oxford, Dept. Engineering Science, Dec. 1999. D.Phil. thesis.

[29] T. Danner and L. Kavraki. Randomized planning for short inspection paths. In *Proc. Int. Conf. Robotics and Automation*, 2000.

[30] J. Davis. Mosaics of scenes with moving objects. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 1998.

[31] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs. In *SIGGRAPH*, 1996.

[32] Paul E. Debevec. *Modeling and Rendering Architecture from Photographs.* Ph.D. dissertation, University of California at Berkeley, Computer Science Division, Berkeley CA, 1996.

[33] Defense Advanced Research Projects Agency (DARPA). DARPA Grand Challenge. Available at `http://www.darpa.mil/grandchallenge`. Accessed July 22, 2003.

[34] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the Condensation algorithm for robust, vision-based mobile robot localization. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, pages 588–594, 1999.

[35] A. Dick, P. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *Int. J. Computer Vision*, 60(2), November 2004.

[36] Anthony Dick. *Modelling and Interpretation of Architecture from Several Images.* Ph.D. dissertation, University of Cambridge, 2001.

[37] Gyuri Dorkó and Cordelia Schmid. Object class recognition using discriminative local features. Rapport de recherche RR-5497, INRIA - Rhone-Alpes, February 2005.

[38] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. Fragment-based image completion. *ACM Transactions on Graphics*, 22(3):303–312, 2003.

[39] R. Duda, P. Hart, and D. Stork. *Pattern Classification, 2nd ed.* John Wiley and Sons, 2001.

[40] A. Efros and W. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001.

[41] A.F. Elaksher and J.S. Bethel. Reconstructing 3d buildings from lidar data. In *Photogrammetric Computer Vision (PCV02)*, page A: 102, 2002.

[42] D. Farin, P. de With, and W. Effelsberg. Robust background estimation for complex video sequences. In *Proceedings of the IEEE International Conference on Image Processing*, 1997.

[43] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE Computer Vision and Pattern Recognition*, 2004.

[44] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, October 2005.

[45] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[46] A. W. Fitzgibbon and A. Zisserman. Automatic 3D model acquisition and generation of new images from video sequences. In *Proceedings of European Signal Processing Conference (EUSIPCO '98), Rhodes, Greece*, pages 1261–1269, 1998.

[47] C. Frueh and A. Zakhor. 3-D model generation for cities using aerial photographs and ground level laser scans. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2001.

[48] C. Frueh and A. Zakhor. Constructing 3-D city models by merging ground-based and airborne views. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2003.

[49] Christian Früh, Siddharth Jain, and Avideh Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *Int. J. Comput. Vision*, 61(2):159–184, 2005.

[50] Christian Früh and Avideh Zakhor. An automated method for large-scale, ground-based city model acquisition. *Int. J. Comput. Vision*, 60(1):5–24, 2004.

[51] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. Improving feature tracking with robust statistics. In *Pattern Analysis and Applications*, 1999.

[52] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[53] Atanas Georgiev and Peter K. Allen. Vision for mobile robot localization in urban environments. In *Int. Conf. Intelligent Robots and Sytems (IROS)*, 2002.

[54] Atanas Georgiev and Peter K. Allen. Vision for mobile robot localization in urban environments. In *Proc. of IEEE Int. Conference on Intelligent Robots and Systems, Lausanne, Switzerland*, pages 472–477, October 2002.

[55] H. Gonzalez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th ACM Symp. on Computational Geometry*, 2001.

[56] Peter. J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

[57] Branko Grunbaum and Geoffrey Shephard. *Tilings and Patterns.* W.H. Freeman & Co., 1987.

[58] L. Hamey and T. Kanade. Computer analysis of regular repetitive textures. In *DARPA Image Understanding Workshop*, 1989.

[59] Feng Han and Song-Chun Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In *Proc. of the IEEE International Conference on Computer Vision (ICCV05)*, 2005.

[60] G. M. Hans. Fast determination of parametric house models from dense airborne laser scanner data. *Int'l Archives Photogrammetry and Remote Sensing (IAPRS)*, 32, part 2W1:1–6, 1999.

