Understanding and Recreating Visual Appearance Under Natural Illumination

Jean-François Lalonde

CMU-RI-TR-11-01

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January 2011

Acknowledgements

I first want to thank my two advisors, Alyosha and Srinivas, for always providing me with the right balance of guidance and freedom. The seemingly unlikely mix of both your "data-driven" and "physically-based" backgrounds ended up being a very productive combination: our fruitful discussions always led to way more ideas than I ever could try! Thank you for your dedication, mentorship, and for always challenging me to go beyond.

I also thank the numerous friends, colleagues, and officemates I had over the years at CMU. I consider myself lucky to have gotten to know and work with David Bradley, Derek Hoiem, Ranjith Unnikrishnan, Hongwen Kang, Minh Hoai Nguyen, Andrew Stein, Tom Stepleton, Santosh Divvala, Abhinav Gupta, and Tomasz Malisiewicz. I also had a great time hanging out in the graphics lab with my friends Jim McCann, Ronit Slyper, Chris Twigg, Jernej Barbič, James Hays, Eakta Jain, and Laura Trutoiu.

A special thanks goes to my good friend Chris Bartley for our joint "crushing of the competition" during our very first semester at CMU. Thanks also to R. Coulter, for your advice, friendship, and for so generously welcoming us into your family.

I thank my parents, Hélène and Gilles, for always encouraging me to follow my dreams and my sisters Catherine and Geneviève for your love and support. And last but certainly not least, I thank my wife Barbara and daughter Héloïse: I couldn't have done it without your unconditional love.

Credits This work has been partially supported by a Microsoft Research Graduate Fellowship, an Okawa Foundation Grant, ONR grants N00014-08-1-0330 and DURIP N00014-06-1-0762, as well as NSF IIS-0643628, IIS-0546547, CCF-0541307, and CCF-0541230. We would like to thank Andrew Stein, Derek Hoiem, and Olga Veksler for making their code available online, Nathan Jacobs for sharing his webcam dataset and for fruitful discussions, Danielle Millett for her help with the single image relighting example (Fig. 4.11), Yue (Annie) Zhao of the HCI Institute at CMU for helping us design and implement the GUI (Fig. 7.5), and Bryan Russell, Antonio Torralba and the rest of the LabelMe team for setting up and maintaining the database. We thank our photo contributors: Antonio Torralba, Robert Kitay, and the Flickr users who placed their work under Creative Commons License: Dey Alexander, M. Caimary, Dan Johansson, and `slightly-less-random`. We also thank Chris Cook (`www.chriscookartist.com`) for letting us use his painting *Blue Liquor Store* in Fig. 7.9. And special thanks to Henry Rowley of Google Inc. for finding our needle in their haystack.

To Barbara and Héloïse

Abstract

The appearance of an outdoor scene is determined to a great extent by the prevailing illumination conditions. However, most practical computer vision applications treat illumination more as a nuisance rather than a source of signal. In this dissertation, we suggest that we should instead *embrace* illumination, even in the challenging, uncontrolled world of consumer photographs.

Our first main contribution is an *understanding* of natural illumination from images. This is, in general, a hard problem given the wide appearance variation in scenes. Fortunately, natural illumination, while complex, is far from being completely arbitrary. It has a structure that is well understood in atmospheric optics, but which has hardly been exploited in vision and graphics. We introduce methods for automatically estimating the illumination conditions from two types of uncontrolled outdoor image datasets: webcams and single images. The variation in sun position and sky appearance over time can be exploited to obtain viewing and illumination geometry in webcam sequences. For single images, the sky is combined in a probabilistic way with other scene features such as cast shadows and shading on vertical surfaces and convex objects, as well as with illumination priors from large image collections.

Our second main contribution is to exploit the knowledge of illumination in order to *synthesize* novel, realistic visual content. Instead of creating appearance using the traditional computer graphics pipeline, we propose to *borrow* the appearance of the world that is contained in existing photo collections and webcam datasets. We also demonstrate realistic 3-D object insertion by creating plausible high-dynamic range environment maps. This can be done in image sequences, and even in single images, completely automatically. Addressing such questions has implications in a broad range of applications including intelligent transportation, surveillance, human-robot interaction, and digital entertainment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The journey of a thousand leagues begins with a single step.

Lao Tzu (600–531 BC)

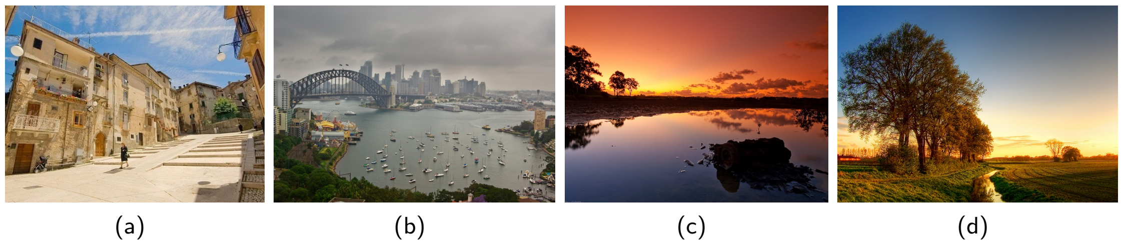


Figure 1.1: Natural illumination creates beautiful effects, much coveted by photographers. Whether it is sunny (a), overcast (b), at sunset (c) or at the golden hour (d), natural illumination creates challenging cases for computer vision and graphics researchers alike.

The appearance of an outdoor scene is determined to a great extent by the prevailing illumination conditions, as illustrated in Fig. 1.1. Is it sunny or overcast, morning or noon, clear or hazy? Claude Monet, a fastidious student of light, observed: "A landscape does not exist in its own right . . . but the surrounding atmosphere brings it to life . . . For me, it is only the surrounding atmosphere which gives subjects their true value." Within the Grand Vision Problem, illumination is one of the key variables that must be untangled in order to get from pixels to image understanding. But while a lot of work has been done on modeling and using illumination in a laboratory setting, relatively little is known about it "in the wild", i.e., in a typical outdoor scene. In fact, most practical vision applications treat illumination more as a nuisance—something that one strives to be invariant to—rather than a source of signal. Examples include illumination adaptation in tracking and surveillance (e.g., [Stauffer, 1999]), or contrast normalization schemes in popular object detectors (e.g., [Dalal and Triggs, 2005]). Alas, the search for the ultimate illumination invariant might be in vain [Chen *et al.*, 2000].

In this dissertation, we suggest that we should instead *embrace* illumination, even in the challenging, uncontrolled world of consumer photographs. Our first main contribution[1] is:

> ***Understanding* of natural illumination** We propose algorithms to automatically estimate the illumination conditions in 1) time-lapse videos captured by static cameras, and 2) single images. We capture the position of the sun with respect to the camera, sun visibility, and sky color.

Understanding illumination from images is a hard problem which has received significant attention [Ramamoorthi and Hanrahan, 2001; Tappen *et al.*, 2005]. What makes it especially hard is that light undergoes a series of complex interactions with the scene, which depend on its geometry and reflectance properties, before being captured by a camera to form an image. Fortunately, natural illumination, while complex, is far from being completely arbitrary. It has a structure that is well understood in atmospheric optics [Judd *et al.*, 1964; Perez *et al.*, 1993], but which has hardly been used in vision and graphics (notable exceptions being [Sato and Ikeuchi, 1995; Sato *et al.*, 2003; Preetham *et al.*, 1999]). Here we propose to take advantage of its structure, and use the sky appearance as well as other cues such as shadows and shading as sources of information.

This knowledge of illumination in images also enables a new class of computer graphics applications. One of the ultimate goals of computer graphics is to make the creation and manipulation of novel, photo-realistic imagery simple and intuitive to the casual user. Despite the recent advances, visual realism in complex, real-world scenes remains largely elusive for traditional computer graphics. The main culprit appears to be the sheer com-

[1]Originally introduced in [Lalonde *et al.*, 2008b; Lalonde *et al.*, 2009a; Lalonde *et al.*, 2010a; Lalonde *et al.*, 2010c]

plexity of our visual experience—a realistic scene requires much more detailed geometry and lighting than could possibly be provided by hand.

In this thesis, we propose an alternative approach to traditional computer graphics techniques for creating realistic, lighting-consistent visual content—by leveraging the enormous amount of photographs that have *already been captured*, and analyzing them using our novel illumination estimation techniques. We now state our second main contribution[2]:

> **Data-driven, illumination-consistent *synthesis* of novel visual content** Instead of creating appearance with the usual computer graphics pipeline, we propose to *borrow* the appearance of the world that is contained in existing pictures, while keeping the illumination conditions consistent. Whereas image-based rendering techniques use photographs of a specific, confined space, our ultimate goal is to use unconstrained photographs of the entire world.

In "traditional" computer graphics, the main problem is not the lack of good algorithms—recent advances in global illumination, material modeling, rendering and lighting have allowed for synthesis of beautifully realistic and detailed imagery. Instead, the biggest obstacle is the sheer richness and complexity of our visual world. Every object to be rendered requires painstaking work by a skilled artist to specify detailed geometry and surface properties. And while there are a few databases of pre-made object models, typically the number of objects is quite small and the quality is far from photo-realistic. Various image-based approaches can also be used to acquire object models, but the data acquisition process is not straightforward and is typically not suitable for relatively large, outdoor, potentially moving objects (cars, pedestrians, etc). While it is possible to add synthetic objects into real scenes [Debevec, 1998], this requires estimating camera orientation and capturing environment maps—tasks too difficult for a casual user.

Instead of restricting ourselves to small datasets, what about exploiting *all* the available visual data in the world, no matter where it was captured? to build a universal *photo clip art* library that can be used for manipulating visual content "on the fly", in an illumination-consistent fashion? While this is a very ambitious goal, the recent emergence

[2]Originally introduced in [Lalonde *et al.*, 2007; Lalonde and Efros, 2007; Lalonde *et al.*, 2009b]

of large, peer-labeled object datasets [Russell *et al.*, 2008; von Ahn *et al.*, 2006], advances in geometric scene interpretation [Hoiem *et al.*, 2005; Hoiem *et al.*, 2006], as well as our novel understanding of illumination in images suggest that efforts in this direction are very timely.

1.1 Our Approach

Our ultimate goals are to 1) estimate natural illumination; and 2) synthesize realistic content, both from *single* images captured in the *real world*, so we need to find ways to deal with the inherent uncertainty in the data. This uncertainty may come from multiple sources: the geometric and radiometric properties of the camera (orientation, zoom, response function, dynamic range, etc.), the scene structure (scene geometry, material properties, object identities, depth discontinuities, occlusions, etc.) and the illumination conditions (sun position, weather, etc.) all interact in highly complex ways to form an image. We now present the key ideas which allows us to deal with such ill-posed problems.

Our first key idea is to constrain the problem by analyzing the case of time-lapse image sequences, where the scene is kept constant while the illumination conditions vary over time. This allows us to de-couple the effects due to illumination from those due to scene changes. An unexpected result of this analysis reveals that the visible portion of the sky, when observed over time, is sufficient to recover the natural illumination conditions in each frame in the sequence. Our approach combines physically-based modeling with robust non-linear parameter fitting to deal with common sources of distortion such as vignetting, non-linear camera response functions, and sensor noise.

While useful to understand the sky, physical models are unfortunately of limited use when estimating illumination in single images. For example, existing physical models for detecting cast shadows—strong cues for illumination understanding—fail when applied to consumer-quality images. In this case, the models are not robust enough to compensate for the strong sources of distortion in the capture process. Other illumination cues such as shading on flat and convex objects also cannot be estimated using physics-based models because of the uncertainty in scene structure recovery. To compensate for these shortcomings, our second key idea is to use data-driven, machine learning methods to model the statistics of the world. For this, we introduce novel datasets of image sequences and single images, in which the illumination is manually (or automatically) labelled. We use these datasets to train classifiers on novel illumination-sensitive features, and show that their performance is surprisingly good on a wide range of challenging images.

These illumination estimation techniques can subsequently be used to *generate* novel, illumination-consistent images. Instead of trying to generate appearance from the ground up as in the traditional computer graphics pipeline, our third key idea is to borrow visual content from existing images, and transfer them onto other images. Our illumination understanding tools are used to create novel databases of single or time-lapse objects which are aware of their illumination conditions (and other parameters such as camera geometry, depth, etc.).

1.2 Overview

This thesis is divided into two main parts that mirror the natural progression from image sequences to single images. Each part follows the same structure: natural illumination is first estimated; then used to synthesize realistic, illumination-consistent images.

Part I considers the case of image sequences captured by static cameras. First, Ch. 3 shows that the natural illumination conditions can be estimated by analyzing the sky region only. Subsequently, Ch. 4 demonstrates how this knowledge of illumination can be used to synthesize new image sequences where illumination effects such as reflections, shading, and cast shadows can accurately be transferred and generated.

Part II then explores the case of a single image. First, because shadows are such an important cue in illumination understanding [Koenderink *et al.*, 2004], we present in Ch. 5 a method for automatically detecting shadow boundaries in single, consumer-quality photographs. Ch. 6 shows how to estimate the natural illumination conditions from a single image by combining several cues computed over different portions of the image. Ch. 7 then presents a system for creating novel, illumination-consistent images in a data-driven way. Ch. 8 concludes this part of the dissertation by presenting an analysis of the role of color in assessing the realism of image composites created by such a system.

Just before Part I commences, Ch. 2 presents an overview of the existing representations for natural illumination. We then contrast them with our own, which we will keep throughout this document.

1.3 Background

We first provide a short review of related work for the tasks of estimating illumination, and data-driven methods for image compositing. In addition to this section, each chapter will also provide supplementary relevant background references as needed.

1.3.1 Natural Illumination Estimation

The color and geometry of illuminants can be directly observed by placing probes, such as mirror spheres [Stumpfel *et al.*, 2004], color charts or integrating spheres, within the scene. But, alas, most of the photographs captured do not contain such probes and thus, we are forced to look for cues within the scene itself. There is a long and rich history in computer vision about understanding the illumination from images. We will briefly summarize relevant works here, but also talk about relevant references in each subsequent chapter of this thesis.

Color constancy These approaches strive to extract scene representations that are insensitive to the illuminant color. For this, several works either derive transformations between scene appearances under different source colors (e.g., [Finlayson *et al.*, 1993]), or transform images into different color spaces that are insensitive to illuminant colors (e.g., [Zickler *et al.*, 2006]). Our work focuses on a complementary representation of outdoor illumination (see Ch. 2).

Model-based reflectance and illumination estimation Several works estimate illumination (light direction and location), in conjunction with model-based estimation of object shape and reflectances (Lambertian, Dichromatic, Torrance-Sparrow), from one or more images of the scene [Sato *et al.*, 2003; Basri *et al.*, 2007]. Of particular interest, Sun *et* *al.* [2009] estimate illumination conditions in complex urban scenes by registering the photographs to 3-D models of the scene. Our work does not rely on specific reflectance models of outdoor surfaces or exact estimation of 3-D geometry.

Shadow extraction and analysis Many works detect and remove shadows using one or more images [Weiss, 2001; Finlayson *et al.*, 2002; Wu and Tang, 2005]. The extracted shadows have also been used to estimate the sun direction in constrained settings [Kim and Hong, 2005] or in webcams [Junejo and Foroosh, 2008]. But shadows are only weakly informative about illumination when their sizes in the image are too small or their shapes are complex or blurred. Our work, for the first time, combines such weak cues with other semi-informative cues to better estimate illumination from a single image.

Illumination estimation from time-lapse sequences Narasimhan *et al.* [2002] introduced a dataset of high quality registered and calibrated images of a fixed outdoor scene captured every hour for a year. By fixing the scene and viewing geometry, it is possible to analyze the effect of illumination, weather and aging conditions more directly. This idea has recently been exploited by several researchers. For example, Koppal and Narasimhan [2006] showed that scene points can be clustered according to their surface normals, irrespective of material and lighting, by observing their appearance over time, as illumination changes. On a global scale, Jacobs *et al.* [2007a] observed that the main variations in scene appearance are common across scenes by applying PCA on a large dataset of webcam sequences. Sunkavalli *et al.* [2008] have also demonstrated impressive color constancy results, by fitting an illumination model to such an image sequence.

Sun position analysis The sun position has been exploited mostly in the robotics community. Cozman and Krotkov [1995] extract sun altitudes from images and use them to estimate camera latitude and longitude. Trebi-Ollennu *et al.* [2001] describe a system that estimates camera orientation in a celestial coordinate system, that is used to infer the attitude of a planetary rover. Both these techniques yield precise estimates, but require expensive additional hardware (digital inclinometer [Cozman and Krotkov, 1995] and custom sun sensor [Trebi-Ollennu *et al.*, 2001]). In comparison, our method recovers the viewing geometry from the sun position annotated in images captured by any ordinary camera.

Sky appearance analysis The sky appearance has long been studied by physicists. One of the most popular physically-based sky model was introduced by Perez *et al.* [1993], and was built from measured sky luminances. This model has been used in graphics for relighting architectural models [Yu and Malik, 1998], and for developing an efficient sky rendering algorithm [Preetham *et al.*, 1999]. Alternatively, Stumpfel *et al.* [2004] proposed to capture the sky directly into an HDR environment map format, and used it for both rendering and relighting [Debevec *et al.*, 2004]. Surprisingly however, very little work has been done on extracting information from the visible sky. One notable exception is the work of Jacobs *et al.* [Jacobs *et al.*, 2008] where they use the Perez sky model to infer the camera azimuth by using a correlation-based approach. In our work, we address a broader question: what does the sky tell us about the camera? We show how we can recover three camera extrinsic and intrinsic geometric properties simultaneously, from the sun position and the sky appearance.

1.3.2 Data-driven Image Synthesis

A number of papers have dealt with issues involved in compositing an object into an image in a visually pleasing way [Porter and Duff, 1984; Pérez *et al.*, 2003; Jia *et al.*, 2006; Wang and Cohen, 2006]. However, they all assume that the user has already chosen the object that would be a good fit (geometrically and photometrically) for the target image. For instance, most blending techniques [Pérez *et al.*, 2003; Rother *et al.*, 2004; Li *et al.*, 2004; Jia *et al.*, 2006] rely on the assumption that the area surrounding the object in the source and target images are similar. Playing with these methods, one quickly realizes how very important the selection of a good source object is to the quality of the result and how difficult it is to get it right if one is not artistically gifted.

The work closest to our own is Semantic Photo Synthesis [Johnson *et al.*, 2006] (which itself has been inspired by [Diakopoulos *et al.*, 2004]). This paper develops a very ambitious approach for letting the user "design a new photograph" using textual and sketch queries into a database of real, automatically annotated images. Graphcut optimization [Boykov *et al.*, 2001] is used to stitch together parts of different retrieved photographs to get the final resulting image. While this work is very sophisticated technically and loaded with good, novel ideas, unfortunately, the visual results are not very compelling. We believe that the reasons for this are three-fold: 1) the synthesis process is done purely in the 2D image plane, without regard to scene geometry; 2) the image parts being stitched together are often not well defined, i.e., not corresponding to physical objects; 3) there is no attempt to match illumination conditions between image parts. Our present work addresses all of the above issues. We employ a more incremental approach to image synthesis. Instead of starting from scratch and hoping to get a photo-realistic picture in the end, we begin with an image that is already real and try to alter it in ways that change the content without breaking this "realness". Some surprisingly successful applications of this simple idea have since then [Lalonde *et al.*, 2007] been demonstrated [Hays and Efros, 2007; Bitouk *et al.*, 2008; Chen *et al.*, 2009]. Closely related are efforts that perform "appearance mining" of photo collections to recover visual content like virtual walk-throughs [Snavely *et al.*, 2008], skies [Tao *et al.*, 2009] and even 3D models [Goesele *et al.*, 2007].

1.4 Links to Online Material

To facilitate further progress in this area and encourage transparency in the research process, we provide MATLAB code, data, and demos for all of the main chapters of this thesis. Please refer to the URLs provided in Table 1.1.

Ch.	Content	URL
3	Code	`http://graphics.cs.cmu.edu/projects/sky`
4	Webcam dataset	`http://graphics.cs.cmu.edu/projects/webcamclipart`
5	Code, shadow dataset	`http://graphics.cs.cmu.edu/projects/shadows`
6	Code, dataset	`http://graphics.cs.cmu.edu/projects/outdoorillumination`
7	Live demo	`http://graphics.cs.cmu.edu/projects/photoclipart`
8	Color realism dataset	`http://graphics.cs.cmu.edu/projects/realismcolor`

Table 1.1: Links to online material available for each chapter of this thesis.

Chapter 2

Representations for Natural Illumination

And not by eastern windows only,
When daylight comes, comes in the light,
In front the sun climbs slow, how slowly,
But westward, look, the land is bright.

Arthur Hugh Clough (1819–1861)

Because of its predominant role in any outdoor setting, it has been of critical importance to understand natural illumination in many fields such as computer vision and graphics, but also in other research areas such as biology [Hill, 1994], architecture [Reinhart *et al.*, 2006], solar energy [Mills, 2004], and remote sensing [Healey and Slater, 1999]. Consequently, researchers have proposed many different representations for natural illumination adapted to their respective applications. In this chapter, we present an overview of popular representations divided into three main categories: "physically-based", "environment maps", and "statistical". We conclude this chapter by describing the representation that we use in this work.

2.1 Physically-based Representations

The type of representation that is probably the most popular in the literature is what we name here "physically-based" representations: mathematical expressions that model the physics of natural illumination. They typically stem from the following equation (adapted from [Slater and Healey, 1998]), which models the ground spectral irradiance $L(\lambda)$ as a function of the sun and sky radiances E_{sun} and E_{sky} respectively:

$$L(\lambda) = KE_{sun}(\theta_s, \phi_s, \lambda)\cos(\theta_s) + \int_{\phi=0}^{2\pi}\int_{\theta=0}^{\frac{\pi}{2}} E_{sky}(\theta, \phi, \lambda)\cos(\theta)\sin(\theta)d\theta d\phi\,, \tag{2.1}$$

where K is a binary constant which accounts for occluding bodies in the solar to surface path, and E_{sky} is integrated over the entire sky hemisphere. While (2.1) may look simple, its complexity lies in the characterization of the sun and sky radiance components which depend on the form and amount of atmospheric scattering. As a result, a wide variety of ways to model the sun and the sky have been proposed; we summarize a few here that are most relevant to our work.

Being the dominant light source outdoors, the sun has been studied extensively. For example, in architectural design, the relative position of the sun is an important factor in calculating the heat gain of buildings and in radiance computations that determine the quantity of natural light received inside each of the rooms [Ward, 1994]. Understanding the quantity of energy delivered by the sun at ground level is also critical to predict the quantity of electricity that can be produced by photovoltaic cells [Mills, 2004]. As such, highly precise physical models for sunlight transport through the atmosphere (E_{sun} in (2.1)) have been proposed [Bird, 1984; Kasten and Young, 1989].

The second main illuminant outdoors, the sky, has also long been studied by physicists. One of the first parametric models of the sky was introduced by Pokrowski [1929], based on theory and sky measurements. The model was improved by Kittler [1967], and subsequently adopted as a standard by the Commission Internationale de l'Éclairage (CIE) [CIE, 1994]. The CIE model has found applications in computer graphics [Ward, 1994]. One of the most popular physically-based sky model was introduced by Perez *et al.* [1993], and was built from measured sky luminances. It is a generalization of the CIE formula, and it has been found to be more accurate for a wider range of atmospheric conditions [Ineichen *et al.*, 1994]. This model has been used in graphics for relighting architectural models [Yu and Malik, 1998], and for developing an efficient sky rendering algorithm [Preetham *et al.*, 1999]. This is also the model we will ourselves use to understand the sky appearance (see Secs 2.4.2 and 3.3).

In computer vision, Sato and Ikeuchi [1995] use a model similar to (2.1) along with simple reflectance models to estimate the shape of objects lit by natural illumination. This has led to applications in outdoor color representation and classification [Buluswar and Draper, 2002], surveillance [Tsin *et al.*, 2001], and robotics [Manduchi, 2006]. A similar flavor of the same model has also been used in the color constancy domain by [Maxwell *et al.*, 2008]. Because physical models possess many parameters, recovering them from images is not an easy task. Therefore, researchers have resorted to simplifying assumptions, approximations, or to the use of image sequences [Sunkavalli *et al.*, 2008] to make the estimation problem more tractable.

2.2 Environment Map Representations

An alternative representation is based on the accurate measurement and direct storage of the quantity of natural light received at a point that comes in from all directions. As opposed to "physically-based" representations, here no compact formula is sought and all the information is stored explicitly. Originally introduced in the computer graphics community by Blinn and Newell [1976a] to relight shiny objects, the representation (also dubbed "light probe" in the community) was further developed by Debevec [Debevec, 1998; Debevec and Malik, 1997] to realistically insert virtual objects in real images. It is typically captured using a high-quality, high-dynamic range camera equipped with an omni-directional lens (or a spherical mirror), and requires precise calibration. Stumpfel *et al.* [2004] more recently proposed to employ this representation to capture the sky over an entire day into an environment map format, and used it for both rendering and relighting [Debevec *et al.*, 2004].

In comparison to its "physically-based" counterpart, the "environment map" representation has no parameters to estimate: it is simply measured directly. However, its main drawback—aside from large memory requirements—is that physical access to the scene of interest is necessary to capture it. Thus, it cannot be used on images which have already been captured.

2.3 Statistical Representations

The third type of representation for natural illumination has its roots in data mining and dimensionality reduction techniques. The idea is to extract the low-dimensional trends from datasets of measurements of natural illumination. This “statistical”, or “data-driven” representation is typically obtained by gathering a large number of observations—either from physical measurements of real-world illumination captured around the world [Judd *et al.*, 1964], or by generating samples using a physical model [Slater and Healey, 1998]—and performing a dimensionality reduction technique (e.g., PCA) to discover which dimensions explain most of the variance in the samples.

One of the first to propose such an idea was Judd *et al.* [1964], and is still to this date considered as one of the best experimental analysis of daylight [Slater and Healey, 1998]. Their study considered 622 samples of daylight measured in the visible spectrum, and observed that most of them could be approximated accurately by a linear combination of three fixed functions. Subsequently, Slater and Healey [1998] reported that a 7-dimensional PCA representation capture 99% of the variance of the spectral distribution of natural illumination in the visible and near-infrared spectra, based on a synthetically-generated dataset of spectral lighting profiles. Dror *et al.* [2004] performed a similar study by using a set of HDR environment maps as input.

Linear representations for illumination have been successful in many applications, notably in the joint estimation of lighting and reflectance from images of an object of known geometry. Famously, Ramamoorthi and Hanrahan [2001] used a spherical harmonics representation (linear in the frequency domain) of reflectance and illumination and expresses their interaction as a convolution. More recently, Romeiro and Zickler [2010] also use a linear illumination basis to infer material properties by marginalizing over a database of real-world illumination conditions.

Linear representations have also been used in graphics, to render skies much faster than with computationally expensive physical simulations. For example, Dobashi *et al.* [1995] employed a series of Legendre basis functions for specific sky data, and Nimeroff *et al.* [1996] employed steerable basis functions to fit multiple sky models, including the CIE [CIE, 1994]. Finally, we mention the work of Preetham *et al.* [1999], who optimized a simple linear model on the Perez sky formula [Perez *et al.*, 1993], which we will ourselves use in this thesis.

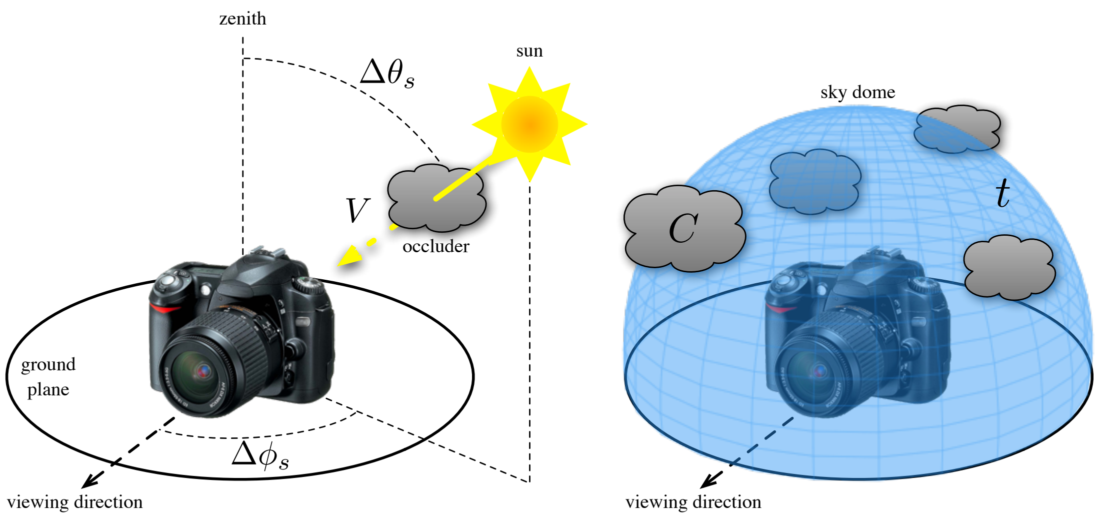
2.4 An Intuitive Representation for Natural Illumination

All the previous representations assume that the camera (or other sensing modalities) used to capture illumination are of very high quality—there must be a linear relationship between the illumination radiance and sensor reading, they have high dynamic range, etc.—and that the images contain very few, easily recognizable objects. However, most consumer photographs that are found on the Internet do not obey these rules, and we have to deal with acquisition process issues like non-linearity, vignetting, compression and resizing artifacts, limited dynamic range, out-of-focus blur, etc. In addition, scenes contain occlusions, depth discontinuities and distortions due to perspective projection. Finally, the representations presented above assume that the entire sky hemisphere can be directly observed by hyper-spectral sensors or wide-angle cameras. This of course does not correspond to the capture conditions of regular cameras, which have narrow fields of view—the sky typically occupies only a restricted portion of the image (if any). In general, these issues make the previous illumination representations mathematically impossible to estimate from single, real-world images.

We propose to use a hybrid representation that is more amenable to understand this type of images. As in (2.1), we also model the sun and the sky separately, but we do so using the intuitive parameters shown in Fig. 2.1.

2.4.1 Sun Component

We model the sun component (Fig. 2.1a) using two variables:



(a) Sun component (b) Sky component

Figure 2.1: Graphical illustration of the natural illumination model used in this thesis. The sun (a) is described by its angular position $(\Delta\theta_s, \Delta\phi_s)$ with respect to the camera, and its visibility V. The sky (b) is represented by its turbidity t (degree of scattering in the atmosphere) and cloud cover C.

V its visibility, or whether or not it is shining on the scene;

S its angular position relative to the camera. In spherical coordinates, $S = (\Delta\theta_s, \Delta\phi_s)$, where $\Delta\theta_s = \theta_s - \theta_c$ is the sun relative zenith angle with respect to the camera; and $\Delta\phi_s = \phi_s - \phi_c$ the sun relative azimuth angle with respect to the camera (see Fig. 2.1a). The s and c indices denote the sun and camera respectively. S is specified only if $V = 1$, and undefined if $V = 0$.

In contrast to existing approaches, this representation for natural illumination is more simple and intuitive, therefore making it better-suited for single image interpretation. For example, if the sun visibility V is 0 (for example, when the sky is overcast), then the sun relative position S is undefined. Indeed, if the sun is not shining on the scene, then all the objects in the scene are lit by the same area light source that is the sky. It is not useful to know (or recover) the sun direction in this case. If V is 1 however, then knowing the sun position S is important since it is responsible for illumination effects such as cast shadows, shading, specularities, etc. which might strongly affect the appearance of the scene.

2.4.2 Sky Component

For the sky component (Fig. 2.1b)), we employ the Perez sky model [Perez *et al.*, 1993], of which we provide a brief overview here (please refer to Sec. 3.3.1 for a more comprehensive presentation). This model expresses the *relative* luminance l_p of a sky element as a function of its zenith angle θ_p and angle γ_p with respect to the sun:

$$l_p = f(\theta_p, \gamma_p) = [1 + a \exp(b / \cos\theta_p)] \times \left[1 + c \exp(d\gamma_p) + e \cos^2 \gamma_p\right] . \tag{2.2}$$

Here, the weather coefficients a, b, c, d, e quantify the atmospheric conditions, which can range from clear to overcast. Following Preetham *et al.* [1999], we parameterize the model using a single variable: the turbidity t. Intuitively, the turbidity encodes the amount of scattering in the atmosphere, and replaces the weather coefficients in (2.2) like so: $a = f_a(t), b = f_b(t), \ldots$, where the functions $f(t)$ are simple linear transformations (see Sec. 3.6 for more details).

Even though its "physically-based" nature will require us to carefully analyze the image to compensate for some sources of distortion like the non-linear response function, there are several advantages to using this physical model to represent the sky. First, because there is a single parameter to estimate (t), this can be done from only the observed portion of the sky in an image: the entire sky dome needs not be visible. Moreover, once the turbidity t is recovered, the sky appearance can be extrapolated outside of the field of view of the camera, so the full sky dome can be generated. Third, it can be used to extract the visible cloud cover C (Fig. 2.1b, see Sec. 3.6) in order to obtain a rich description of the natural illumination conditions.

To summarize, we use two additional variables to model the sky component (Fig. 2.1b):

t its turbidity, which encodes the amount of scattering in the atmosphere, so the lower the t, the clearer the sky;

C its cloud cover, which we represent as the difference between the observed and predicted sky under the physical model.

Naturally, the sky component can only be represented if the sky is actually visible in the image. If it is not, we retain only the sun component to characterize the natural illumination conditions.

Part I

Image Sequences

In the first part of this thesis, we consider sequences of images acquired by a static outdoor camera. Assuming the scene itself is mostly static, the main variations in appearance are due to changes in the natural illumination conditions. Studying such sequences, commonly available worldwide via online webcams, will allow us to gain insight on an important source of information about illumination seldom explored in computer vision and graphics: the sky.

Chapter 3

Estimating Illumination in Image Sequences

The sky is that beautiful old parchment in which the sun and the moon keep their diary.

Alfred Kreymborg (1883–1966)

When presented with an outdoor photograph (such as the images on Fig. 3.1), an average person is able to infer a good deal of information just by looking at the sky. Is it morning or afternoon? Do I need to wear a sunhat? Is it likely to rain? A professional, such as a sailor or a pilot, might be able to tell even more: time of day, temperature, wind conditions, likelihood of a storm developing, etc. As the main observed illuminant in an outdoor image, the sky is a rich source of information about the scene. However it is yet to be fully explored in computer vision. The main obstacle is that the problem is woefully under-constrained. The appearance of the sky depends on a host of factors such as the position of the sun, weather conditions, photometric and geometric parameters of the camera, and location and direction of observation. Unfortunately, most of these factors remain unobserved in a single photograph; the sun is not often visible in the picture, the camera parameters and location are usually unknown, and worse yet, only a small fraction of the full hemisphere of sky is actually seen. However, if we observe the same small portion of the sky *over time*, we

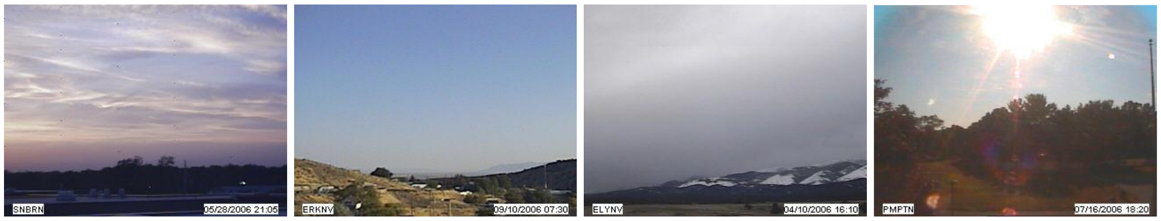


Figure 3.1: The sky appearance is a rich source of information about the scene illumination.

would see changes in sky appearance due to the sun and weather that cannot be perceived within a single image. In short, this is exactly the type of problem that might benefit from analyzing a time-lapse image sequence, which is acquired by a static camera observing the same scene over a period of time.

3.1 Overview

information about the natural illumination conditions is available in the visible portion of the sky in a time-lapse image sequence, and how to extract it. For this, we exploit two important cues: the sun position and the appearance of the sky. Our analysis demonstrates that it is possible to recover our full illumination model from Sec. 2.4—the relative angular direction of the sun with respect to the camera, the sun visibility, the sky turbidity and cloud cover—just by looking at the sky region in an image sequence.

In particular, we will demonstrate that for a static camera, if the GPS coordinates (latitude and longitude) of the camera, as well as the date and time of capture of each image are given as input, then recovering the angular sun direction $(\theta_s, \Delta\phi_s)$ with respect to the camera is equivalent to estimating the following camera parameters: its focal length f_c, its zenith (with respect to vertical) and azimuth (with respect to North) angles θ_c and ϕ_c respectively. Note that the GPS coordinates and the date and time of capture are commonly available in online webcams. As a result, we will first focus on showing how the sun and the sky can be used to recover these 3 camera parameters, and then describe how the sun visibility, sky turbidity and cloud cover are estimated.

We present an overview of the 2 main results of this chapter in Table 3.1[1]. Algorithm 1,

[1]Code is available at `http://graphics.cs.cmu.edu/projects/sky`.

Algorithm	Section	Inputs	Outputs
Alg. 1	Sec. 3.2	Sun position GPS coordinates Date and time	Camera parameters (f_c, θ_c, ϕ_c)
Alg. 2	Sec. 3.3	Clear sky images GPS coordinates Date and time	Camera parameters (f_c, θ_c, ϕ_c)
Alg. 3	App. A.1	Clear sky images Sun position Date and time	Camera parameters (f_c, θ_c, ϕ_c) GPS coordinates

Table 3.1: Overview of the different algorithms introduced in this chapter, which extract various information about the camera from the sun and the sky.

introduced in Sec. 3.2, computes the focal length, zenith and azimuth angles of the camera given as input the sun coordinates in some images, the GPS coordinates (latitude and longitude) of the camera, as well as the date and time of capture of each image. This algorithm requires the sun to be manually identified in a few frames throughout the entire sequence, a process which takes only a few minutes per sequence.

Algorithm 2, presented in Sec. 3.3, uses the sky appearance as its only input. From several images of the clear sky, GPS coordinates, date and time of capture of the images, the same camera parameters can also be estimated, this time without requiring any user input. Thus, the sky alone can be used to estimate the angular sun direction with respect to the camera in each frame in an image sequence.

Finally, we include in Appendix A.1 Algorithm 3, which demonstrates that by combining the sun position with the sky appearance, the GPS coordinates can also be estimated, along with the camera focal length, zenith and azimuth angles. In short, the sun and the sky can be used to locate and calibrate the camera.

An immediate practical result of our work is the recovery of the camera orientation and zoom level, even if we do not have physical access to the camera. We validate Algorithms 1 and 2 on a sequence where the ground truth camera parameters are known, and demonstrate that our methods make error of less that 1% in focal length, at most 3° in zenith angle, and at most 1° in azimuth angle. In addition, we also evaluate these algorithms on 22 real, low-quality webcam sequences from the AMOS (Archive of Many Outdoor Scenes) database [Jacobs *et al.*, 2007a], which contains image sequences taken by static webcams over more than a year. The selected sequences cover a wide range of latitudes ($28° - 48°$) and longitudes ($74° - 124°$), and are composed of a total of over a quarter of a million daytime images which were given as input to our algorithms. Unfortunately, ground truth is not available for these sequences, and we do not have physical access to the cameras. Instead, we analyze the consistency between parameters obtained from Algorithms 1 and 2, and show that the mean deviation is 4% for the focal length, and less than 1.5° and 3° for the zenith and azimuth angles respectively.

For all these algorithms, we assume that a static camera is observing the same scene over time, with no roll angle (i.e., the horizon line is parallel to the image horizontal axis). We also assume that the sky region has been segmented, either manually or automatically [Jacobs *et al.*, 2007a; Hoiem *et al.*, 2007], and that the sun position has already been identified in images. For the algorithms that exploit the sky appearance (2 and 3), we also assume that the camera has been radiometrically calibrated, an important step for which we have evaluated several existing approaches. The method of Kuthirummal *et al.* [2008] cannot be used in our context since they assume a random sampling of scenes. The technique proposed by Kim *et al.* [2008] also cannot be applied on all sequences since it relies on lambertian surfaces. Instead, we use the method of Lin *et al.* [2004] which was designed to estimate the inverse camera response function by using color edges gathered from a single image. For robustness, we detect edges across several images, and run the optimization on all of them.

As we mentioned earlier on, estimating the camera parameters effectively results in the recovery of the direction of main illumination (the sun) with respect to the camera. Once these parameters are estimated, we then show how we can estimate the other parameters in our natural illumination model: the sun visibility V, the sky turbidity t as well as the cloud cover C from each image in a sequence.

3.2 Camera Geometry from the Sun Position

We begin by introducing our sun-based algorithm, where the camera parameters are estimated from the sun *position* (i.e., its pixel coordinates) in a sequence of images. For this section, we assume that the sun position has already been identified in images.

3.2.1 Sun Position Model

Let us illustrate the geometry of the problem in Fig. 3.2. The sun, indicated by its zenith and azimuth angles (θ_s, ϕ_s), is observed by a camera and its center is projected at pixels (u_s, v_s) in the image. The camera, whose local reference frame is denoted by $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$, is rotated by angles (θ_c, ϕ_c) and centered at the world reference frame $(\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w)$. We assume that the camera has no roll angle, i.e., its horizon line is parallel to the image $\mathbf{u}$-axis.

The coordinates of the sun in the image (u_s, v_s) can be obtained by multiplying its 3-D coordinates $\mathbf{s}$ by the camera projection matrix $\mathbf{M}$, where

$$\mathbf{s} = \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} = \begin{bmatrix} \sin\theta_s \cos\phi_s \\ \sin\theta_s \sin\phi_s \\ \cos\theta_s \\ 1 \end{bmatrix} , \tag{3.1}$$

in homogeneous coordinates and $\mathbf{M}$ has the form

$$\mathbf{M} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} . \tag{3.2}$$

Here $\mathbf{t} = \mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}$ since the center of projection is located at the origin. The rotation matrix R is given by:

$$\mathbf{R} = \begin{bmatrix} \cos(\theta_c - \frac{\pi}{2}) & 0 & -\sin(\theta_c - \frac{\pi}{2}) \\ 0 & 1 & 0 \\ \sin(\theta_c - \frac{\pi}{2}) & 0 & \cos(\theta_c - \frac{\pi}{2}) \end{bmatrix} \begin{bmatrix} \cos\phi_c & \sin\phi_c & 0 \\ -\sin\phi_c & \cos\phi_c & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.3}$$

and assuming a simple camera model (no skew, square pixels), we can express the intrinsic

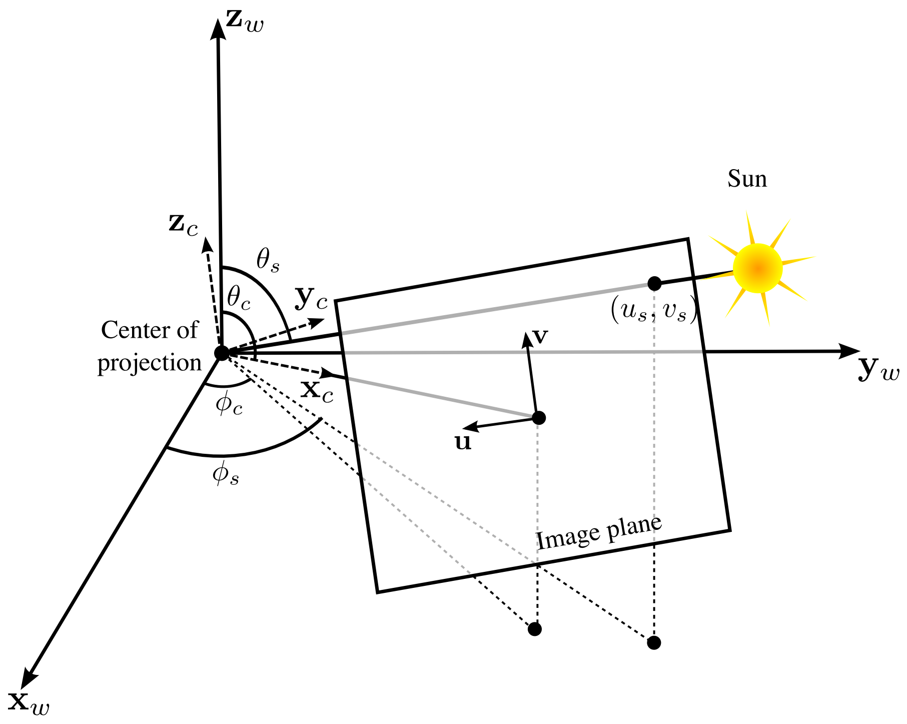


Figure 3.2: Geometry of the sun labeling problem. When the sun is visible in an image, its center gets projected at coordinates (u_s, v_s), which correspond to angles (θ_s, ϕ_s) with respect to the world reference frame $(\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w)$. The camera has zenith and azimuth angles (θ_c, ϕ_c), and its focal length f_c, not shown here, is the distance between the origin (center of projection), and the image center. The camera local reference frame is denoted by $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$ and is centered at the world reference frame.

parameters matrix $\mathbf{K}$ as:

$$\mathbf{K} = \begin{bmatrix} 0 & -f_c & 0 \\ 0 & 0 & f_c \\ 1 & 0 & 0 \end{bmatrix} . \tag{3.4}$$

With these definitions in hand, we now see how we can recover the parameters (f_c, θ_c, ϕ_c) of a camera that is observing the sun *over time*.