[61] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt. Real-time scene stabilization and mosaic construction. In *DARPA Image Understanding Workshop*, 1994.

[62] Karsten Hartelius and Jens Michael Carstensen. Bayesian grid matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):162–173, 2003.

[63] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

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

[65] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[66] James H. Hays, Marius Leordeanu, Alexei A. Efros, and Yanxi Liu. Discovering texture regularity as a higher-order correspondence problem. In *9th European Conference on Computer Vision*, May 2006.

[67] Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986. revised from Graphics Interface '86 version.

[68] J. Hoschek and D. Lasser. *Fundamentals of Computer-Aided Geometric Design.* A.K. Peters, 1993.

[69] Jinhui Hu, Suya You, and Ulrich Neumann. Approaches to large-scale urban modeling. *IEEE Computer Graphics and Applications*, 23:62–69, 2003.

[70] Terrance L. Huntsberger, Hrand Aghazarian, Yang Cheng, Eric T. Baumgartner, Edward Tunstel, Chris Leger, Ashitey Trebi-Ollennu, and Paul S. Schenker. Rover autonomy for long range navigation and science data acquisition on planetary surfaces. In *IEEE Intl. Conf. on Robotics and Automation*, pages 3161–3168, 2002.

[71] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the European Conference on Computer Vision*, pages 343–356, 1996.

[72] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29:5–28, 1998.

[73] M. Isard and A. Blake. A mixed-state Condensation tracker with automatic model-switching. In *Proc. Int. Conf. Computer Vision*, pages 107–112, 1998.

[74] J. Jia, T. Wu, Y. Tai, and C. Tang. Video repairing: Inference of foreground and background under severe occlusion. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2004.

[75] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999.

[76] N. Jojic and B. Frey. Learning flexible sprites in video layers. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2001.

[77] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, 1960.

[78] Y. Ke and R. Suthanker. A more distinctive representation for local image descriptors. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2004.

[79] Zia Khan, Tucker Balch, and Frank Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence*, 27(11):1805–1918, November 2005.

[80] Seon Joo Kim and Marc Pollefeys. Radiometric alignment of image sequences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004.

[81] A. Kokaram, B. Collis, and S. Robinson. A Bayesian framework for recursive object removal in movie post-production. In *Proceedings of the IEEE International Conference on Image Processing*, 2003.

[82] A.C. Kokaram, B. Collis, and S. Robinson. Automated rig removal with bayesian motion interpolation. *IEEE Proc. - Vision, Image and Signal Processing*, 152(4):407–414, August 2005.

[83] Anil Kokaram. Practical, unified, motion and missing data treatment in degraded video. *J. Math. Imaging Vis.*, 20(1-2):163–177, 2004.

[84] Nikos Komodakis and Georgios Tziritas. Image completion using global optimization. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[85] T. Korah and C. Rasmussen. PCA-based recognition for efficient inpainting. In *Proc. Asian Conf. Computer Vision*, 2006.

[86] Thommen Korah and Christopher Rasmussen. Improving spatiotemporal inpainting with layer appearance models. In *International Symposium on Visual Computing*, 2006.

[87] S. Kumar, M. Biswas, and T. Nguyen. Spatio-temporal texture synthesis and image inpainting for video applications. In *Proceedings of the IEEE International Conference on Image Processing*, 2005.

[88] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH*, August 2005.

[89] Vivek Kwatra, Arno Schdl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH*, 22(3):277–286, July 2003.

[90] F. De la Torre and M. Black. A framework for robust subspace learning. *Int. J. Computer Vision*, 54 (Aug 2003):117–142, 2003.

[91] S. Lee and Y. Liu. Psu Near-Regular Texture Database. Available at `http://vivid.cse.psu.edu/texturedb/gallery/`. Accessed November, 2006.

[92] Sung Chun Lee and Ram Nevatia. Extraction and integration of window in a 3d building model from ground view images. *Computer Vision and Pattern Recognition*, 02:113–120, 2004.

[93] T. Lee. Image representation using 2D Gabor wavelets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

[94] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robotics and Automation*, 7(3):376–382, 1991.