3.2.2 Recovering Camera Parameters from the Sun Position

This process is akin to general camera calibration (see [Forsyth and Ponce, 2003] for more details), except that $\mathbf{M}$ is constrained to be of the form in (3.2). Suppose the sun is visible in N images taken from a sequence, and that we know the coordinates $\mathbf{p} = \begin{bmatrix} u_s & v_s \end{bmatrix}^\mathsf{T}$ of its center in the images. From the GPS location and the time of capture of each image, we can also obtain the corresponding sun angular positions (θ_s, ϕ_s) by using [Reda and Andreas, 2005].

If $\mathbf{m}_i$ is the ith row of $\mathbf{M}$, then following the standard camera calibration procedure [Forsyth and Ponce, 2003] we get that each image defines two equations:

$$\begin{aligned} \left(\mathbf{m}_1 - u_s^{(i)}\mathbf{m}_3\right) \cdot \mathbf{s}^{(i)} &= 0 \ , \\ \left(\mathbf{m}_2 - v_s^{(i)}\mathbf{m}_3\right) \cdot \mathbf{s}^{(i)} &= 0 \ . \end{aligned} \tag{3.5}$$

As detailed in Appendix A.2, this results in a system of $2N$ equations and 8 unknowns. When $N \geq 4$, homogeneous linear least-squares can be used to compute the value of the matrix $\mathbf{M}$ as the solution of an eigenvalue problem. The camera parameters can then be retrieved from the individual entries of $\mathbf{M}$ (see Appendix A.2 for details).

We observe that we can improve the results quality by using these estimates as initial guess in a non-linear minimization which optimizes the camera parameters directly:

$$\min_{f_c, \theta_c, \phi_c} \sum_{i=1}^{N} \left((\mathbf{m}_1 - u_s^{(i)}\mathbf{m}_3) \cdot \mathbf{s}^{(i)} \right)^2 + \left((\mathbf{m}_2 - v_s^{(i)}\mathbf{m}_3) \cdot \mathbf{s}^{(i)} \right)^2 \ . \tag{3.6}$$

This non-linear least-squares minimization can be solved iteratively using standard optimization techniques such as Levenberg-Marquadt or `fminsearch` in MATLAB.

The entire sun-based calibration process is summarized in Algorithm 1: from a set of sun positions in images, with the corresponding GPS location and date and time of capture of each image, the algorithm recovers the camera parameters (f_c, θ_c, ϕ_c).

3.2.3 Validation Using Synthetic Data

In order to thoroughly validate our approach, we test it under a very wide variety of conditions using synthetically-generated data.

3.2.3.1 Synthetic Data Generation

For a given set of camera parameters, sun coordinates $\mathbf{p}$ are generated by uniformly sampling the visible sky area (above the horizon line). Their corresponding 3-D sun vectors $\mathbf{s}$ are

```
Algorithm 1: Camera geometry from the sun position
    Input: Sun position in images: $(u_s, v_s)$;
    Input: GPS location of the camera;
    Input: Date and time of capture of each image.
  1 Compute the sun angles $(\theta_s, \phi_s)$ from the GPS, date and time of capture using [Reda
    and Andreas, 2005];
  2 Build matrix $\mathbf{M}$;
  3 Solve linear system (A.7);
  4 Refine estimate by minimizing (3.6).
    Output: Camera parameters: $(f_c, \theta_c, \phi_c)$.
```

Variable name	Range	Default value	Comments
Focal length f_c	$[100, 2000]$ px	1000 px	18° field of view
Zenith angle θ_c	$[80°, 100°]$	90°	Looking straight
Azimuth angle ϕ_c	$[-180°, 180°]$	0°	Facing North
Number of images N	$[5, 50]$	20	–
Sun detection noise σ^2	$[0, 10]$ px	–	–

Table 3.2: Range and default values for each variable used in the experiments on synthetic data. Note that the range for θ_c is limited such that the horizon line remains visible at the given focal length.

then found by applying the sequence of equations opposite to the one shown in the previous section. Gaussian noise is added to $\mathbf{p}$ to simulate error in sun center detection. All images have dimension 320×240 pixels.

We evaluate the influence of five variables on the quality of the results: the three camera parameters (f_c, θ_c, ϕ_c), the number of available images N, and the labeling noise variance σ^2. From this space of variables, we sample 213 different combinations by varying each one at a time, while maintaining the others constant at a default value, shown in Table 3.2. The range of each variable is also shown in that table. In order to account for the randomness in point selection, each experiment is performed 20 times.

3.2.3.2 Quantitative Evaluation

Fig. 3.3 shows error curves for 6 different scenarios, illustrating the effect of varying each parameter presented in the previous section on estimating the camera parameters. Figs 3.3a and 3.3b show that small focal lengths (or, alternatively, wide fields of view) result in larger estimation error, although it still remains below 3% error for f_c, and below 0.7° for θ_c, even in the high noise case ($\sigma^2 = 10$) Similar results are obtained for ϕ_c (not shown here). As expected, increasing N also decreases estimation error, as shown in Fig. 3.3d and 3.3f.

Plots obtained from varying θ_c and ϕ_c are shown in Fig. 3.3c and 3.3e, and result in mostly constant curves over the parameter range. θ_c is varied such that the horizon line remains visible in the image at the given focal length (from Table 3.2). In short, Fig. 3.3 demonstrates that this algorithm can simultaneously recover the focal length, zenith and azimuth angles of a very wide set of cameras, given a sequence of sun positions.

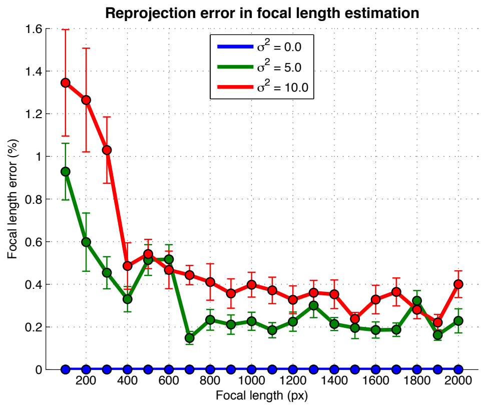
3.2.4 Validation Using Ground Truth Camera Geometry

In addition to synthetic data, we also evaluate our algorithm on a sequence of images of the sky, captured by a calibrated camera with ground truth parameters: focal length, zenith and azimuth angles. We first present the acquisition and calibration setup, then show that both algorithms estimate camera parameters that are very close to ground truth.

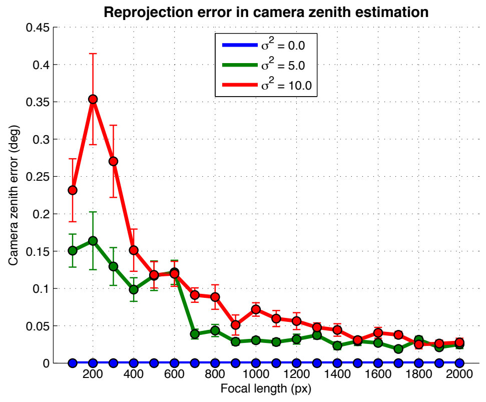
3.2.4.1 Acquisition Setup

We captured a high-quality sequence of the sky by placing a Canon Digital Rebel XT equipped with a 18mm lens outdoors during an entire day (see Fig. 3.4a). Using a computer-controlled script, the camera continuously captured images at 5-minute intervals between 10:45 until 20:25, on April 17th, 2009. Fig. 3.4b shows example images from the resulting sequence. The camera was placed at the GPS coordinates of 40.367° of latitude, and −80.057° of longitude. The images were captured in RAW mode, which allows us to convert them to the JPEG format with a linear response function. We kept 8-bit dynamic range to simulate the behavior of a typical webcam.

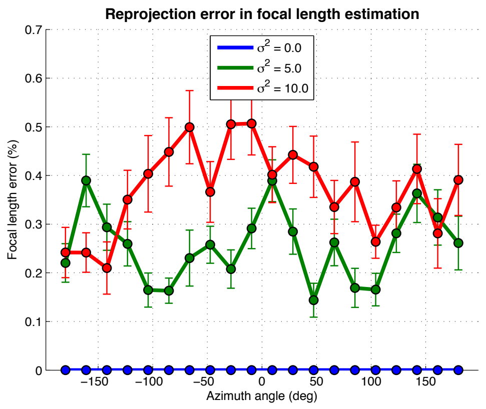
The intrinsic camera parameters were recovered using the MATLAB® camera calibration toolbox. The ground truth focal length was determined to be 2854 pixels. To compute the ground truth zenith angle, we placed a leveled calibration target in the field of view of the camera, and computed the intersection of parallel lines from the target. This intersection point indicates the horizon line in the image, from which the zenith angle can be computed using (3.18). The resulting zenith angle is 71.3°. The ground truth azimuth angle was



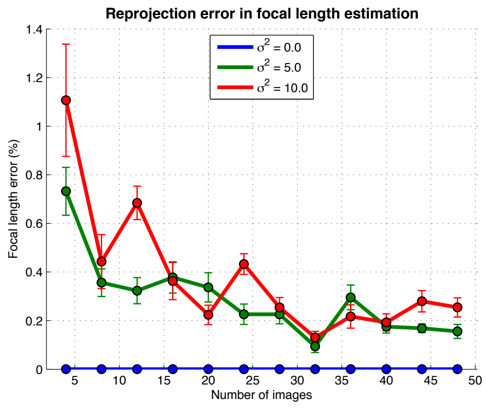
(a) $f_c \to f_c$



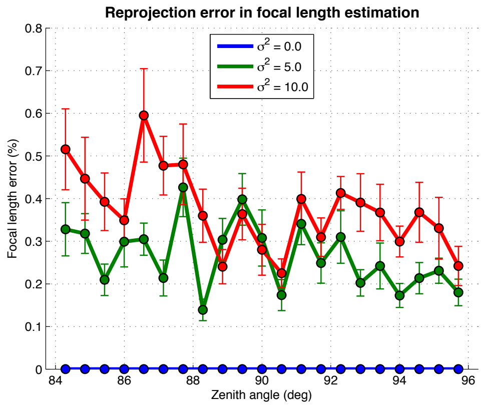
(b) $f_c \to \theta_c$



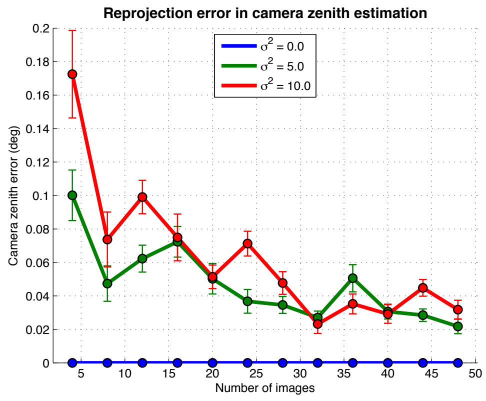
(c) $\theta_c \to f_c$



(d) $N \to f_c$

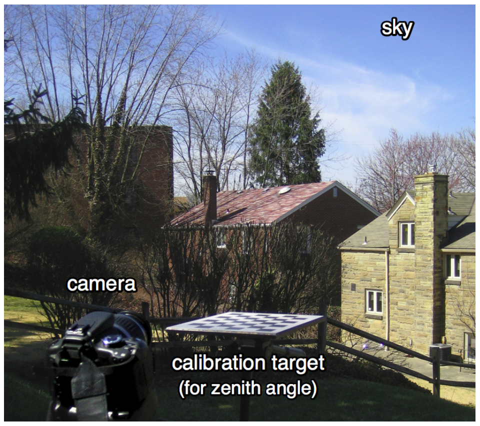


(e) $\phi_c \to f_c$

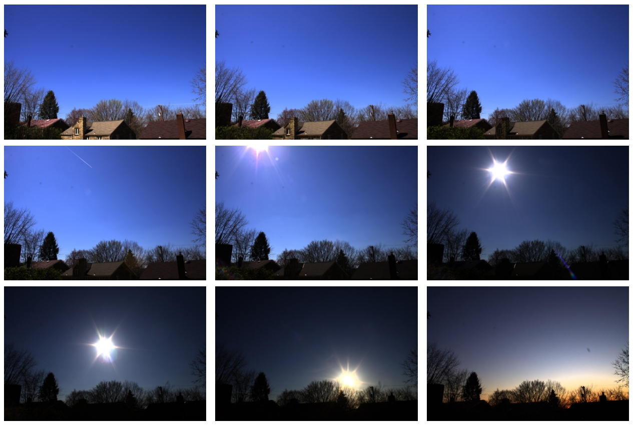


(f) $N \to \theta_c$

Figure 3.3: Synthetic data evaluation of camera parameters estimation from the sun position. A representative sample of the entire set of experiments performed is shown, the remaining plots can be found in [Lalonde *et al.*, 2008a].



(a)



(b)

Figure 3.4: Ground truth data acquisition. (a) Acquisition setup. The calibration target, placed on a tripod in front of the camera and leveled with the ground, is used to recover the camera zenith angle. (b) Example images from 12:00 (top-left) to 20:00 (bottom-right), in one-hour increments. The images are shown with a linear camera response function.

Parameter	Ground truth	Sun estimate	Error
Focal length f_c	2854 px	2881 px	0.9%
Zenith angle θ_c	71.3°	70.2°	1.1°
Azimuth angle ϕ_c	93.5° W	94.3° W	0.8°

Table 3.3: Comparison with ground truth.

computed using a satellite map of the capture location and was found to be 93.5° West.

3.2.4.2 Calibration Results

After manually labeling the sun in all the frames in which it is visible, the sun-based calibration algorithm was applied to recover the camera parameters. The results, shown in Table 3.3, demonstrate that the focal length can be recovered within 0.9% of its true value, and the zenith and azimuth angles are obtained within 1.1° and 0.8° of their true values respectively.

3.3 Camera Geometry from the Sky Appearance

Unfortunately, the sun might not always be visible in image sequences, so the previous algorithm applicability might be limited in practice. Therefore, we now focus our attention on a different source of information more widely available: the sky *appearance*. We will consider clear skies only, and address the more complicated case of clouds at a later point.

3.3.1 Physically-based Model of the Sky Appearance

First, we introduce the physically-based model of the sky that lies at the foundation of our approach. We will first present the model in its general form, then in a useful simplified form, and finally demonstrate how it can be written as a function of camera parameters.

3.3.1.1 Perez Sky Model

The Perez sky model [Perez *et al.*, 1993] describes the luminance of any arbitrary sky element as a function of its elevation, and its relative orientation with respect to the sun. It is a generalization of the CIE standard clear sky formula [CIE, 1994], and it has been found to be more accurate for a wider range of atmospheric conditions [Ineichen *et al.*, 1994]. Consider the illustration in Fig. 3.5. The *relative* luminance l_p of a sky element is a function of its zenith angle θ_p and the angle γ_p with the sun:

$$l_p = f(\theta_p, \gamma_p) = [1 + a \exp(b / \cos\theta_p)] \times \left[1 + c \exp(d\gamma_p) + e \cos^2 \gamma_p\right] \quad , \tag{3.7}$$

where the 5 constants a, b, c, d, e specify the current atmospheric conditions, and all angles are expressed in radians. As suggested in [Preetham *et al.*, 1999], those constants can also be approximated by a linear function of a single parameter, the turbidity t. Intuitively, the turbidity encodes the amount of scattering in the atmosphere, so the lower the t, the clearer the sky. For clear skies, the constants take on the following values: $a = -1$, $b = -0.32$, $c = 10$, $d = -3$, $e = 0.45$, which corresponds approximately to $t = 2.17$.

The model expresses the *absolute* luminance L_p of a sky element as a function of another arbitrary reference sky element. For instance, if the zenith luminance L_z is known, then

$$L_p = L_z \frac{f(\theta_p, \gamma_p)}{f(0, \theta_s)} \quad , \tag{3.8}$$

where θ_s is the zenith angle of the sun. Fig. 3.6 illustrates the luminance predicted by the Perez sky model for different values of t and θ_s.

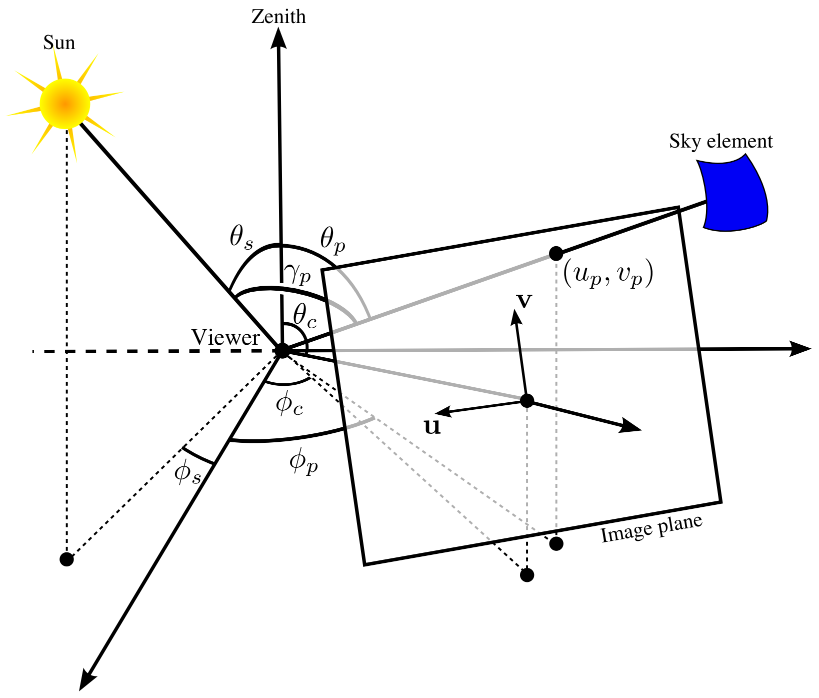


Figure 3.5: Geometry of the problem when a camera is viewing a sky element (blue patch in the upper-right). The sky element is imaged at pixel (u_p, v_p) in the image, and the camera is rotated by angles (θ_c, ϕ_c). The camera focal length f_c, not shown here, is the distance between the origin (center of projection), and the image center. The sun direction is given by (θ_s, ϕ_s), and the angle between the sun and the sky element is γ_p. Here (u_p, v_p) are known because the sky is segmented.

3.3.1.2 Azimuth-independent Sky Model

By running synthetic experiments, we were able to determine that the influence of the second factor in (3.7) becomes negligible when the sun is more than 100° away from a particular sky element (see Appendix A.4). In this case, the sky appearance can be modeled by using only the first term from (3.7):

$$l'_p = f'(\theta_p) = 1 + a \exp(b / \cos\theta_p) \ . \tag{3.9}$$

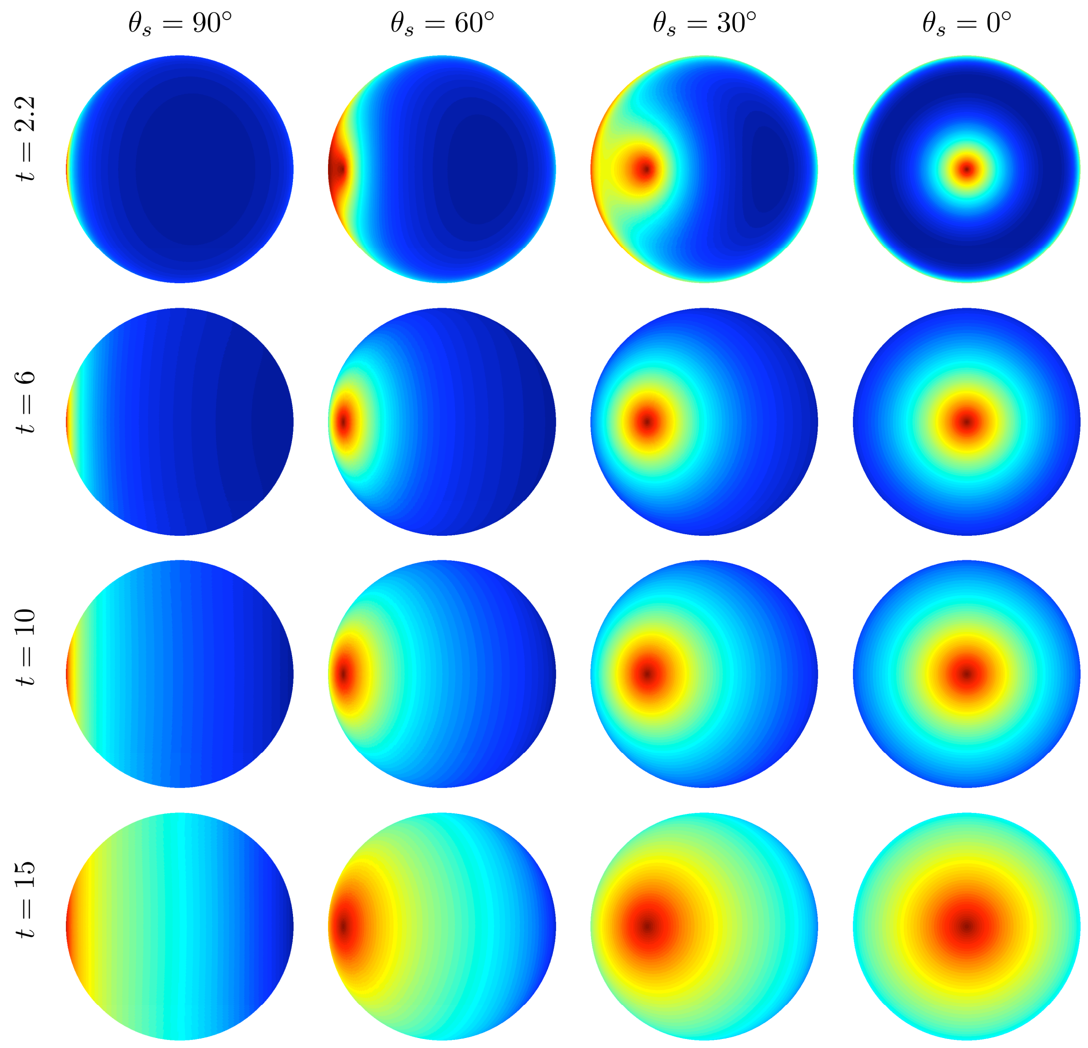


Figure 3.6: Perez sky model (3.7), plotted for different turbidities t (rows), for different sun positions θ_s (columns). Each plot represents the entire hemisphere, centered at the zenith (i.e., looking straight up). For example, the sun is at the horizon in the left-most column ($\theta_s = 90^\circ$), and at the zenith in the right-most column ($\theta_s = 0^\circ$).

This equation effectively models the sky *gradient*, which varies from light to dark from horizon to zenith on a clear day. L'_p is obtained in a similar fashion as in (3.8):

$$L'_p = L_z \frac{f'(\theta_p)}{f'(0)} \ . \tag{3.10}$$

3.3.1.3 Expressing the Sky Model as a Function of Camera Parameters

Now suppose a camera is looking at the sky, as in Fig. 3.5. We can express the general (3.7) and azimuth-independent (3.9) models as functions of camera parameters. Let us start with the simpler azimuth-independent model.

From (3.9), we see that we only need to find an expression for θ_p, since a and b are fixed. We obtain the following relation (see Appendix A.3 for the full derivation):

$$\theta_p = \arccos\left(\frac{v_p \sin\theta_c + f_c \cos\theta_c}{\sqrt{f_c^2 + u_p^2 + v_p^2}}\right) \quad , \tag{3.11}$$

where u_p and v_p are the image coordinates of a sky element, f_c is the camera focal length, and θ_c is its zenith angle (see Fig. 3.5). We substitute (3.11) into (3.9) to obtain the final equation. Throughout this chapter, we will refer to this model using:

$$l'_p = g'(u_p, v_p, f_c, \theta_c) \ . \tag{3.12}$$

In the general sky model case (3.7), we also need to express γ_p as a function of camera parameters:

$$\gamma_p = \arccos\left(\cos\theta_s \cos\theta_p + \sin\theta_s \sin\theta_p \cos(\phi_p - \phi_s)\right) \quad , \tag{3.13}$$

where θ_s and ϕ_s are the sun zenith and azimuth angles. We already found the expression for θ_p in (3.11), so the only remaining unknown is ϕ_p, the azimuth angle of the sky element. Following the derivation from Appendix A.3, we obtain:

$$\phi_p = \arctan\left(\frac{f_c \sin\phi_c \sin\theta_c - u_p \cos\phi_c - v_p \sin\phi_c \cos\theta_c}{f_c \cos\phi_c \sin\theta_c + u_p \sin\phi_c - v_p \cos\phi_c \cos\theta_c}\right) \ . \tag{3.14}$$

We substitute (3.11), (3.13), and (3.14) into (3.7) to obtain the final equation. For succinctness, we omit writing it in its entirety, but present its general form instead:

$$l_p = g(u_p, v_p, \theta_c, \phi_c, f_c, \theta_s, \phi_s) \ . \tag{3.15}$$

Before we present how we use the models presented above, recall that we are dealing with *ratios* of sky luminance, and that a reference element is needed. Earlier, we used the zenith luminance L_z as a reference in (3.8) and (3.10), which unfortunately is not always visible in images. Instead, we can treat this as an additional unknown in the equations. Since the denominators in (3.8) and (3.10) do not depend on camera parameters, we can combine them with L_z into a single unknown scale factor k.

3.3.2 Recovering Camera Parameters from the Sky Appearance

In the previous section, we presented a physically-based model of the clear sky that can be expressed as a function of camera parameters. Now, if we are given a set of images taken from a static camera, can we use the clear sky as a calibration target and recover the camera parameters from the sky appearance only?

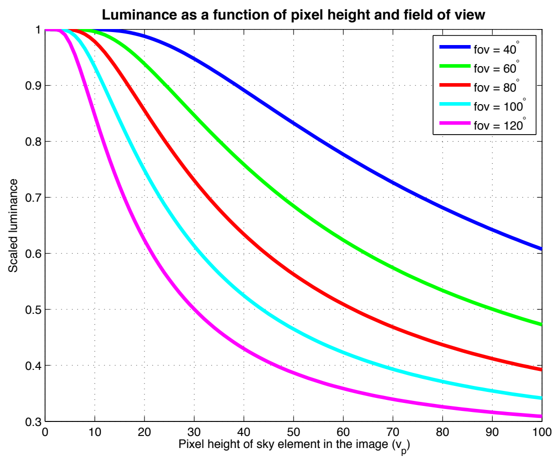
3.3.2.1 Recovering the Focal Length and Zenith Angle

Let us first consider the simple azimuth-independent model (3.12). If we plot the predicted luminance profile for different focal lengths as in Fig. 3.7a (or, equivalently, for different fields of view), we can see that there is a strong dependence between the focal length f_c and the shape of the luminance gradient. Similarly, the camera azimuth θ_c dictates the vertical offset, as in Fig. 3.7b.

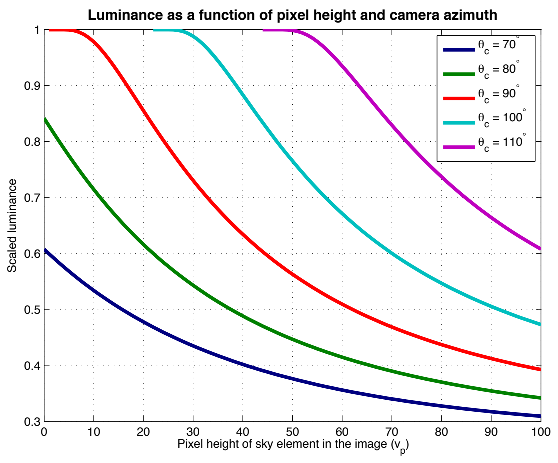
From this intuition, we devise a method of recovering the focal length and zenith angle of a camera from a set of images where the sun is far away from its field of view (i.e., at least 100° away). Suppose we are given a set $\mathcal{I}$ of such images, in which the sky is visible at pixels in set $\mathcal{P}$, also given. We seek to find the camera parameters (θ_c, f_c) that minimize

$$\min_{\theta_c, f_c, k^{(i)}} \sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}} \left(y_p^{(i)} - k^{(i)} g'(u_p, v_p, \theta_c, f_c) \right)^2 \quad , \tag{3.16}$$

where $y_p^{(i)}$ is the observed intensity of pixel p in image i, and $k^{(i)}$ are unknown scale factors (Sec. 3.3.1.3), one per image. f_c is initialized to a value corresponding to a 35° field of view, and θ_c is set such that the horizon line is aligned with the lowest visible sky pixel. All $k^{(i)}$'s are initialized to 1. This process is used to initialize Algorithm 2.



(a)



(b)

Figure 3.7: Luminance profiles predicted by the azimuth-independent model (3.12). For clear skies, intensity diminishes as pixel height above the horizon (x-axis) increases. (a) The camera zenith angle is kept constant at $\theta_c = 90°$, while the field of view is varied. (b) The field of view is kept constant at $80°$, while the camera zenith angle is varied. Both parameters have a strong influence on the shape and offset of the predicted sky gradient. Note that the curves in (b) are not translations of the same curve along the x-axis, because they are expressed in pixels in the image, not angles.

3.3.2.2 Recovering the Azimuth Angle

From the azimuth-independent model (3.12) and images where the sun is far from the camera field of view, we were able to estimate the camera focal length f_c and its zenith angle θ_c. If we consider the general model (3.15) that depends on the sun position, we can also estimate the camera azimuth angle using the same framework as before.

Suppose we are given a set of images $\mathcal{J}$ where the sky is clear, but where the sun is now closer to the camera field of view. Similarly to (3.16), we seek to find the camera azimuth angle which minimizes

$$\min_{\phi_c, k^{(j)}} \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}} \left(y_p^{(j)} - k^{(j)} g(u_p, v_p, \theta_c, \phi_c, f_c, \theta_s, \phi_s) \right)^2 \quad . \tag{3.17}$$

We already know the values of f_c and θ_c, so we do not need to optimize over them. Additionally, if the GPS coordinates of the camera and the time of capture of each image are known, the sun zenith and azimuth (θ_s, ϕ_s) can be computed using [Reda and Andreas, 2005]. Therefore, the only unknowns are $k^{(j)}$ (one per image), and ϕ_c. Since this equation is highly non-linear, we have found that initializing ϕ_c to several values over the $[-180°, 180°]$ interval and keeping the result that minimizes (3.17) works the best. This process is summarized in Algorithm 2.

Algorithm 2: Camera geometry from the sky appearance

```
  Input: Clear sky images;
  Input: GPS location of the camera;
  Input: Date and time of capture of each image.
1 Find set $\mathcal{I}$ of images where the sun is far away from the field of view;
2 Solve the non-linear minimization (3.16) to recover $f_c$ and $\theta_c$;
3 Find set $\mathcal{J}$ of images where the sun is close to the field of view;
4 Compute the sun angles $(\theta_s, \phi_s)$ from the GPS, date and time of capture using [Reda
  and Andreas, 2005];
5 Solve the non-linear minimization (3.17).
  Output: Camera parameters: $(f_c, \theta_c, \phi_c)$.
```

3.3.3 Validation Using Synthetic Data

In order to thoroughly evaluate our model, we have performed extensive tests on synthetic data generated under a wide range of operating conditions. We now present our data generation algorithm, followed by plots showing that we can effectively recover the parameters of a diverse set of cameras.

3.3.3.1 Synthetic Data Generation

We evaluate the influence of seven variables on the quality of the results: the three camera parameters (f_c, θ_c, ϕ_c), the number of available clear sky images N, the sky visibility (percentage of unoccluded sky above the horizon), the camera latitude, and the noise variance σ^2. Given a set of camera parameters and sun positions, we generate synthetic images by using our sky model (3.15). Sun positions are generated by sampling every hour over an entire year at a given latitude, and applying [Reda and Andreas, 2005]. Note that longitude does not affect the results since daytime images can be chosen such that $\theta_s < 90°$. We simulate limited visibility by occluding the sky from the horizon line, one row at a time.

Variable name	Symbol	Default value	Comments
Focal length	f_c	750 px	24° field of view
Zenith angle	θ_c	90°	Looking straight
Azimuth angle	ϕ_c	0°	Facing North
Number of images	N	15	
Sky visibility	–	100%	No occlusion
Latitude	–	0°	Equator

Table 3.4: Default values for each variable used in the experiments on synthetic data.

We build set $\mathcal{I}$ by randomly picking N sun positions that are at least 100° away from the camera field of view. If no such point is available given the geometry, we select those that are furthest away from the camera. We build set $\mathcal{J}$ by randomly selecting N sun positions. In both cases, we make sure that the sun is never directly visible by the camera. 1000 points are then randomly picked for each image, and used in the optimization. In order to evaluate the influence of each variable independently, we vary one at a time, while keeping the others at their default values shown in Table 3.4. Each experiment is performed 15 times to account for the randomness in point and sun position selection.

3.3.3.2 Quantitative Evaluation

Fig. 3.9 shows the effect of varying the variables on the camera parameters estimation error. As expected, Fig. 3.9a shows that increasing the visible portion of the sky decreases the estimation error.

Fig. 3.9b represents the influence of the camera azimuth angle ϕ_c on the estimation error of the same parameter, and shows that error is typically higher when $\phi_c = 0°$ (North) or $\phi_c = 180°$ (South) in noisy conditions. In this configuration, the sun is always relatively far from the camera, so its influence is not as visible, see Fig. 3.8 for a graphical illustration of the situation.

Figs 3.9c and 3.9d shows the effect of varying focal length f_c on estimating f_c and θ_c respectively. We note the higher the f_c (or, equivalently, the narrower the field of view), the larger error. In this situation, the visible portion of the sky hemisphere is smaller, and variations in intensity more subtle to capture.

Finally, we note that the latitude does not seem to affect the estimation error, as varying

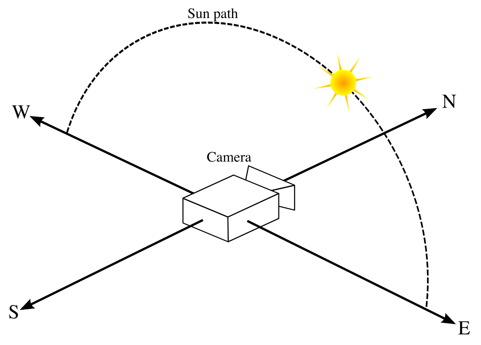


Figure 3.8: Graphical illustration of the influence of camera azimuth. If the camera is at the Equator (latitude=0°) and facing North (or South, equivalently), the sun path is further away from the camera than if it was facing East (or West). Its influence is less noticeable, therefore it is harder to recover the camera azimuth.

Parameter	Ground truth	Sky estimate	Error
Focal length f_c	2854 px	2845 px	0.3%
Zenith angle θ_c	71.3°	74.4°	3.1°
Azimuth angle ϕ_c	93.5° W	94.4° W	0.9°

Table 3.5: Comparison with ground truth for the sky-based algorithm.

it over the $[0°, 90°]$ range yields mostly constant curves as shown in Fig. 3.9f. The same behavior is also observed in Fig. 3.9e for θ_c, which is varied such that the horizon line remains visible in the image at the given focal length.

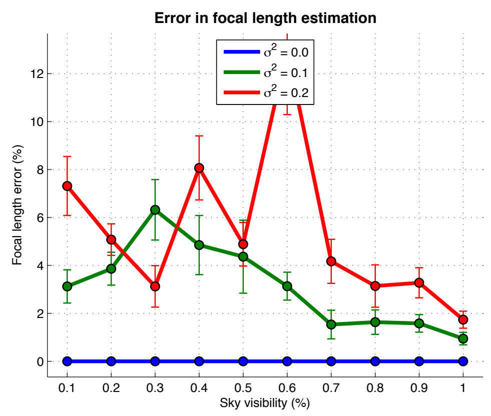
3.3.4 Validation Using Ground Truth Camera Geometry

In addition to synthetic data, we also evaluate our algorithm on the same sequence of images of the sky as in Sec. 3.2.4, captured by a calibrated camera with ground truth parameters.

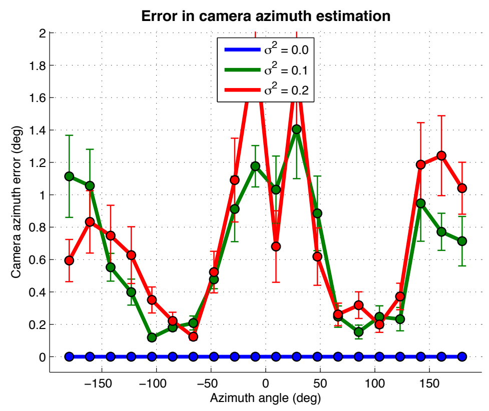
After manually segmenting the sky, we applied our sky-based calibration algorithm which yielded the camera parameters estimates shown in Table 3.5. The algorithm is able to recover, *entirely automatically*: the focal length within 0.3% of the true value, the zenith angle within 3.1°, and the azimuth angle within 0.9°.

3.4 Calibrating the Webcams of the World

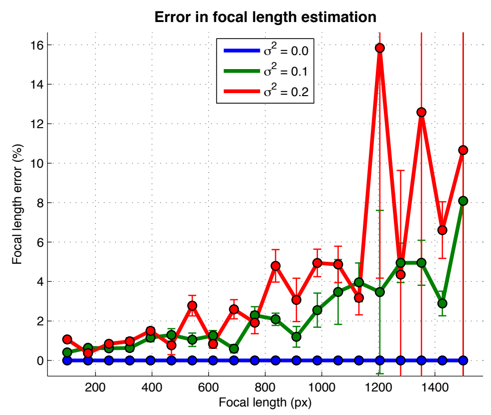
We have already shown that our algorithms achieve good estimation results on synthetic data and on one set of high-quality images. In this section, we demonstrate how our



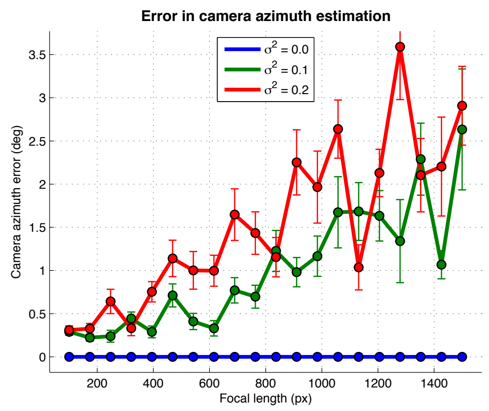
(a) $v \longrightarrow f_c$



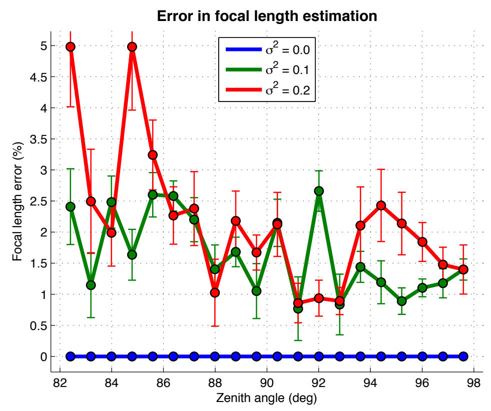
(b) $\phi_c \longrightarrow \phi_c$



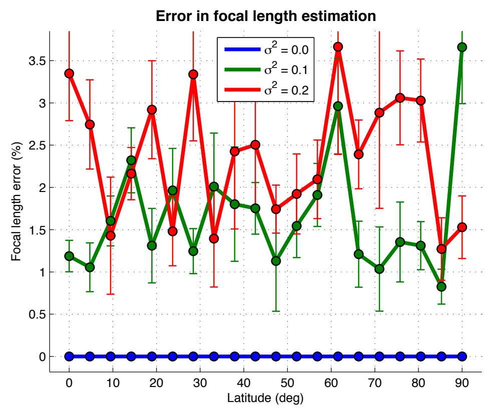
(c) $f_c \longrightarrow f_c$



(d) $f_c \longrightarrow \phi_c$



(e) $\theta_c \longrightarrow f_c$



(f) latitude $\longrightarrow f_c$

Figure 3.9: Synthetic data evaluation of camera parameters estimation from the sky appearance. Each of the 6 representative plots are shown at three different noise levels.

algorithms perform on a wide range of operating conditions, by testing them on a large set of real, typical low-quality webcam data such as the ones found in the AMOS database [Jacobs *et al.*, 2007a]. We first evaluate each technique independently, and then compare their results together to establish their consistency, which confirms that both can reliably be used in a real application setting.

We test our algorithms on 22 webcam sequences from the AMOS database in which the sun is visible, which amounts to a total of approximately a quarter of a million individual daytime images. The selected webcams are located in the continental United States, their individual GPS locations are shown in Fig. 3.10. Although they are all in the same country, they cover a wide range of latitudes ($28° - 48°$) and longitudes ($74° - 124°$).

3.4.1 Using the Sun Position

We first present results obtained by using the sun position to recover the camera parameters. Although no ground truth is available, we can assess the estimation quality by displaying the predicted sun position on every image, as well as the estimated horizon line v_h, and inspect the results visually. v_h is obtained by:

$$v_h = -f_c \tan(90° - \theta_c) \quad . \tag{3.18}$$

Fig. 3.11 shows examples of sun position and horizon line prediction on several frames for different image sequences, where the sun was not manually labeled. The predicted and actual sun positions overlap, which indicates that the camera parameters are recovered successfully.

A useful by-product of our approach is that the sun position can be predicted for all frames in the sequences, even if it is not visible. For example, in Fig. 3.11 Seq. 347, the sun position in the second and fourth frames can be predicted even if it is occluded by the scene.

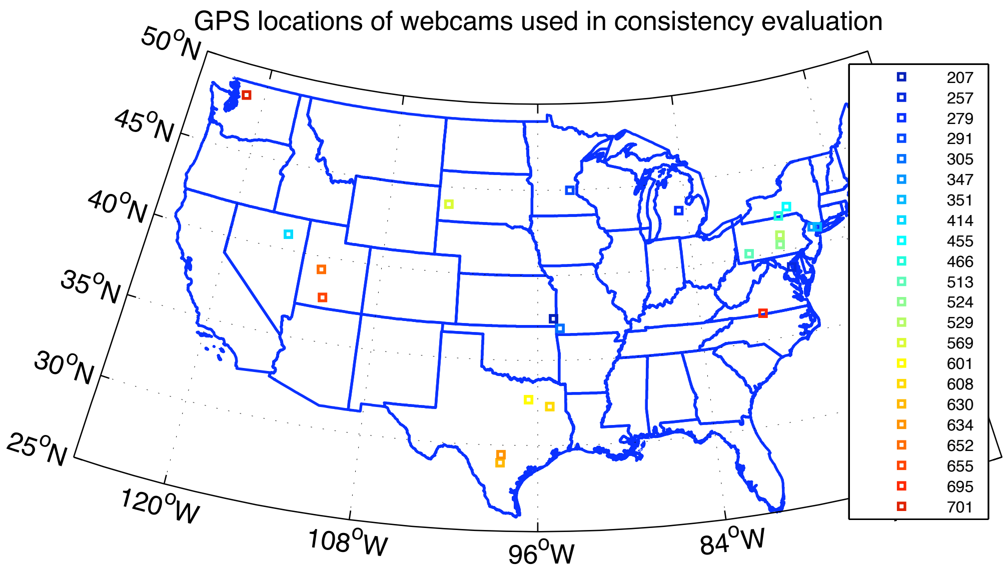


Figure 3.10: Map of GPS coordinates of each webcam used in the consistency evaluation. Since the sequences selected from the AMOS database come from the US, only this area of the world is shown. However, as we have shown in the synthetic evaluation, the algorithms are not limited to this area. Each webcam is represented by a different color.