[95] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, volume 2, pages 1482 – 1489, October 2005.

[96] Thomas K. Leung and Jitendra Malik. Detecting, localizing and grouping repeated scene elements from an image. In *Proc. of European Conference on Computer Vision (ECCV)*, 1996.

[97] Thomas K. Leung and Jitendra Malik. Recognizing surfaces using three-dimensional textons. In *International Conference on Computer Vision (ICCV)*, 1999.

[98] S. Z. Li. *Markov random field modeling in computer vision.* Springer-Verlag, 1995.

[99] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20:127–150, 2001.

[100] W. Lin and Y. Liu. Tracking dynamic near-regular textures under occlusions and rapid movements. In *Proceedings of the European Conference on Computer Vision*, 2006.

[101] Wen-Chieh Lin and Yanxi Liu. Tracking dynamic near-regular textures under occlusion and rapid movements. In *9th European Conference on Computer Vision*, 2006.

[102] Lingyun Liu and Ioannis Stamos. Automatic 3d to 2d registration for the photorealistic rendering of urban scenes. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[103] Lingyun Liu, Ioannis Stamos, Gene Yu, George Wolberg, and Siavash Zokai. Multiview geometry for texture mapping 2d images onto 3d range data. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2293–2300, 2006.

[104] Yanxi Liu, Robert Collins, and Yanghai Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):354 – 371, March 2004.

[105] Yanxi Liu, Wen-Chieh Lin, and James H. Hays. Near regular texture analysis and manipulation. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3):368 – 376, August 2004.

[106] Yanxi Liu, Yanghai Tsin, and Wen-Chieh Lin. The promise and perils of near-regular texture [publication grayscale version]. *International Journal of Computer Vision*, 62(1-2):145 – 159, April 2005.

[107] Le Lu, Kentaro Toyama, and Gregory D. Hager. A two level approach for scene recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[108] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, 2002.

[109] T. Matsuyama, S. Miura, and M. Nagao. A structural analysis of natural textures by fourier transformation. In *Proceedings of the International Conference on Pattern Recognition*, 1982.

[110] H. Mayer and S. Reznik. Building faade interpretation from image sequences. In *Proc. of the ISPRS Workshop CMRT 2005 - Object Extraction for 3D City Models, Road Databases and Traffic Monitoring - Concepts, Algorithms and Evaluation*, 2005.

[111] H. Mayer and S. Reznik. Mcmc linked with implicit shape models and plane sweeping for 3d building facade interpretation in image sequences. In *Photogrammetric Computer Vision (PCV06)*, 2006.

[112] Pragyana Mishra. *Image and Depth Coherent Surface Description*. Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2005.

[113] M. Morgan and K. Tempeli. Automatic building extraction from airborne laser scanning data. In *19th Int'l Soc. Photogrammetry and Remote Sensing Congress (ISPRS)*, 2000.

[114] Pascal Mller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Transactions on Graphics (SIGGRAPH '06)*, 25(3):614–623, 2006.

[115] Pascal Mller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. In *Proceedings of ACM SIGGRAPH 2007*, New York, NY, USA, 2007. ACM Press.

[116] Maan E. El Najjar and Philippe Bonnifait. A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Autonomous Robots*, 19:173–191, 2005.

[117] R. M. Neal. Probabilistic inference using markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[118] H. Norbert and B. Claus. Generation of 3d city models from airborne laser scanning data. In *3rd European Assoc. Remote Sensing Laboratories (EARSEL) Workshop Lidar Remote Sensing of Land and Sea, A.A. Balkema*, 1997.

[119] F. Odone, A. Fusiello, and E. Trucco. Layered representation of a video shot with mosaicing. *Pattern Analysis and Applications*, 5:296–305, 2002.

[120] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[121] K. Patwardhan, G. Sapiro, and M. Bertalmo. Video inpainting of occluding and occluded objects. In *IEEE International Conference on Image Processing (IEEE-ICIP)*, 2005.

[122] P. Perez. Markov random fields and images. Technical Report 1196, Irisa, July 1998.