3.4.2 Using the Sky Appearance

We now present results obtained by using the sky appearance to recover the camera parameters from real image sequences. Intensity information can be corrupted in several ways in low-quality webcam sequences: non-Gaussian noise, slight variations in atmospheric conditions, vignetting, saturation, under-exposure, etc. Most importantly however, the camera response function may be non-linear, yielding significant distortions in the sky appearance, thus leading our estimation process astray. We rely on [Lin *et al.*, 2004] which estimates the inverse response function by using color edges gathered from a single image. For additional robustness, we detect edges across several frames.

Additionally, recall that the optimization procedures (3.16) and (3.17) require clear sky image sets $\mathcal{I}$ and $\mathcal{J}$, where the sun is far and close to the camera respectively. We first give more details about how this is done, and then present results on real image sequences.

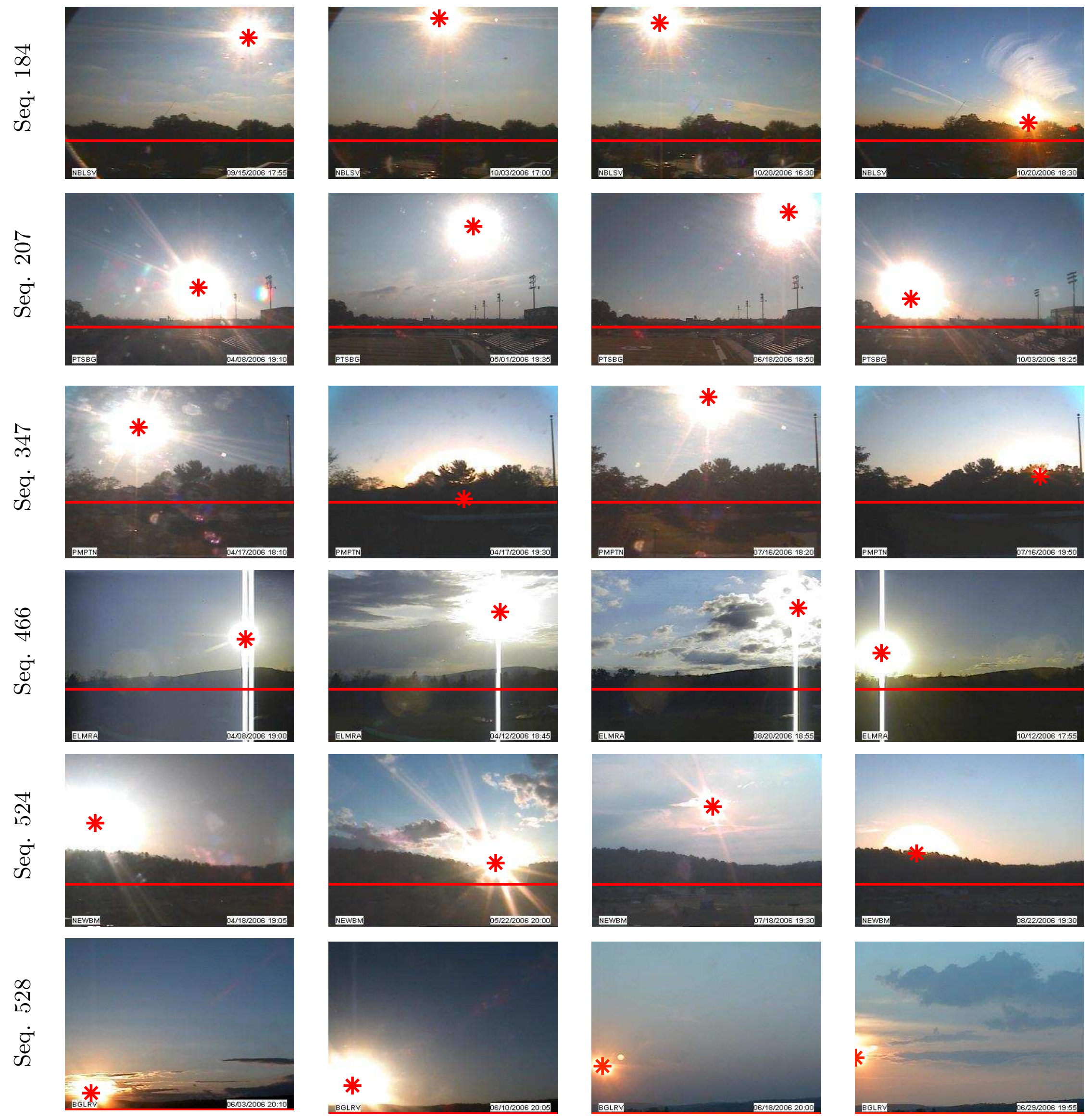


Figure 3.11: Visual evaluation of our camera calibration algorithm from the sun position. The camera parameters determined by our method are used to predict where the sun (red star) and the horizon (red line) will appear in images. Note that the sun position can be predicted even if it is occluded by the scene protruding above the horizon. The images shown here were not used in the optimization.

3.4.2.1 Finding Clear Sky Images Automatically

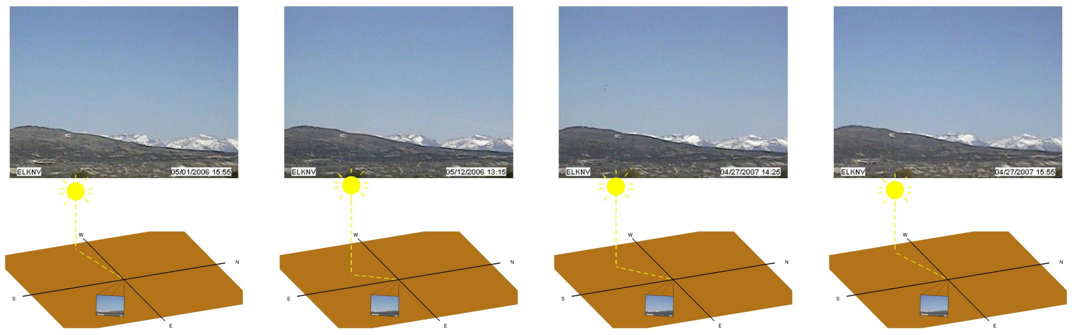
The algorithm presented in Sec. 3.3.2 uses clear sky images as input. In order to avoid having to painstakingly select such images by hand from a long image sequence, we propose an algorithm which does not require any knowledge of the camera parameters to do so automatically.

To build set $\mathcal{I}$, we mentioned that the sun should be at least $100°$ away from the camera field of view. Unfortunately, this is impossible to know a priori, so we propose an algorithm that finds clear sky images that do not seem affected by the sun. To do so, we approximate the sky model (3.12) by a vertical quadratic of the form:

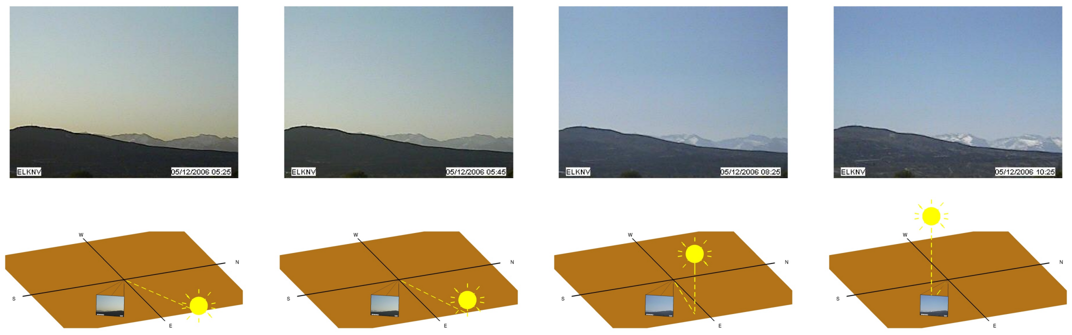
$$\alpha(v_p - v_{min})^2 + \beta = 0 \ , \tag{3.19}$$

where v_{min} is the lowest visible sky pixel, and α and β are the quadratic coefficients. We then fit this simple model to all the images in the sequence using linear least-squares. Images with low residual error and $\alpha < 0$ should exhibit a smooth vertical gradient that correspond to a clear sky, since it varies from light to dark from horizon to zenith. We found that retaining the top 10% of images with $\alpha < 0$ based on their residual error, and then keeping the top N by sorting them in decreasing order of $|\alpha|$ yields consistently good results across image sequences. Note that if the sun was close to the camera, it would most likely create a horizontal gradient in the image, thereby reducing the quality of a fit to a vertical quadratic. We show example images recovered by this algorithm in Fig. 3.12a.

Building set $\mathcal{J}$ requires finding images where the sun influence is visible at varying degrees. This is a harder problem than before for three main reasons: 1) the sun might never be very close to the camera (see Fig. 3.8), so its influence might be subtle; 2) it is hard to model the appearance of the sun influence on individual images because it may come from different directions, unlike the sky gradient that is always vertical (3.19); and 3) $\mathcal{J}$ needs to contain images where the sun is at different positions to insure robustness in the estimation. Therefore, we must enforce that the recovered images be taken at different times of day. Instead of trying to find individual clear images, our strategy is to find clear *days*. We score each day in a sequence by counting how many images with $\alpha < 0$, and select the top 4 days, which should contain mostly clear skies. To filter out clouds that may appear, we then select the N smoothest images, based on their average u and v gradients (we use finite differences). Example images recovered by this algorithm are shown in Fig. 3.12b.



(a)

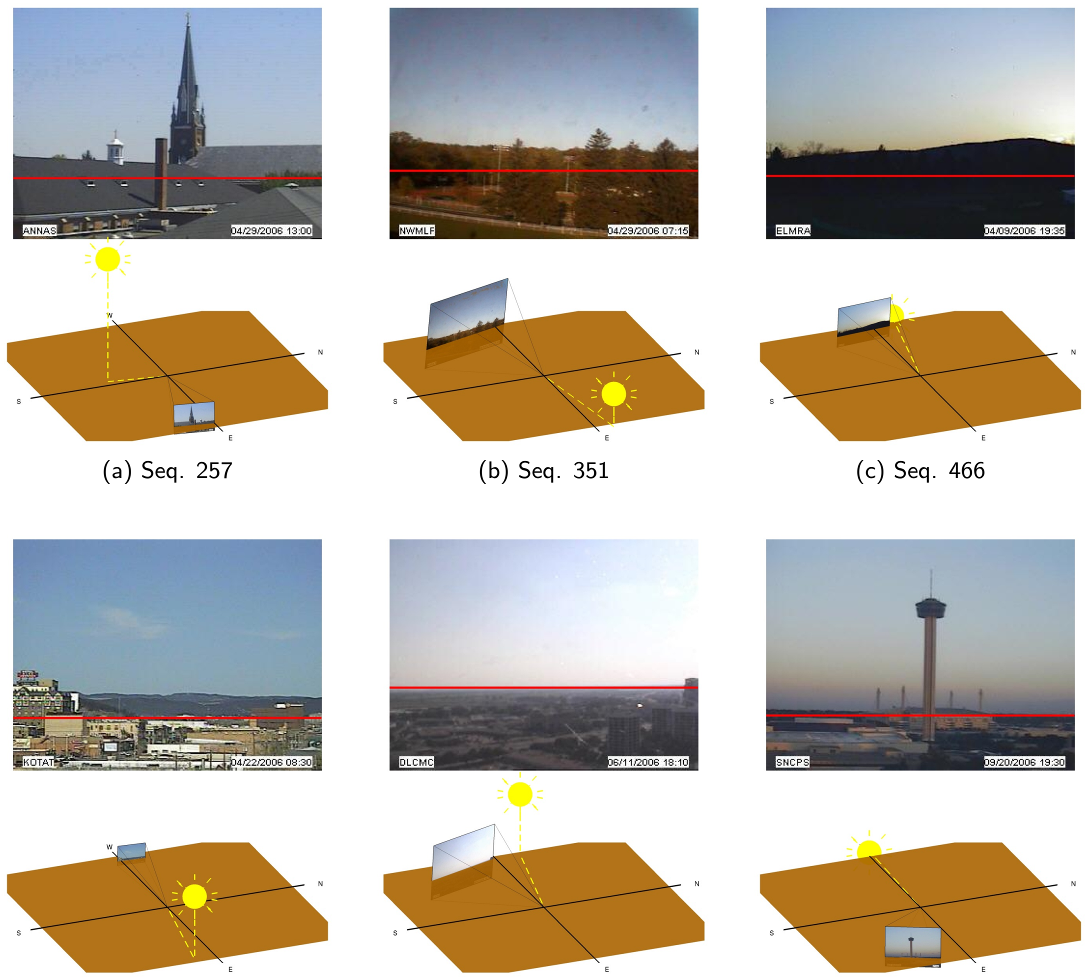


(b)

Figure 3.12: Clear sky images recovered automatically for sequence 414. Samples from the sets (a) $\mathcal{I}$, where the sun influence is not noticeable, and from (b) $\mathcal{J}$, where the sun induces changes in the sky appearance throughout the set. The actual sun position, relative to the camera, is shown in the bottom rows. Note that the relative sun position is never explicitly known to our image selection algorithm.

3.4.2.2 Visualizing a Webcam Dataset

Since the sun position might not be visible in the sequence, we cannot apply the same method for validation as we did when the sun is visible. Instead, we must rely on visible cues in the images, such as the horizon line, shadows, differently-lit building surfaces, etc. Fig. 3.13 shows qualitative results obtained by applying our method on 6 different sequences taken from AMOS database. The recovered camera parameters are consistent with the sky appearance. For instance, the sun appears to be just to the right of the image in Fig. 3.13c-(e), which is reflected in the diagrams. In Fig. 3.13d, the buildings are brightly lit, which indicates that the sun must be behind the camera. Similarly, the tower in Fig. 3.13f is front-lit by the sun, but it has an orange hue, so the sun must be near the horizon line,



(a) Seq. 257 (b) Seq. 351 (c) Seq. 466

(d) Seq. 569 (e) Seq. 601 (f) Seq. 634

Figure 3.13: Camera parameters recovered from the sky appearance. The horizon line is represented by the red line on the images. Below each image is a drawing illustrating the recovered camera-sun geometry. The sun is drawn at the location corresponding to the date and time of the image. The brown plane is horizontal, and intersects with the image plane at the horizon line.

and behind the camera as well. Shadows also hint to the sun position: right of the camera in Fig. 3.13a, and behind in Fig. 3.13b. We also note that the recovered horizon lines are aligned with the real one when visible, as in Fig. 3.13e-(f). The horizon line position in the image is predicted by (3.18).

3.4.3 Validation by Consistency Between the Two Approaches

When the sun is visible, we can run both algorithms on the same sequence and compare their estimates. Since both methods rely on two different sources of data (sun position and sky appearance), we quantify their performance by analyzing their consistency.

Numerical results of the estimates for the sun-based algorithm $(f_{sun}, \theta_{sun}, \phi_{sun})$, the sky-based algorithm $(f_{sky}, \theta_{sky}, \phi_{sky})$, and their agreement $(\Delta f, \Delta\theta, \Delta\phi)$ for all 22 sequences are shown in Table 3.6. Results for sequences indicated by ** are obtained *entirely automatically*. Automatic sky segmentation is performed by running the geometric context algorithm [Hoiem *et al.*, 2005] on 40 randomly chosen images from the sequence, and averaging the sky probability map. The other sequences require some degree of manual intervention, either for specifying the sky segmentation, or for retrieving images for set $\mathbf{J}$. In the latter case, the intervention consists of manually choosing 3 or 4 mostly clear days from the sequence (see Sec. 3.7 for more details).

Consistency in focal length is evaluated by using $\Delta f = \frac{|f_{sky} - f_{sun}|}{f_{sun}} \times 100$, and is found to be at most 9.3%, but typical values range from 1.8% to 6.2%. Consistency in zenith and azimuth angles is evaluated by computing the angular deviation. For the zenith angle, deviation is at most $6.5°$, with typical values from $0.3°$ to $2.5°$. For the azimuth angle, it is at most $8.1°$, with typical values ranging from $1.2°$ to $4.3°$.

Fig. 3.14 shows all the cameras drawn on the same plot, illustrating their difference in focal length, zenith and azimuth angles. The horizontal plane is shown in brown, and crosses each image at the horizon line. The parameters used to generate this drawing are recovered from the sky appearance algorithm. Since the sun is visible in all of them, the cameras either face East (sunrise) or West (sunset). This visualization provides an intuitive way to explore the cameras of a large dataset.

3.5 Estimating the Sun Visibility

While the approach presented so far gives robust estimates of the sky and camera parameters, it is not enough to describe the illumination conditions of the *scene*. This is because the visible portion of the sky is typically a small fraction of the entire sky hemisphere, and

Sequence name and location	Focal length f_c (px)			Zenith angle θ_c (°)			Azimuth angle ϕ_c (°)		
	f_{sun}	f_{sky}	Δf (%)	θ_{sun}	θ_{sky}	$\Delta\theta$	ϕ_{sun}	ϕ_{sky}	$\Delta\phi$
207 (Cherokee, KS)	447.1	430.2	3.8	97.9	97.6	0.3	88.8	90.5	1.7
257** (Riva, MD)	941	951.4	1.1	96.9	93.4	3.5	257.9	260.5	2.6
279 (Lafayette, MI)	713.4	684.9	4	91.5	91.7	0.2	114.7	118.3	3.6
291 (Minneapolis, MN)	356.8	338.7	5.1	88.8	90.3	1.5	38.9	32.3	6.6
305 (Neosho, MO)	1039.5	973.3	6.3	98.1	98.7	0.6	108.8	109.3	0.5
347** (Butler, NJ)	387.5	372.3	3.9	96.3	97.5	1.4	80.2	81.8	1.6
351** (New Milford, NJ)	381.7	393.9	3.2	97.4	97.2	0.2	104.7	107.4	2.7
414** (Elburz, NV)	1067.4	1032.4	3.3	95.9	97.1	1.2	240.6	239.1	1.5
455 (Marathon, NY)	757.3	816.3	7.8	98.3	95.8	2.5	134.1	131.8	2.3
466 (Elmira, NY)	651.6	669.4	1.1	1.45	-0.06	0.1	93.4	95.5	2.1
513 (Hunker, PA)	1204	1134	5.8	92.3	92	0.3	73	71.9	1.1
524 (N. Bloomfield, PA)	670.3	664.7	0.8	95.4	95.3	0.1	71.2	68.6	2.6
529 (Mifflinburg, PA)	884.9	918.9	3.9	89.9	90.5	0.6	56	60.3	4.3
569 (Rapid City, SD)	1389.1	1362.6	1.9	93.4	92.8	0.6	84.2	85.1	0.9
601 (Mesquite, TX)	442.8	484.3	**9.3**	93.3	94.1	0.8	105.8	109.4	3.6
608** (Tyler, TX)	355.9	357.4	0.4	95.3	97.5	2.4	47.6	52	4.4
630 (Pleasanton, TX)	474.7	473.6	0.2	90.5	91.9	1.4	100.9	102.1	1.2
634 (San Antonio, TX)	665.8	706.6	6.1	98.2	95.2	3	242.4	244.4	2
652 (Delta, UT)	363.6	395.5	8.8	103.6	100.7	2.9	265.2	265.3	0.1
655 (Tropic, UT)	1430.8	1363.6	4.7	94.2	95.4	1.2	289.9	292.2	2.3
695** (Danville, VA)	700	714.8	2.1	92.4	92.4	0	59.3	51.3	**8**
701 (Darrington, WA)	632.2	652.2	3.2	105.2	98.7	**6.5**	124.8	130.4	5.6

Table 3.6: Quantitative comparison of camera calibration results obtained by the two methods presented in this chapter: the sun position and the sky appearance. For each method, estimated values for f_c, θ_c and ϕ_c are shown and indicated by their corresponding subscripts. Consistency is evaluated by comparing the values together and is indicated by Δ. Worst consistency results are indicated in bold. All images have 320×240 pixels dimension. Results were automatically recovered for sequences indicated by **, the others required manual intervention at some stage in the process, either to define the sky region, or to select images for set $\mathcal{J}$.

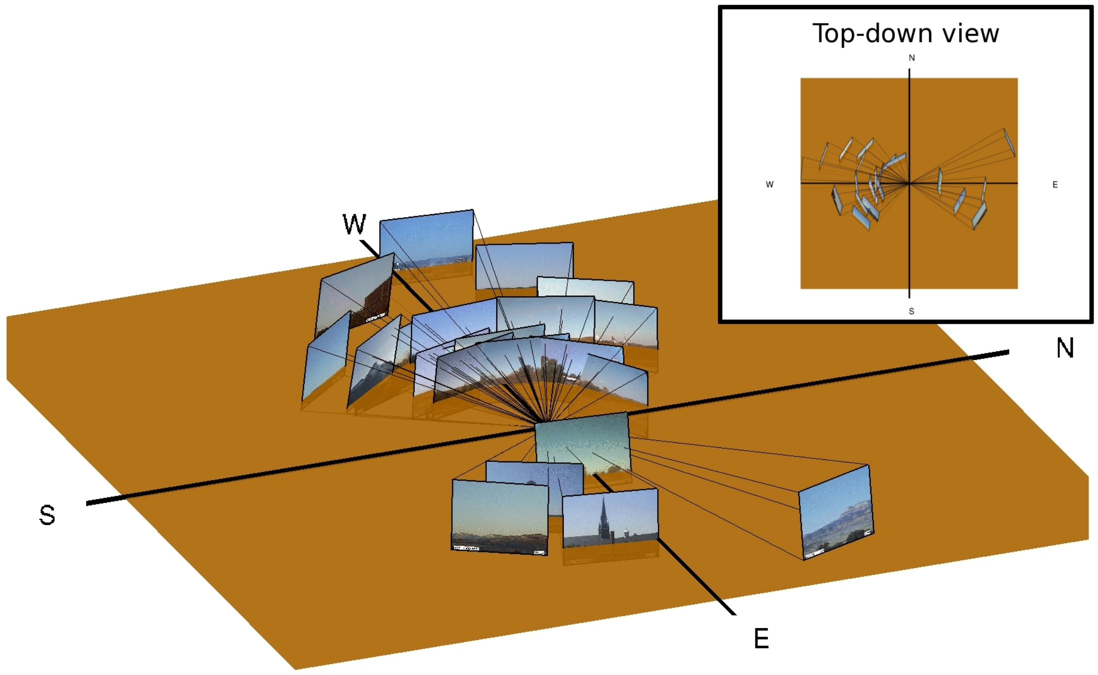


Figure 3.14: All the 22 cameras used in the consistency evaluation shown on the same display. The parameters used to generate the figure were obtained by using the sky-based algorithm. All cameras are either facing East or West, so the sun is visible at dawn or dusk respectively. The inset shows a top-down view for better visualization of the cameras azimuths.

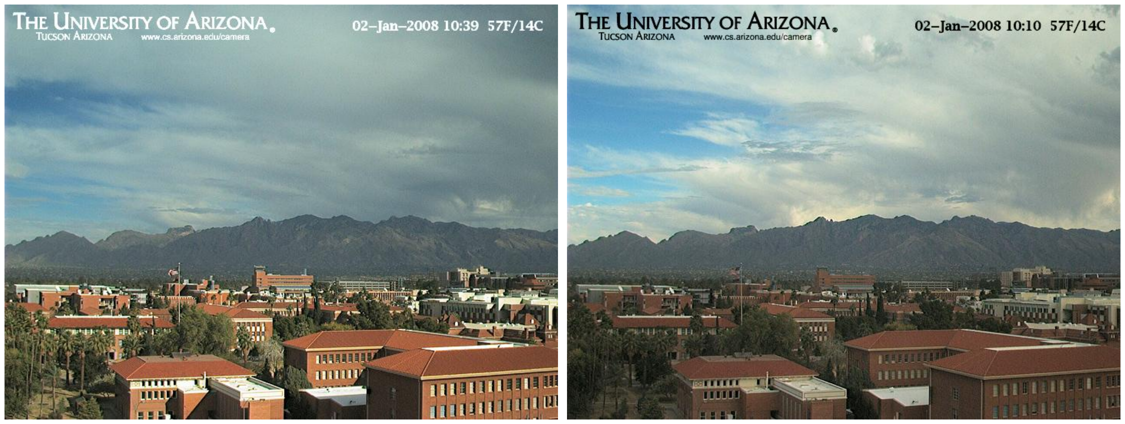

THE UNIVERSITY OF ARIZONA
TUCSON ARIZONA www.cs.arizona.edu/camera
02–Jan–2008 10:39 57F/14C
THE UNIVERSITY OF ARIZONA
TUCSON ARIZONA www.cs.arizona.edu/camera
02–Jan–2008 10:10 57F/14C

Figure 3.15: An example of similar sky appearance, but very different scene illumination. We rely on scene-based features to determine whether the sun is visible or not.

much of it remains unobserved within the field of view. For instance, Fig. 3.15 shows two similarly-looking cloudy skies with scenes illuminated differently: in the first, the sun shines from the right of the camera on the scene, but not in the second. How can we then determine whether the sun is occluded by clouds or other objects when the full sky hemisphere is not observed? For this, we must rely on cues provided by the visible scene itself.

Consider the mean image of a large webcam sequence: it shows no preferred illumination direction and resembles a scene captured under overcast skies (where sun is hidden behind clouds). Only small deviations from this mean image are likely when the sun is not visible to the scene. But larger deviations (bright or dark) from the mean are more likely when the sun is visible. Based on this observation, we compute scene features that can be used as a measure of sun visibility. In order to avoid the scene-dependent hue, the features are computed in the saturation-value space. We first compute the ratios of saturation and value of the current image w.r.t the mean image, and then compute a 2-D joint histogram of these ratio images. For an image where the sun is not visible, the joint histogram will have a dominant peak near (1,1). On the other hand, when the sun is visible, the shadow regions result in multiple peaks or a wider spread in the joint histogram.

We validated the effectiveness of our representation by evaluating it on a set of 400 randomly chosen images that cover a wide range of scene and sky appearances. We manually labelled each image as to whether or not the scene appears to be sunny. We compare the performance of our features against three others, also computed over the saturation-value space—(a) 2-D joint histogram of saturation and value of a single image; (b) Histogram of gradients of an image as a measure of local contrast, computed on the strong edges only (determined by canny edge detector with high threshold of 0.1); (c) first four moments of each of the marginal histograms of the previous features. A k-NN classifier is used to test the performance of the features, where the optimal k is determined by 10-fold cross-validation. The k nearest neighbors are found by using the χ^2 distance for the histogram-based features, and L2 for the moments. The visibility classification accuracy for each of these features are shown in Fig. 3.16. Our features are computed relative to the mean image and outperform the others.

This nearest-neighbor approach results in a binary estimate for V: the sun is either

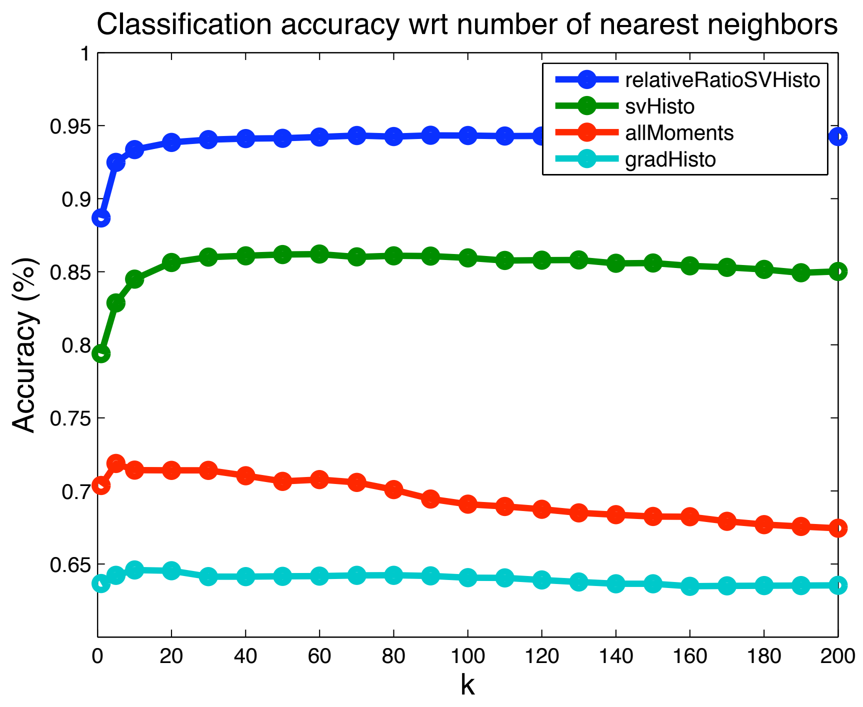


Figure 3.16: Classification accuracy (in %) for each feature tested on our training set, using a k nearest neighbor algorithm. Each feature is evaluated by varying k and performing 10-fold cross validation on the training set.

completely visible, or completely occluded. We can also use the same approach to provide an estimate of the *partial* sun visibility, i.e., how much is the sun being occluded. We estimate the sun visibility coefficient V by taking the ratio of the number of nearest neighbors with fully visible suns divided by k. Note that we currently cannot evaluate our partial visibility estimation method quantitatively since obtaining ground truth would require knowing the thickness of occluding clouds even if they are not visible. Qualitatively however, results are satisfying: the shadows generated under a partially-occluded sun match those of the scene (see Fig. 4.9a for example).

3.6 Estimating the Sky Turbidity and Cloud Cover

Now that we have recovered camera parameters, either from the sun position or sky appearance, we demonstrate how to use the same physically-based model to estimate the last remaining parameters in our natural illumination model—the sky turbidity and cloud cover—even in challenging weather conditions. Until now, we have only dealt with clear skies, but alas, this is not always true! In this section, we present a novel cloud segmentation algorithm which will allow us to deal with any type of weather.

Clouds exhibit a wide range of textures, colors, shapes, and even transparencies. Segmenting the clouds from the sky cannot be achieved with simple heuristics such as color-based thresholding as they are easily confounded by the variation in their appearances. On the other hand, our physically-based model predicts the sky appearance, so any pixel that differs from it is an outlier and is likely to correspond to a cloud. Using this intuition, we now consider two ways of fitting our model to skies that may contain clouds. We perform all processing in the xyY color space because it was determined that it offers the best agreement with the Perez sky model in [Preetham *et al.*, 1999].

3.6.1 Least-squares Fitting

The first idea is to follow a similar approach as we did previously and fit the model (3.15) in a non-linear least-squares fashion by adjusting the coefficients a, b, c, d, e and the unknown scale factor k independently in each color channel. This approach was proposed in [Yu and Malik, 1998], and works quite well in the absence of clouds. When clouds are present, we observe that fitting 5 coefficients gives too much freedom to the model, so we constrain the optimization and reduce the number of variables by following [Preetham *et al.*, 1999] and expressing the five weather coefficients as a linear function of a single value, the turbidity t. Strictly speaking, this means minimizing over $\mathbf{x} = \begin{bmatrix} t & k^{(1)} & k^{(2)} & k^{(3)} \end{bmatrix}$:

$$\min_{\mathbf{x}} \sum_{l=1}^{3} \sum_{p \in \mathcal{P}} \left(y_p^{(l)} - k^{(l)} g(u_p, v_p, \theta_s, \phi_s, \tau^{(l)}(t)) \right)^2 \quad , \tag{3.20}$$

where l indexes the color channel. Here the camera parameters are fixed, so we omit them for clarity. The vector $\tau^{(l)}(t)$ represents the coefficients $(a, \ldots, e)$ obtained by multiplying the turbidity t with the linear transformation $M^{(l)}$: $\tau^{(l)}(t) = M^{(l)}[t \quad 1]^\mathsf{T}$. The entries of $M^{(l)}$ for the xyY space are given in the appendix in [Preetham *et al.*, 1999]. The $k^{(l)}$ are initialized to 1, and t to 2 (low turbidity). Unfortunately, solving this simplified minimization problem yields unsatisfying results. The L2-norm is not robust to outliers, so even a small amount of clouds will bias the results.

3.6.2 Regularized Fitting

In order to increase robustness to outliers, we compute a data-driven prior model of clear skies $\mathbf{x_c}$, which we use to add 2 terms to (3.20): 1) we assign more weight to pixels we believe are part of the sky; and 2) we penalize parameters with a large L2 divergence from the prior. Equation (3.20) becomes

$$\min_{\mathbf{x}} \sum_{l=1}^{3} \sum_{p \in \mathcal{P}} w_p \left(y_p^{(l)} - k^{(l)} g(u_p, v_p, \theta_s, \phi_s, \tau^{(l)}(t)) \right)^2 + \beta \|\mathbf{x} - \mathbf{x_c}\|^2 \ , \tag{3.21}$$

where, $w_p \in [0, 1]$ is a weight given to each pixel, and $\beta = 0.05$ controls the importance of the prior term in the optimization. We initialize $\mathbf{x}$ to the prior $\mathbf{x_c}$.

Let us now look at how $\mathbf{x_c}$ is obtained. We make the following observation: clear skies should have low turbidities, and they should be smooth (i.e., no patchy clouds). Using this insight, if minimizing (3.20) on a given image yields low residual error and turbidity, then the sky must be clear. We compute a database of clear skies by keeping all images with turbidity less than a threshold (we use 2.5), and then keep the best 200 images, sorted by residual error. Given an image, we compute $\mathbf{x_c}$ by taking the mean over the K nearest neighbors in the clear sky database, using the angular deviation between sun positions as a distance measure (we use $K = 2$). This allows us to obtain a prior model of what the clear sky should look like at the current sun position. Note that we simply could have used the values for $(a, \ldots, e)$ from Sec. 3.3.1.1 and fit only the scale factors $k^{(l)}$, but this tends to over-constrain, so we fit t as well to remain as faithful to the data as possible. For example, the mean estimated turbidity is $t = 2.06$ for Sequence 257, very close to the clear sky model $t = 2.17$ used in Sec. 3.3.1.1.

To obtain the weights w_p in (3.21), the color distance λ between each pixel and the prior model is computed and mapped to the $[0, 1]$ interval with an inverse exponential: $w_p = \exp\{-\lambda^2/\sigma^2\}$ (we use $\sigma^2 = 0.01$ throughout this chapter). After the optimization is over, we re-estimate w_p based on the new parameters $\mathbf{x}$, and repeat the process until convergence, or until a maximum number of iterations is reached. The process typically

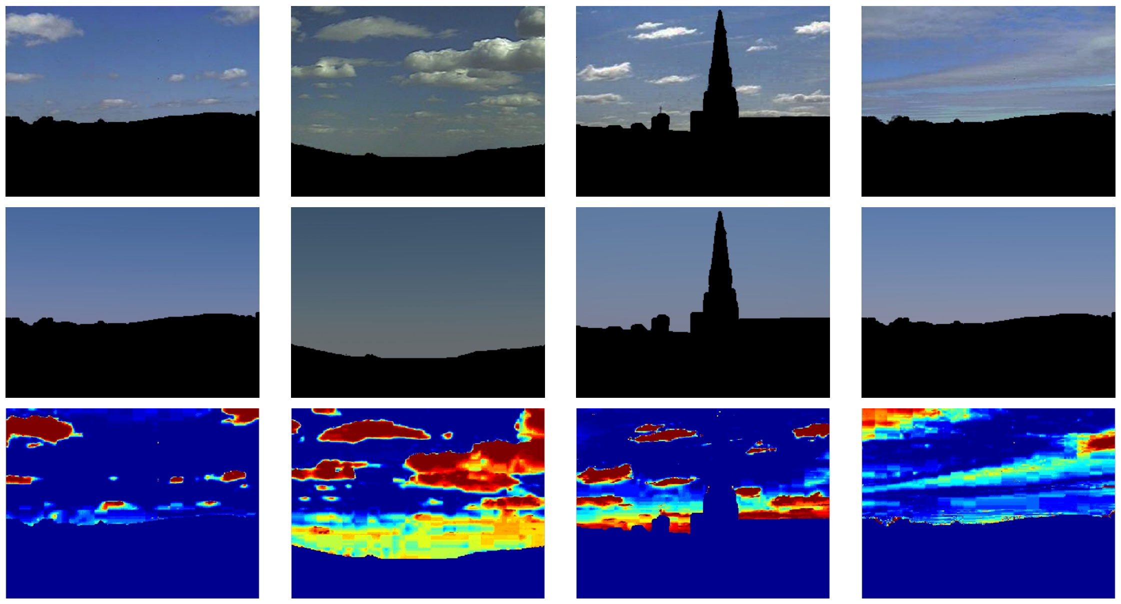


Figure 3.17: Sky-cloud separation example results. *First row*: input images (radiometrically corrected). *Second row*: sky layer. *Third row*: cloud segmentation. The clouds are color-coded by weight: 0 (blue) to 1 (red). Our fitting algorithm is able to faithfully extract the two layers in all these cases.

converges in 3 iterations, and the final value for w_p is used as the cloud segmentation. Cloud coverage is then computed as $\frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}} w_p$.

3.6.3 Results

Fig. 3.17 shows typical results of cloud layers extracted using our approach. Note that unweighted least-squares (3.20) fails on all these examples because the clouds occupy a large portion of the sky, and the optimization tries to fit them as much as possible, since the quadratic loss function is not robust to outliers. A robust loss function behaves poorly because it treats the sky pixels as outliers in the case of highly-covered skies, such as the examples shown in the first two columns of Fig. 3.18. Our approach injects domain knowledge into the optimization by using a data-driven sky prior, forcing it to fit the visible sky. Unfortunately, since we do not model sunlight, the estimation does not converge to a correct segmentation when the sun is very close to the camera, as illustrated in the last two columns of Fig. 3.18.

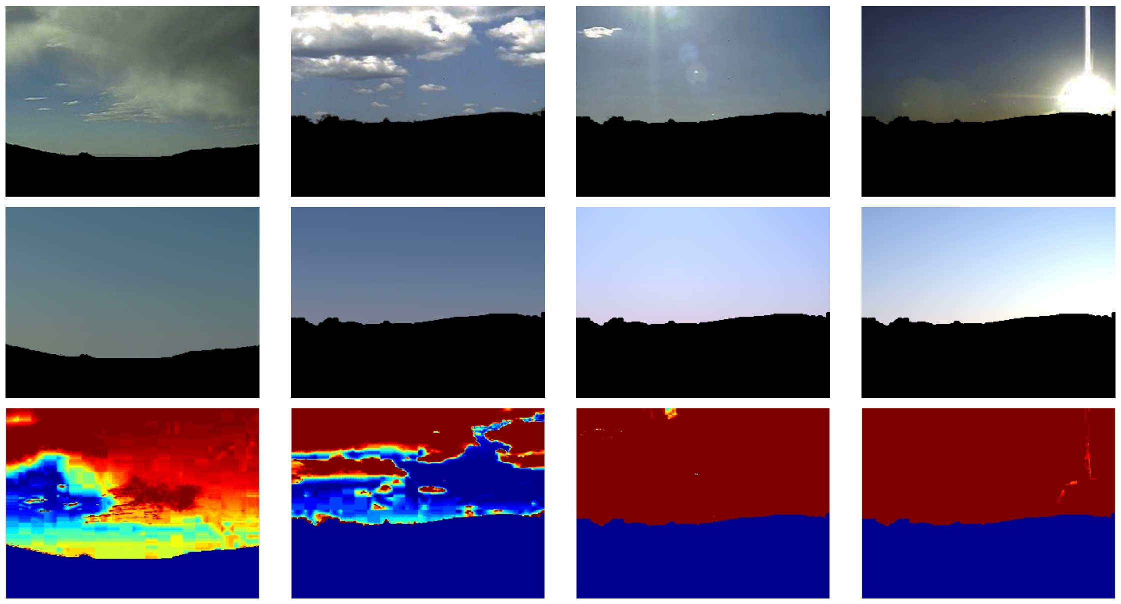


Figure 3.18: More challenging cases for the sky-cloud separation, and failure cases. *First row*: input images (radiometrically corrected). *Second row*: sky layer. *Third row*: cloud layer. The clouds are color-coded by weight: 0 (blue) to 1 (red). Even though the sky is more than 50% occluded in the input images, our algorithm is able to recover a good estimate of both layers. The last two columns illustrate a failure case: the sun (either when very close or in the camera field of view) significantly alters the appearance of the pixels such that they are labeled as clouds.

3.7 Discussion

We discuss three important problems related to the sky-based calibration algorithm that arise in practice, namely radiometric issues, varying weather conditions, and the need for date and time of capture. It is important to understand the various elements other than the camera parameters that may also affect the sky appearance, in order to factor out their influence and isolate the effects solely due to camera parameters.

3.7.1 Radiometric Issues

Our sky-based algorithm relies on the sky pixel intensities and assumes that they faithfully represent the real sky radiance. Unfortunately, radiance undergoes a series of unknown transformations [Kim and Polleyfeys, 2008] before being observed as pixel intensities. In particular, we must consider gain, dynamic range, camera response function, vignetting, and sensor noise. In this section, we discuss how each one of these unknown transformations are dealt with in this work.

Gain is an unknown scale factor which is applied to radiance, and can vary from one image to the next because of Automatic Gain Control (AGC). Our approach is insensitive to AGC because it estimates an unknown scale factor k at each image (see Sec. 3.3.1.3), in which the gain gets incorporated.

The exposure controls the amount of light that is captured by the camera, and may or may not vary across images depending on the camera. A particular exposure may result in under-exposed or saturated pixels when the corresponding scene is too dark or too bright, respectively. These incorrectly-exposed pixel values have been truncated to fit the dynamic range of the camera, therefore are not accurate representations of the scene radiance. This problem can be solved by ignoring pixels that have intensity less than $2/255$ or higher than $254/255$ in the optimizations.

The camera response function is a (typically non-linear) transformation that maps radiance values to pixel intensities. This is usually computed by acquiring several images of the same scene at different exposures [Debevec and Malik, 1997]. Unfortunately, we cannot assume this is the case in an image sequence because the frequency of acquisition might be too low, and illumination conditions might be different from one frame to the next which breaks the constant radiance assumption of such methods. Instead, we mentioned that we rely on [Lin *et al.*, 2004], which estimates the response function from color edges computed over several images. This method suffers from two important drawbacks: 1) selection of the weight λ, which controls the relative importance between the data and prior terms in the optimization, has to be done empirically; and 2) the images might not have enough different colors to cover the entire RGB cube, so the set of available edges might be restricted to a small region in the color space. Future work includes recovering the camera response function from techniques which are geared towards using multiple images from the same [Kim *et al.*, 2008] or different [Kuthirummal *et al.*, 2008] scenes as input.

Vignetting is a common issue that arises when dealing with low-quality, wide-angle lenses typical of webcams. It can significantly alter the intensity of pixels located near the corners of the image. Although elegant vignetting removal solutions have been proposed [Kim and Polleyfeys, 2008; Kuthirummal *et al.*, 2008; Zheng *et al.*, 2006], we simply ignore pixels that are far from the image center (e.g., 120 pixels for a 320×240 pixel image), and have found this approximation to be sufficient with our test sequences.

The last issue is the one of sensor noise, which can be significant in low-quality webcam images. Because our algorithm operates on several randomly-chosen sky pixels gathered across many images, and the Gaussian noise assumption underlying our least-squares minimization approach, we have found our algorithm to be robust to the noise level present in our test sequences, which is also confirmed by the synthetic experiments performed in Sec. 3.3.3.2.

3.7.2 Weather Conditions

Although we already presented in Sec. 3.6 how we can represent challenging weather conditions once the camera parameters have been recovered, recall that finding these very parameters relied on clear sky images in the first place. We presented in Sec. 3.4.2.1 two algorithms that automatically select clear skies to build sets $\mathcal{I}$ and $\mathcal{J}$ from a large set of images. Unfortunately, because camera parameters are unknown initially, we had to rely on image-based heuristics to guide these algorithms. We now discuss how these algorithms are affected by slight variations in weather conditions, which in turn has an effect on the camera parameters estimation. Luckily, both of them need not be perfect, and we observe that they are robust to clouds being present in roughly $10-15\%$ of their respective input image sets.

Empirically, we observe that building set $\mathcal{I}$ (clear skies where the sun does not affect the sky appearance) succeeds in approximately 90% of the time. The main failure case is when a thin layer of semi-transparent clouds cover the entire image, and smoothly modify the vertical sky gradient. Additionally, presence of large amounts of haze close to the horizon is another source of noise because it is not predicted by the clear sky model. For the set $\mathcal{J}$ (clear skies where the moving sun effect is visible), performance decreases and the automatic method is used in only 25% of the sequences in Table 3.6. The algorithm typically fails when the sun is very close to the camera field of view, and induces very large changes in the sky appearance. Unfortunately, these failure cases can only be detected by manually inspecting the resulting images, so future work includes determining a better way of finding clear sky images that is more robust to stronger variations in weather.

3.7.3 Are Date and Time of Capture Necessary?

We show in Appendix A.1 that, given only clear sky images and date and time of capture, it is possible to estimate the camera focal length, zenith and azimuth angles, and even GPS coordinates. But can we go further? Could we also recover all this information, given just the images as input?

One possibility would be to have a webcam which captures at precise regular intervals, closely spaced in time (e.g., every minute), over a very long period of time (e.g., one year). In short, this regular time spacing gives their time of capture up to translation and scale. From only the images of such a webcam, it should be possible to track the sun position and get a good estimate of sunset and sunrise (i.e., when the predicted θ_s given θ_c and f_c is equal to $90°$). Given many sunset or sunrise estimates, it might be possible to recover the time translation and scale factor by correlating their *relative* sunset/sunrise times with real times gathered from an astronomical almanac. This could be used to recover the actual date and time of capture of each image.

Unfortunately, real webcams are not so regular: their capture frequency may vary slightly, they might become unavailable for a period of time, etc. Dropping a single frame would adversely effect the algorithm, so the feasibility of such an approach imposes undue restrictions on data capture. The date and time of capture are stored with virtually all captured images and hence can be exploited, thus avoiding such restrictions on image acquisition.

3.8 Conclusion

In this chapter, we analyze two sources of information available within the visible portion of the sky region: the sun *position*, and the sky *appearance*. In particular, we show how to use them to estimate our entire natural illumination model of Sec. 2.4 in image sequences: the sun angular direction with respect to the camera, the sun visibility, the sky turbidity and cloud cover. From the sun coordinates in images, we show how we can extract the camera focal length and its zenith and azimuth angles. For the sky appearance, we express a well-known physically-based sky model in terms of these camera parameters and fit it to clear sky images using standard minimization techniques. We test our methods on a high-quality image sequence with known camera parameters, and obtain errors of less that 1% for the focal length, 1° for azimuth angle and 3° for zenith angle. We then show that both these techniques consistently recover the same parameters on synthetic and real image sequences. We evaluate their performance by calibrating 22 real, low-quality image sequences distributed over a wide range of latitudes and longitudes. Finally, we demonstrate that by combining the information available within the sun position and the sky appearance, we can also estimate the camera geolocation, as well as its geometric parameters. Our method achieves a mean localization error of 110km on real, low-quality Internet webcams. Once the camera parameters are estimated, we show how we can use the same model to segment out clouds from sky and build a novel bi-layered representation. In the next chapter, we will see how this newfound knowledge about illumination allows us to synthesize image sequences with consistent lighting.

Chapter 4

Synthesizing Illumination-Consistent Sequences

Always remember, sir, that light and shadow never stand still.

Benjamin West (1738–1820)

In this chapter, we exploit the illumination model estimated from image sequences presented in Ch. 3, in computer graphics. For the first time, we demonstrate how objects can be transferred across webcams in a lighting-consistent manner, and even how the lighting conditions of an image can be used to realistically relight a virtual 3-D model.

4.1 Background

Despite the recent advances, visual realism in complex, real-world scenes remains largely elusive for traditional computer graphics. The main culprit appears to be the sheer complexity of our visual world—a realistic scene requires much more detailed geometry and lighting than could possibly be provided by hand. Of course, if one happens to have access to the physical site to be modeled (e.g., a monument or a movie set), then image-based modeling and rendering techniques can be employed *in situ*, producing excellent results (e.g., [Arnold *et al.*, 2003]). However, often it is not possible to travel "on location" in order

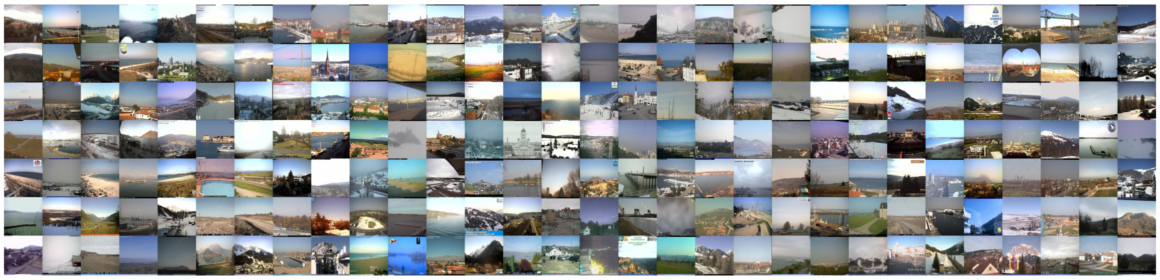


Figure 4.1: 180 images randomly selected from our dataset of 1.2 million images from 1350 webcams.

to capture the needed geometric and photometric data in person. Fortunately, over the last few years, the Internet has developed into a gargantuan depository of visual data (photos, videos, webcams, annotations, etc) captured by people all over the globe. A pressing research question is how this data could be useful in graphics as a way of "crowd-sourcing" visual realism.

Recently, the concept of *appearance transfer* has been proposed as a way of employing large datasets for synthesizing novel imagery. The basic idea is to: *match* various aspects of the given scene to one or more previously seen examples from the dataset; and *transfer* the appearance information associated with the examples onto the scene. Compared to more traditional techniques, the main difference is that the algorithmic burden is shifted from a synthesis task to a (hopefully easier) matching task. Some surprisingly successful applications of this simple idea have been demonstrated for several types of data such as unordered photo collections [Hays and Efros, 2007; Bitouk *et al.*, 2008; Chen *et al.*, 2009]. Closely related are efforts that perform "appearance mining" of photo collections to recover visual content like virtual walk-throughs [Snavely *et al.*, 2008], skies [Tao *et al.*, 2009] and even 3D models [Goesele *et al.*, 2007].

One intriguing source of visual data that has yet to be fully explored are *Internet webcams*. A webcam is a stationary camera that continuously captures a time-lapse sequence, typically at one frame every 5-10 minutes. There are tens of thousands of such webcams all over the world, providing a rich round-the-clock visual record of some of the most interesting, as well as some of the most mundane, places on the planet. The big advantage of webcam data is that, by observing *the same scene under many different lighting and weather* *conditions*, it contains much more information than can be extracted from either a single image or an unstructured video. As a result, working with this kind of data allows for the use of more principled physics-based methods to complement the mostly heuristic data-driven approaches. Several research groups have explored this source of information, recovering various scene properties including scene geometry, surface reflectance, shadows, illumination, camera parameters, and even camera geo-location [Koppal and Narasimhan, 2006; Sunkavalli *et al.*, 2007; Weiss, 2001; Sunkavalli *et al.*, 2008; Kim *et al.*, 2008; Jacobs *et al.*, 2007b]. These efforts have prepared the ground for the next step: actually using the wealth of webcam data as a source of visual realism for computer graphics applications.

4.2 Overview

In this chapter, we propose a combined data-driven / physics-based approach for using a library of calibrated outdoor webcams as a "Webcam clip art"—to inject varied, realistic appearance into the user's own scenes. These scenes could themselves be time-lapse sequences, as well as 3D models, 2D Light Stage objects, or, in some cases, even single photographs. Our aim is to transfer the right data from the appropriate source images while avoiding, as much as possible, synthesizing new appearance and illumination. In this chapter, we focus on a single aspect of appearance—illumination—where time-lapse data can provide much more information than a single image, allowing for a more physics-based approach as well as a much richer set of applications. In Ch. 7, we will see how to match other important parameters, such as geometry, scale, and local context of an object in the case of creating single images. For now, we divide these applications into two broad categories:

Illumination-consistent appearance transfer: Webcams see and record the visual life of many interesting things—buildings, monuments, mountains, beautiful skies, etc. We would like to be able to insert an object from our Webcam Clip Art library directly into a user's own time-lapsed sequence in such a way that the lighting of the object is consistent with the overall illumination of the scene *at each time instance*. For instance, using a Paris webcam, one could create a time-lapse Eiffel Tower clip art object which, when inserted into a new scene, will automatically become correctly lit.

Illuminant transfer: A webcam library could also act as a rich source of realistic, time-varying natural illumination for relighting the user's own virtual objects. By estimating the sun and sky light from the webcam, we can create a library of sky probe sequences, to be used for relighting scenes under many different illumination and weather/cloud conditions. Moreover, by hypothesizing full environment maps for a given sequence, virtual objects can be seamlessly inserted into real scenes. For instance, an architect or a sculptor wishing to see how their proposed creations will harmonize with the existing surroundings could simply install a webcam at the site (or several alternative sites), and visualize their model *in situ*, lit by a wide range of real-world illumination conditions.

These are demanding goals. To succeed will require addressing three major challenges: **1) Building a high-quality, calibrated webcam database.** Current webcam collections such as the one introduced in [Jacobs *et al.*, 2007a] are either much too small or too low-resolution to be suitable for graphics applications. Building a truly large dataset will require that Internet crawling and camera calibration all be done completely automatically. **2) Matching illumination conditions across different webcams.** This is hard because the webcams can depict completely different scenes in terms of geometric structure, color, and materials and weather, making a purely data-driven approach unlikely to succeed. **3) Estimating sky probes from a webcam.** The small field of view of a typical camera means that webcams show only a tiny part of the scene, from which the algorithm must recreate its global lighting environment.

Our approach to addressing these challenges combines the use of data-driven appearance transfer paradigm with a physics-based model of outdoor illumination. We will assume that all our webcams have only natural (daylight) illumination, that they are filmed by static cameras, and that their GPS locations are known.

While illumination has been analyzed extensively in laboratory settings, little has been done on outdoor illumination in unconstrained scenes observed in the real world by off-the-shelf cameras. [Lalonde *et al.*, 2007] also model illumination within a data-driven framework, but they do it in a purely heuristic way—via color histogram of the sky region. This coarse representation loses spatial information and can only match the global "feel" of the scene (e.g., sunny vs. overcast). A similar behavior is obtained by the sky matching approach of [Tao *et al.*, 2009]. A physics-based illumination model has been proposed by [Bitouk *et al.*, 2008] but only for the specific domain of faces (which are assumed to be Lambertian) and does not generalize to arbitrary outdoor scenes. [Haber *et al.*, 2009] apply inverse rendering on several images of the same object to estimate reflectance and illumination, but require a priori knowledge of object shape.

This chapter is organized as follows. We first discuss how we collected our webcam library (Sec. 4.3). The remainder is devoted to deriving algorithms for our two main applications, appearance transfer (Sec. 4.4) and illuminant transfer (Sec. 4.5).

4.3 Building the Webcam Clip Art Database

Creating a webcam database suitable for our needs poses several challenges. The database must be large enough to contain a wide range of appearances and illumination conditions as to be useful for graphics applications. The webcams must be of high resolution and stationary (e.g., no excessive shaking due to wind). Furthermore, all cameras within the database must be calibrated, both geometrically and radiometrically. Geometric calibration is crucial for determining the illumination direction with respect to the camera. Radiometric calibration is needed to linearize camera responses in order to fit the sky appearance model. In this section, we will briefly describe our approach for automatically collecting and calibrating our webcam database, the first of its kind in computer graphics[1].

4.3.1 Data Collection

While the number of Internet webcams is vast, many are not suitable as sources of interesting appearance data (e.g., TrafficCams, OfficeCams, SportsCams). Additionally, many (especially older) webcams use cheap cameras and offer poor resolution and image quality. For example, we initially experimented with Archive of Many Outdoor Scenes (AMOS) [Jacobs *et al.*, 2007a], the only existing dataset containing over 1,000 webcams, but rejected it due to low image resolution (320×240 for most sequences). Luckily, webcam quality and resolution have been steadily going up in recent times. For our task, we have been able

[1]Dataset available at `http://graphics.cs.cmu.edu/projects/webcamclipart`.

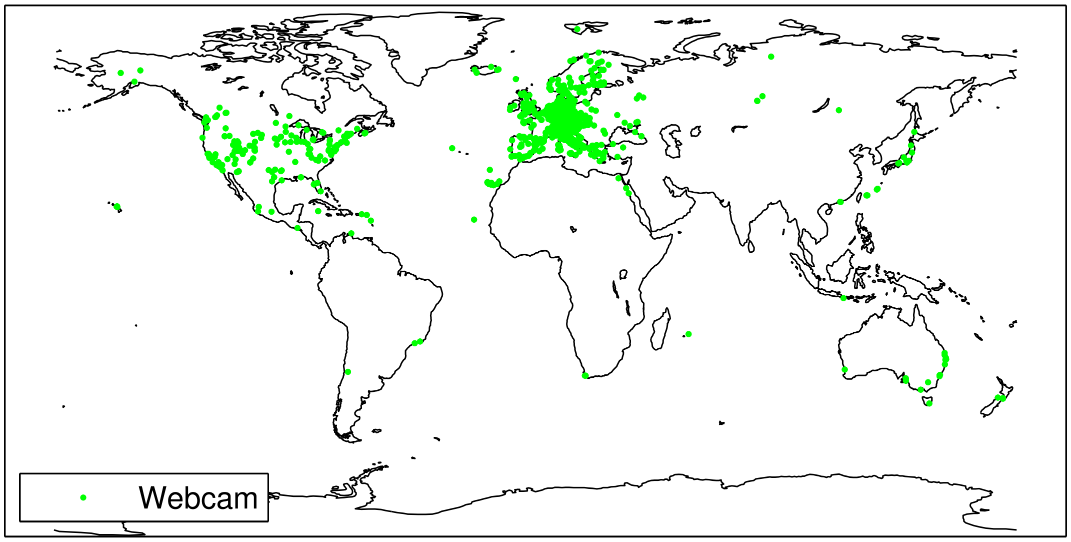


Figure 4.2: World map showing the location of all 1350 webcams in the dataset. The main concentration is in Europe and North America, but there are also several in Asia and Oceania.

to identify a source of high-quality webcams—the "webcams.travel" website [Opperman *et al.*, 2009], which contains a set of live links to 11,500 geo-tagged webcams worldwide at the time of writing.

We have developed a system that crawls the entire website and automatically finds webcam sequences where 1) the resolution is sufficient, 2) the sky is visible, and 3) the camera is static. To determine if the sky is visible, we use the geometric context algorithm [Hoiem *et al.*, 2005] and average the segmentations over several images to yield a robust sky mask. To determine if the camera is static, we first extract features (using SIFT [Lowe, 2004]) from each image of a webcam. We select only those webcams with no translation between the feature locations across frames, as in [Jacobs *et al.*, 2008].

After the crawling, we ended up with 1350 suitable webcams, for a total of over 1.2 million images (so far). The images were downloaded every 15 minutes from sunrise to sunset over at least two months, and more than one year in some cases (we continue the downloads, aiming to have at least 6 months of data from every webcam). For all webcams the location information (GPS) is provided, and every image is tagged with time-of-capture. All of our sequences have at least 640×480 pixel spatial resolution, with 10% of them with

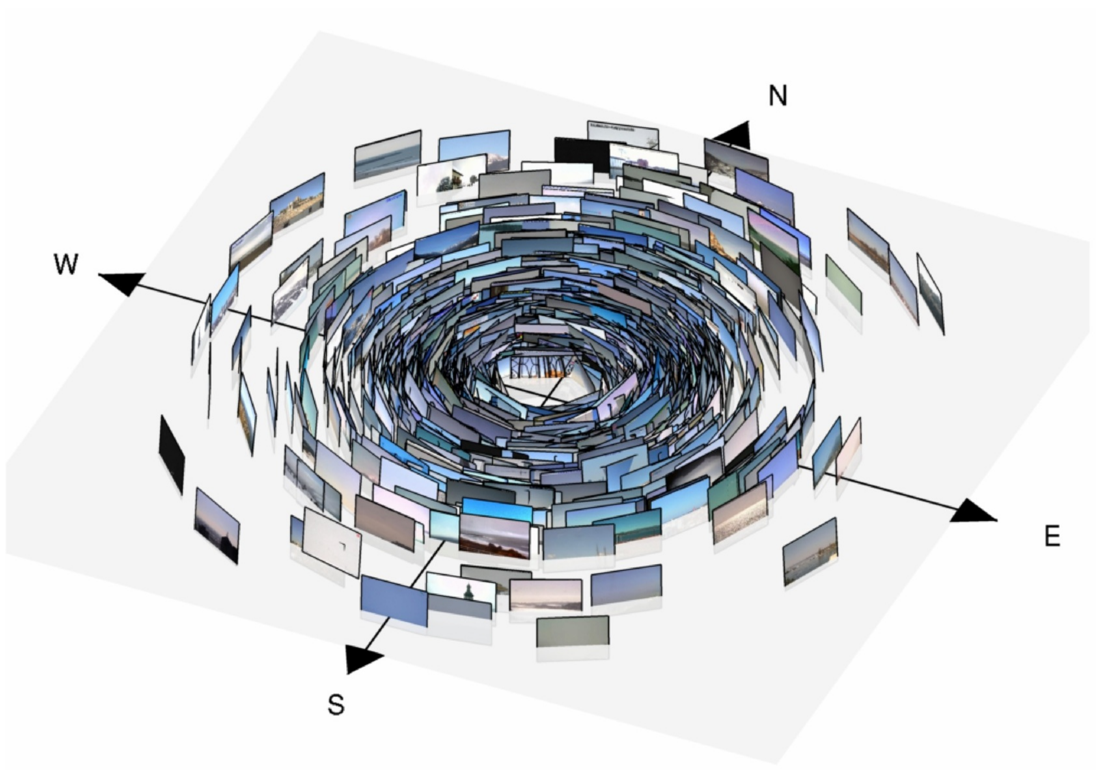


Figure 4.3: Our calibrated webcam dataset. It comprises 1350 webcams downloaded from the Internet, and calibrated using the technique introduced in Ch. 3. The display depicts their fields of view and orientations.

greater than 1024×768 resolution.

Fig. 4.1 shows a random sample of 180 sequences from our dataset. It illustrates the wide variety of scenes that are being captured: mountains, cities, landmarks, buildings, public spaces, parks, beaches, fields, airfields, streets, etc. The webcams are distributed throughout a wide range of geographic locations, as illustrated in Fig. 4.2. While the main concentrations are in Europe and North America, all five continents (except Antarctica) are represented.

4.3.2 Recovering the Natural Illumination Conditions

In addition to the raw image data, we employ the method presented in Ch. 3 to estimate the natural illumination conditions for each image in each webcam sequence. This results in a rich database of images for which we know the relative angular position of the sun with respect to the camera $(\Delta\theta_s, \Delta\phi_s)$, the sun visibility V, the turbidity t in the atmosphere, and the cloud layer C. In addition, we also have camera parameters for each sequence, namely the focal length f_c, zenith and azimuth angles (θ_c, ϕ_c). The resulting camera parameters are shown in Fig. 4.3.

4.4 Appearance Transfer Across Image Sequences

The world's webcams record the visual appearance of cities, mountains, beaches, forests and skies, under a variety of illumination and weather conditions. But while the space of all possible scenes may well be infinite, the space of illumination conditions they experience might not be that large. Indeed, given a large-enough database of scenes it is extremely likely that many will share similar orientation with respect to the sun and, given long-enough recording time, similar skies and weather. And if a similar scene can be found, then appearance can simply be transferred from it to the target scene, without having to explicitly model the complex physics and material properties. Thus, the collection of webcams can be treated as a "clip art" from which users can insert objects into their own time-lapse photographs in an illumination-consistent way. As with any data-driven approach, just growing the number of webcams in the clip art library will further improve performance by increasing the number of illumination matches for transfer. In this section, we describe our techniques for matching illumination and transferring objects from one webcam to another.

4.4.1 Matching Illumination Across Scenes

Matching illuminations of completely different scenes is a challenging problem. One scene may be a natural landscape and the other may consist of skyscrapers. In this case, simple image-based heuristics will not suffice because they do not take into account either the viewing and illumination geometry, the type of weather, or the sun visibility. However, this is exactly the type of information we can estimate from webcam sequences, as described in Sec. 4.3.2.

Consider two images captured by different webcams. In order to test whether the appearance of an object in one can be transferred to another, we must match:

1. the camera azimuth angle relative to the sun $\Delta\phi_s$, which ensures, for example, whether the sun is either to the left or right or behind the cameras;

2. the camera zenith angles θ_c, which measures the orientation of the camera with respect to the ground plane, thereby accounting for variations in viewpoint for objects;

3. the sun zenith angle θ_s which effects sky color especially during sunrise and sunset;

4. the visibility V of the sun, captured by the features from Sec. 3.5;

5. turbidity t in the atmosphere;

6. the cloud coverage denoted by the norm $||C||$ of the cloud layer.

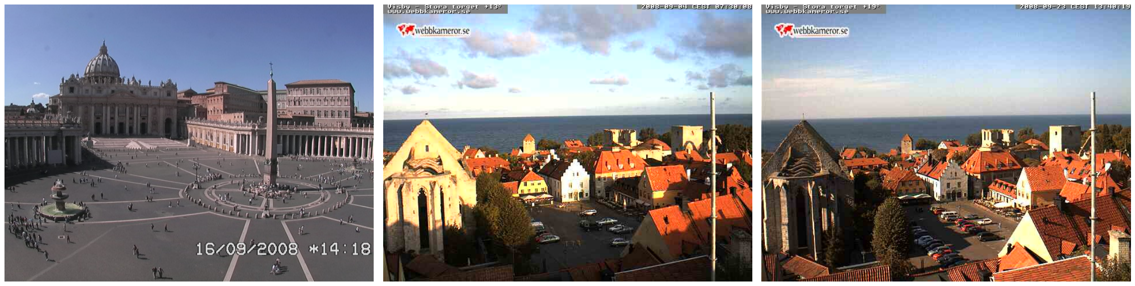
Since we do not expect different webcams to exactly match all of these five quantities, we define an aggregate matching function M between the skies of source and destination images i and j, as the weighted linear combination:

$$\begin{aligned} M = w_1\angle(\Delta\phi_s^i, \Delta\phi_s^j) + w_2\angle(\theta_c^i, \theta_c^j) + w_3 w_z \angle(\theta_s^i, \theta_s^j) + \\ w_4\chi^2(V^i, V^j) + w_5||(C^i, C^j)||_2 + w_6||(t^i, t^j)||_2 \, , \end{aligned} \tag{4.1}$$

where, $\angle(\cdot,\cdot)$ is the angular difference, and $\chi^2(\cdot,\cdot)$ the chi-square distance between two histograms.

The weights $w_1 = 2$, $w_2 = 1.5$, $w_3 = 1.5$, $w_4 = 1.5$, $w_5 = 1$, and $w_6 = 1$ are used in all the examples shown in the chapter. Note that errors in sun and camera angles or sun visibility are more easily perceived that turbidity or cloud cover, hence their larger relative weights. Furthermore, matching the colors of the sky during sunrise or sunset is critical for perception. We set an additional weight $w_z = \sin(\theta_s^i)$ that increases the importance of the term when the sun is close to the horizon ($\theta_s^i = 90°$). Each of the terms in the above functions are equalized by normalizing them to a range of [0–1]. A nearest neighbor algorithm is used to match the function M of one webcam against the library of source webcams.

Fig. 4.4 compares our new illumination matching function M with the one proposed by Lalonde *et al.* [2007]. While their approach is good at representing the general scene mood



(a) Input image (b) [Lalonde *et al.*, 2007] (c) Ours

Figure 4.4: Sky matching comparison against [Lalonde *et al.*, 2007]. The input image from the Vatican sequence (a) is matched against frames from the Visby sequence using Photo Clip Art (b) and our technique (c). While Photo Clip Art can capture the overall "feel" of the scene (sunny, in this case), note that the shadow direction is completely wrong.

(e.g., sunny vs. cloudy), it does not accurately capture the sun position, so it cannot be used to transfer the appearance of objects and scenes across webcams in a physically-consistent way.

4.4.2 2-D Appearance Transfer Across Webcams

Once the illuminations of the source and destination webcams are matched, the object of interest can be matted from the source webcam and composited into the destination. Fig. 4.5 shows three different appearance transfer results: 1) skyscrapers from Tokyo are placed into an image sequence in Berkeley, CA (Fig. 4.5a); 2) the Eiffel Tower in Paris is composited into the Berlin skyline (Fig. 4.5b); and 3) the sky (it's just another object!) can likewise be transferred, here from Berkeley to Portugal (Fig. 4.5c). To compensate for color differences that might exist between cameras, we post-process the object by re-coloring according to the scene colors, following [Reinhard *et al.*, 2001]. When the sky is transferred as in Fig. 4.5c, pixels are automatically warped according to the field of view of the cameras. Notice the beautiful sunset effect in Fig. 4.5a where the reflection of the sun, the colors of the buildings and the sky appearance are all consistent. The consistency between the illumination conditions of the transferred object and those of the destination scene is what makes the results appear so realistic.

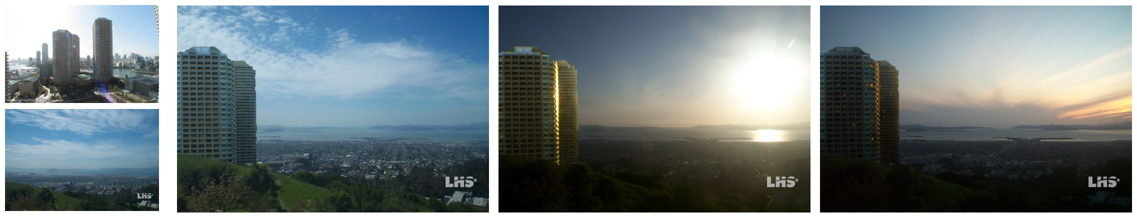
Illumination can be matched for each frame of the source / destination webcam. However, in order to ensure smooth transitions between frames adjacent in time, we add a constraint which penalizes large changes in appearance in the resulting video. We first compute the mean histogram of scene intensity differences over all pairs of adjacent frames in the sequence containing the source object. We then penalize images whose histogram of object intensity differences with respect to the previous frame differs from the mean histogram (using the χ^2 distance measure). We ensure global consistency using dynamic programming as in [Schöld *et al.*, 2000].

The success of appearance transfer depends on the quality of matches recovered from the database. Rare illumination conditions such as storms and dense fog for example, are harder to find and may result in erroneous relighting results. Since not every webcam can be matched to every other webcam (e.g., sun positions at the earth's equator and pole do not overlap), we assume that the webcams are distributed across the entire globe. As with other data-driven approaches, many current limitations will be addressed by simply adding more data. In addition, while this type of compositing maintains consistency in the appearance of the object, shadows cast by the object onto others (or vice versa) are hard to simulate without knowing 3D geometry of the scene. While shadow detection and transfer solutions have been proposed [Weiss, 2001; Lalonde *et al.*, 2007], they may not work in arbitrarily complex scenes with occlusions, moving objects, etc. While this topic is part of our future investigations, in the following section we present a different way to accurately handle object shadows—by using 3D virtual object models.

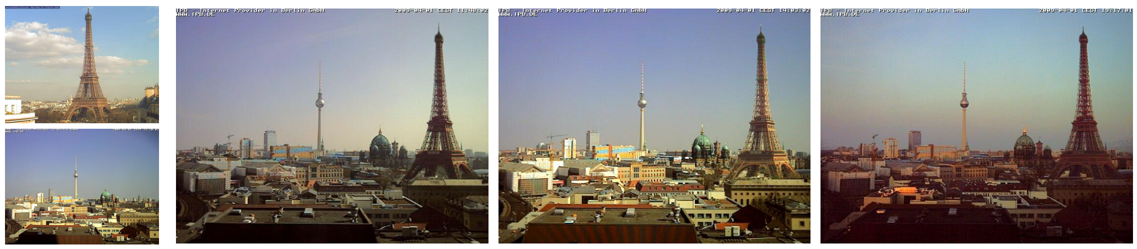
4.5 Illuminant Transfer in Image Sequences

Webcam sequences can be used to transfer pixel data across scenes, as we demonstrated earlier, but we would also like to move beyond pixels, and capture as much of the illumination information as we can. This will allow us to insert synthetic 3D objects seamlessly into a webcam sequence, and also to use the natural light from a webcam to relight a novel 3D scene. In this case, knowledge of geometry will result in physically-consistent shadows.

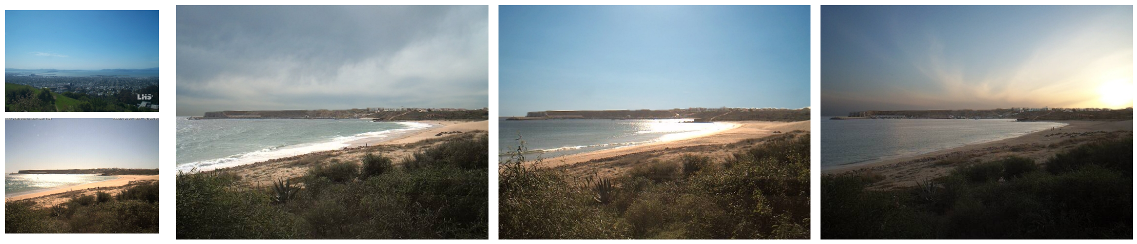
The way that the relighting problem is typically addressed in graphics is by capturing natural light from the real scene and using it as an *environment map* to light virtual ob-



(a) Object transfer (building)



(b) Object transfer (Eiffel tower)



(c) Sky transfer

Figure 4.5: Appearance transfer between webcams. The first column shows example frames from the webcams corresponding to the object (top) and destination (bottom). The other three columns show (a) buildings transferred from Tokyo to Berkeley and "re-lit" by finding an image in the Tokyo sequence with illumination that matches that of Berkeley; (b) the Eiffel Tower transferred in Berlin, notice how the shading effects are consistent across the scene; (c) the sky transferred from Berkeley to Portugal, in which case the scene can be automatically "re-lit" in an illumination-consistent way, observe how the reflections on the water are consistent with the sun position.

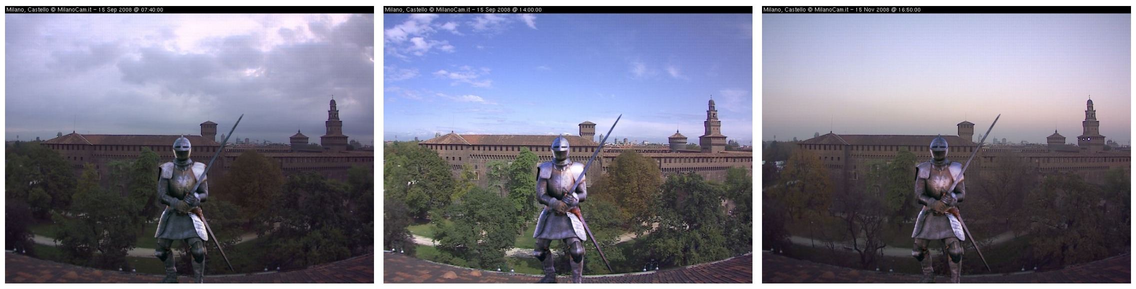


Figure 4.6: Rain or shine, the knight stands guard at the castle. Our technique can be used to render objects captured in a Light Stage [Debevec *et al.*, 2002].

jects [Blinn and Newell, 1976b; Debevec, 1998]. An environment map is a sample of the plenoptic function at a single point in space capturing the full sphere of light rays incident at that point. It is typically captured by either taking a high dynamic range (HDR) panoramic photograph from the point of view of the object, or by placing a mirrored ball at the desired location and photographing it. Such an environment map can then be used as an extended light source for rendering synthetic objects.

Given only an image or a webcam sequence however, it is generally impossible to recover the true environment map of the scene, since an image will only contain a small visible portion of the full map (and from the wrong viewpoint besides). However, there is psychophysical evidence suggesting that an approximation to the environment map which is consistent with the target image is typically enough to create believable lighting effects [Dror *et al.*, 2004]. Consequently, a simple technique for environment map estimation from a single image has been proposed by Khan *et al.* [2006] that work reasonably well for its intended setting. However, we have found it inadequate for our task mainly due to its low dynamic range and inability to estimate even approximate illumination direction. Both problems stem from the fact that the sun and the sky—the main illuminants in an outdoor environment—are not being explicitly modeled. Coincidentally, this is exactly the information that we can reliably recover from webcam sequences (Sec. 4.3.2). Here we propose to use our model of natural illumination to estimate high dynamic range environment maps at each frame. Since we are dealing with outdoor images, we can divide the environment map into two parts: the *sky probe* and the *scene probe*. We now detail the process of building a realistic approximation to the real environment map for these two parts.

4.5.1 Creating a Sky Probe

Our sky probe is composed of the sky layer, the sun layer, and the cloud layer. The sky layer is generated in a straightforward way from the sky parameters that were computed in Sec. 4.3.2. The estimated turbidity and camera parameters allow us to use (3.8) and extrapolate the sky appearance on the entire hemisphere, even though only a small portion of it was originally visible to the camera.

The camera calibration procedure also yields the relative position of the sun with respect

Color channel	α	β
R	1.4243×10^{10}	0.2187
G	1.4463×10^{10}	0.2399
B	1.3×10^{10}	0.2889

Table 4.1: Recovered parameters after fitting our parametric model to the sun data from the HDR sky database of Stumpfel *et al.* [2004].

to the camera. For the sun layer, we can use this to position a synthetic sun at the desired location. But how should it look, and how bright should it be? We rely on the HDR sky dataset of Stumpfel *et al.* [2004] to estimate sun appearance and brightness. In all frames of the dataset where the sun is not occluded by clouds, we can automatically detect the location of its center by fitting a 2-D Gaussian in log-intensity space. We compute the sun appearance by rotating the sky probe so that all the sun locations are at the origin, and averaging them.

We also compute the sun brightness in each color channel across the full database as a function of its height and fit a parametric model of the form $\alpha \exp(-\beta m(\theta_s))$, where $m(\theta_s)$ is relative optical path length through Earth's atmosphere [Kasten and Young, 1989]:

$$m(\theta_s) = \frac{1}{\cos(\theta_s) + 0.50572(96.07995 - \frac{180}{\pi}\theta_s)^{-1.6364}} \ . \tag{4.2}$$

Here, α is an arbitrary scale factor and β is a scattering coefficient. Table 4.1 shows the recovered values for parameters (α, β) for the RGB color channels, and Fig. 4.7 illustrates that the model provides a good fit to the sun data.

The final sun appearance placed in the sun layer of the sky probe is generated by taking the mean sun image and rescaling it by $s(\theta_s)$ which will take care of both sun height and sun occlusion by clouds:

$$s(\theta_s) = s_{max} V \frac{\alpha \exp(-\beta m(\theta_s))}{\alpha \exp(-\beta)} \ , \tag{4.3}$$

since $m(0) = 1$, and where V is the partial sun visibility coefficient (defined in Sec. 3.5) and s_{max} is the maximum sun brightness value, reached when the sun is at zenith $\theta_s = 0$ ($s_{max} = 1 \times 10^5$ in all our experiments).

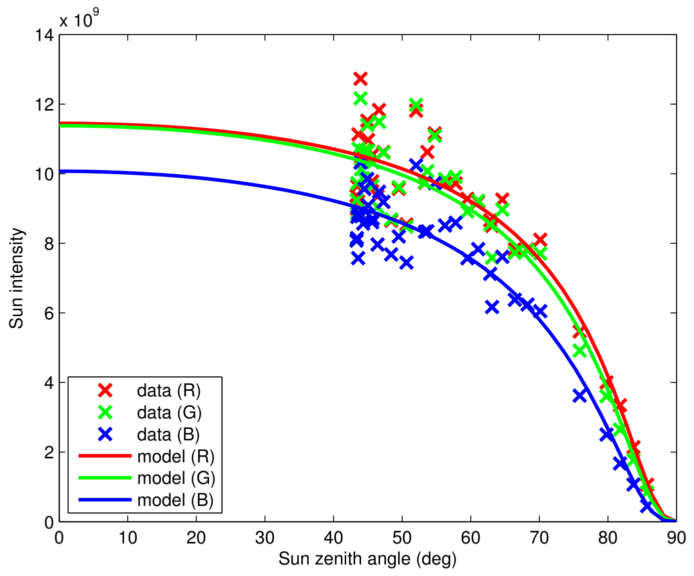


Figure 4.7: Sun intensity as a function of sun zenith angle. The data points are obtained from the HDR sky database [Stumpfel *et al.*, 2004], and the solid lines indicate the prediction by our low-dimensional parametric model (4.3).

Finally, we must define the cloud layer. Its role is mostly decorative (since we have already taken care of sun occlusion by clouds), useful mainly for relighting mirror-like objects which can reflect the sky. We make an assumption that the cloud cover of the full sky is statistically similar to the cloud cover of the visible sky. Therefore, we can use texture synthesis to create plausible cloud cover for the missing portion of the sky. We use Image Quilting [Efros and Freeman, 2001] to synthesize more of the cloud layer C, but we run it in the gradient domain to preserve the color of the clear sky layer underneath. Quilting proceeds along the longitude first, and then the "north cap" of the map is filled in by planar projection at the pole. The final sky probe is the sum of the clear sky, sun, and the cloud layers. It captures the approximate illuminant of the scene in high dynamic range and can be used directly in rendering programs.

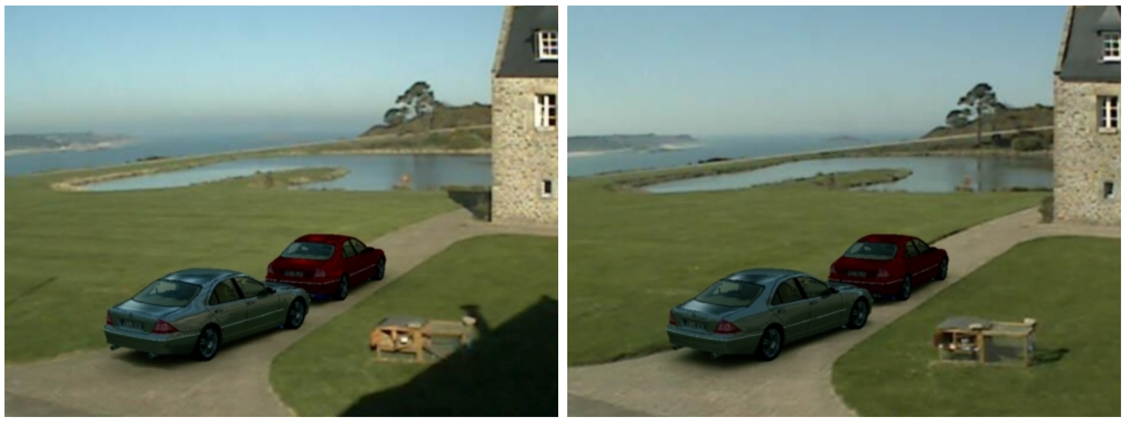
4.5.2 Estimating the Environment Map

To insert a synthetic 3D object into an image from a given webcam, we need the full spherical environment map. We have already estimated the main illuminant in the scene — the sky probe, and the rest will be filled-in from the input image directly. First, we use the approach of Khan *et al.* [2006] to map the pixels below the horizon line in the image onto the bottom hemisphere. This way, we create a map with the sky in the top hemisphere and the mirrored spherical projection of the ground in the bottom hemisphere. However, we have not yet accounted for objects that protrude above the horizon line and occlude part of the sky from view. For this we use another simple approximation—find all non-sky objects above the horizon (using the sky mask defined previously) and project them onto a cylinder around the equator of the environment map. Our experiments [Lalonde and Efros, 2010] have shown that, unless highly-reflective objects are inserted, this approximation results in renderings that are perceptually indistinguishable compared to employing a more accurate representation of the scene surrounding the inserted object (e.g., flat ground plane).

4.5.3 Relighting Virtual Objects

We can now insert virtual objects into a real scene under various illuminations. First, we show how a light-field object captured in a Light Stage apparatus [Debevec *et al.*, 2002] can easily be inserted with the correct lighting. The object is synthesized by summing images of the object taken under 256 different light directions, and weighting them according to the light intensities specified in our estimated environment maps. Fig. 4.6 shows the result of inserting a light-field knight into a castle scene. To insert 3-D objects, we follow the image-based lighting method of Debevec [1998]. This assumes knowledge of the geometry of the scene, which can be acquired in several ways, using photometric stereo [Sunkavalli *et al.*, 2007], or manual intervention [Debevec *et al.*, 1996; Horry *et al.*, 1997] for example. We approximate the scene with a ground plane surrounding the object. Since geometry is known, complex shadow effects can be generated by most off-the-shelf rendering packages.

We now demonstrate practical examples of our approach. Fig. 4.9 shows a scenario where architects wish to know how their newest design will look in three different settings: on a beachfront property (Fig. 4.9a), on a farm (Fig. 4.9b), or in the Swiss mountains (Fig. 4.9c). Our large database provides a wide variety of such environments, which can be used to preview how an object would appear at different times of day and under different weather conditions. Alternatively, one can also visualize how an object will harmonize with existing surroundings by installing a webcam at that site. For example, Fig. 4.10 illustrates


(a) [Khan *et al.*, 2006]

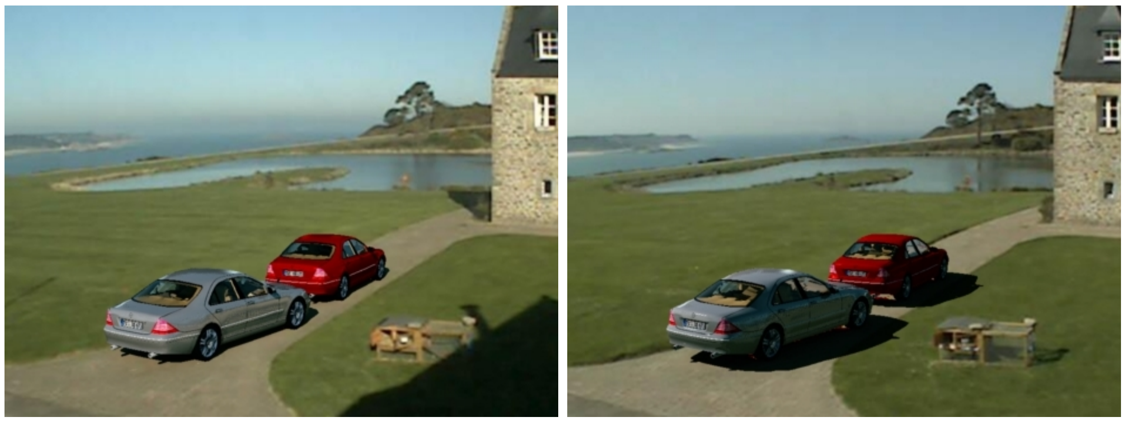

(b) Our result

Figure 4.8: Comparison between objects relit by environment maps obtained with (a) the method of Khan *et al.* [2006], and (b) ours. The dynamic range and sun orientation captured by our method make the inserted object appear much more realistic.

what would happen if the Madrid city planners chose to insert a statue of Venus de Milo in a public square. The statue can be visualized *in situ*, lit by real-world illumination conditions specific to that location.

We compare our approach to that of Khan *et al.* [2006] in Fig. 4.8. The dynamic range and sun orientation captured by our method make the inserted object appear much more realistic.