[123] P. Perez, A. Blake, and M. Gangnet. Jetstream: Probabilistic contour extraction with particles. In *Proc. Int. Conf. Computer Vision*, pages 524–531, 2001.

[124] P. Perez, M. Gangnet, , and A. Blake. Patchworks: Example-based region tiling for image editing. Technical report, Microsoft Research Report TR-2004-04, 2004.

[125] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on Graphics (SIGGRAPH'03)*, pages 313–318, 2003.

[126] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. Computer Vision*, 59(3):207–232, 2004.

[127] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 90, Washington, DC, USA, 1998. IEEE Computer Society.

[128] Kari Pulli, Habib Abi-Rached, Tom Duchamp, Linda G. Shapiro, and Werner Stuetzle. Acquisition and visualization of colored 3d objects. In *International Conference on Pattern Recognition (ICPR '98)*, 1998.

[129] C. Rasmussen. Grouping dominant orientations for ill-structured road following. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2004.

[130] C. Rasmussen and T. Korah. On-vehicle and aerial texture analysis for vision-based desert road following. In *IEEE International Workshop on Machine Vision for Intelligent Vehicles*, 2005.

[131] C. Rasmussen and T. Korah. Spatiotemporal inpainting for recovering texture maps of partially occluded building facades. In *IEEE Int. Conf. on Image Processing*, 2005.

[132] Realviz. *Realviz ImageModeler V4.0 product information. http://www.realviz.com/*.

[133] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, volume 1, pages 10–17, 2003.

[134] Stephan A Roth, Bradley Hamner, Sanjiv Singh, and Myung Hwangbo. Results in combined route traversal and collision avoidance. In *International Conference on Field & Service Robotics (FSR '05)*, July 2005.

[135] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[136] S. Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems*, 1997.

[137] F. Schaffalitzky and A. Zisserman. Geometric grouping of repeated elements within images. In *Proceedings of the 9th British Machine Vision Conference, Southampton*, 1998.

[138] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[139] V. Sequeira and J. Goncalves. 3d reality modelling: photo-realistic 3d models of real world scenes. In *3D Data Processing Visualization and Transmission*, pages 776–783, 2002.

[140] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 1994.

[141] David Silver, Boris Sofman, Nicolas Vandapel, James Bagnell, and Anthony (Tony) Stentz. Experimental analysis of overhead data processing to support long range navigation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2443 – 2450, October 2006.

[142] Sanjiv Singh, Reid Simmons, Trey Smith, Anthony (Tony) Stentz, Vandi Verma, Alex Yahja, and Kurt Schwehr. Recent progress in local and global traversability for planetary rovers. In *Proceedings of the IEEE International Conference on Robotics and Automation, 2000*. IEEE, April 2000.

[143] S. Soatto, G. Doretto, and Y. N. Wu. Dynamic textures. In *Proceedings of the International Conference on Computer Vision*, volume 2, 2001.

[144] Kevin Sookocheff and David Mould. One-click lattice extraction from near-regular texture. In *GRAPHITE '05: Proc. of Intl. Conf. on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 265–268, New York, NY, USA, 2005. ACM Press.

[145] J. Spletzer and C. Taylor. A framework for sensor planning and control with applications to vision guided multi-robot systems. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2001.

[146] Ioannis Stamos. *Geometry and Texture Recovery of Scenes of Large Scale: Intergration of Range and Intensity Sensing.* Ph.D. dissertation, Columbia University, 2001.

[147] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, 2000.

[148] J. Sun, W. Zhang, X. Tang, and H. Shum. Background cut. In *Proceedings of the European Conference on Computer Vision*, 2006.

[149] Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. Image completion with structure propagation. *ACM Transactions on Graphics*, 24:861–868, 2005.

[150] Y. Sun, J. K. Paik, A. Koschan, and M. A. Abidi. 3d reconstruction of indoor and outdoor scenes using a mobile range scanner. In *International Conference on Pattern Recognition (ICPR'02)*, 2002.