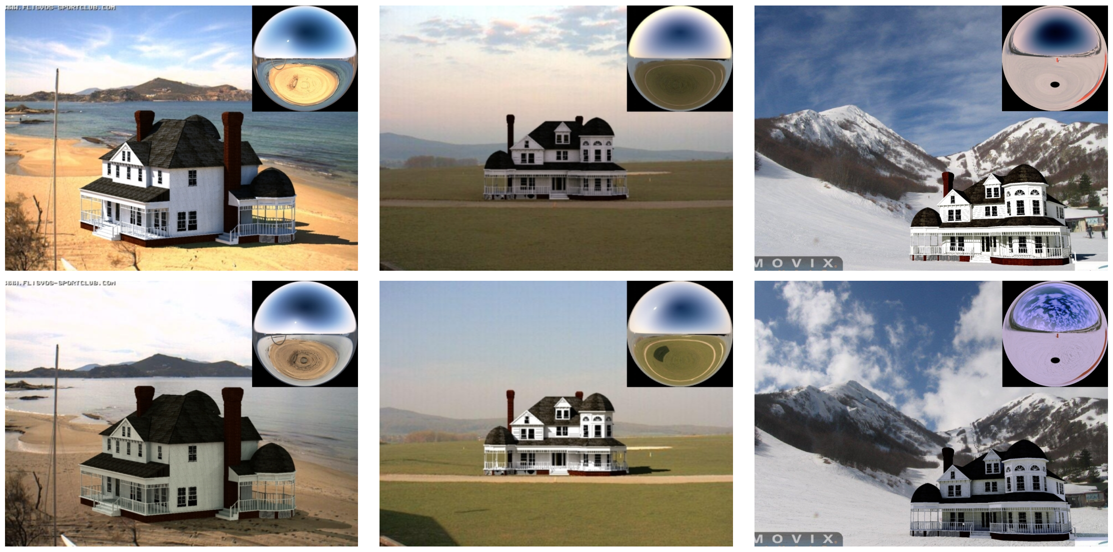


Figure 4.9: An architect wants to show what his architectural model will look like in different settings: (a) on the beach, (b) on a farm, and (c) in the mountains; and under different illumination conditions. It is easy to do so using our rich webcam dataset and our automatic environment map extraction algorithm. The corresponding environment maps are shown as insets.

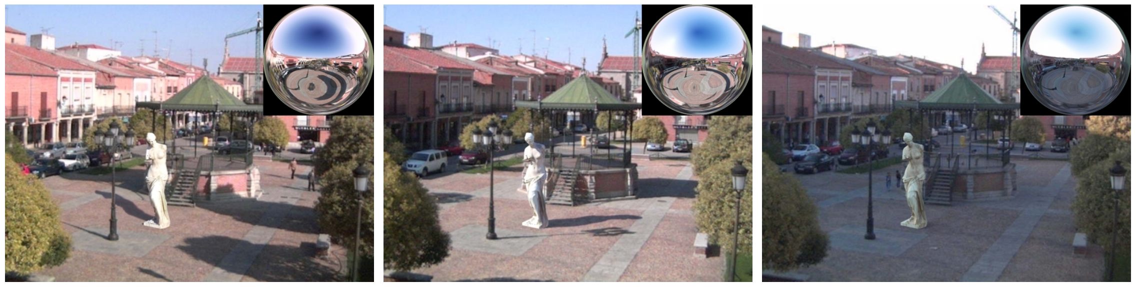


Figure 4.10: The Madrid city planners are considering adding a *Venus de Milo* statue to this square. Using our approach, it is easy to visualize what it will look like in that environment. Notice how the cast shadows on the ground follow the shadows of the other objects in the scene, and how the sun visibility is estimated properly. Shadows can be inserted behind objects by manual layering (as in the left-most image). The corresponding environment maps are shown as insets.

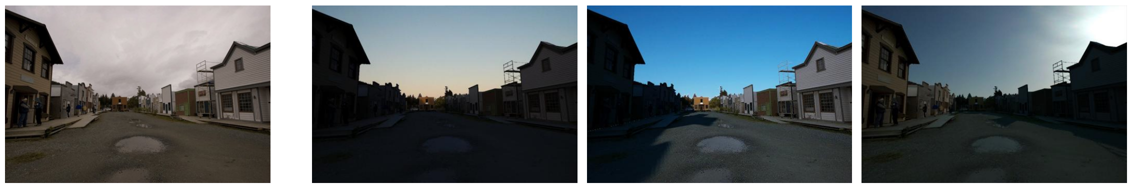


Figure 4.11: Single image relighting. The overcast input image (left) is relit by a clear Berkeley sky and shown here in the morning, noon, and evening.

4.5.4 Relighting in a Single Image

Finally, we demonstrate an application of illuminant transfer for relighting a scene from a *single image* (provided it was taken on overcast day). We start by modeling the geometry of the scene. For simplicity, we used a slightly modified "Tour into the Picture" approach [Horry *et al.*, 1997] in which there is no ceiling, but any other single view modeling approach may be used. The geometry allows us to compute a sky visibility map for each pixel. Since there are no shadows present in the overcast image, the albedo of every scene point can be estimated using the known illumination and scene geometry, based on the Lambertian reflectance model. We transfer a new sky probe as described above into the scene. Using a simple ray tracing algorithm we are then able to relight the scene with the new sky probe, as shown in Fig. 4.11.

4.6 Discussion

We now discuss two important problems which arise when synthesizing appearance in image sequences: the availability of good matches in a large database and the challenge of transferring cast shadows.

4.6.1 Matching Illumination in Image Sequences

The success of appearance transfer depends on the quality of matches recovered in the database. Since not every webcam can be matched to every other webcam, the number of object candidates to be transferred depends on the number of cameras with similar illumination directions. One practical way of recovering potential objects to be inserted in a given webcam is to first match the zenith and azimuth angles of the camera (i.e., only keeping the first two terms of (4.1) for matching). This effectively sorts the cameras by how likely it will be to find similar sun directions.

If the object to be inserted is vertically symmetric, it is also possible to artificially increase the number of available webcams by a factor of two by flipping the image along its vertical axis, and reversing the sun azimuth ($-\Delta\phi_s$).

Finally, one future exciting research direction is to retrieve matches that are *perceptually* consistent, as opposed to *physically* correct. The weights w_i could be correlated to match our perception of illumination consistencies. For example, Koenderink *et al.* [2004] have shown that we are much more sensitive to disparities in sun azimuth than zenith. Is this always the case? For example, matching the sun zenith was important at specific configurations, like the reflection at sunset in Fig. 4.5a. Further research is required to determine how sensitive humans are to illumination inconsistencies, in real-world imagery.

4.6.2 Transferring Cast Shadows

One of the limitations of the appearance transfer application introduced in Sec. 4.4 is the handling of cast shadows. Although there already exists methods for detecting shadows in image sequences [Weiss, 2001], our current implementation does not actually transfer cast shadows to the destination image. The main challenge here is due to the fact that large objects may cast shadows on other objects in the scene (see Fig. 4.12, so unless the geometry in the target image is similar to that of the source, transferring cast shadows would generate uncanny artifacts.

One potential solution to this problem is to estimate the geometry of the object to be transferred, either automatically [Sato *et al.*, 2003; Jacobs *et al.*, 2010] or with user-guided modeling tools [Debevec *et al.*, 1996], and realistically *synthesize* the cast shadow onto the target image. Once the geometry of the object is acquired, the rendering process for shadows is akin to the one presented in Sec. 4.5, for inserting virtual 3-D objects. In both cases, the scene geometry in the target image needs to be modeled as well. Fig. 4.13 presents a behind-the-scenes view of the process. In order to generate realistic cast shadow interactions, such as the one in Fig. 4.13a, we model the target scene using a simple ground plane model (Fig. 4.13b). We could achieve even more complex results, such as the statue

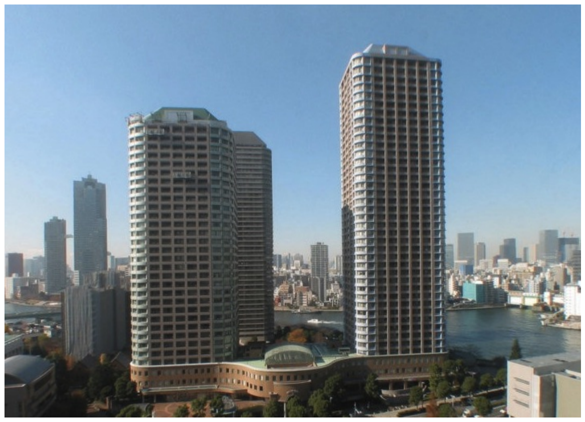


Figure 4.12: Challenging case for transferring cast shadows. In this example, the left building casts a shadow onto the right one, so transferring the shadow would require extracting, and reasoning about the scene geometry.

casting its shadow on the gazebo, by including more details on the geometric model.

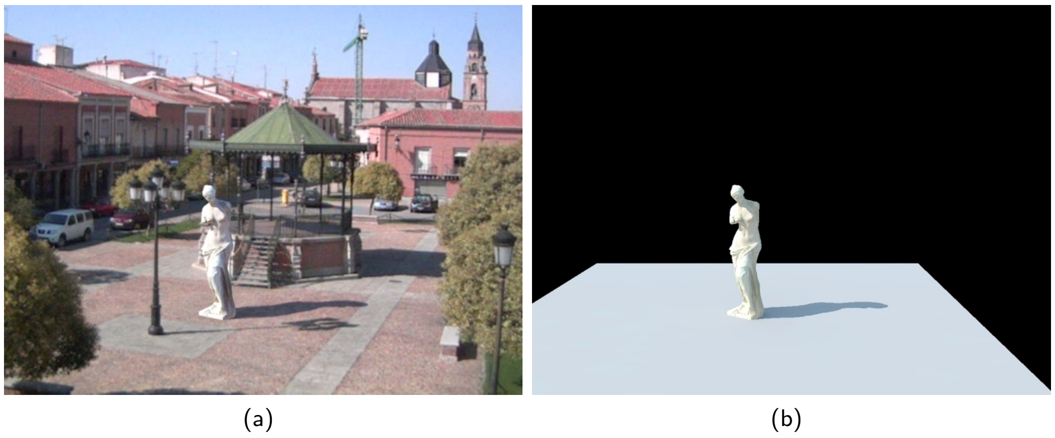


Figure 4.13: Behind-the-scenes look at object insertion. To generate the realistic cast shadow effect in (a), a virtual 3-D model of the scene is needed. In this case, a simple ground plane (b) was used.

4.7 Conclusion

We have shown how to exploit the abundance of Internet webcam data that is available to us for relighting applications. Achieving visual realism by synthesizing appearance is a hard problem. Instead, for the first time, we are able to transfer appearances (sky, objects) from one scene into another and achieve a wide range of relighting effects using webcam data. The large repository of webcams serves as a new form of ”clip art” from which objects can be inserted into a user’s time-lapse sequence or even a single photograph in an illumination-consistent way. We have shown several applications of relighting and appearance transfer between two webcams, and webcams and single images. We believe this work presents a first step in a promising new direction—using the webcams of the world as a viable source of computer graphics content.

Part II

Single Images

Aside from the availability of temporal information, webcam sequences are inherently different than single images. Webcams are typically installed at high vantage points, giving them a broad overlook on large-scale panoramas such as natural or urban landscapes. On the other hand, single images are most often taken at human eye level, where earthbound "objects" like cars, pedestrians, bushes or trees, occupy a much larger fraction of the image. In the second part of this dissertation, we will present approaches to estimate natural illumination, and synthesize illumination-consistent content, from a single image alone.

Chapter 5

Detecting Shadows in a Single Image

Shadow is a color as light is, but less brilliant; light and shadow are only the relation of two tones.

Paul Cézanne (1839–1906)

When the sun is shining on a scene, the most prominent result is probably the cast shadows that it creates. Take a look at some of the images in this document: shadows are everywhere! Yet, the human visual system is so adept at filtering them out [Rensink and Cavanagh, 2004], that we never give shadows a second thought. However, they typically provide strong indication about the illumination conditions of an image. We begin, in this chapter, by addressing the challenging problem of detecting shadows in single images.

5.1 Background

Since the very beginning of computer vision, the presence of shadows has been responsible for wreaking havoc on a wide variety of applications, including segmentation, object detection, scene analysis, stereo, tracking, etc. On the other hand, shadows play a crucial role in determining the type of illumination in the scene [Sato *et al.*, 2003] and the shapes of objects that cast them [Matsushita *et al.*, 2004]. But while standard approaches, software, and evaluation datasets exist for a wide range of important vision tasks, from edge detection to face recognition, there has been comparatively little work on shadows in the last 40 years. Approaches that use multiple images [Finlayson *et al.*, 2007], time-lapse image sequences [Weiss, 2001; Huerta *et al.*, 2009] or user inputs [Wu and Tang, 2005; Bousseau *et al.*, 2009; Shor and Lischinski, 2008] have demonstrated impressive results, but detecting shadows reliably and automatically from a single image remains an open problem. This is because the appearances and shapes of shadows outdoors depend on several hidden factors such as the color, direction and size of the illuminants (sun, sky, clouds), the geometry of the objects that are casting the shadows and the shape and material properties of objects onto which the shadows are cast.

Most works for detecting shadows from a single image are based on computing illumination invariants that are physically-based and are functions of individual pixel values [Finlayson *et al.*, 2002; Finlayson *et al.*, 2004; Finlayson *et al.*, 2009; Maxwell *et al.*, 2008; Tian *et al.*, 2009] or the values in a local image neighborhood [Narasimhan *et al.*, 2005]. Unfortunately, reliable computations of these invariants require high quality images with wide dynamic range, high intensity resolution and where the camera radiometry and color transformations are accurately measured and compensated for. Even slight perturbations (imperfections) in such images can cause the invariants to fail severely (see Fig. 5.4). Thus, they are ill-suited for the regular consumer-grade photographs such as those from Flickr and Google, that are noisy and often contain compression, resizing and aliasing artifacts, and effects due to automatic gain control and color balancing. Since much of current computer vision research is done on consumer photographs (and even worse-quality photos from mobile phones), there is an acute need for a shadow detector that could work on such images.

Our goal is to build a reliable shadow detector for consumer photographs of outdoor scenes. While detecting all shadows is expected to remain hard, we explicitly focus on the shadows cast by objects onto the ground plane. Fortunately, the types of materials constituting the ground in typical outdoor scenes are (relatively) limited, most commonly including concrete, asphalt, grass, mud, stone, brick, etc. Given this observation, our key hypothesis is that the appearances of shadows on the ground are not as widely varying as the shadows everywhere in the scene and can be learned from a set of labelled images of real world scenes. This restriction by no means makes the problem trivial: the ground shadow detector still needs to contend with myriad other non-shadow visual manifestations such as markings and potholes on the roads, pavement/road boundaries, grass patterns on lawns, etc. Further, since many objects (pedestrians, vehicles, traffic signs, etc) of interest to vision applications, are attached to the ground and cast shadows onto the ground, we believe such a ground shadow detector will find wide applicability.

5.2 Overview

Our approach consists of three stages depending on the information in the image used. In the first stage, we will exploit local information around edges in the image. For this, we compute a set of shadow sensitive features that include the ratios of brightness and color filter responses at different scales and orientations on both sides of the edge. These features are then used with a trained decision tree classifier to detect whether an edge is a shadow or not. The idea is that while any single feature may not be useful for detecting all ground shadows, the classifier is powerful enough to choose the right features depending on the underlying edge region. In order to make the classifier robust to non-shadow edges, a negative training set is constructed from a set of edges not on the ground and those arising due to road markings, potholes, grass/mud boundaries, etc. Surprisingly, this simple procedure yields 80% classification accuracy on our test set of images randomly chosen from Flickr and LabelMe [Russell *et al.*, 2008].

In the second stage, we enforce a grouping of the shadow edges using a Conditional Random Field (CRF) to create longer contours. This is similar in spirit to the classical constrained label propagation used in mid-level vision tasks [Freeman *et al.*, 2000]. This procedure connects likely shadow edges, discourages T-junctions which are highly unlikely on shadow boundaries, and removes isolated weak edges. But how do we detect the ground in an image? For this, in the third stage, we incorporate a global scene layout descriptor within our CRF, such as the 3-way ground-vertical surface-sky classifier by Hoiem et. al [Hoiem *et al.*, 2007]. Since the scene layout classifier is trained on the general features of the scene and not the shadows, we are able to reduce the number of false-positive (non-shadow) detections outside the ground. Our results show that accuracy improves by 5% with this step.

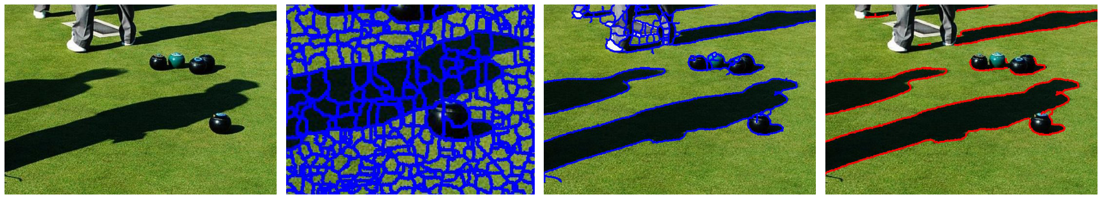

(a) Input image (b) Boundaries (c) Strong boundaries (d) Output

Figure 5.1: Processing stages for the local classifier. The input image (a) is over-segmented into thousands of regions to obtain boundaries (b). Weak boundaries are filtered out by a Canny edge detector [Canny, 1986] (c), and the classifier is applied on the remainder. (d) shows the boundaries i for which $P(y_i = 1|\mathbf{x}) > 0.5$. Note the correct classification of occlusion contours around the person's legs and the reflectance edges in the white square between the person's feet.

We demonstrate successful shadow detection on several images of natural scenes that include beaches, meadows and forest trails, as well as urban scenes that include numerous pedestrians, vehicles, trees, roads and buildings, captured under a variety of illumination conditions (sunny, partly cloudy, overcast). Similarly to the approach of Zhu et al. [Zhu *et al.*, 2010], our method relies on learning the appearance of shadows based on image features, but does so by using full color information. We found that using color features and incorporating knowledge of the ground location improve classification results as much as 10% on our test set. While our technique can be used as a stand-alone shadow detector, we believe it can also be tightly integrated into higher level scene understanding tasks.

5.3 Learning Local Cues for Shadow Detection

Our approach relies on a classifier which is trained to recognize ground shadow edges by using features computed over a local neighborhood around the edge. We show that it is indeed possible to obtain good classification accuracy by relying on local cues, and that it can be used as a building block for subsequent steps. In this section, we describe how to build, train, and evaluate such a classifier.

5.3.1 From Pixels to Boundaries

We first describe the underlying representation on which we compute features. Since working with individual pixels is prone to noise and computationally expensive, we propose to instead reason about *boundaries*, or groups of pixels along an edge in the image. To obtain these boundaries, we first smooth the image with a bilateral filter [Tomasi and Manduchi, 1998], compute gradient magnitudes on the filtered image, and then apply the watershed segmentation algorithm on the gradient map. Fig. 5.1b shows a close-up example of such boundaries.

An undesirable consequence of the watershed segmentation is that it generates boundaries in smooth regions of the image (Fig. 5.1b). To compensate for this, we retain only those boundaries which align with the strong edges in the image. For this, we use the Canny edge detector [Canny, 1986] at 4 scales to account for blurry shadow edges ($\sigma^2 = \{1, 2, 4, 8\}$), with a high threshold empirically set to $t = 0.3$. Under these conditions, we verified that the initial set of boundaries contain more than 97% of the true shadow edges in our dataset. For example, Fig. 5.1c shows the set of boundaries on which our classifier is evaluated for that image.

5.3.2 Local Shadow Features

We now describe the features computed over each boundary in the image. A useful feature to describe a shadow edge is the ratio of color intensities on both sides of the edge (e.g., min divided by max) [Barnard and Finlayson, 2000]. The intuition is that shadows should have a specific ratio that is more or less the same across an image, since it is primarily due to the differences in natural lighting inside and outside the shadow. Since it is hard to manually determine the best color space [Khan and Reinhard, 2005] or best scale to compute features, we use 3 different colors spaces (RGB, LAB, [Chong *et al.*, 2008]) and 4 different scales, and let the classifier automatically select the relevant features during the training phase. Although color-based features are bound to be affected by camera non-linearities, we found these ratios to work well across a wide range of cameras and capture conditions.

For a pixel along a boundary, we compute the intensity on one side of the edge (say, the left) by evaluating a weighted sum of pixels on the left of the edge. But which pixels to choose? We could use the watershed segments, but they do not typically extend very far. Instead, we use an oriented gaussian derivative filter of variance σ^2, but keep only its values which are greater than zero. We align the filter with the boundary orientation such that its positive weights lie on the left of the boundary and convolve it with the image to obtain f_l. The same operation is repeated with the filter rotated by 180° to obtain the weighted mean of pixels on its right f_r. Color ratios can then be computed at pixel p by $\frac{\min(f_l(p), f_r(p))}{\max(f_l(p), f_r(p))}$. This is done independently for each color channel of the RGB, LAB, and illumination-invariant [Chong *et al.*, 2008] color spaces. To account for edge sharpness, we compute each filter at 4 different scales $\sigma^2 = \{1, 2, 4, 8\}$ and size $2\sigma^2$, to obtain 36 ratios in total.

We also employ two features suggested in [Zhu *et al.*, 2010] which capture the texture and intensity distribution differences on both sides of a boundary. The first feature computes a histogram of textons at 4 different scales, and compares them using the χ^2-distance. The texton dictionary was computed on a non-overlapping set of images. The second feature computes the difference in skewness of pixel intensities, again at the same 4 scales.

Finally, we concatenate the minimum filter response $\min(f_l(p), f_r(p))$ computed over the intensity channel to obtain the final, over-complete, 48-dimensional feature vector at every pixel. Boundary feature vectors are obtained by averaging the features of all pixels that belong to it.

5.3.3 Classifier

Having computed the feature vector $\mathbf{x}_i$ at each strong boundary in the image, we can now use them to train a classifier to learn the probability $P(y_i|\mathbf{x}_i)$ that boundary i is due to a shadow (which we denote with label y_i). We estimate that distribution using a logistic regression version of Adaboost [Collins *et al.*, 2002], with twenty 16-node decision trees as weak learners. This classification method provides good feature selection and outputs probabilities, and has been successfully used in a variety of other vision tasks [Hoiem *et al.*, 2007; Hoiem *et al.*, 2010].

To train the classifier, we selected 170 images from LabelMe [Russell *et al.*, 2008], Flickr, and the dataset introduced in [Zhu *et al.*, 2010], with the only conditions being that the

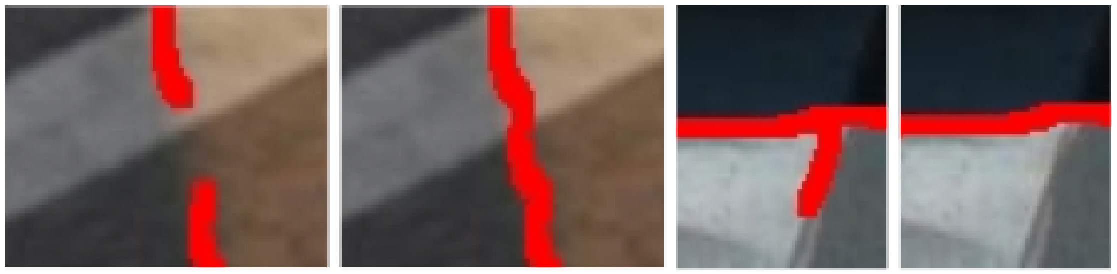


Figure 5.2: Creating shadow contours by enforcing local consistency. Our CRF formulation may help to (a) bridge the gap across X-junctions where the local shadow classifier might be uncertain, and (b) remove spurious T-junctions which should not be caused by shadows.

ground must be visible, and there must be shadows. The positive training set contains manually labelled shadow boundaries, while the negative training set is populated with an equal amount of strong non-shadow boundaries on the ground (e.g., street markings) and occlusion boundaries.

We obtain a per-boundary classification accuracy of 79.7% (chance is 50%, see Fig. 5.5 for a breakdown per class). See Fig. 5.1d for an example. This result support out hypothesis: while the appearance of shadows on *any* type of material in *any* condition might be impossible to learn, the space of shadow appearances on the ground in outdoor scenes may not be that large after all!

5.4 Creating Shadow Contours

Despite encouraging results, our classifier is limited by its locality since it treats each boundary independently of the next. However, the color ratios of a shadow boundary should be consistent with those of its neighbors, since the sources illuminating nearby scene points should also be similar. Thus, we can exploit higher order dependencies across local boundaries to create longer shadow contours as well as remove isolated/spurious ones.

To model these dependencies, we construct a graph with individual boundaries as nodes (such as those in Fig. 5.1b) and drawing an edge across boundaries which meet at a junction point. We then define a CRF on that graph, which expresses the log-likelihood of a particular labeling $\mathbf{y}$ (i.e., assignment of shadow/non-shadow to each boundary) given observed data

$\mathbf{x}$ as a sum of unary $\phi_i(y_i)$ and pairwise potentials $\psi_{i,j}(y_i, y_j)$:

$$-\log P(\mathbf{y}|\mathbf{x}; \lambda, \beta) = \lambda \sum_{i \in \mathcal{B}} \phi_i(y_i) + \sum_{(i,j) \in \mathcal{E}} \psi_{i,j}(y_i, y_j) - \log Z_{\lambda,\beta} \ , \tag{5.1}$$

where $\mathcal{B}$ is the set of boundaries, $\mathcal{E}$ the set of edges between them, and λ and β are model parameters. In particular, λ is a weight controlling the relative importance of the two terms. $Z_{\lambda,\beta}$ is the partition function that depends on the parameters λ and β, but not on the labeling $\mathbf{y}$ itself.

Intuitively, we would like the unary potentials to penalize the assignment of the "shadow" label to boundaries which are not likely to be shadows according to our local classifier. This can be modeled using

$$\phi_i(y_i) = -\log P(y_i|\mathbf{x}_i) \ . \tag{5.2}$$

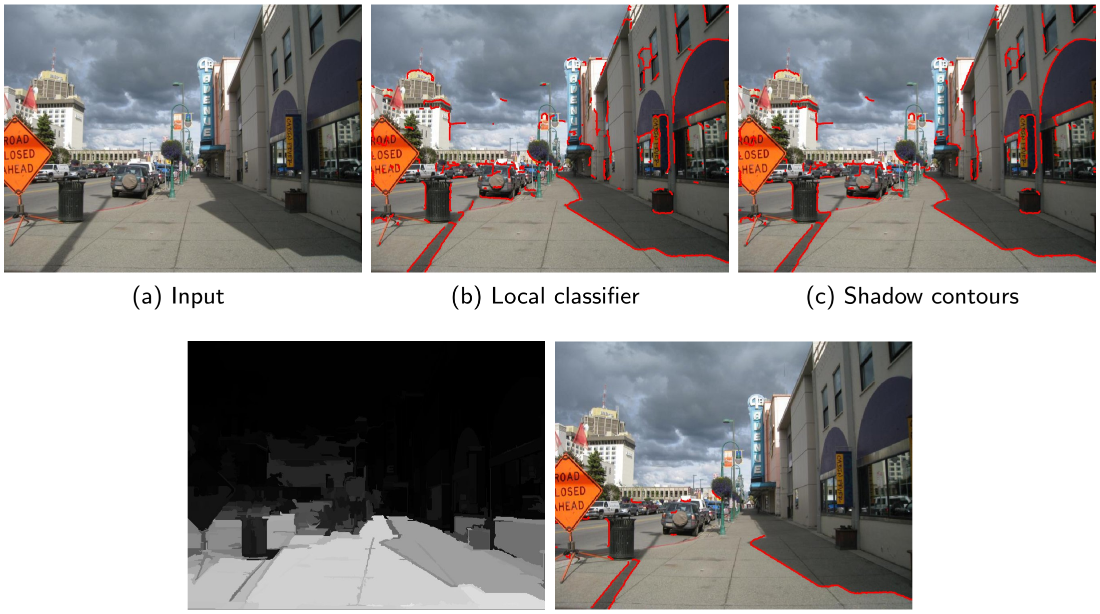
We would also like the pairwise potentials to penalize the assignment of different labels to neighboring boundaries that have similar features, which can be written as

$$\psi_{i,j}(y_i, y_j) = \mathbf{1}(y_i \neq y_j) \exp(-\beta \|\mathbf{x}_i - \mathbf{x}_j\|_2^2) \ , \tag{5.3}$$

where $\mathbf{1}(\cdot)$ is the indicator function, and β is a contrast-normalization constant as suggested in [Boykov and Jolly, 2001]. In other words, we encourage neighboring shadows which have similar features and strong local probabilities to be labelled as shadows.

The negative likelihood in (5.1) can be efficiently minimized using graph cuts [Boykov *et al.*, 2001; Kolmogorov and Zabih, 2004; Boykov and Kolmogorov, 2004]. The free parameters were assigned the values of $\lambda = 0.5$ and $\beta = 16$ obtained by 2-fold cross-validation on a non-overlapping set of images.

Applying the CRF on our test images results in an improvement of roughly 1% in total classification accuracy, for a combined score of 80.5% (see Fig. 5.5-(b)). But more importantly, in practice, the way the CRF is setup encourages continuity, crossing through X-junctions, and discourages T-junctions as shown in Fig. 5.2. Since shadows are usually signaled by the presence of X-junctions and the absence of T-junctions [Sinha and Adelson, 1993], this reduces the number of false positives.



(a) Input (b) Local classifier (c) Shadow contours

(d) Ground likelihood [Hoiem *et al.*, 2007] (e) Combining (c) and (d)

Figure 5.3: Incorporating scene layout for detecting cast shadows on the ground. Applying our shadow detector on a complex input image (a) yields false detections in the vertical structures because of complex effects like occlusion boundaries, self-shadowing, etc. (b) & (c). Recent work in scene layout extraction from single images [Hoiem *et al.*, 2007] can be used to estimate the location of the ground pixels (d). We show how we can combine scene layout information with our shadow contour classifier to automatically detect cast shadows on the ground (e).

5.5 Incorporating Scene Layout

Until now, we have been considering the problem of detecting cast shadow boundaries on the ground with a classifier trained on local features and a CRF formulation which defines pairwise constraints across neighboring boundaries. While both approaches provide good classification accuracy, we show in Fig. 5.3 that applying them on the entire image generates false positives in the vertical structures of the scene. Reflections, transparency, occlusion boundaries, self-shadowing, and complex geometry [Sinha and Adelson, 1993] are common phenomena that can confuse our classifier. This results in image-wide classification results which might not be useful for complex scenes (see Figs 5.3b-(c)).

The advent of recent approaches which estimate a qualitative layout of the scene from a single image (e.g., splitting an image into three main geometric classes: the sky, vertical surfaces, and ground [Hoiem *et al.*, 2007]) may provide explicit knowledge of where the ground is. Since such a scene layout estimator is specifically trained on general features of the scene and not the shadows, combining its output with our shadow detector should reduce the number of false positive (non-shadow) detections outside the ground. We now consider how such high-level scene reasoning can be used within our shadow detection framework.

5.5.1 Combining Scene Layout With Local Shadow Cues

To combine the scene layout probabilities with our local shadow classifier, we can marginalize the probability of shadows over the three geometric classes sky $\mathcal{S}$, ground $\mathcal{G}$, and vertical surfaces $\mathcal{V}$:

$$P_{comb}(y_i|\mathbf{x}) = \sum_{c \in \{\mathcal{G},\mathcal{V},\mathcal{S}\}} P(y_i|c_i, \mathbf{x}_i) P(c_i|\mathbf{x}_i) \ , \tag{5.4}$$

where c_i is the geometric class label of boundary i, $P(y_i|c_i, \mathbf{x}_i)$ is given by our local shadow classifier, and $P(c_i|\mathbf{x}_i)$ by the scene layout classifier (we use the geometric context algorithm [Hoiem *et al.*, 2007]). Unfortunately, this approach does not actually improve classification results because while it gets rid of false positives in the vertical structures, it also loses true positives on the ground along the way. This is due to the fact that shadow likelihoods get down-weighted by low-confidence ground likelihoods. Thus, we need a different approach.

5.5.2 Combining Scene Layout With Shadow Contours

Intuitively, we would like to penalize an assignment to the shadow class when the probability of being on the ground is low. When it is high, however, we should let the shadow classifier decide. We can encode this behavior simply by modifying the unary potentials $\phi_i(y_i)$ from (5.2) in our CRF formulation:

$$\phi_i(y_i) = \begin{cases} -\log P(c_i = \mathcal{G}|\mathbf{x}_i) - \log P(y_i = 1|\mathbf{x}_i) & \text{if } y_i = 1 \text{ (shadow)} \\ (1 - P(c_i = \mathcal{G}|\mathbf{x}_i)) - \log P(y_i = 0|\mathbf{x}_i) & \text{if } y_i = 0 \text{ (non-shadow)} \ . \end{cases} \tag{5.5}$$

Here, $\lambda = 0.5$ and $\beta = 16$ was found by cross-validation. They yield a good compromise between local evidence and smoothness constraints.

This approach effectively combines local and mid-level shadow cues with high-level scene interpretation results, and yields an overall classification accuracy of 84.8% on our test set (see Fig. 5.5) without adding to the complexity of training our model. Observe how the results are significantly improved in Fig. 5.3e as compared to the other scenarios in Fig. 5.3b-(c).

5.6 Experimental Results

We evaluate our approach on 135 consumer photographs downloaded from LabelMe [Russell *et al.*, 2008], Flickr, and images from the dataset introduced in [Zhu *et al.*, 2010]. In all cases, we have no control over the acquisition settings, so images contain the typical sources of distortions [Chakrabarti *et al.*, 2009] such as jpeg compression, sharpening, sampling due to resizing, non-linear response functions, image noise, etc. We first compare our method with the current state of the art [Finlayson *et al.*, 2009], then show shadow detection and removal results on several challenging images.

5.6.1 Comparison With Previous Work

The current state of the art in color-based shadow detection and removal in single images is the approach of Finlayson et al. [Finlayson *et al.*, 2009], which relies on a physics-based model of shadows to compute an illumination-invariant image. Shadows are then obtained by finding edges in the original image which are not present in the invariant image. The first row of Fig. 5.4 shows that our implementation of their approach successfully recovers a shadow-free invariant image from the same high-quality, linear image used in their original paper [Finlayson *et al.*, 2002]. Note that our approach is able to extract similar results to theirs (Fig. 5.4d). When applying their method on an image from our set, the performance degrades because the invariant still contains strong traces of shadows (second row of Fig. 5.4).

This is also demonstrated in the quantitative comparison shown in Fig. 5.5-(a), which compares our approach with the original shadow detection technique of [Finlayson *et al.*, 2004], and the most recent version [Finlayson *et al.*, 2009]. Note here that we cannot

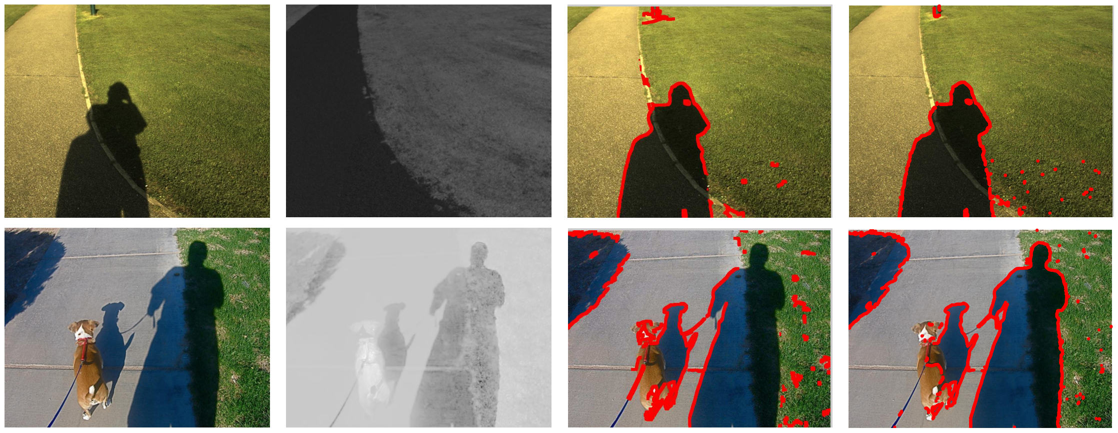
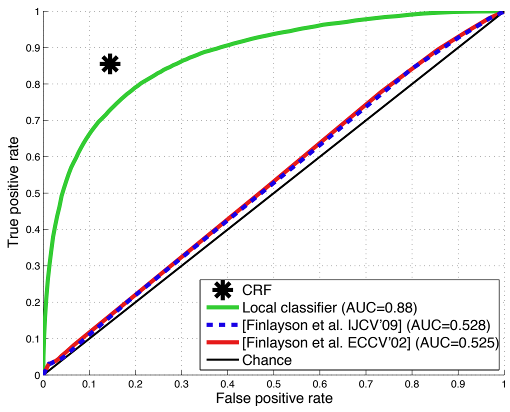

(a) Original image
(b) Invariant img. [Finlayson *et al.*, 2009]
(c) Result with [Finlayson *et al.*, 2009]
(d) Our result

Figure 5.4: Comparison with the shadow detection method of [Finlayson *et al.*, 2009]. *First row*: Using a high-quality linear image as input (a), our implementation of their method successfully recovers a shadow-free 1-D invariant image (b), which is used to detect shadow edges (c). *Second row*: However, if the input is not linear and corrupted by noise or jpeg compression typical of consumer photographs, the 1-D invariant image still contains some shadows (b), making it hard to tell them apart from other types of edges (c). Our method detects shadows both in high and low quality images (d).

generate an ROC curve with the results of our CRF formulation, since the output is binary, so we plot a point at its classification accuracy. To obtain ROC scores for the competing methods, we first estimate the invariant image, and compute the difference of gradient magnitudes between the original and the invariant images. For fairness, we evaluate this score only at the strong boundaries in the image. The ROC curve shows that our results greatly outperform the previous work. This is most likely due to the use of features which are robust to artifacts common in consumer photographs.

5.6.2 Ground Shadow Detection and Removal

We summarize the quantitative results obtained by the technique presented in this chapter on our test set in Fig. 5.5-(b), which shows results obtained on the entire image by the local classifier and the boundary CRF (Secs 5.3 and 5.4), and those obtained by combining the geometric context ground likelihoods (Sec. 5.5). The best performance (84.8% accuracy) is obtained by our CRF formulation which combines the scene layout results with our local



(a) ROC curve comparison with previous work

	Shadows	Non-shadows	Combined
Local	78.3%	81.0%	79.7%
CRF	78.7%	82.3%	80.5%
CRF + scene layout	73.1%	96.4%	**84.8%**

(b) Quantitative ground shadow classification results

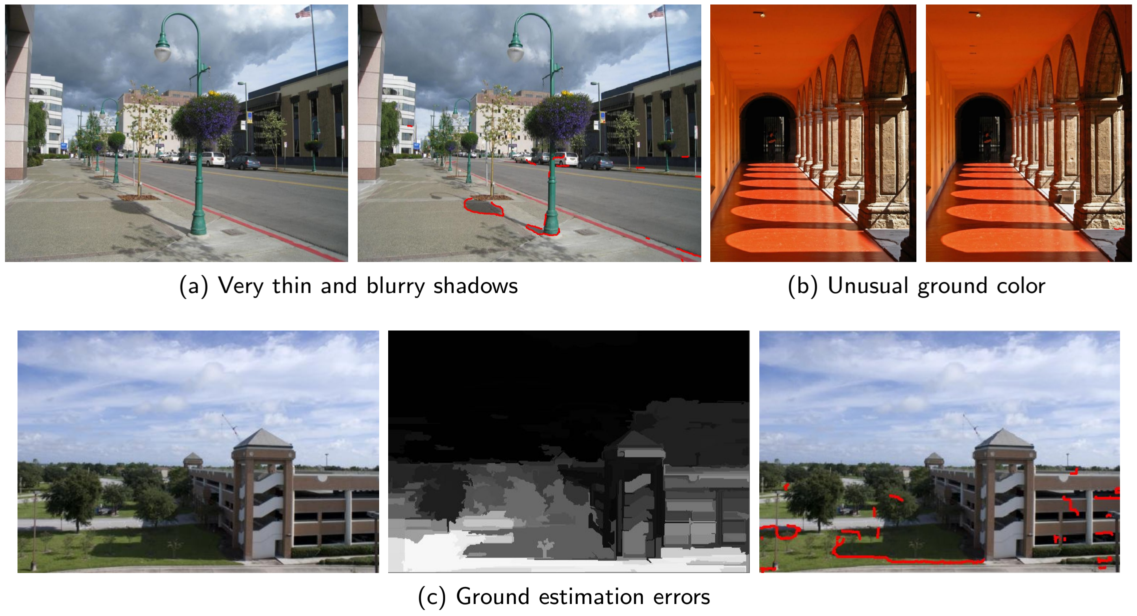
Figure 5.5: Quantitative results. We compare our local classifier with the methods proposed by Finlayson et al. [Finlayson *et al.*, 2004; Finlayson *et al.*, 2009] (a). The table in (b) show the results obtained with the approaches presented in Secs 5.3 (local), 5.4 (CRF) and 5.5 (CRF + scene layout). Integrating scene layout information from [Hoiem *et al.*, 2007] results in ground shadow classification accuracy of 84.8%.

shadow boundary classifier.

Fig. 5.7 shows ground shadow detection results on several images from our dataset. It demonstrates that our method works on challenging outdoor images with varying illumination conditions, ground colors and textures, and clutter.

The typical errors made by our method are shown in Fig. 5.6. It may fail to detect shadows cast by thin structures like the lamppost in Fig. 5.6a. Another failure case arises when the ground has a color that is vastly different from all the other images in the training set, as in Fig. 5.6b. This can likely be improved by increasing the size and diversity of the training set. A third failure mode is due to errors in the estimated scene layout probabilities as in Fig. 5.6c.

Once we have detected shadow boundaries, we can, as an application, use the technique introduced in [Finlayson *et al.*, 2002] to remove them and recover a shadow-free image. There have been improvements proposed since then [Fredembach and Finlayson, 2006], but we chose the original method for its simplicity. This approach involves setting the derivatives of the image at shadow boundaries to zero, and reintegrating the result by solving the Poisson equation with Neumann boundary conditions. Fig. 5.8 shows shadow-free images that were computed using the boundaries detected by our method.


(a) Very thin and blurry shadows (b) Unusual ground color

(c) Ground estimation errors

Figure 5.6: Failure cases. The downside of using boundaries from an over-segmentation is the trade-off between spatial support obtained from longer boundaries and the size of shadow regions that can be detected. In our current setting, it may miss thin and blurry edges, like the lamppost in (a). Our approach is also sensitive to vastly different ground colors, which have never been seen by the classifier (b). Although our ratio-based features are somewhat color independent, they are not able to compensate for such drastic differences. Increasing the variety of ground colors in the training set would likely improve performance on such extreme cases. (c) Errors in the scene layout probabilities can lead to false positives.

5.7 Discussion

Before seeing how this novel knowledge about shadows in single images can help us reason about the illumination conditions, we first discuss a few caveats of detecting shadows using our approach. In this section, we consider the challenge of detecting non-ground shadow boundaries, achieving robustness to varying camera parameters, and the case for detecting shadow boundaries as opposed to regions.

5.7.1 Non-ground Shadow Boundaries

The insight behind our approach is to learn the statistical distribution of the appearance of shadow boundaries from data. Our assumption is that the appearance of shadows on the ground lives in a restricted space of color ratios, which is compact enough to be learned

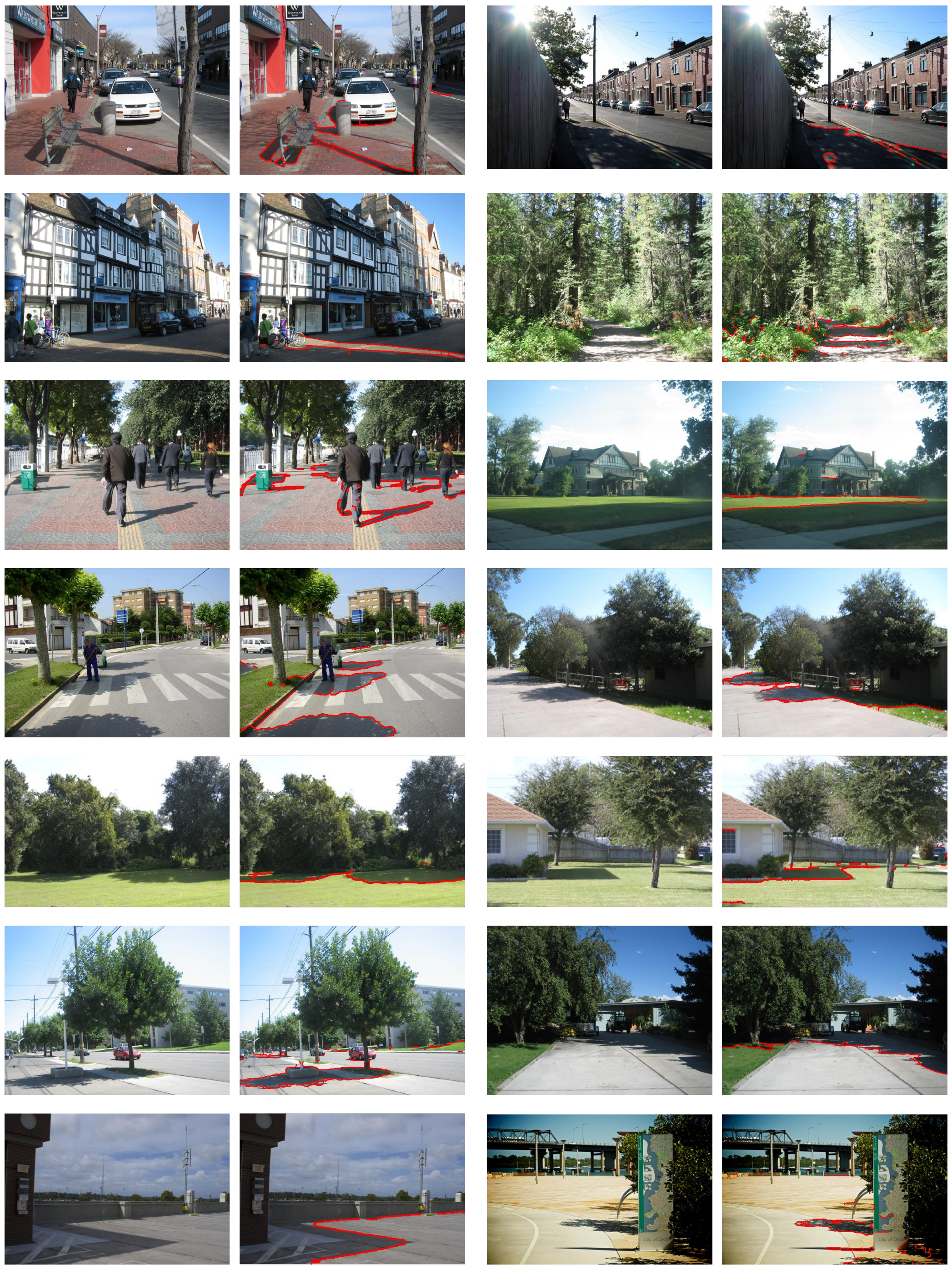


Figure 5.7: Ground shadow detection results on images downloaded from the web (Flickr, LabelMe [Russell *et al.*, 2008]), and the dataset from [Zhu *et al.*, 2010]. First and third columns: input images; second and fourth columns: detected ground shadows. Our approach successfully detects ground shadows in many challenging, real-world conditions.

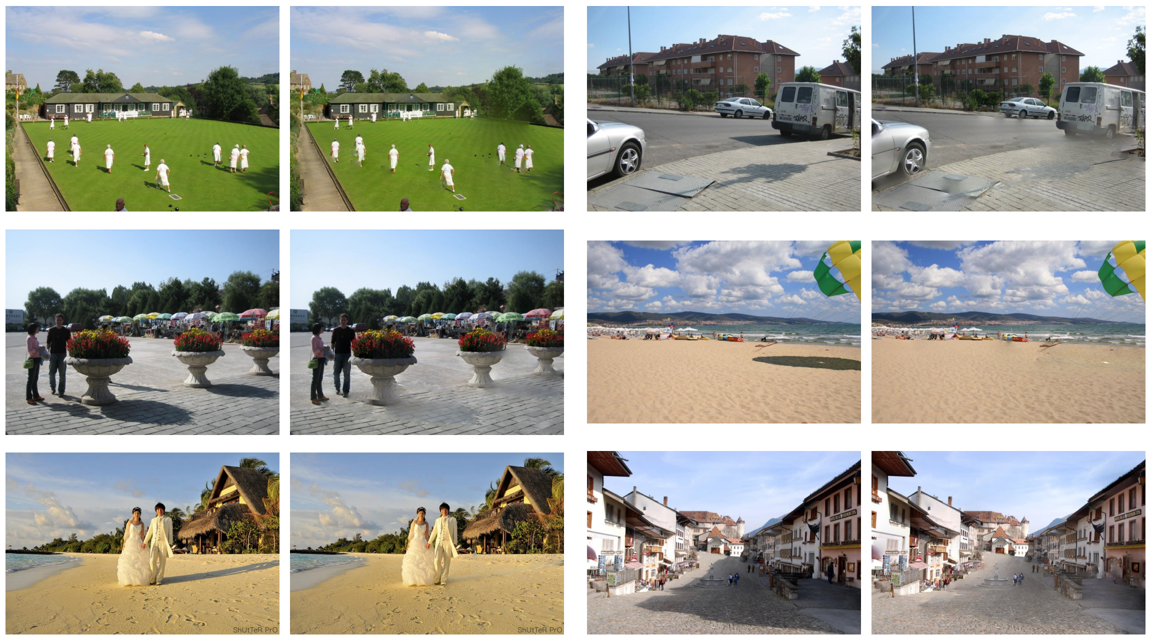


Figure 5.8: Automatic ground shadow removal from a single image. We apply the original gradient reintegration method of [Finlayson *et al.*, 2002] on the boundaries detected by our method. The shadows are either completely removed or greatly attenuated, with few visual artifacts.

by a classifier. Because our approach relies on such an assumption, it is likely to fail when presented with an image completely different from the training set (such an example is shown in Fig. 5.6b). Typically, such cases can be handled by adding more training data.

One interesting future research direction is to explore the problem of detecting shadows everywhere in an image, not just on the ground. This task is very challenging because there are several other types of boundaries which might confuse such a local classifier: material changes, occlusion boundaries, and geometry (normal) discontinuities all create strong edges in the scene. Future work in this area will likely have to jointly consider all these types of edges, and reason about them all simultaneously. As a starting point, one could start by trying to integrate a shadow-specific detector, such as the one presented in this chapter, to an occlusion boundary classifier, such as the one in [Hoiem *et al.*, 2010].

5.7.2 Robustness to Camera Parameters

The appearance of an image depends strongly on the parameters of the camera that is capturing it. Manual and automatic settings such as the non-linear response function [Grossberg and Nayar, 2004; Lin *et al.*, 2004; Mitsunaga and Nayar, 1999], white balance [Hsu *et al.*, 2008; Kawakami *et al.*, 2005], gain, ISO setting [Hasinoff *et al.*, 2010], exposure [Hasinoff and Kutulakos, 2008], etc. are all responsible for determining the intensity of the pixels in the resulting image. Our approach already demonstrates robustness to changes in some of these camera parameters—the images in our training and test sets were taken from independent sources, such as LabelMe [Russell *et al.*, 2008], Flickr, and the dataset of Zhu *et al.* [2010], and we had no control over the capture conditions. Unfortunately, given the sheer number of camera models and their capacity to adapt to varying illumination conditions, it is very challenging to capture the space of all possible camera parameters [Grossberg and Nayar, 2004; Chakrabarti *et al.*, 2009].

Shadow detection, as with most computer vision tasks, would certainly benefit from canceling the effects of camera parameters and *normalizing* all the images to a common color reference frame. Unfortunately, estimating these parameters from a single image is very much under-constrained, and while ideas have already been proposed [Lin *et al.*, 2004; Dale *et al.*, 2009], this largely remains an unsolved problem in vision.

One key observation to note here is that while they can vary across images, most of these parameters stay constant over an image (one notable exception being vignetting [Zheng *et al.*, 2006]). In addition, shadows should share similar characteristics in a given image: the sun and the sky colors also remain constant (only their respective ratios change because of occlusion). Therefore, one interesting future research direction would be to study what properties do shadow boundaries *have in common* within a single image. What makes a shadow boundary similar to other shadow boundaries, but different than, say, material boundaries? Answering this question would provide robustness to varying camera parameters, without having to explicitly recover them.

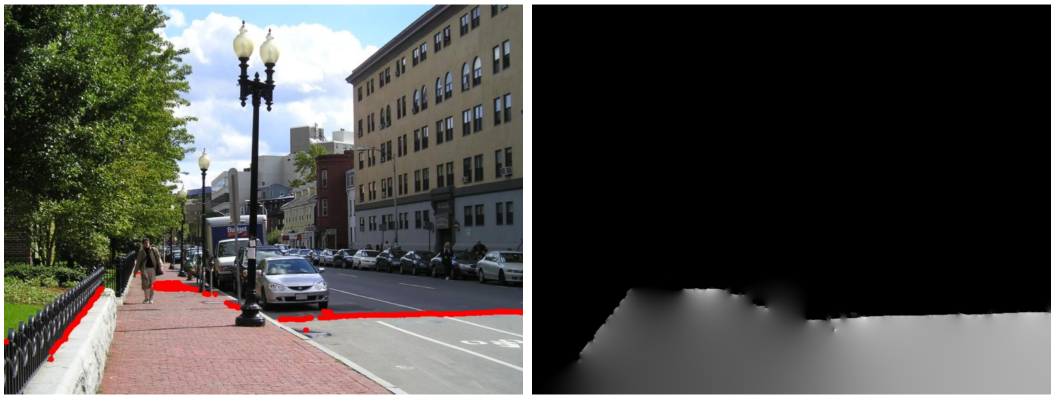

(a) (b)

Figure 5.9: Estimating sunny regions on the ground. From our detected shadow boundaries (a), we generate a map of the ground region that is in sunlight by reintegrating the shadow boundaries only (b).

5.7.3 Shadow Boundaries vs Regions

In the literature, researchers have proposed ways to detect both shadow *boundaries* [Barnard and Finlayson, 2000; Figov *et al.*, 2004; Finlayson *et al.*, 2002] as well as *regions* [Zhu *et al.*, 2010; Tian *et al.*, 2009; Shor and Lischinski, 2008]. While neither has shown tremendous advantages over the other, our decision of focusing on boundaries has its roots in psycho-physical studies which have shown that humans are more sensitive to *variations* in luminance (i.e., contrast) than to absolute values [Campbell and Robson, 1968]. Therefore, it is perceptually more important to interpret edges as opposed to regions of similar intensities, and our work provides a computational way of doing so.

On the other hand, the spatial support obtained by computing shadow regions provide advantages over their boundaries only. For instance, consistency checks can be performed on an entire region (i.e., an entire shadow region must be darker than its surroundings). In addition, while retrieving boundaries from regions is an easy task, the opposite problem is significantly more challenging. However, we believe there is still hope. Fig. 5.9 shows a preliminary result of using the shadow boundaries detected by our approach to obtain an estimate of the ground region that is lit by sunlight. The region is obtained by reintegrating [Finlayson *et al.*, 2002] the ground shadow boundaries only.

In addition, the definition of a cast shadow region is sometimes ill-defined: if a large building casts a shadow on the ground as in Fig. 5.6c, where does the shadow region begin and end? Should it encompass the entire building facade in the shade, or should it stop at its boundary with the ground? High-level scene reasoning might be necessary to obtain a coherent shadow segmentation of the image.

5.8 Conclusion

While our technique succeeds in detecting ground shadows with good accuracy, shadows that are not on the ground exhibit significantly larger appearance variations, so detecting them will be challenging. The method introduced in this chapter can directly be used as an off-the-shelf, stand-alone shadow detector[1], but we believe it can also be tightly integrated into higher level scene understanding tasks. In the next chapter, we will show how this detector, in combination with others, is used to estimate our natural illumination model from Sec. 2.4 from a single image.

[1]Data and code are available at `http://graphics.cs.cmu.edu/projects/shadows`.

Chapter 6

Estimating Illumination in Single Images

Of all creatures in this visible world, light is the most glorious; of all light, the light of the sun without compare excels the rest.

William Burnall (1617–1679)

In this chapter, we propose a method for estimating natural illumination from a single outdoor image, that will rely on the shadows detected in the preceding chapter, but on other scene cues as well. To be sure, this is an extremely difficult task, even for humans [Cavanagh, 2005]. In fact, the problem is severely under-constrained in the general case—while some images might have enough information for a reasonably precise estimate, others will be completely uninformative. Therefore, we will take a probabilistic approach, estimating illumination parameters using as much information as may be available in a given image and producing the maximum likelihood solution.

6.1 Introduction

What information about illumination is available in a single image? Unfortunately, there is no simple answer. When we humans perform this task, we look at different parts of the image for clues. The appearance of the sky can tell us if it is clear or overcast (i.e.,

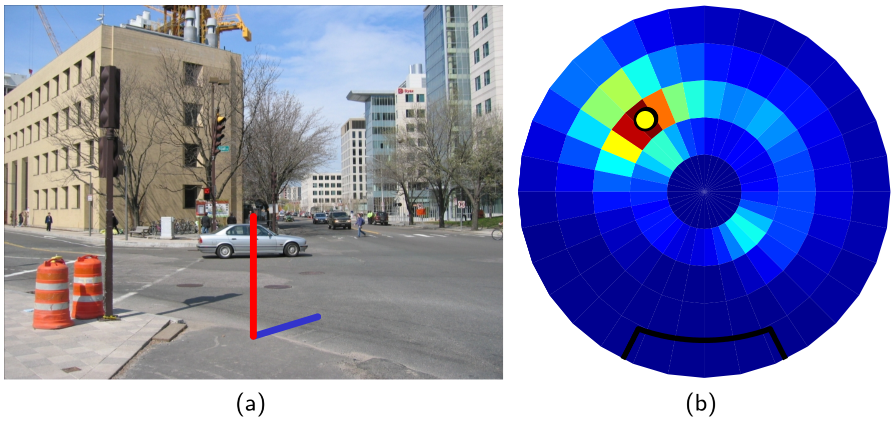


Figure 6.1: Example of the results obtained in this chapter. From a single input image (a), we estimate the probability of the relative sun position $P(\theta_s, \Delta\phi_s)$ with respect to the camera (b). Throughout the paper, the sun position probability is displayed as if the viewer is looking straight up (center point is zenith), with the camera field of view drawn at the bottom. Probabilities are represented with a color scale that vary from blue (low probability) to red (high probability). In this example, the maximum likelihood sun position (yellow circle) is estimated to be at the back-left of the camera. We sometimes illustrate the results by inserting a virtual sun dial (red stick in (a)) and drawing its shadow corresponding to the most likely sun position.

is the sun visible?). On a clear day, the sky might give some weak indication about the sun position. The presence of shadows on the ground plane can, again, inform us about sun visibility, while the direction of shadows cast by vertical structures can tell us about sun direction. The relative shading of surfaces at differing orientations (e.g., two building facades at a right angle), can also give a rough indication of sun direction, and so can the shading effects on convex objects populating the scene (e.g., pedestrians, cars, poles, etc.).

Our approach is based on implementing some of these intuitions into a set of illumination cues. Of course, each one of these cues by itself is rather weak and unreliable. The sky might be completely saturated, or might not even be present in the image. The ground might not be visible, be barren of any shadow-casting structures, or be deprived of any recognizable objects. Shading information might, likewise, be inaccessible due to lack of appropriate surfaces or large differences between surface reflectances. Furthermore, computing these cues will inevitably lead to more noise and error (misdetected shadows, poor segmentation, incorrect camera parameters, etc). Hence, in this work, we combine the information obtained from these weak cues together, while applying a data-driven prior computed over a set of 6 million Internet photographs.

The result sections (Secs 6.2.2 and 6.5) will show that the sun visibility can be estimated with 83.5% accuracy, and the combined estimate is able to successfully locate the sun within an octant (quadrant) for 40% (55%) of the real-world images in our very challenging test set, and as such outperforms any of the cues taken independently. Fig. 6.1 shows an example taken from our test set. While this goes to show how hard the problem of illumination from single images truly is, we believe this can still be a useful result for a number of applications[1]. For example, just knowing that the sun is somewhere on your left might be enough for a point-and-shoot camera to automatically adjust its parameters, or for a car detector to be expecting cars with shadows on the right.

6.2 Is the sun visible or not?

Using the above representation, we estimate $P(I|\mathcal{I})$, the probability distribution over the illumination parameters I, given a single image $\mathcal{I}$. Because our representation defines the sun position S only if it is visible, we propose to first estimate the sun visibility V:

$$P(I|\mathcal{I}) = P(S, V|\mathcal{I}) = P(S|V, \mathcal{I})P(V|\mathcal{I})\,. \tag{6.1}$$

In this section, we present how we estimate the distribution over the sun visibility variable given the image $P(V|\mathcal{I})$. The remainder of the chapter will then focus on how we estimate $P(S|V = 1, \mathcal{I})$, that is, the probability distribution over the sun position if it was determined to be visible. We propose a supervised learning approach to learn $P(V|\mathcal{I})$, where a classifier is trained on features computed on a manually-labelled dataset of images. We first describe the features that were developed for this task, then provide details on the classifier used.

6.2.1 Image cues for predicting the sun visibility

When the sun is shining on the scene, it usually creates noticeable effects in the image. Consider for a moment the first and last images of Fig. 6.2. When the sun is visible (Fig. 6.2a), it creates hard cast shadow boundaries, bright and dark areas corresponding to

[1]Dataset and code available at `http://graphics.cs.cmu.edu/projects/outdoorIllumination`.

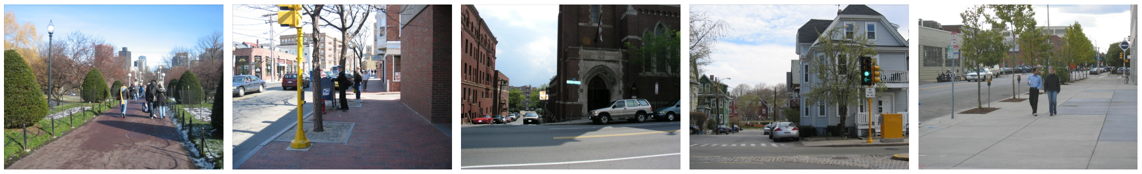

(a) $P(V|\mathcal{I}) = 0.95$ (b) $P(V|\mathcal{I}) = 0.75$ (c) $P(V|\mathcal{I}) = 0.50$ (d) $P(V|\mathcal{I}) = 0.25$ (e) $P(V|\mathcal{I}) = 0.05$

Figure 6.2: Results of applying the sun visibility classifier on single images to estimate $P(V|\mathcal{I})$, sorted by decreasing order of probability.

sunlit and shadowed regions, highly-saturated colors, and clear skies. On the other hand, when the sun is occluded (Fig. 6.2e), the colors are dull, the sky is gray or saturated, and there are no visible shadows.

We devise sun visibility features based on these intuitions. We first split the image into three main geometric regions: the ground $\mathcal{G}$, the sky $\mathcal{S}$, and the vertical surfaces $\mathcal{V}$ using the geometric context classifier of Hoiem *et al.* [2007]. We use these regions when computing the following set of features:

Bright and dark regions: We compute the mean intensity of the brightest of two clusters on the ground $\mathcal{G}$, where the clustering is performed with k-means with $k = 2$; The same is done for the vertical surfaces $\mathcal{V}$;

Saturated colors: We compute 4-dimensional marginal histograms of the scene $(\mathcal{G} \cup \mathcal{V})$ in the saturation and value color channels; To add robustness to variations in exposure, white balance, and gamma response functions across cameras, we follow [Dale *et al.*, 2009] and also compute a 4-dimensional histogram of the scene in the normalized log-RGB space;

Sky: If the sky is visible, we compute its average color in RGB;

Contrast: We use the contrast measure proposed in [Ke *et al.*, 2006] (a measure of the histogram spread), computed over the scene pixels only $(\mathcal{G} \cup \mathcal{V})$;

Shadows: We apply the shadow detection method of Ch. 5, and count the fraction of pixels that belong to the ground $\mathcal{G}$ and scene $(\mathcal{G} \cup \mathcal{V})$ and are labelled as shadows.

	Not visible	Visible
Not visible	87.6%	12.4%
Visible	20.5%	79.5%

Table 6.1: Confusion matrix for the sun visibility classifier. Overall, the class-normalized classification accuracy is 83.5%.

These features are concatenated in a 20-dimensional vector which is used in the learning framework described next.

6.2.2 Sun visibility classifier

We employ a classical supervised learning formulation, in which the image features are first pre-computed on a set of manually-labelled training images, and then fed to a classifier. We now provide more details on the learning setup used to predict whether or not the sun is visible in an image.

We selected a random subset of outdoor images from the LabelMe dataset [Russell *et al.*, 2008], split into 965 images used for training, and 425 for testing. Because LabelMe images taken from the same folders are likely to come from the same location or taken by the same camera, the training and test sets were carefully split to avoid folder overlap. Treating V as a binary variable, we then manually labelled each image with either $V = 1$ if the sun is shining on the scene, or $V = 0$ otherwise. Notice that this binary representation of V is only a coarse approximation of the physical phenomenon: in reality, the sun may have fractional visibility (e.g., due to partially-occluding clouds, for example). In practice, however, we find that it is extremely difficult, even for humans, to reliably estimate a continuous value for the sun visibility from a single image. Additionally, precisely measuring this value requires expensive equipment that is not available in databases of existing images.

We then train a binary, linear SVM classifier using this training dataset, and evaluate its performance on the test set. We use the libsvm package [Chang and Lin, 2001], and convert the SVM scores to probabilities using the sigmoid fitting method of Platt [1999]. Overall, we have found this method to work quite well on a variety of images, with a resulting class-normalized test classification accuracy obtained is 83.5%, and the full confusion matrix is shown in Table 6.1. Qualitative results are also shown in Fig. 6.2.

Now that we have a classifier that determines whether or not the sun is visible in the image $P(V|\mathcal{I})$, we consider how we can estimate the remaining factor in our representation (6.1): the distribution over sun positions S given that the sun is visible $P(S|V=1,\mathcal{I})$.

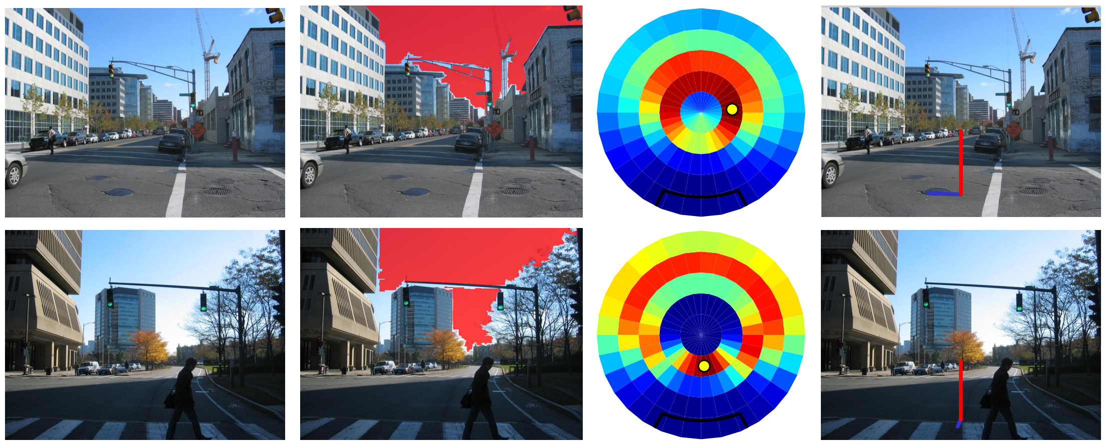
6.3 Image cues for predicting the sun direction

When the sun is visible, its position affects different parts of the scene in very different ways. In our approach, information about the sun position is captured from four major parts of the image—the sky pixels $\mathcal{S}$, the ground pixels $\mathcal{G}$, the vertical surface pixels $\mathcal{V}$ and the pixels belonging to pedestrians $\mathcal{P}$—via four cues. To partition the image in this way, we use the approach of Hoiem *et al.* [2007], which returns a pixel-wise labeling of the image, together with the pedestrian detector of Felzenszwalb *et al.* [2010] which returns the bounding boxes of potential pedestrian locations. Both these detectors also include a measure of confidence of their respective outputs.

As introduced in Sec. 2.4, we represent the sun position $S = \{\theta_s, \Delta\phi_s\}$ using two parameters: θ_s is the sun zenith angle, and $\Delta\phi_s$ the sun azimuth angle with respect to the camera. This section describes how we compute distributions over these parameters given the sky, the shadows, the shading on the vertical surfaces, and the detected pedestrians individually. Afterwards, in Sec. 8.5, we will see how to combine these cues to estimate the sun position given the entire image.

6.3.1 Sky

In order to estimate the sun position angles from the sky, we take inspiration from the insight we gained by looking at image sequences (Ch. 6), and see how we can use the same physically-based model of the sky [Perez *et al.*, 1993] in the context of a single image. If we, for now, assume that the sky is completely clear, our solution is then to discretize the parameter space and try to fit the sky model for each parameter setting. For this, we assume that the sky pixel intensities $s_i \in \mathcal{S}$ are conditionally independent given the sun position, and are distributed according to the following generative model, function of the Perez sky



(a) Image and estimated horizon (b) Sky mask [Hoiem *et al.*, 2007] (c) $P(\theta_s, \Delta\phi_s|\mathcal{S})$ (d) Inserted sun dial

Figure 6.3: Illumination cue from the sky only. Starting from the input image (a), we compute the sky mask (b) using [Hoiem *et al.*, 2007]. The resulting sky pixels are then used to estimate $P(\theta_s, \Delta\phi_s|\mathcal{S})$ (c). The maximum likelihood sun position is shown with a yellow circle. We use this position to artificially synthesize a sun dial in the scene (d).

model $g(\cdot)$ [Perez *et al.*, 1993], the image coordinates (u_i, v_i) of s_i, and the focal length f_c:

$$s_i \sim \mathcal{N}(k\, g(\theta_s, \Delta\phi_s, u_i, v_i, f_c), \sigma_s^2)\,, \tag{6.2}$$

where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2; and k is an unknown scale factor (see Ch. 3 for details). We obtain the distribution over sun positions by computing

$$P(\theta_s, \Delta\phi_s|\mathcal{S}) \propto \exp\left(\sum_{s_i \in \mathcal{S}} \frac{-(s_i - k\, g(\theta_s, \Delta\phi_s, \ldots))^2}{2\sigma_s^2}\right) \tag{6.3}$$

for each bin in the discrete $(\theta_s, \Delta\phi_s)$ space, and normalizing appropriately. Note that since k in (6.2) and (6.3) is unknown, we first optimize for k for each sun position bin independently using a non-linear least-squares optimization scheme similar to the one introduced in (3.20) from Sec. 3.6, but optimizing over k instead of $\mathbf{x}$.

As it is indicated in (6.2), the sky model $g(\cdot)$ also requires knowledge of two important camera parameters: its zenith angle θ_c (needed to compute θ_s and its focal length f_c. If we assume that f_c is available via the EXIF tag of the photograph, then θ_c can be computed by finding the horizon line v_h in the image (assuming the camera has no roll angle). We circumvent the hard problem of horizon line estimation by making a simple approximation: select the row midway between the lowest sky pixel and the highest ground pixel as horizon. Note that all the results shown in this chapter have been obtained using this approximation, which we have found to work quite well in practice. Fig. 6.3 demonstrates the distribution over sun positions obtained using the sky cue.

So far, we have assumed that the sky is completely clear, which might not always be the case! Even if the sun is shining on the scene, the sky might be covered with clouds, thus rendering the physical model $g(\cdot)$ useless. To deal with this problem, we classify the sky into one of three categories: clear, partially cloudy, or completely overcast. For this, we build a small database of representative skies for each category from images downloaded from Flickr, and compute the illumination context feature [Lalonde *et al.*, 2007] on each. We then find the k nearest neighbors in the database, and assign the most common label (we use $k = 5$). If the sky is found to be overcast, the sun position distribution $P(\theta_s, \Delta\phi_s|\mathcal{S})$ is left uniform. For partly cloudy scenes, we remove the clouds with a simple binary color segmentation of the sky pixels (keeping the cluster that is closer to blue) and fit the sky model described earlier only to the clear portion of the sky.

6.3.2 Ground shadows cast by vertical objects

Shadows cast on the ground by vertical structures can essentially serve as "sun dials" and are often used by humans to determine the sun direction [Koenderink *et al.*, 2004]. Unfortunately, it is extremely hard to determine if a particular shadow was cast by a vertical object. Luckily, it turns out that due to the statistics of the world (gravity makes a many things stand-up straight), the majority of long shadows should be, in fact, produced by vertical things. Therefore, if we can detect a set of "long and strong" shadow lines (edges), we can use them in a probabilistic sense to determine a likely sun azimuth (up to the directional ambiguity). While shadows have also been used to estimate the sun direction with user input [Kim and Hong, 2005] or in webcams [Junejo and Foroosh, 2008], no technique so far has proposed to do so automatically from a single image.

We use the technique presented in Ch. 5 to detect the ground shadow boundaries in the image. From the resulting boundaries, we detect the long shadow lines on the ground $l_i \in \mathcal{G}$ by applying the line detection algorithm of [Košecká and Zhang, 2002]. Let us consider one shadow line l_i. If its orientation on the ground plane is denoted α_i, then the angle between the shadow line orientation and the sun azimuth angle is

$$\angle(\alpha_i, \phi_s) = \min\left\{\angle(\alpha_i, \phi_s), \angle(\alpha_i + 180^\circ, \phi_s)\right\} , \tag{6.4}$$

with the 180° ambiguity due to our assumption that we do not know which object is casting this shadow. Here where $\angle(\cdot, \cdot)$ denotes the angular difference. We obtain a non-parametric estimate for $P(\phi_s|\alpha_i)$ by detecting long lines on the ground truth shadow boundaries in 74 images from our shadow boundary dataset [Lalonde *et al.*, 2010b], in which we have manually labeled the sun azimuth. The distribution obtained from the resulting 1,700 shadow lines found is shown in Fig. 6.4a. The strongest peak, at $\angle(\alpha_i, \phi_s) < 5^\circ$, confirms our intuition that long shadow lines align with the sun direction. Another, smaller peak seems to rise at $\angle(\alpha_i, \phi_s) > 85^\circ$. This is explained by the roof lines of buildings, quite common in our database, which cast shadows that are perpendicular to the sun direction (Fig. 6.4b).

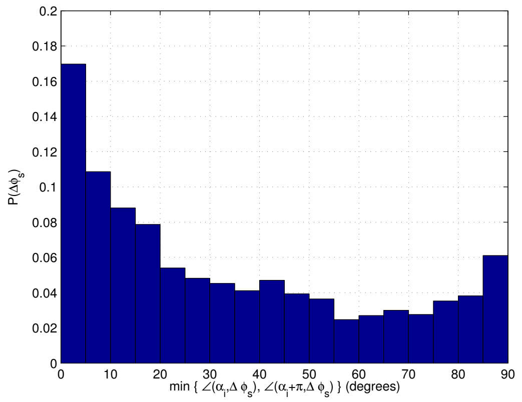
All the shadow lines are combined by making each one vote for its preferred sun direction:

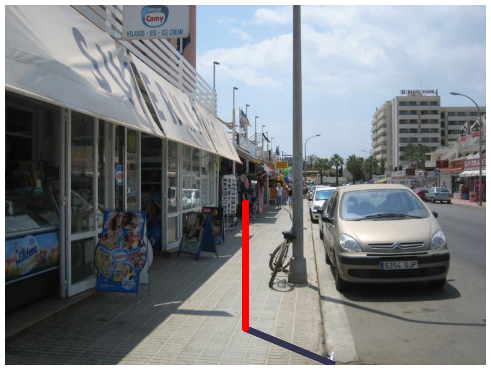
$$P(\Delta\phi_s|\mathcal{G}) \propto \sum_{l_i \in \mathcal{G}} P(\Delta\phi_s|\alpha_i) . \tag{6.5}$$

Of course, computing the shadow line orientation α_i on the ground requires knowledge of the zenith angle θ_c and focal length f_c of the camera. For this we use the estimates obtained in the previous section. Fig. 6.5 illustrates the results obtained using the shadow cue only.

6.3.3 Shading on vertical surfaces

If the rough geometric structure of the scene is known, then analyzing the shading on the main surfaces can often provide an estimate for the possible sun positions. For example, a brightly lit surface indicates that the sun may be pointing in the direction of its normal, or at least in the vicinity. Of course, this reasoning also assumes that the albedos of the surfaces are either known or equal, neither of which is true. However, we have experimentally found





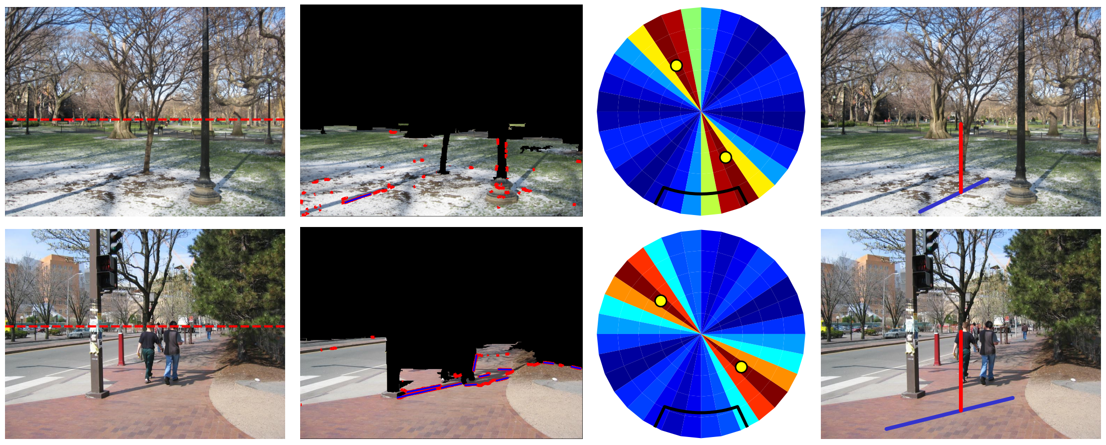
(a) (b)

Figure 6.4: How do shadow lines predict the sun azimuth? We use our shadow boundary dataset [Lalonde *et al.*, 2010b] to compute a non-parametric estimate for $P(\Delta\phi_s|\angle(\alpha_i, \phi_s))$ (a). A total of 1,700 shadow lines taken from 74 images were used to estimate the distribution. (b) Example image from the shadow dataset which contains shadows both aligned with and perpendicular to the sun (lamppost and roof line respectively). For visualization, the ground truth sun direction is indicated by the red stick and its shadow.

that, within a given image, the albedos of the major *vertical* surfaces are often relatively similar (e.g., different sides of the same house, or similar houses on the same street), while the ground is quite different. Therefore, we use the three coarse vertical surface orientations (front, left-facing, and right-facing) computed by Hoiem *et al.* [2007] and attempt to estimate the azimuth direction only.

Intuitively, we assume that a surface $w_i \in \mathcal{V}$ should predict that the sun lies in front of it if it is bright. On the contrary, the sun should be behind it if the surface is dark. We discover the mapping between the average brightness of the surface and the relative sun position with respect to the surface normal orientation β_i by manually labeling the sun azimuth in the Geometric Context dataset [Hoiem *et al.*, 2007]. In particular, we learn the relationship between surface brightness b_i and whether the sun is in front of or behind the surface, i.e., whether $\angle(\beta_i, \Delta\phi_s)$ is less, or greater than 90°. This is done by computing the average brightness of all the vertical surfaces in the dataset, and applying logistic regression to learn $P(\angle(\beta_i, \Delta\phi_s|b_i) < 90°)$. This effectively models the probability in the following fashion:

$$P(\angle(\beta_i, \Delta\phi_s) < 90°|b_i) = \frac{1}{1 + e^{-(x_1 + x_2 b_i)}} , \tag{6.6}$$



(a) Image and estimated horizon (b) Ground [Hoiem *et al.*, 2007] and shadows boundaries (c) $P(\theta_s, \Delta\phi_s|\mathcal{G})$ (d) Inserted sun dial

Figure 6.5: Illumination cue from the shadows only. Starting from the input image (a), we compute the ground mask using [Hoiem *et al.*, 2007], estimate shadow boundaries using [Lalonde *et al.*, 2010a] (shown in red), and finally extract long lines [Košecká and Zhang, 2002] (shown in blue) (b). The long shadow lines are used to estimate $P(\theta_s, \Delta\phi_s|\mathcal{G})$ (c). Note that shadow lines alone can only predict the sun relative azimuth angle up to a $180°$ ambiguity. For visualization, the two most likely sun positions (shown with yellow circles) are used to artificially synthesize a sun dial in the scene (d).

where, after learning, $x_1 = -3.35$ and $x_2 = 5.97$. Fig. 6.6 shows the computed data points and fitted model obtained with this method. As expected, a bright surface (high b_i) predicts that the sun is more likely to be in front of it than behind it, and vice-versa for dark surfaces (low b_i). In practice, even if the sun is shining on a surface, a large portion of it might be in shadows because of occlusions. Therefore, b_i is set to be the average brightness of the brightest cluster, computed using k-means with $k = 2$.

To model the distribution of the sun given a vertical surface of orientation β_i, we use:

$$P(\Delta\phi_s|w_i) \sim \begin{cases} \mathcal{N}(\beta_i, \sigma_w^2) & \text{if } (6.6) \geq 0.5 \\ \mathcal{N}(\beta_i + 180°, \sigma_w^2) & \text{if } (6.6) < 0.5 \end{cases}, \tag{6.7}$$

where σ_w^2 is such that the fraction of the mass of the gaussian $\mathcal{N}$ that lies in front of (behind) the surface corresponds to $P(\angle(\beta_i, \Delta\phi_s) < 90°|b_i)$ if it is greater (less) than 0.5. Note that $\beta_i \in \{-90°, 90°, 180°\}$ since we assume only 3 coarse surface orientations. We combine each

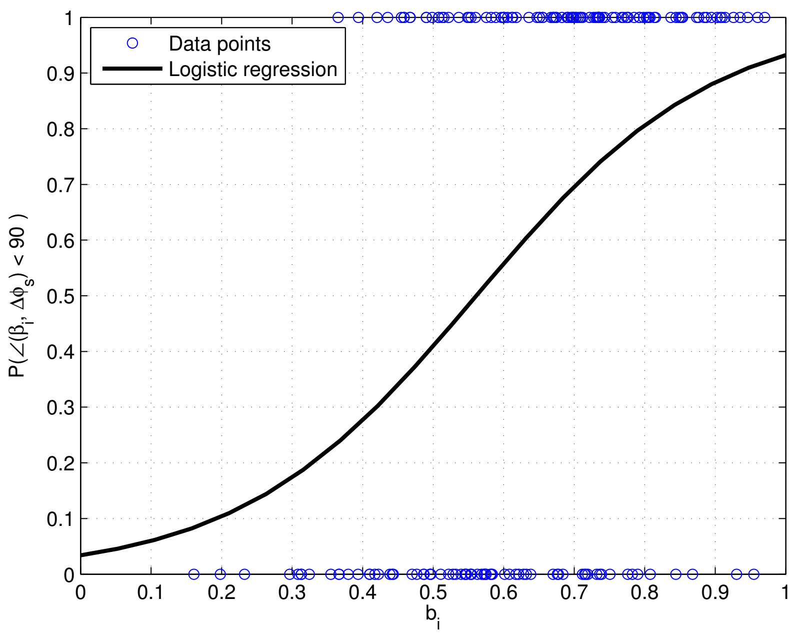


Figure 6.6: How do the vertical surfaces predict the sun? From the Geometric Context dataset [Hoiem *et al.*, 2007], the mapping between the probability of the sun being in front of a surface and its average brightness b_i is learned using logistic regression. Bright surfaces (high b_i) predict that the sun is in front of them (high probability), and vice-versa for dark surfaces (low b_i).

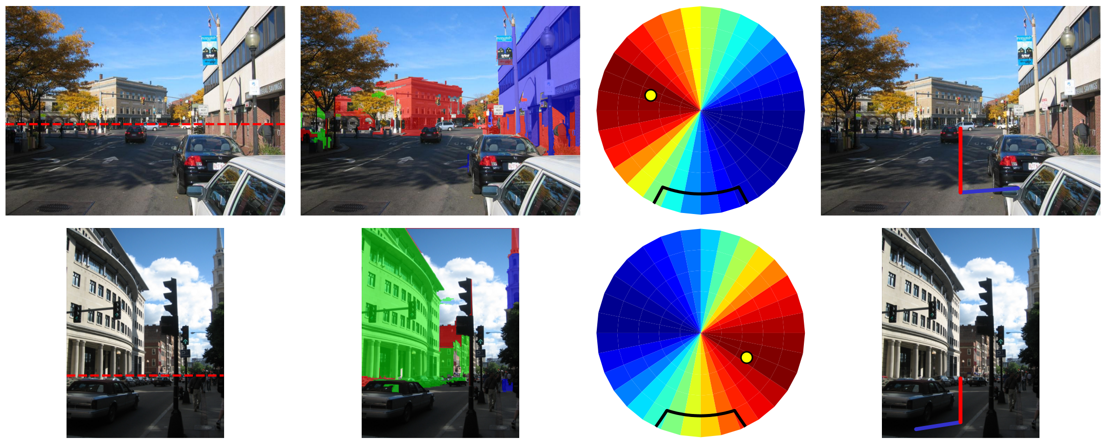
surface by making each one vote for its preferred sun direction:

$$P(\Delta\phi_s|\mathcal{V}) \propto \sum_{w_i \in \mathcal{V}} P(\Delta\phi_s|w_i)\,. \tag{6.8}$$

Fig. 6.7 shows the sun azimuth prediction results obtained using the shading on vertical surfaces only. We find that this cue can often help resolve the ambiguity arising with shadow lines.

6.3.4 Pedestrians

When convex objects are present in the image, their appearance can also be used to predict where the sun is [Langer and Büelthoff, 2001]. One notable example of this idea is the work of Bitouk *et al.* [2008] which uses faces to recover illumination in a face swapping application. However, front-looking faces are rarely of sufficient resolution in the type of



(a) Image and estimated horizon (b) Vertical surfaces [Hoiem *et al.*, 2007] (c) $P(\theta_s, \Delta\phi_s|\mathcal{V})$ (d) Inserted sun dial

Figure 6.7: Illumination cue from the vertical surfaces only. Starting from the input image (a), we compute the vertical surfaces mask (b) using [Hoiem *et al.*, 2007] (blue = facing left, green = facing right, red = facing forward). The distribution of pixel intensities on each of these surfaces are then used to estimate $P(\theta_s, \Delta\phi_s|\mathcal{V})$ (c). Note that in our work, vertical surfaces cannot predict the sun zenith angle θ_s. In these examples, the sun zenith is set to $45°$ for visualization. We then find the most likely sun position (shown with a yellow circle), which is used to artificially synthesize a sun dial in the scene (d).

images we are considering (they were using mostly high resolution closeup portraits), but entire people more often are. As shown in Fig. 6.8, when shone upon by the sun, pedestrians also exhibit strong appearance characteristics that depend on the sun position: shadows are cast at the feet, a horizontal intensity gradient exists on the persons body, the wrinkles in the clothing are highlighted, etc. It is easy for us to say that one person is lit from the left (Fig. 6.8a), or from the right (Fig. 6.8b).