[151] S. Syed and M. E. Cannon. A fuzzy logic-based map matching algorithm for vehicle navigation systems in urban canyons. In *National Technical Meeting of The Institute of Navigation, San Diego*, 2004.

[152] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, 1996.

[153] Franck Taillandier. Texture and relief estimation from multiple georeferenced images. Master's thesis, DEA Algorithmique, Ecole Polytechnique, Paris 6&7, ENS-Cachan, and ENS-Ulm, 2000.

[154] Ashit Talukder and David P. Casasent. Multiscale gabor wavelet fusion for edge detection in microscopy images. In Harold H. Szu, editor, *Wavelet Applications V*, volume 3391, pages 336–347. SPIE, 1998.

[155] Hai Tao, Harpreet S. Sawhney, and Rakesh Kumar. A global matching framework for stereo computation. In *International Conference on Computer Vision (ICCV01)*, 2001.

[156] S. Teller. Scalable, controlled image capture in urban environments. Technical Report MIT LCS Technical Report 825, Massachusetts Institute of Technology, 2001.

[157] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master. Calibrated, registered images of an extended urban area. *Int. J. Computer Vision*, 53:93–107, 2003.

[158] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

[159] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto. Making good features to track better. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, pages 178–183, 1998.

[160] Paul Tompkins. *Mission-Directed Path Planning for Planetary Rover Exploration.* Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2005.

[161] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers:. Principles and practice of background maintenance. In *Proc. Int. Conf. Computer Vision*, 1999.

[162] A. Turina, T. Tuytelaars, and L. Van Gool. Efficient grouping under perspective skew. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2001.

[163] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 1991.

[164] F. van den Heuvel. *Automation in Architectural Photogrammetry; Line-Photogrammetry for the Reconstruction from Single and Multiple Images.* Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, 2003.

[165] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision: Special Issue on Texture Analysis and Synthesis*, 62(1–2):61–81, April 2005.

[166] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Trans. Image Processing*, 3(5):625–638, 1994.

[167] Jingbin Wang, Erdan Gu, and Margrit Betke. Mosaicshape: Stochastic region grouping with shape prior. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[168] T. Werner and A. Zisserman. Model selection for automated architectural reconstruction from multiple views. In *Proceedings of the British Machine Vision Conference*, 2002.

[169] T. Werner and A. Zisserman. New techniques for automated architecture reconstruction from photographs. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 2, pages 541–555, 2002.

[170] Y. Wexler, A. Fitzgibbon, and A. Zisserman. Bayesian estimation of layers from multiple images. In *Proceedings of the European Conference on Computer Vision*, 2002.

[171] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2004.

[172] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 2*, 2005.

[173] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Transactions on Graphics*, 22:669–677, 2003.

[174] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cut. In *Proceedings of the National Conference on Computer Vision and Pattern Recognition*, 2004.

[175] L. Xu and A. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Transactions on Neural Networks*, 6(1):131 – 143, 1995.

[176] Liron Yatziv, Guillermo Sapiro, and Marc Levoy. Lightfield completion. In *International Conference on Image Processing*, pages 1787–1790, 2004.

[177] Suya You, Jinhui Hu, Ulrich Neumann, and Pamela Fox. Urban site modeling from lidar. In *Computational Science and Its Applications ICCSA03*, 2003.

[178] A. Yuille and J. Coughlan. Fundamental limits of Bayesian inference: Order parameters and phase transitions for road tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(2):160–173, 2000.

[179] W. Zhang and J. Kosecka. Extraction, matching and pose recovery based on dominant rectangular structures. In *High Level Knowledge in Vision Workshop, ICCV*, 2003.

[180] B. Zhao and J. Trinder. Integrated approach based automatic building extraction. In *19th Int'l Soc. Photogrammetry and Remote Sensing Congress (ISPRS)*, 2000.

[181] H. Zhao and R. Shibasaki. Reconstructing a textured cad model of an urban environment using vehicle-borne laser range scanners and line cameras. *Machine Vision and Applications*, 14(1):35–41, 2003.

[182] Wenyi Zhao, David Nister, and Steve Hsu. Alignment of continuous video onto 3d point clouds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1305–1318, 2005.