Because several applications require detecting pedestrians in an image (surveillance or safety applications in particular), they are a class of objects that has received significant attention in the literature [Dalal and Triggs, 2005; Dollár *et al.*, 2009; Park *et al.*, 2010; Felzenszwalb *et al.*, 2010], and as such many efficient detectors exist. In this section, we present a novel approach to estimate the azimuth angle of the sun with respect to the camera given detected pedestrian bounding boxes in an image. In our work, the pedestrians are detected using [Felzenszwalb *et al.*, 2010] which uses the standard Histogram of Gradients (HOG) features [Dalal and Triggs, 2005] for detection. The detector is operated at a high-

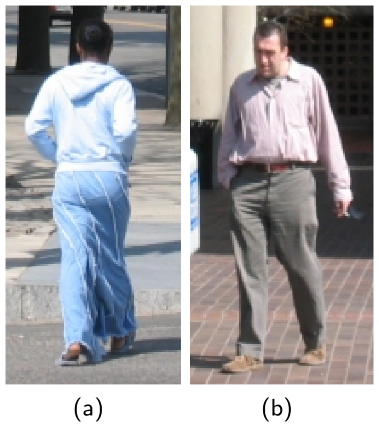
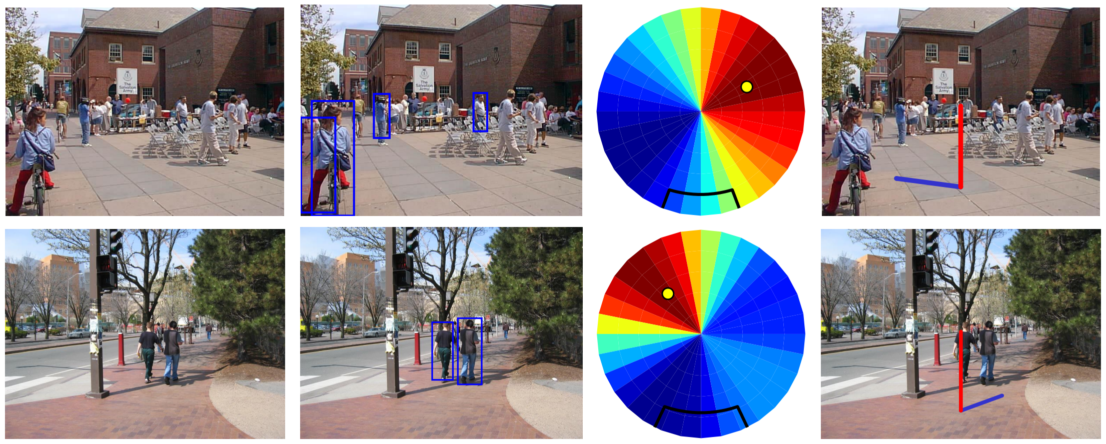


Figure 6.8: Looking at these pedestrians, it is easy to say that (a) is lit from the left, and (b) from the right. In this work, we train a classifier that predicts the sun direction based on the appearance of pedestrians.

precision, low-recall setting to ensure that only very confident detections are used.

We employ a supervised learning approach to the problem of predicting the sun location given the appearance of a pedestrian. In particular, a set of 2,000 random images were selected from the LabelMe dataset [Russell *et al.*, 2008], which contain the ground truth location of pedestrians, and for which we also manually labelled the ground truth sun azimuth. To make the problem more tractable, the space of sun azimuths $\Delta\phi_s \in [-180^\circ, 180^\circ]$ is discretized into four bins of 90° intervals: $[-180^\circ, -90^\circ]$, $[-90^\circ, 0^\circ]$, and so forth. We then train a multi-class SVM classifier which uses the same HOG features used for detection. However, the difference now is that the classifier is *conditioned* on the presence of a pedestrian in the bounding box, so it effectively learns different feature weights that capture effects only due to illumination. In practice, the multi-class SVM is implemented as a set of four one-vs-all binary SVM classifiers. The SVM training is done with the libsvm library [Chang and Lin, 2001], and the output of each is normalized via a non-linear least-squares sigmoid fitting process [Platt, 1999] to obtain a probability for each class.

Of course, the illumination effects on pedestrians shown in Fig. 6.8 and captured by the classifier only arise when a pedestrian is actually lit by the sun. However, since buildings or scene structures frequently cast large shadows upon the ground, pedestrians may very well be in the shade. To account for that, we train another binary SVM classifier to predict



(a) Image (b) Detected pedestrians [Felzenszwalb *et al.*, 2010] (c) $P(\theta_s, \Delta\phi_s|\mathcal{P})$ (d) Inserted sun dial

Figure 6.9: Illumination cue from the pedestrians only. Starting from the input image (a), we detect pedestrians using the detector of Felzenszwalb *et al.* [2010] (b). Each of these detections is used to estimated $P(\Delta\phi_s|\mathcal{P})$ (c) by running our pedestrian-based illumination predictor, and combining all predictions via a voting framework. The most likely sun azimuth (shown with a yellow circle) is used to artificially synthesize a sun dial in the scene (d). For display purposes, the most likely sun zenith is set to be $45°$.

whether the pedestrian is in shadows or not. This binary classifier uses simple features computed on the bounding box region, such as coarse 5-bin histograms in the HSV and RGB colorspaces, and a histogram of oriented gradients within the entire bounding box to capture contrast. This simple "sunlit pedestrian" classifier results in more than 82% classification accuracy. Fig. 6.9 shows the sun azimuth estimation results obtained using the automatically-detected sunlit pedestrians only.

As with the previous cues, we compute the probability $P(\Delta\phi_s|p_i)$ of the sun azimuth given a single pedestrian p_i by making each one vote for its preferred sun direction:

$$P(\Delta\phi_s|\mathcal{P}) \propto \sum_{p_i \in \mathcal{P}} P(\Delta\phi_s|p_i) \tag{6.9}$$

6.4 Estimating the sun position

Now that we are equipped with several features that can be computed over an image, we show how we combine them in order to get a more reliable estimate. Because each cue can be very weak and might not even be available in any given image, we combine them in a probabilistic framework that captures the uncertainty associated with each of them.

6.4.1 Cue combination

We are interested in estimating the distribution $P(S|V = 1, \mathcal{I})$ over the sun position $S = \{\theta_s, \Delta\phi_s\}$, given the entire image $\mathcal{I}$ and assuming the sun is visible (see Sec. 6.2). We saw in the previous section that the image $\mathcal{I}$ is divided into features computed on the sky $\mathcal{S}$, the shadows on the ground $\mathcal{G}$, the vertical surfaces $\mathcal{V}$ and pedestrians $\mathcal{P}$, so we apply Bayes rule and write

$$P(S|\mathcal{S}, \mathcal{G}, \mathcal{V}, \mathcal{P}) \propto P(\mathcal{S}, \mathcal{G}, \mathcal{V}, \mathcal{P}|S)P(S) \,. \tag{6.10}$$

We make the Naive Bayes assumption that the image pixels are conditionally independent given the illumination conditions, and that the priors on each region of the image ($P(\mathcal{S})$, $P(\mathcal{G})$, $P(\mathcal{V})$, and $P(\mathcal{P})$) are uniform over their own respective domains. Applying Bayes rule twice, we get

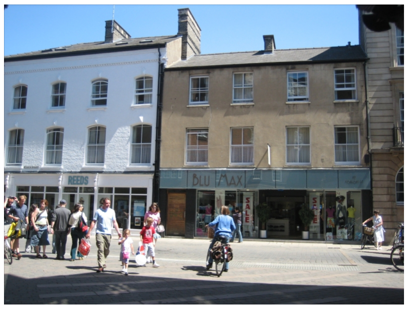
$$P(S|\mathcal{S}, \mathcal{G}, \mathcal{V}, \mathcal{P}) \propto P(S|\mathcal{S})P(S|\mathcal{G})P(S|\mathcal{V})P(S|\mathcal{P})P(S) \,. \tag{6.11}$$

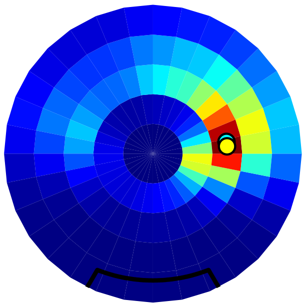
The process of combining the cues according to (6.11) for the sun position is illustrated in Fig. 6.10. We have presented how we compute the conditionals $P(S|\mathcal{S})$, $P(S|\mathcal{G})$, $P(S|\mathcal{V})$, and $P(S|\mathcal{P})$ in Sec. 6.3.1, 6.3.2, 6.3.3 and 6.3.4 respectively. We now look at how we can compute the prior $P(S)$ on the sun position themselves.

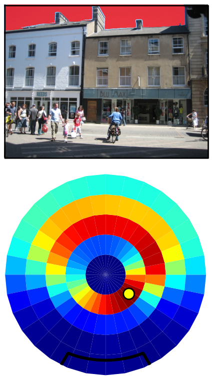
6.4.2 Data-driven illumination prior

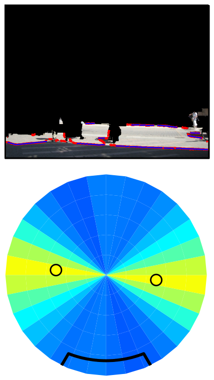
The prior $P(S) = P(\theta_s, \Delta\phi_s)$ captures the typical sun positions in outdoor scenes. We now proceed to show how we can compute it given a large dataset of consumer photographs.

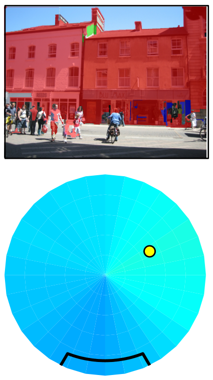
The sun position $(\theta_s, \Delta\phi_s)$ depends on the latitude L of the camera, its azimuth angle

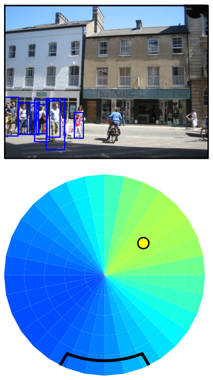

(a) Input image


(b) $P(\theta_s, \Delta\phi_s|\mathcal{I})$


(c) $P(\theta_s, \Delta\phi_s|\mathcal{S})$


(d) $P(\theta_s, \Delta\phi_s|\mathcal{G})$


(e) $P(\theta_s, \Delta\phi_s|\mathcal{V})$


(f) $P(\theta_s, \Delta\phi_s|\mathcal{P})$

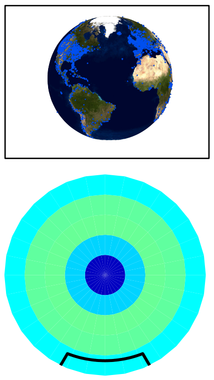

(g) $P(\theta_s, \Delta\phi_s)$

Figure 6.10: Combining illumination features computed from image (a) yields a more confident final estimate (b). We show how (c) through (f) are estimated in Sec. 6.3, and how we compute (g) in Sec. 6.4.2. To visualize the relative confidence between the different cues, all probability maps in (c) through (g) are drawn on the same color scale.

ϕ_c, the date D and the time of day T expressed in the local timezone:

$$P(\theta_s, \Delta\phi_s) = P(f(L, D, T, \phi_c)) \,, \tag{6.12}$$

where $f(\cdot)$ is a non-linear function defined in [Reda and Andreas, 2005]. To estimate (6.12), we can sample points from $P(L, D, T, \phi_c)$, and use $f(\cdot)$ to recover θ_s and $\Delta\phi_s$. But estimating this distribution is not currently feasible since it requires images with known camera orientations ϕ_c, which are not yet available in large quantities. On the other hand, geo- and time-tagged images do exist, and are widely available on photo sharing websites such as Flickr. The database of 6 million images from Hays and Efros [2008] is used to compute the empirical distribution $P(L, D, T)$. We compute (6.12) by randomly sampling

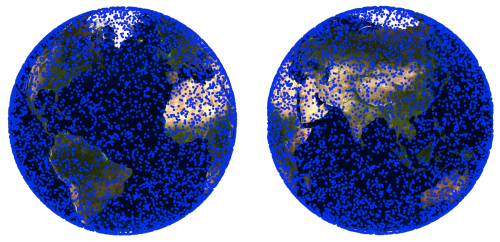
1 million points from the distribution $P(L, D, T)P(\phi_c)$, assuming $P(\phi_c)$ to be uniform in the $[-180°, 180°]$ interval. As a consequence, (6.12) is flat along the $\Delta\phi_s$ dimension and is marginalized.

Fig. 6.11c shows 4 estimates for $P(\theta_s)$, computed with slightly different variations. First, a uniform sampling of locations on Earth and times of day is used as a baseline comparison. The three other priors use data-driven information. Considering non-uniform date and time distributions decrease the likelihood of having pictures with the sun taken close to the horizon (θ_s close to 90°). Interestingly, the red and green curve overlap almost perfectly, which indicates that the three variables L, D, and T seem to be independent.

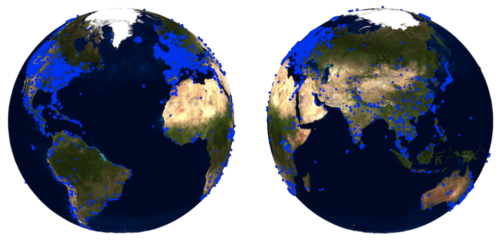
This database captures the distribution of where photographs are most likely to be taken on the planet, which is indeed very different than considering each location on Earth as equally likely (as shown in Figs 6.11a and 6.11b). We will show in the next section that this distinction is critical to improve our estimation results. Finally, note that the assumption of uniform camera azimuth is probably not true in practice since a basic rule of thumb of good photography is to take a picture with the sun to the back. With the advent of additional sensors such as compasses on digital cameras, this data will surely become available in the near future.

6.5 Evaluation and results

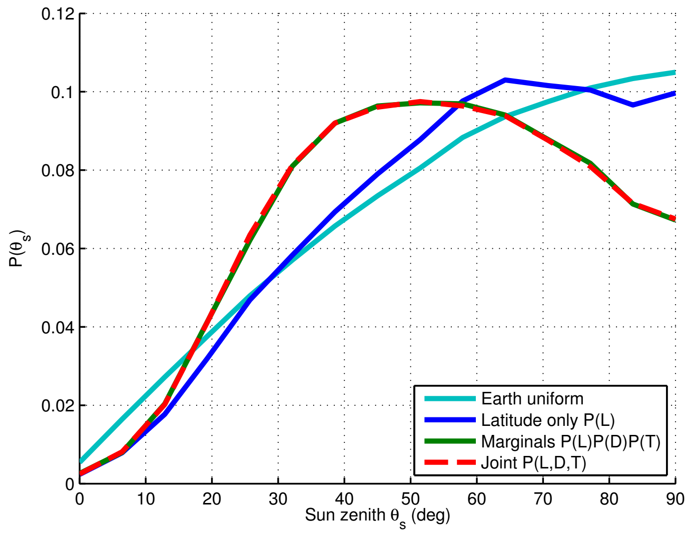
We evaluate our technique in three different ways. First, we quantitatively evaluate our sun estimation technique for both zenith and azimuth angles using images taken from calibrated webcam sequences. Second, we show another quantitative evaluation this time for the sun azimuth only, but performed on a dataset of manually labelled single images downloaded from Internet. Finally, we also provide several qualitative results on single images that demonstrate the performance of our approach. These results will show that, although far from being perfect, our approach is still able to exploit visible illumination cues to reason about the sun.



(a) Uniform



(b) Data-driven



(c)

Figure 6.11: Illumination priors (6.12) on the sun zenith angle. The Earth visualizations illustrate the (a) uniform and (b) data-driven sampling over GPS locations that are compared in this work. The data-driven prior is learned from a database of 6M images downloaded from Flickr. (c) The priors are obtained by sampling 1 million points (1) uniformly across GPS locations and time of day (cyan); from (2) the marginal latitude distribution $P(L)$ only (blue); (3) the product of independent marginals $P(L)P(D)P(T)$ obtained from data (green); and (4) the joint $P(L, D, T)$, also obtained from data (red). The last two curves overlap, indicating that the three variables L, D, and T indeed seem to be independent.

6.5.1 Quantitative evaluation using webcams

We select 984 images taken from 15 different time-lapse image sequences from our webcam dataset [Lalonde *et al.*, 2009c]. Each webcam is calibrated using the technique presented in Ch. 3. Two example images from our dataset are shown in Fig. 6.12a and 6.12b. Our algorithm is applied to every image from the sequences independently and the results are compared against ground truth.

Fig. 6.12 reports the cumulative histogram of errors in sun position estimation for different scenarios: chance, making a constant prediction of $\theta_s = 0$ (straight up), using only the priors from Sec. 6.4.2 (we tested both the data-driven and Earth uniform priors), scene cues only, and using our combined measure $P(\theta_s, \Delta\phi_s|\mathcal{I})$ with both priors as well. Fig. 6.12 highlights the performance at errors of less than 22.5° (50% of images) and 45° (71% of images), which correspond to accurately predicting the sun position within an octant (e.g., North vs. North-West), or a quadrant (e.g., North vs West) respectively.

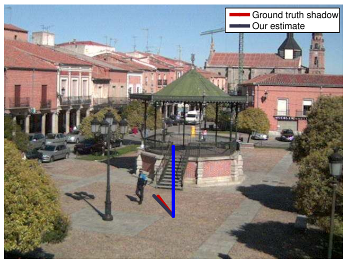
The cues contribute to the end result differently for different scenes. For instance, the sky in Fig. 6.12a is not informative, since it is small and the sun is always behind the camera. It is more informative in Fig. 6.12b, as it occupies a larger area. Pedestrians may appear in Fig. 6.12a and may be useful; however they are too small in Fig. 6.12b to be of any use.

6.5.2 Quantitative azimuth evaluation on single images

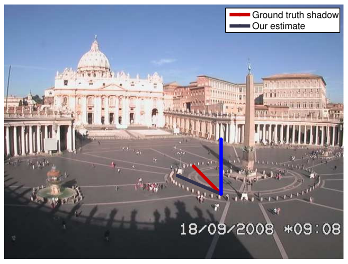
When browsing popular publicly available webcam datasets [Lalonde *et al.*, 2009b; Jacobs *et al.*, 2007a], one quickly realizes that the types of scenes captured in such sequences are inherently different than those found in single images. Webcams are typically installed at high vantage points, giving them a broad overlook on large-scale panoramas such as natural or urban landscapes. On the other hand, single images are most often taken at human eye level, where earthbound "objects" like cars, pedestrians, bushes or trees, occupy a much larger fraction of the image.

In addition, the scene in a webcam sequence does not change over time (considering static cameras only), it is the illumination conditions that vary. In single images, however, both the scenes *and* illumination conditions differ from one image to the next. In this section, we evaluate our method on a dataset of single images. Admittedly, this task is much harder than the case of webcams, but we still expect to be able to extract meaningful information across a wide variety of urban and natural scenes.

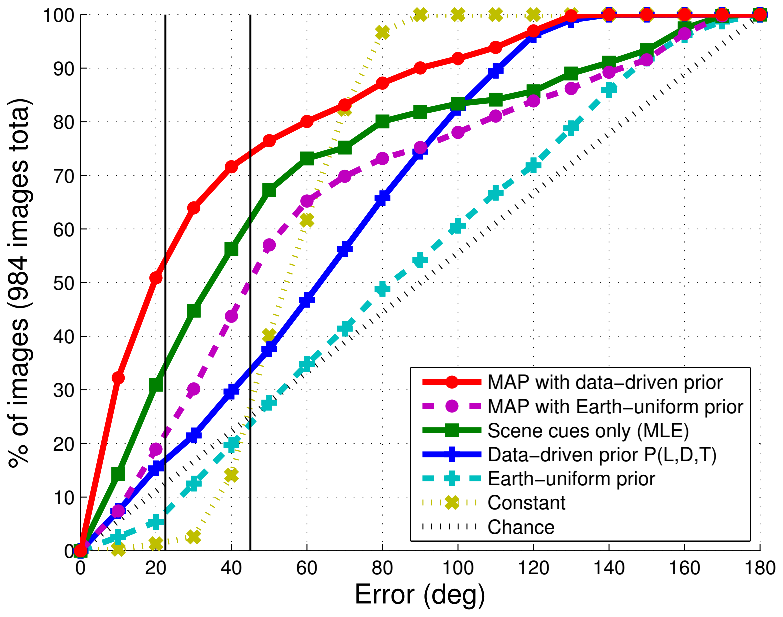
The main challenge we face here is the unavailability of ground truth sun positions in single images: as of this day, there exist no such publically available dataset. Additionally, as discussed in Sec. 6.4.2, while the GPS coordinates and time of capture of images are



(a) Madrid sequence



(b) Vatican sequence



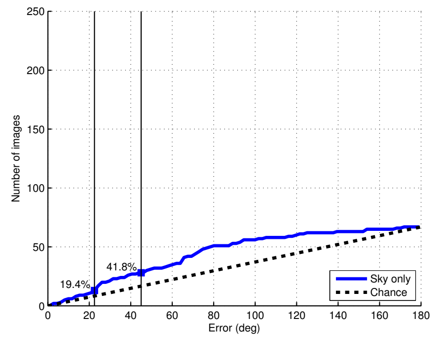
(c)

Figure 6.12: Quantitative evaluation using 984 images taken from 15 webcam sequences, calibrated using the method presented in Ch. 3. Two example images, from (a) Madrid and (b) Vatican City, compare sun dials rendered using our estimated sun position (gray), and the ground truth (red). (c) Cumulative sun position error (angle between estimate and ground truth directions) for different methods. Our result, which combines both the scene cues and the data-driven illumination prior, outperforms the others. The data-driven prior surpasses the Earth-uniform one (both alone and when combined with the scene cues), showing the importance of being consistent with likely image locations. Picking a constant value of $\theta_s = 0$ results in error of at least $20°$.

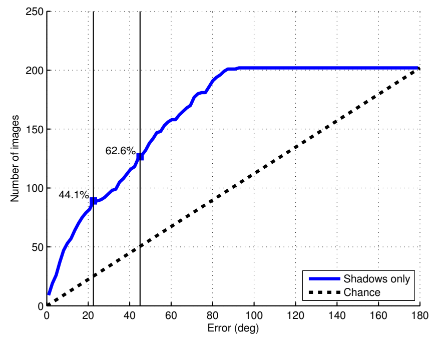
commonly recorded by modern-day cameras, the camera azimuth angle ϕ_c is not. We randomly selected 300 outdoor images from the LabelMe dataset [Russell *et al.*, 2008] where the sun appears to be shining on the scene (i.e., is not occluded by a cloud or very large building), and manually labelled the sun position in each one of them. The labeling was done using an interactive graphical interface that resembled the "virtual sun dials" used throughout this chapter (see Fig. 6.4b for example), where the task of the human labeler was to orient the shadow so that it aligned with the perceived sun direction. If the labeler judged that he or she cannot identify the sun direction with sufficient accuracy in a given image, that image was discarded from the dataset. After the labeling process, 239 images were retained for evaluation. In addition, we have found that reliably labeling the sun zenith angle θ_s is very hard to do in the absence of objects of known heights [Koenderink *et al.*, 2004], so we ask the user to label the sun relative azimuth $\Delta\phi_s$ only.

Fig. 6.13 reports the cumulative histograms of errors in sun azimuth estimation for each of the cues independently (Figs 6.13a through 6.13d), and jointly (Fig. 6.13e). We also report the percentage of images which have less than 22.5° and 45° errors, corresponding to correctly estimating the sun azimuth within an octant and quadrant, respectively. Since each cue is not available in all images, we also report the percentage of images for which each cue is available.

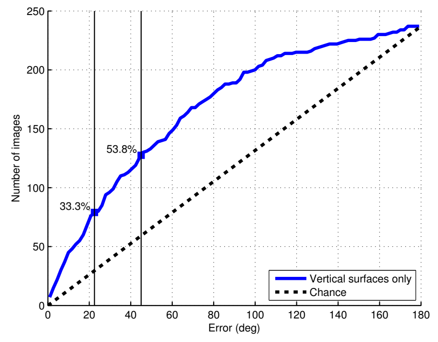
Let us point out a few noteworthy observations. First, since the shadow lines alone (Fig. 6.13b) have a 180° ambiguity in azimuth estimation, therefore we keep the minimum error between each predicted (opposite) direction and the ground truth sun azimuth. This explains why the minimum error is 90°, and as such should not be compared directly with the other cues. A second note is that the vertical surfaces (Fig. 6.13c) are available in 99% of the images in our dataset, which indicates that our dataset might be somewhat biased towards urban environments, much like LabelMe currently is. Also observe how the pedestrians are, independently, one of the best cues when available (Fig. 6.13d). Their performance is the best of all cues. As a side note, we also observed that there is a direct correlation between the number of pedestrians visible in the image and the quality of the azimuth estimate: the more the better. For example, the mean error is 41° when there is only one pedestrian, 22° with two, and 20° with three or more. Finally, the overall system (Fig. 6.13e) estimates the sun azimuth angle within 22.5° for 40.2% of the images, and within 45° for 54.8% of the images, which is better than any of the cues taken independently.



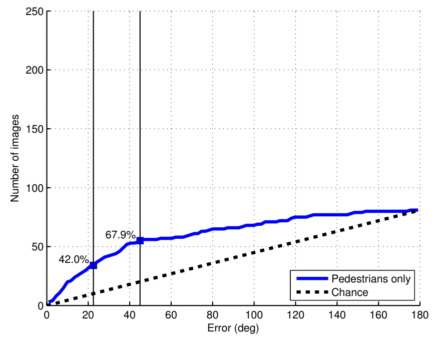
(a) Sky only (28% of images)



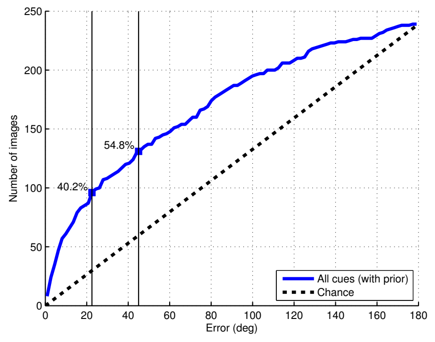
(b) Shadows only (86% of images)



(c) Vertical surfaces only (99% of images)



(d) Pedestrians only (34% of images))



(e) All cues (with prior, 100% of images)

Figure 6.13: Cumulative sun azimuth error (angle between estimate and ground truth directions) on a set of 239 single images taken from the LabelMe dataset [Russell *et al.*, 2008], for the individual cues presented in Sec. 6.3: (a) the sky, (b) the shadows cast on the ground by vertical objects, (c) the vertical surfaces, and (d) the pedestrians. The plot in (e) shows the result obtained by combining all cues together with the sun prior, as discussed in Sec. 8.5. The percentages indicated represent the fraction of images in the test set for which the cue is available.

6.5.3 Qualitative evaluation on single images

Fig. 6.14 shows several example results of applying our algorithm on typical consumer-grade images taken from our test set introduced in Sec. 6.5.2. The rows are arranged in order of decreasing order of estimation error. Notice how our technique recovers the entire *distribution* over the sun parameters $(\theta_s, \Delta\phi_s)$ which also captures the degree of confidence in the estimates. High confidence cases are usually obtained when one cue is very strong (i.e., large intensity difference between vertical surfaces of different orientations in the 5th row, 2nd column), or when all 4 cues are correlated (as in Fig. 6.10). On the other hand, highly cluttered scenes with few vertical surfaces, shadowed pedestrians, or shadows cast by vegetation (4th row, 2nd column of Fig. 6.14) usually yield lower confidences, and the most likely sun positions might be uncertain. This is emphasized in Fig. 6.15, which shows two typical failure cases.

6.6 Discussion

We now discuss three important aspects of our approach: the distribution over the sun positions, how we can exploit additional sources of information when available, and higher-order interactions between cues.

6.6.1 Distribution Over Sun Positions

The quantitative evaluation presented in Sec. 6.5.2 revealed that our method successfully predicts the sun azimuth within $22.5°$ for 40% of the images in our test set, within $45°$ for 55% of them, and within $90°$ for 80% (see Fig. 6.13e). Admittedly, this is far from perfect, and we provide ideas on how to improve upon these results in Ch. 9. However, we believe this is still a very useful result for applications which might not require a very precise estimate. For instance, all that might be required in some cases is to correctly identify the sun quadrant ($< 45°$ error), or distinguish between left and right, or front and back ($< 90°$ error). In these cases, our method obtains very reasonable results. But these most likely sun positions used to generate these results actually hide an important result of our method which is not captured by this evaluation.

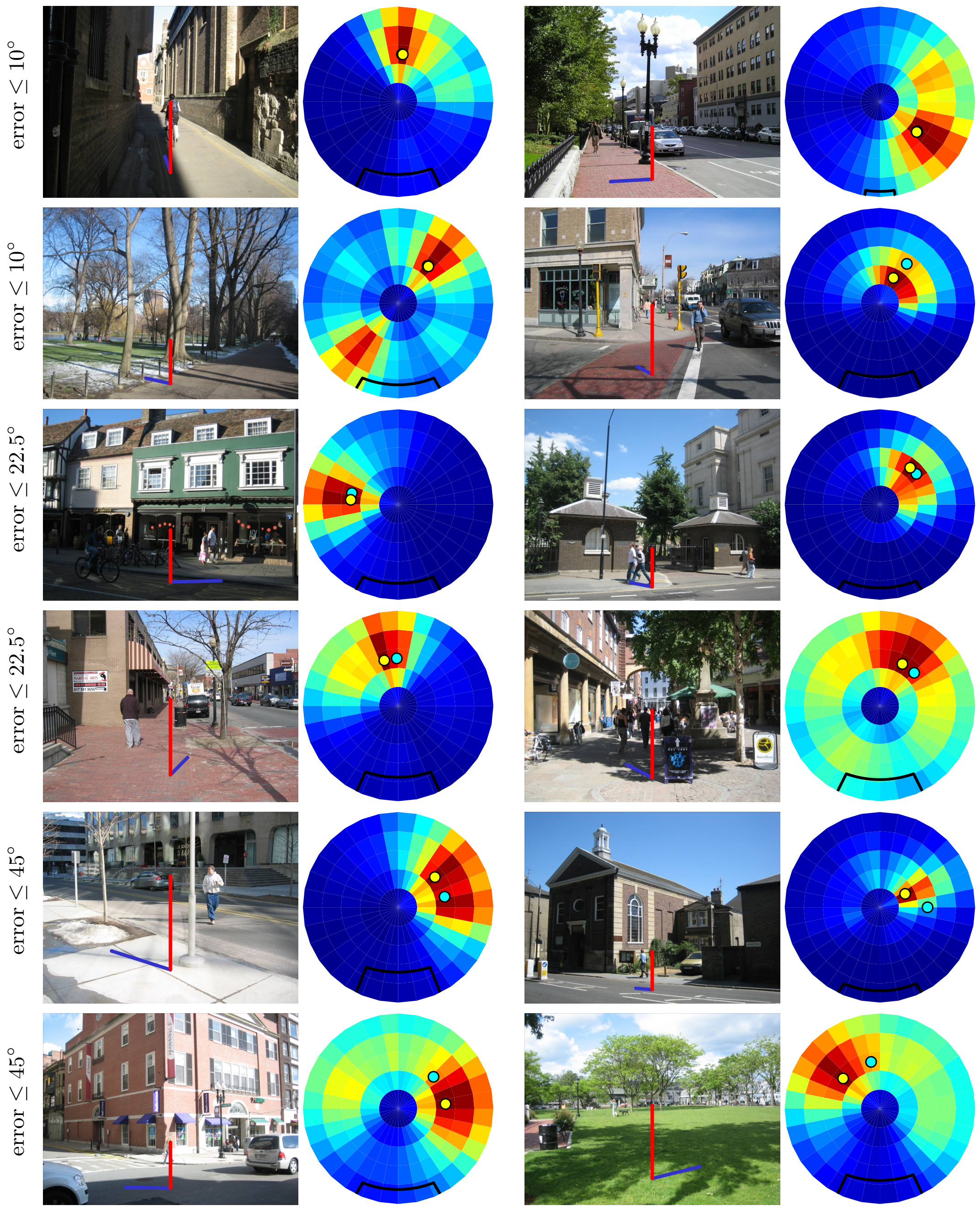
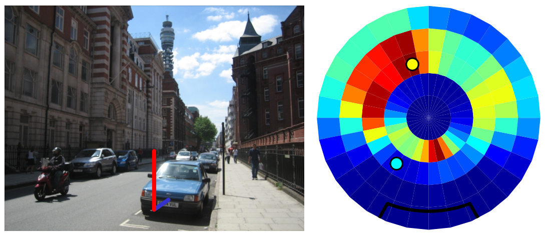


Figure 6.14: Sun direction estimation from a single image. A virtual sun dial is inserted in each input image (first and third columns), whose shadow correspond to the MAP sun position in the corresponding probability maps $P(\theta_s, \Delta\phi_s|\mathcal{I})$ (second and fourth columns). The ground truth sun azimuth is shown in cyan, and since it is not available, a zenith angle of 45° is used for visualization.


(a) Shadows cast by non-vertical objects

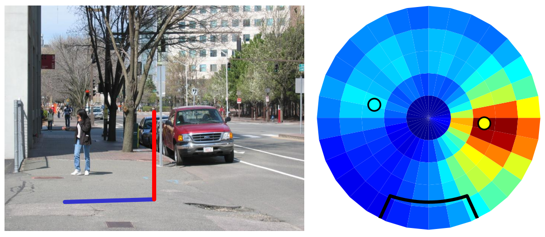
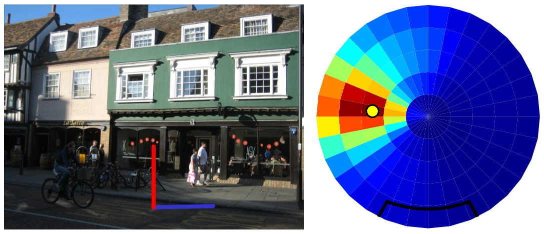

(b) Wrong shadow direction selected

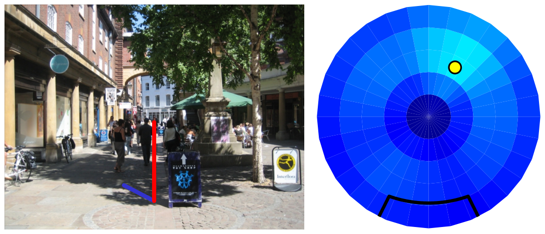
Figure 6.15: Typical failure cases. (a) First, the dominating shadow lines are not cast by thin vertical objects, and are therefore not aligned with sun direction. (b) Mislabeling the vertical surfaces orientation causes the wrong shadow direction to be selected. Note that the pedestrian visible in this example was not a high-confidence detection, thus unused for the illumination estimate. In both cases, none of the other cues were confident enough to compensate, thus yielding erroneous estimates.

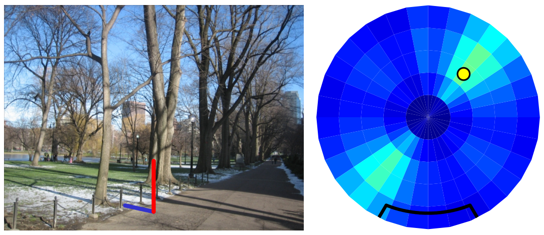
While it makes intuitive sense to report the results in terms of angular errors in sun position estimation, the real "output" of our system is the *probability distribution* over the sun position. Throughout this chapter, we have displayed those using colored circles, indicating the likelihood of the sun being at each position, but without paying much attention to them. A closer look at the distributions themselves reveal very interesting observations about the degree of certainty of the estimate, which itself should be a very useful result.

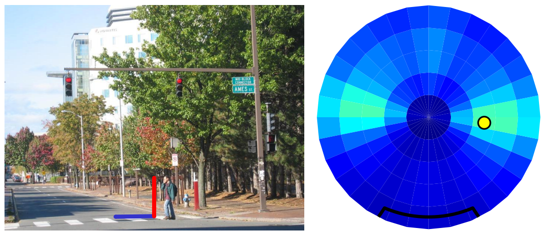
Fig. 6.16 shows examples of common scenarios that arise in practice. In Fig. 6.16a, we observe that strong cues in the scene—bright surfaces, strong cast shadows, visible illumination effects on pedestrians—result in an estimate that is peaked around a sun position which aligns well with the ground truth. When the scene is cluttered as in Fig. 6.16b, the cues become harder to detect, and the resulting estimate is less confident, as evidenced by the bluer colors in the probability map (all the colors in the figure are on the same scale, making it easy to compare them).

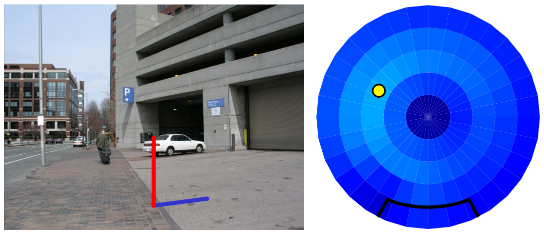
When there are strong cast shadows but the other cues are weaker, the shadow ambiguity remains present in the sun probability distribution as in Figs 6.16c and 6.16d. Finally, when the cues are altogether too difficult to detect or simply uninformative as in Figs 6.16e and 6.16f, the resulting estimate is of more or less constant probability. In this case, the maximum likelihood sun position, used to generate the quantitative evaluation plots of Sec. 6.5.2 and the virtual sun dials used for visualization, is meaningless. A more representative way of evaluating the results could employ a measure of the confidence of the distribution (e.g., variance).


(a) Strong cues, confident estimate


(b) Cluttered scene, less confident


(c) Shadow ambiguity, correct direction


(d) Shadow ambiguity, wrong direction


(e) Weak cues, low confidence

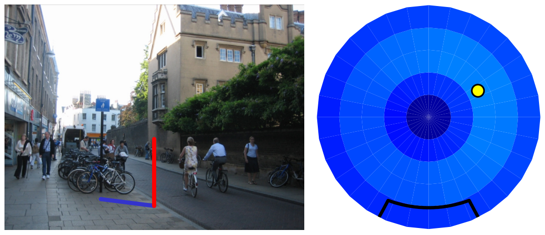

(f) Weak cues, very low confidence

Figure 6.16: Different scenarios result in different confidences in the illumination estimate. When the cues are strong and extracted properly, the resulting estimate is highly confident (a). A scene with more clutter typically results in lower confidence (b). The ambiguity created by shadows is sometimes visible when the other cues are weaker (c)–(d). When the cues are so weak (e.g., no bright vertical surfaces, no strong cast shadows, pedestrians in shadows, etc.), the resulting estimate is not confident, and the maximum likelihood sun position is meaningless (e)–(f). All probability maps are drawn on the same color scale, so confidences can be compared directly.

6.6.2 Complementary Sources of Information

It is sometimes the case that additional sources of information about the camera are available. For example, the EXIF header in image files commonly contain information like the focal length (used in this work), the date and time of capture of the image, and the GPS location. We now discuss what the availability of this information means in the context of our work.

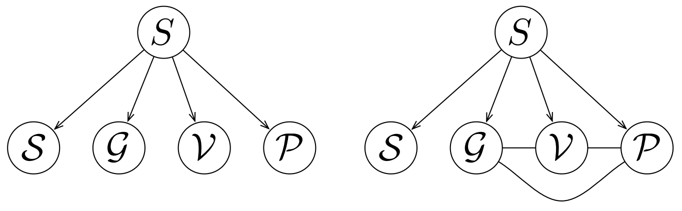
If the date and time of capture of the image as well as the GPS location are available, then it is possible to compute the zenith angle of the sun by using atmospherical formulas [Reda and Andreas, 2005]. Since the camera azimuth is unknown, this effectively restricts the sun to be in a band of constant zenith, so the probability of the sun to be anywhere else can readily be set to zero.

Recently, newer models of smartphone sport an additional sensor which readings are also available via EXIF: a digital compass. This compass records the absolute orientation of the camera, and has been shown to be useful in rudimentary augmented reality applications. In recording the camera azimuth angle, we now have everything we need to actually compute the sun position with respect to the camera. Of course, this does not indicate whether the sun is visible or not (Sec. 6.2), nor does it provide information about the weather conditions, but could be a tremendous tool to capture datasets of images with ground truth sun positions. The availability of large amounts of images with annotated information will surely play an important role in improving our understanding of the illumination in real outdoor images.

6.6.3 Higher-order Interactions Between Cues

In Sec. 8.5, we saw how we can combine together the predictions from multiple cues to obtain a final, more confident estimate. To combine the cues, we rely on the Naive Bayes assumption, which states that all cues *conditionally* independent given the sun direction. While this conditional independence assumption makes intuitive sense for many cues—for example, the appearance of the sky is independent from the shadow directions on the ground if we know the sun direction—it does not apply for all cues. Here we discuss a few dependencies that arise in our framework, and how we could leverage them to better capture interactions across cues.

Even if the sun direction is known, there is still a strong dependency between objects and their shadows. Knowing the position of vertical objects (e.g., pedestrians) tell us that cast shadows should be near their point of contact with the ground. Similarly, knowing the



(a) Naive Bayes model (b) Higher-order model

Figure 6.17: Capturing higher-level interactions across cues. The current approach uses a Naive Bayes model (a), which assumes that all cues are conditionally independent given the sun position. Capturing higher-order interactions across cues would require a more complex model (b), with less restrictive independence assumptions..

location of shadow boundaries on the ground constrain the possible locations of pedestrians, since they must cast a shadow (if they are in sunlight). Capturing this interaction between pedestrians and shadows would be very beneficial: since we know pedestrians are vertical objects, simply finding their shadows would be enough to get a good estimate of the sun position, and the ambiguity in direction from Sec. 6.3.2 could even be resolved.

There is also a dependency, albeit a potentially weaker one, between vertical surfaces and cast shadows on the ground. The top, horizontal edge of vertical surfaces (e.g., roof of buildings) also cast shadows on the ground. Reasoning about the interaction between buildings and shadows would allow us to discard their shadows, which typically point away from the sun (Sec. 6.3.2).

Another interesting cross-cue dependency arises between pedestrians and vertical surfaces. Since large surfaces may create large shadow regions, if the sun comes from behind a large wall, and a pedestrian is close to that wall, it is likely that this pedestrian is in shadows, therefore unpredictive of the sun position.

Capturing these higher-level interactions across cues, while beneficial, would also increase the complexity in the probabilistic model used to solve the problem. Fig. 6.17 shows a comparison between the graphical model that corresponds to our current approach (Fig. 6.17a) and a new one that would capture these dependencies (Fig. 6.17b). The caveat here is that the complexity of the model is exponential in the clique size, which is determined by the number of cues in the image (e.g., number of shadow lines, number of pedestrians, etc). Learning and inference in such a model will certainly be more challenging.

6.7 Conclusion

Outdoor illumination affects the appearances of scenes in complex ways. Untangling illumination from surface and material properties is a hard problem in general. Surprisingly, however, numerous consumer-grade photographs captured outdoors contain rich and informative cues about illumination, such as the sky, the shadows on the ground and the shading on vertical surfaces. Our approach extracts the ”collective wisdom” from these cues to estimate the sun visibility and, if deemed visible, its position relative to the camera. Even when the lighting information within an image is minimal, and the resulting estimates are weak, we believe it can still be a useful result for a number of applications. For example, just knowing that the sun is somewhere on your left might be enough for a point-and-shoot camera to automatically adjust its parameters, or for a car detector to be expecting cars with shadows on the right. Several additional pieces of information can also be exploited to help in illumination estimation. For instance, GPS coordinates, time of day and camera orientation are increasingly being tagged in images. Knowing these quantities can further constrain the position of the sun and increase confidences in the probability maps that we estimate. We will explore these avenues in the future.

Chapter 7

Synthesizing Illumination-Consistent Images

All photos are accurate. None of them is the truth.

Richard Avedon (1923–2004)

We now explore what we can do with the novel knowledge of illumination from single images acquired in Ch. 6. In particular, we present an entire system for realistic image creation, in which lighting-consistent images can be created as easily as three mouse clicks.

7.1 Background

One of the biggest achievements of photography has been in bringing the joy of visual expression to the general public. The ability to depict the world, once the exclusive domain of artists, is now available to anyone with a camera. The sheer number of photographs being taken every day, as evidenced by the explosive growth of websites like Flickr, attests to people's need to express themselves visually. However, the creation of *novel* visual content is still the monopoly of a small select group: painters, computer graphics professionals, and Photoshop artists. One of the "Grand Goals" of computer graphics is to make the creation and manipulation of novel photo-realistic imagery as simple and effortless as using a word-processor to create beautifully typeset text is today.

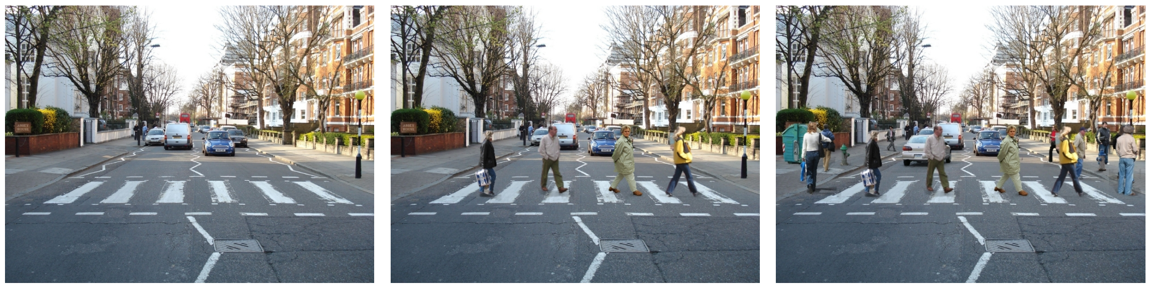


Figure 7.1: Starting with a present day photograph of the famous Abbey Road in London (left), a person using our system was easily able to make the scene much more lively. There are 4 extra objects in the middle image, and 17 extra in the right image. Can you spot them all?

However, there are a number of formidable challenges on the way toward this ambitious goal. For "traditional" computer graphics, the main problem is not the lack of good algorithms—recent advances in global illumination, material modeling, and lighting have allowed for synthesis of beautifully realistic and detailed imagery. Instead, the biggest obstacle is the sheer richness and complexity of our visual world. Every object to be rendered requires painstaking work by a skilled artist to specify detailed geometry and surface properties. And while there are a few databases of pre-made object models, typically the number of objects is quite small and the quality is far from photo-realistic. Various image-based approaches can also be used to acquire object models, but the data acquisition process is not straightforward and is typically not suitable for relatively large, outdoor, potentially moving objects (cars, pedestrians, etc). In this chapter, we are particularly interested in being able to insert new objects into existing images (e.g., creating a movie storyboard or adding street life to a newly built city-block). While it is possible to add synthetic objects into real scenes [Debevec, 1998], this requires estimating camera orientation and capturing environment maps—tasks too difficult for a casual user.

In this work, we advocate an alternative approach for creating novel visual content—by leveraging the enormous amount of photographs that has *already been captured*. A number of recent papers, such as Interactive Digital Photomontage [Agarwala *et al.*, 2004], have demonstrated the power of using stacks of registered photographs to create novel images combining information from the entire stack. The Photo Tourism work [Snavely *et al.*, 2006] showed how hundreds of photographs acquired from the Internet can be used as a novel way to explore famous architectural landmarks in space as well as in time. However, these are all examples of using images taken at the same physical location—a necessarily limited set. What about exploiting all the available visual data in the world, no matter where it was captured, to build a universal *photo clip art* library that can be used for manipulating visual content "on the fly"? While this is a very ambitious goal, the recent emergence of large, peer-labeled object datasets [Russell *et al.*, 2008; von Ahn *et al.*, 2006], as well as advances in geometric scene understanding [Hoiem *et al.*, 2005; Hoiem *et al.*, 2006] suggest that efforts in this direction are very timely.

7.2 Overview

In this chapter we propose a system for inserting new objects into existing photographs in an illumination-consistent way by querying a vast image-based object library, pre-computed using publicly available Internet object datasets[1]. The central goal is to shield the user from all of the arduous tasks typically involved in image compositing—searching for a good object to cut out, manual cropping and resizing, color and contrast adjustment, edge blending, etc. The user is only asked to do two simple things: 1) pick a 3D location in the scene to place a new object; 2) select an object to insert using a hierarchical menu.

In addition to the illumination estimation method introduced in Ch. 6, there are two additional critical elements that form the cornerstone of our approach to solving this challenging task:

Data-driven Object Placement. The difficulty in placing an image of an object (object sprite) into another image is that the camera orientation with respect to the object as well as the lighting conditions must match between the two images. One simple but key intuition is that while placing a *particular* object (e.g., my brown Volvo seen from the side) into a given image is a very difficult task, finding *some* instance of an object class (e.g., a car) that fits well is much easier. The idea, then, is to pose the problem of object insertion as a data-driven, context-sensitive object retrieval task. Instead of attempting to manipulate the object to change its orientation and/or color distribution, we simply retrieve an object of a specified class that has all the required properties (camera pose, lighting, resolution,

[1]Live demo available at `http://graphics.cs.cmu.edu/projects/photoclipart`.

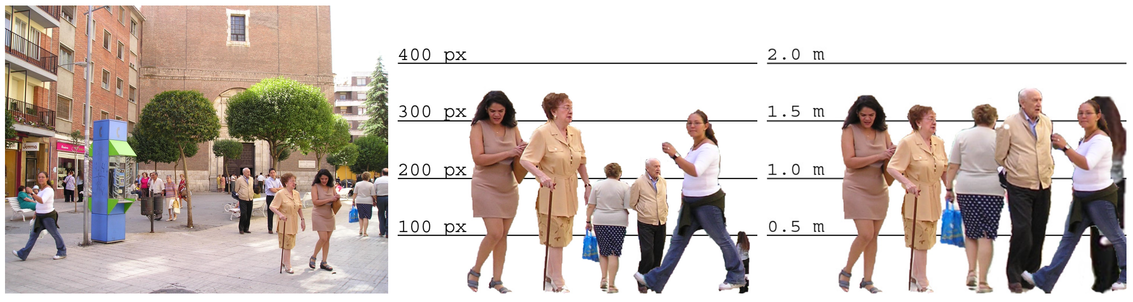


Figure 7.2: Automatic object height estimation. Objects taken from a typical image in LabelMe dataset (a) are first shown in their original pixel size (b), and after being resized according to their automatically estimated 3D heights (c). Image courtesy of D. Hoiem.

etc) from our large object library. This turns an impossibly hard problem into a solvable one, with the results steadily improving as more and more data becomes available.

3D Scene-based Representation: One of the major weaknesses of most current image compositing approaches is that they treat it as a 2D Photoshop-like problem. We believe that any image manipulation must be done in the 3D space of the scene, not in the 2D space of the image. This does not mean that a complete 3D depth representation of a scene is required—only some rough qualitative information about surfaces and their orientations with respect to the camera, plus basic depth ordering. First, this allows the user some intuitive 3D control of the object placement process—placing the car further down the road will automatically make it smaller. More importantly, the 3D information is needed to build the photo clip art library. Each object in the library must be annotated with relative camera pose, whether it's occluded by other objects, whether it's on the ground plane, etc. In the current version, due to the limitations of the object dataset, all of our objects are assumed to reside on the ground plane.

7.2.1 Key Challenges

The main challenge of our proposed system is to make the resulting images look as real as possible. Here we highlight the issues that are critical for the success of this endeavor and summarize our approach for addressing them.

Rich Object Library The data-driven nature of our approach requires a library with very large number of labeled objects. After examining a number of available on-line object datasets, we have chosen LabelMe [Russell *et al.*, 2008] as providing a good mix of object classes, large numbers of objects in each class, and reasonably good initial object segmentations. After post-processing and filtering the data, we end up with an object library currently containing 18 categories of objects with over 13,000 object instances.

Object Segmentation Getting clean object boundaries is critical for seamless object insertion. While the object dataset we use provides rough polygonal object outlines, this is rarely good enough for compositing. To address this problem, we use the automatic segmentation and blending approach of Rother [2007]. It extends the popular graph-cut/Poisson blending framework by introducing a new shape-sensitive prior, adding a crucial local context-matching term for blending, detecting shadows, and modifying the blending process to prevent severe discoloration.

Estimating true Object Size and Orientation A 2D object boundary tells us nothing about the size of the object in the world or its orientation to the viewer. For example, given an outline of a person, we cannot tell if it's a tall man standing far away, or a small boy standing nearby; or whether we are viewing him from street level or from a third-story balcony. But without this information one cannot hope for realistic object insertion. For this, we leverage recent work in scene understanding [Hoiem *et al.*, 2006] and use an automatic algorithm that considers all the objects labeled in the dataset to estimate the camera height and pose with respect to the ground plane in each of the images. For objects resting on the ground (our focus in this chapter), these two parameters are enough to compute object size and orientation.

Estimating Lighting Conditions One of the central contributions of this thesis, consistency in lighting is another very important cue for realistic compositing. A picture of a car taken at sunset will most likely not look right when placed into a midday scene. Unfortunately, we do not have enough information to compute a full environment map for each object. However, such accuracy might not be necessary. It is well-known that painters can often get away with substantially inconsistent lighting [Cavanagh, 2005], and in graphics applications simple approximations have been shown to perform well [Khan *et al.*, 2006]. In this chapter, we use our illumination estimate obtained in Ch. 6, as well as a simpler illumination model based on automatically computing color distributions of major surfaces (e.g., ground, vertical planes, sky) in the scene that is sufficient in some cases.

Intuitive User Interface While not the central focus of this chapter, a good UI is critical for achieving the goal of making our system simple and intuitive. We have designed and implemented a full-functioning GUI that allows for easy browsing of the object library and lets users insert objects with a single click.

The remainder of the chapter is devoted to describing each of these components in more detail. We first discuss our approach for creating the photo clip art object library (Sec. 7.3). We then present our system for inserting objects into existing user photographs (Sec. 7.4). Finally, we demonstrate some results (Sec. 7.5).

7.3 Creating the Photo Clip Art Library

The first issue in creating a photographic clip art library is finding a large dataset of labeled images of objects. Fortunately, the growth of the Internet and the need for large object recognition testbeds for computer vision research have produced a large number of freely available object datasets that we can choose from.

Most datasets (such as Caltech-101 [Fei-Fei *et al.*, 2004], MSRC [Shotton *et al.*, 2006], and PASCAL [Everingham *et al.*, 2006]) have been collected and annotated by small groups of researchers and are therefore limited in size and/or annotation detail (most lack segmentations). Recently, there appeared a new breed of peer-labeled datasets. The most famous example is Flickr, where people can submit photographs and annotate them with semantic image and region tags. A creative approach is presented in an on-line game called Peekaboom [von Ahn *et al.*, 2006], where humans communicate with each other by segmenting objects in images. Finally, LabelMe [Russell *et al.*, 2008] provides as on-line tool for vision researchers to submit, share, and label images.

After experimenting with all these datasets, we decided that LabelMe is the best-suited for our purposes. Its advantages include a large number of images (over 30,000 at present), containing many different objects (over 2,000 object classes) as well as reasonably good segmentations. Also important for us is that most photographs are not close-ups of one individual object but depict scenes containing multiple objects, which enables us to estimate camera orientation. Finally, since this dataset is very popular in the vision community, it is constantly growing in size. At the same time, vision researchers are developing automatic methods for supervised recognition (e.g., [Shotton *et al.*, 2006]) and object discovery (e.g., [Russell *et al.*, 2006]) that should, some day, take over the labeling chore. Success has already been reported on specialized tasks (e.g., an automatically collected database of celebrity faces [Berg *et al.*, 2004]), but further research is needed for more general domains.

The main downside of LabelMe is substantial variation between individual labeling styles as well as semantic tags. Some people include cast shadows, others don't; some people include occluded parts of an object, others don't. Since a semantic tag can be any text string, the same object could be labeled as: "person", "human", "pedestrian", "man smiling", etc. Therefore, one of our tasks in processing the dataset was to deal with such inconsistencies. Additionally, for every object in our library, we need to estimate its size and orientation in the real world as well as its illumination. We will now describe each of these steps in detail.

7.3.1 Estimating Object Size and Orientation

To match and scale objects in our database to the background scene provided by the user, we need to know the true size of the objects in the world and the pose of the camera. As shown by [Criminisi *et al.*, 2000], if we know the vanishing line of the ground plane, we can determine the relative heights of objects that rest on the ground plane. To annotate our database using this method, however, would require painstaking (and often inaccurate) annotation of vanishing lines in thousands of images. Instead, we use the method of Hoiem [2007] (a co-author in this work) to gradually *infer* camera pose and object heights across our database when provided with only the height distribution for *one* object class. We summarize the method here for completeness, but this does not represent one of the contribution of this thesis.

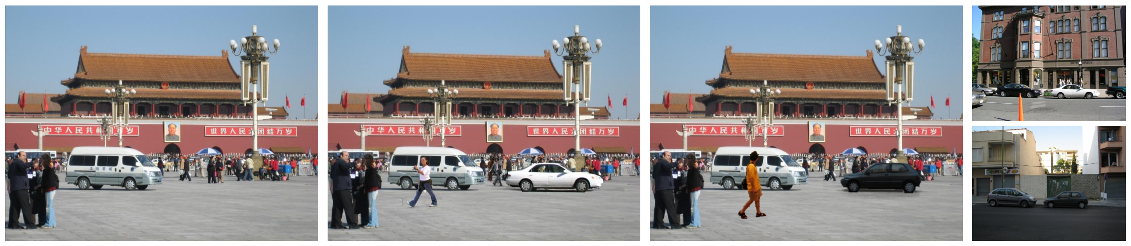

(a) Input image (b) Well matched objects (c) Poorly matched objects (d) Sources

Figure 7.3: Lighting plays crucial role in object insertion. Given an input image (a), objects that are similarly illuminated fit seamlessly into the scene (b), while these with substantially different illumination appear out of place (c). By considering the source image from which an object was taken (d) in relation to the input image (a), our algorithm can predict which objects will likely make a realistic composite.

Let y_i be the 3D height of an object with a 2D height of h_i and a vertical position of v_i (measured from the bottom of the image). The camera pose is denoted by y_c and v_0, which correspond roughly to the camera height and the horizon position. Following [Hoiem *et al.*, 2006], their inference is based on the relationship $y_i = \frac{h_i y_c}{v_0 - v_i}$, assuming that objects stand on the ground and that ground is not tilted from side to side and roughly orthogonal to the image plane. Thus, given the position and height of two objects of known type in the image, we can estimate the camera pose without heavily relying on a prior distribution and use it to compute the 3D height of other objects in the scene. If we know the prior distribution over camera pose, we can provide a good estimate of it, even if only one instance of a known object is present [Hoiem *et al.*, 2006].

The procedure [Hoiem, 2007] is as follows. We first compute the most likely camera pose for images that contain at least two known objects (instances of an object class with a known height distribution). From these, we estimate a prior distribution over camera pose. Using this distribution, we infer the likely camera pose for images that contain only one known object. Finally, we compute heights of objects in the images for which camera pose is known and estimate the height distributions of new classes of objects. We then iterate this process. As the height distributions of new object classes are discovered, they can be used to estimate the camera pose in additional images, which, in turn, allow the discovery of new objects. Fig. 7.2 shows an example of automatically estimate 3D heights for various objects from the same image.

7.3.2 Estimating Lighting Conditions

The appearance of an object is determined by the combination of its reflectance properties and the illumination from the surrounding environment. Therefore, an object taken from one image and inserted into another will look "wrong" if the two scenes have large differences in illumination (see Fig. 7.3). The standard solution of measuring the incident illumination by capturing the object environment map (typically using a mirrored sphere at the object location [Debevec, 1998]), is not possible here since we only have access to a single photograph of the object. We have already described how to estimate the illumination conditions from a single image in Ch. 6, but we provide here another, coarser representation dubbed "illumination context" which was introduced in an earlier version of this work [Lalonde *et al.*, 2007], and used to generate some of the results in Sec. 7.5.

We propose to use scene understanding techniques to compute a very coarsely sampled environment map given a single image as input. The basic idea is to come up with a rough 3D structure of the depicted scene and use it to collect lighting information from the three major zenith angle directions: from above, from below, and from the sides (azimuth direction is not used since there is not enough data in a single image). Our implementation uses the geometric context of [Hoiem *et al.*, 2007] to automatically estimate 3 major surface types: ground plane, vertical planes, and the sky. The distribution of illuminations within each surface type is computed as a joint 3D histogram of pixel colors in the CIE L*a*b* space, for a total of 3 histograms to form our scene *illumination context*. It is important to note that while [Khan *et al.*, 2006] use high dynamic range images, we only have access to regular low dynamic range photographs so cannot capture true luminance. But since we don't use the illumination context to relight a new object, only to compare lighting conditions between images, this has not been a problem.

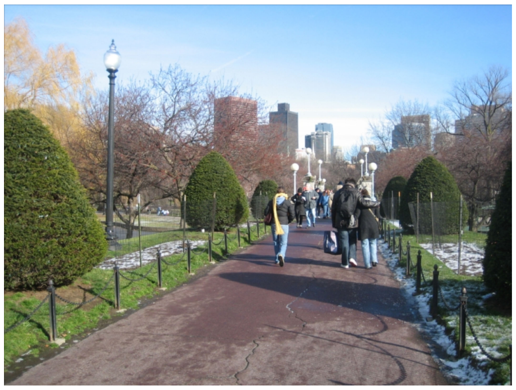
The illumination context is too global and cannot encode location-specific lighting within the scene, i.e., whether a given object is in shadow. Therefore, for each object we also compute a *local appearance context*—a simple color template of its outline (30% of object diameter outward from the boundary, and 5% inward). The local context is used as an additional cue for object placement.

Is it enough to encode only the global "feel" of the scene to generate realistic image composites? In many cases, it seems like this may be sufficient (see Figs 7.9, 7.10 and 7.11). However, when cast shadows are visible such as in Fig. 7.4a, then matching the sun direction actually becomes very important. Although there appears to be some level of disagreement in the visual perception community regarding the physiological process that is responsible for shadow perception [Rensink and Cavanagh, 2004; Elder *et al.*, 2004], it has been shown that illumination consistency is very important to achieve realism [Johnson and Farid, 2007]. For example, Fig. 7.4b shows the compositing result obtained by transferring a person coming from the nearest neighbor image using the the coarse "illumination context" representation described above, which does not match the sun direction. In this case, the best match happens to have the sun in the exactly opposite direction! The resulting composite is therefore noticeably wrong. Instead, matching the sun direction estimated using the technique presented in Ch. 6 results in a composite where the shadows are consistent (Fig. 7.4c), thereby making the result much more realistic.

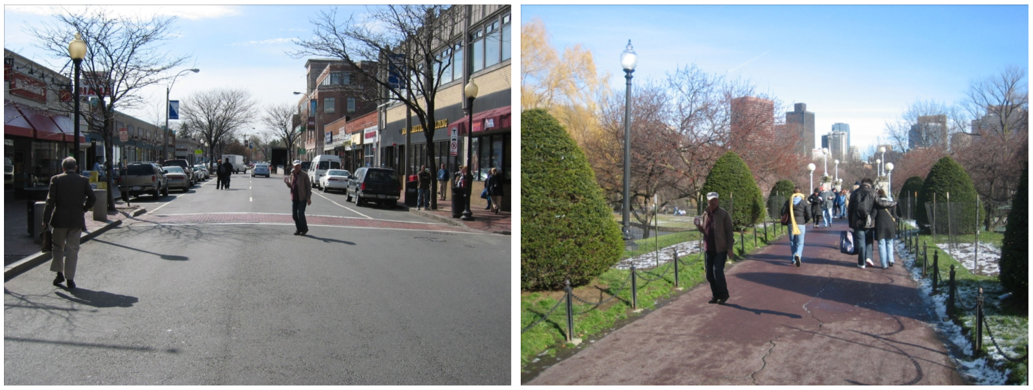
7.3.3 Filtering and Grouping Objects

While the LabelMe dataset has a rich collection of objects and object classes, many are simply not useful for our purposes. A lot of the labeled objects are occluded or are actually object parts (e.g., a wheel, a car window, a head), while others are amorphous (e.g., sky, forest, pavement). The types of objects best suited for a clip art library are semantically whole, spatially compact, with well-defined contact points (i.e., sitting on the ground plane). We attempt to filter out the unsuitable objects from the dataset in a semi-automatic way.

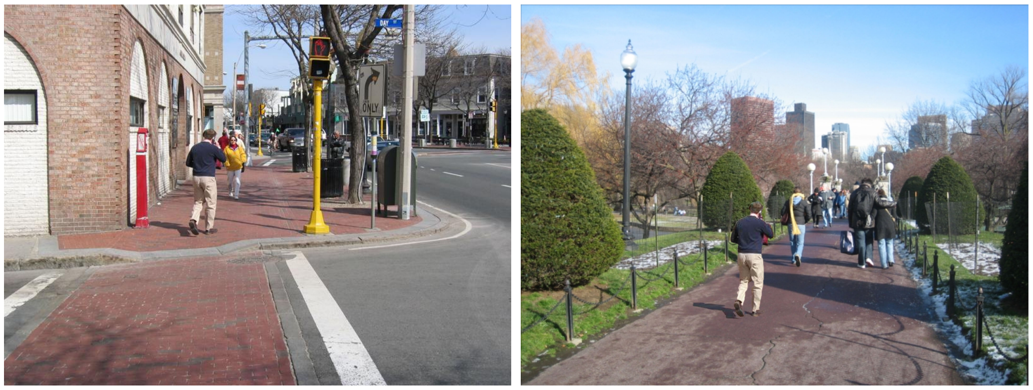
As a first step, we try to remove all non-whole objects by searching the tag strings for words "part", "occlude", "region", and "crop". We also remove any objects that have a boundary right at the image border (likely occluded). Second, we estimate the 3D heights of all objects using the iterative procedure outlined in Sec. 7.3.1. As a by-product, this gives us a list of object classes that have been found useful for camera pose estimation (i.e., having relatively tight height distributions): "bench", "bicycle", "bus", "car" (various orientations), "chair", "cone", "fence", "fire hydrant", "man", "motorcycle", "parking meter", "person", "sofa", "table", "trash can", "truck", "van", "wheel", "woman". Of these, we discard small or unsuitable classes (e.g., "wheel"), and merge others (e.g., "man", "woman",



(a) Input image



(b) Nearest neighbor (left) and composited result (right), illumination context [Lalonde *et al.*, 2007]



(c) Nearest neighbor (left) and composited result (right), matching sun angles (this thesis)

Figure 7.4: Comparison between image synthesis results obtained with the "illumination context" representation from an earlier version of our work [Lalonde *et al.*, 2007] (b) and the representation introduced in this thesis (c). The objects to transfer (pedestrian) are manually selected in the top nearest neighbor from our Photo Clip Art database. Our technique presented in Ch. 6 is able to match shadow directions, crucial for realistic object insertion.

and "person") into consistent groups. We also manually add a few fun classes that were either too small or not very consistent: "plant", "flowers", "tree", and "manhole". Note that most of the recovered classes are of outdoor objects. This mainly reflects the bias in the dataset toward outdoor scenes, but also due to ground-based objects mostly appearing outdoors.

In most cases the variance between objects within the same object class is still too high to make it a useful way to browse our clip art collection. While a few of the object classes have hand-annotated subclass labels (e.g., cars are labeled with their orientation), most do not. In these cases, clustering is used to automatically find visually similar subclasses for each object class. First, all objects are rescaled to their world 3D height (as calculated earlier), so that, for instance, a tree and a bush don't get clustered together. Next, k-means clustering using L2 norm is performed on the binary object outlines. This is done so that object will cluster solely based on shape, and not on their (lighting dependent) appearance. Results show successful subclasses being formed for most of the objects (e.g., see Fig. 7.5).

7.4 Object Insertion

Now that we have built our photo clip art object library, we can describe how to insert objects into user photographs. Given a new photograph, the first task is to establish its camera orientation with respect to the ground plane. To do this, the user is asked to pick the location of the horizon with a simple line interface. Other relevant camera parameters (camera height, focal length, pixel size) are set to their most likely values, or extracted from the EXIF tag, if available. Alternatively, the user can specify the height of one or more known objects in the scene, from which the camera parameters can be estimated as in Sec. 7.3.1. It is also possible to let the user resize a known "reference object", such as a person, placed at different locations in the image, to compute the camera pose and height, although we haven't found this necessary. In addition, lighting conditions must be estimated for the new scene, which is done in the same way as described in Sec. 7.3.2. With the critical parameters of the new photograph successfully estimated, we are now ready for object insertion.

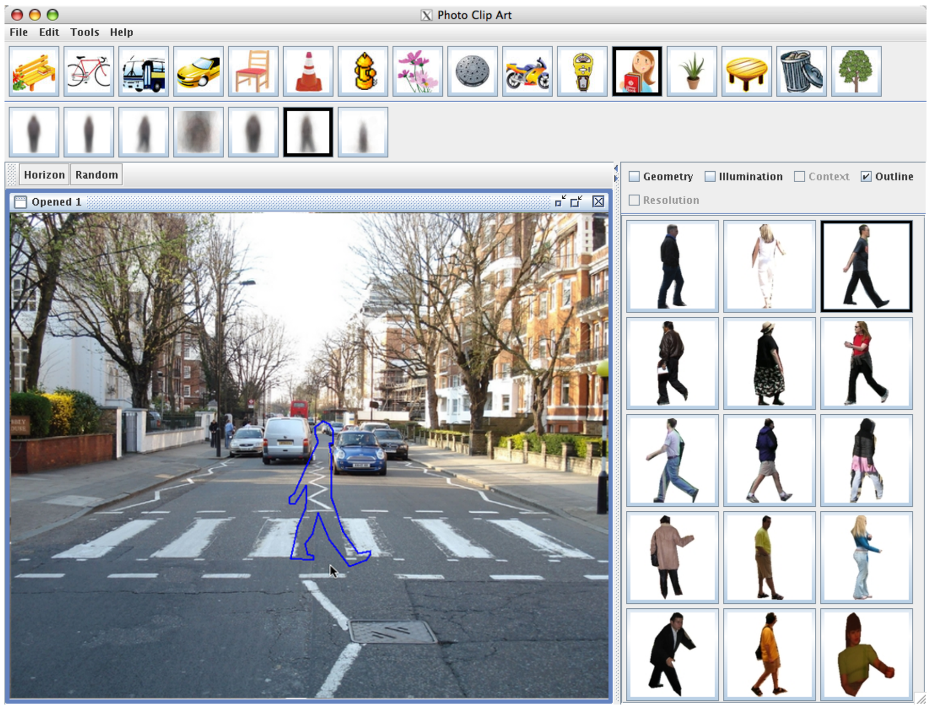


Figure 7.5: User Interface for our system. The two-level menu on the top panel allows users to navigate between different classes and subclasses of the photo clip art library. The side panel on the right shows the top matched objects that can be inserted into the scene.

7.4.1 User Interface

First, we will briefly describe the design of our User Interface. Its job is to let the user 1) easily navigate the entire photo clip art library, and 2) paste an object of choice into the photograph with a single click.

The screenshot of our UI is depicted on Fig. 7.5. The user photograph is loaded in the central panel. The top row of icons shows all the object classes available in the library. The second row of icons shows all object subclasses for a chosen class. The subclasses are generated either from the object annotations, if available, or by automatic clustering (as described previously). The subclass icons are average images of all the objects in the subclass, a technique inspired by [Torralba and Oliva, 2003]. The right-hand-side panel displays the object instances. Initially it contains all the objects in the library, but as the user picks a class of objects, only these instances remain in the panel. The same happens for a subclass. Moreover, the objects in the panel are sorted by how well they match this particular scene.

To insert an object into the scene, the user first chooses an object class (and optionally a subclass), and then picks one of the top matches from the object panel. Now, as the user moves the mouse around the photograph, the outline of the picked object is shown, becoming bigger and smaller according to the correct perspective. A single click pastes the object at the current mouse position, and full segmentation and blending is performed on the fly. Alternatively, the user can first pick a location to paste an object, and then choose among the objects that best fit that particular location. Note that all inserted objects (as well as any labeled ones in the image) preserve their 3D depth information and are rendered back-to-front to accurately account for known occlusions.

7.4.2 Matching Criteria

Given a scene, we need to order all the objects in the library by how natural they would look if inserted into that scene. This way, a user picking an object from near the top of the list would be pretty confident that the compositing operation would produce good results. The ordering is accomplished as a weighted linear combination of several matching criteria, dealing with different aspects of object appearance.

Camera Orientation: The camera orientation with respect to the object's supporting surface has been pre-computed for every object in the library. We also have the estimate for the ground plane orientation for the user's photograph. Geometrically, an object would fit well into the scene if these camera orientation angles are close, and poorly otherwise. If the object placement location is not specified explicitly by the user, a simple difference between the vanishing line heights of the ground planes of the source and destination scenes is used instead.

Global Lighting Conditions: Each object in the library has an associated scene illumination context that was computed from the object's original source image. An object is most likely to appear photometrically consistent with the new scene if its original scene illumination context match that of the new scene. The distance between the illumination contexts is computed as a weighted linear combination of χ^2-distances between the L*a*b* histograms for sky, vertical, and ground. Experimenting with the weights, we have found (unsurprisingly) that the sky portion of the context is the most important, followed by the ground portion. The vertical portion seems to have little effect. When the sun position $(\theta_s, \Delta\phi_s)$ is available, matches can be found by sorting them according to the angular difference in sun angles. Matching the sun visibility V can also be done by keeping only images with the same V as the input.

Local Context: If a user has also specified the location where he wants the object to be placed, local appearance context around (and under) the placement area can also be used as a matching criterion. The context is matched to the corresponding pixels at the placement area using SSD measure. This cue is used primarily as an aid to blending difficult objects, as described in Sec. 7.4.3, and for matching locally-varying lighting effects.

Resolution: Again, if the placement location in the image is given, it is easy to compute the desired pixel resolution for the object so as to minimize resizing.

Segmentation Quality: The initial segmentation quality of objects in our library is highly variable. Various approaches can be used to try separating the good ones from the bad (e.g., alignment with brightness edges, curvature, internal coherence, etc). In this implementation, we use a very simple but effective technique, ordering objects by the number of vertices (i.e., labeling clicks) in their boundary representation.

7.4.3 Object Segmentation and Blending

Once the user picks an object to insert into a given scene, the most important task to insure a quality composite is object matting. Unfortunately, automatic alpha-matting without a user in the loop is not guaranteed to work reliably for all scenarios, despite a lot of recent advances in this field e.g., [Levin *et al.*, 2006]. On the other hand, blending techniques [Pérez *et al.*, 2003; Jia *et al.*, 2006] have shown that impressive results can be achieved without an accurate extraction of the object, assuming that the material surrounding the object and the new background are very similar (e.g., a sheep on a lawn can be pasted onto another lawn, but not on asphalt).

The idea of combining the advantages of blending and alpha-matting have been introduced to hide matting artifacts as much as possible [Wang and Cohen, 2006; Jia *et al.*, 2006]. The key advantage of our system is that the local context criteria (introduced above) has

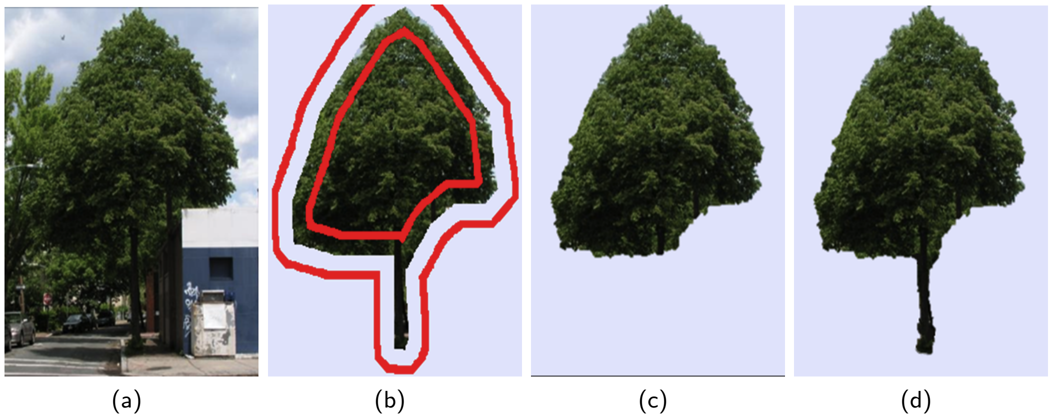


Figure 7.6: Object extraction. Given an image (a), the task is to improve the crude polygon-shaped LabelMe segmentation (b). Standard GrabCut (restricted to a red band in (b)) suffers from a "shrinking" bias (c). This can be overcome by introducing flux, a shape-sensitive prior, into the GrabCut framework (d). Figure courtesy of C. Rother.

already retrieved appropriate objects which match to some degree the given background, so blending and segmentation have a good chance of supporting each other.

Our compositing method is contributed by our collaborator C. Rother and presented in further details in [Rother, 2007], and as such does represent a contribution of this thesis. We however provide some details here for completeness. It runs, as in [Jia *et al.*, 2006], in three steps: first object extraction, then blending mask computation, and finally Poisson image blending. The novel ideas over existing approaches are: a crucial shape prior for segmentation, a new context-matching term for blending mask extraction, and a way to prevent severe discoloration during blending.

7.4.3.1 Object Segmentation With a Shape Prior

Object segmentations provided in the LabelMe dataset have only crude, polygonal boundaries. The task is to find a more faithful outline of the object along with a good alpha mask. For this we extend the popular graph-cut based segmentation approaches [Rother *et al.*, 2004; Li *et al.*, 2004] with a prior for the given shape. Fig. 7.6 gives an example, where the cut is restricted to lie in a small band (red in Fig. 7.6b) around the prior polygon shape. Applying GrabCut without shape prior results in Fig. 7.6c. The tree outline looks convincing, however the tree trunk is lost. This happens because a short expensive bound-

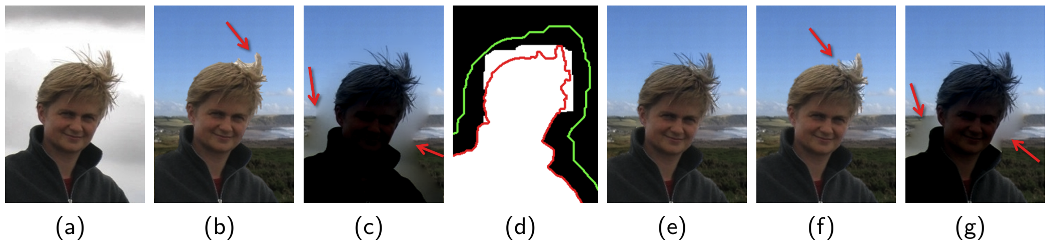


Figure 7.7: Example of image compositing. Source input image (a) and compositing results with different approaches (b), (c), (e)–(g), where red arrows point out problems. Figure courtesy of C. Rother.

ary has a lower cost than a very long cheap one, sometimes called the ”shrinking” bias. To overcome this problem different forms of shape priors, such as shape-based distance transform [Boykov *et al.*, 2006], have been suggested in the past. We use the flux shape prior as introduced in the theoretical work of Kolmogorov and Boykov [2005]. The main advantage over other methods is that it overcomes the shrinking bias while not over-smoothing the whole segmentation. The key idea is that the gradient of the computed contour is similar to the gradient of the given shape. Details on how this is implemented in our context are provided in [Rother, 2007]. The result in Fig. 7.6d shows that the tree trunk is recovered nicely.

7.4.3.2 Context-sensitive Blending

Consider the image compositing task in Fig. 7.7. Given the input image 7.7a, GrabCut [Rother *et al.*, 2004] (based on a rough polygonal outline around the object) produced an alpha matte which is used to paste the object onto a new background in Fig. 7.7b. Here conservative border matting [Rother *et al.*, 2004] was not able to extract the hair perfectly. On the other hand, standard Poisson image blending [Pérez *et al.*, 2003] (with the boundary constraint applied at the green line in 7.7d—a dilation of the GrabCut result) blends the hair correctly 7.7c. However, it produces two artifacts: a severe discoloration due to large differences at the boundary constraint, and a blurry halo around the object where foreground and background textures do not match. The key idea is to find a binary blending mask which either blends in the background color or leaves the segmentation unaltered. Fig. 7.7d shows the blending mask computed by our technique. In places where the mask

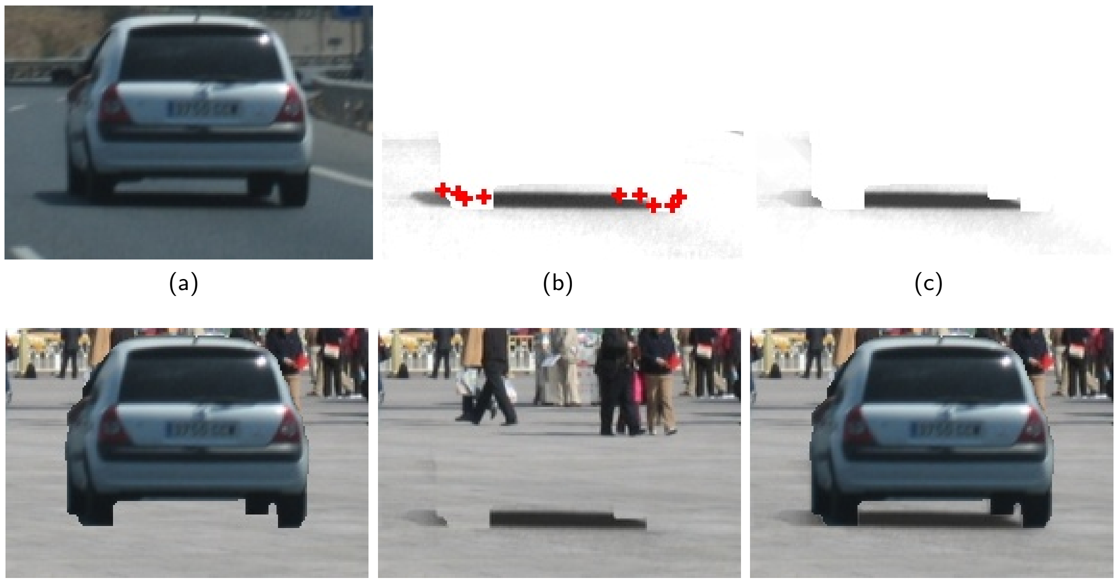


Figure 7.8: Shadow transfer. On the top row, we show the source image (a) for a car with initial (b) and final (c) shadow estimates. Our initial estimate, simply based on intensity relative to surrounding area, is refined by enforcing that the shadow is darker closer to the contact points (red '+'). On the bottom row, we show the background patch with the object (d), with the matted shadow (e), and the combination (f), before blending. Image courtesy of D. Hoiem.

coincides with the object (red line) no blending is performed and when the mask is outside the object (at the hair) it blends in the background. This gives a visually pleasing result (Fig. 7.7e). To achieve this, we extend the approach of Jia *et al.* [2006] by adding an additional regional term that measures the similarity of foreground and background statistics (see [Rother, 2007] for details). This allows the white mask Fig. 7.7d to grow outside the red mask at places with high similarity. In comparison, results using digital photomontage [Agarwala *et al.*, 2004] and drag-and-drop pasting [Jia *et al.*, 2006] (both using our implementation) show more artifacts as seen on Figs 7.7f and 7.7g.

7.4.4 Transferring Shadows

While psychologists have shown that humans are not very sensitive to incorrect cast shadow direction [Cavanagh, 2005], the mere presence of a shadow is a critical cue for object/surface contact [Kersten *et al.*, 1996]. Without it, objects appear to be floating in space. Shadow extraction is a very challenging task and most existing work either requires controlled lighting conditions, e.g., [Chuang *et al.*, 2003], or makes restrictive assumptions about camera, lighting, and shadow properties (e.g., [Finlayson *et al.*, 2006]).

Because we do not know the 3D structure of our objects, synthetically generated shadows rarely look convincing. Instead, we take an image-based approach, transferring plausible shadows from the source image to the background image (illustrated in Fig. 7.8). Unfortunately, we cannot systematically use our single image shadow boundary detection method introduced in Ch. 5 because hard cast shadow boundaries do not always exist below objects. Instead, we employ the simple heuristic described in [Lalonde *et al.*, 2007] which detects dark regions below the objects.

7.5 Appearance Transfer Across Single Images

Figs 7.9, 7.10 and 7.11 show more object insertion results that a user was able to create with our system. One needs to look very carefully to notice all the objects that have been added—there are several that are quite difficult to spot. As can be seen, the system handles input images with wide variation in scene geometry as well as lighting conditions. In addition to photographs, our approach can successfully be applied to paintings (Fig. 7.9, bottom center) and CG renderings (Fig. 7.11). The important thing to note is that all these examples were produced in a few minutes by users who are not at all artistically skilled.

In this chapter, we have argued for a philosophically different way of thinking about image compositing. In many situations, it is not important which particular instance of an object class is pasted into the image (e.g., when generating realistic architectural renderings as shown on Fig. 7.11). In such cases, what is desirable is a kind of photo-realistic clip art library. Of course, regular clip art is a simple 2D iconic drawing, whereas the real 3D world-based clip art is necessarily more complex. However, we have shown that with the right approach, and a lot of data, this complexity could be successfully hidden from the end user, making it seem as simple and intuitive as regular clip art.

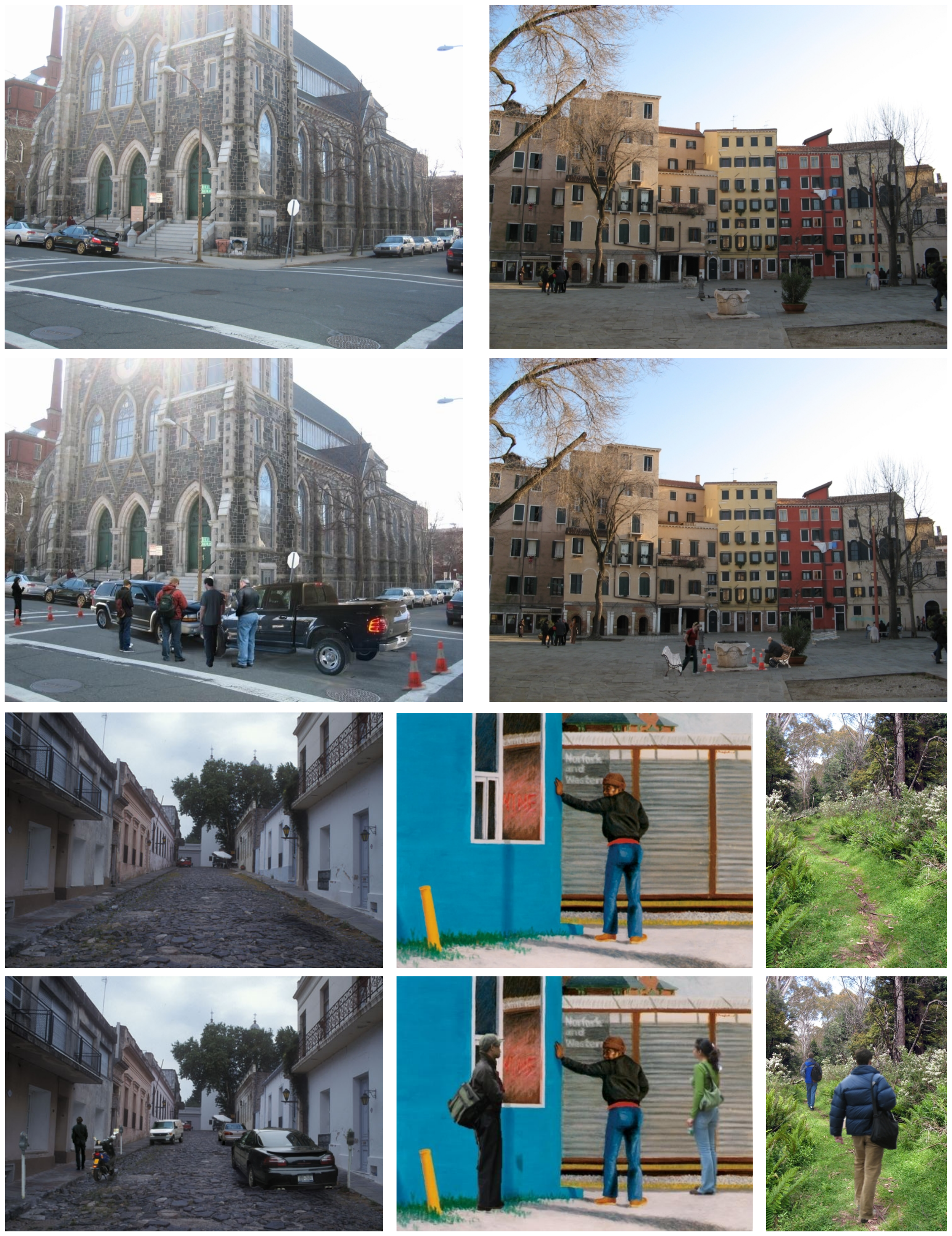


Figure 7.9: Some example images that were created by a user with our system. Can you find all the inserted objects?

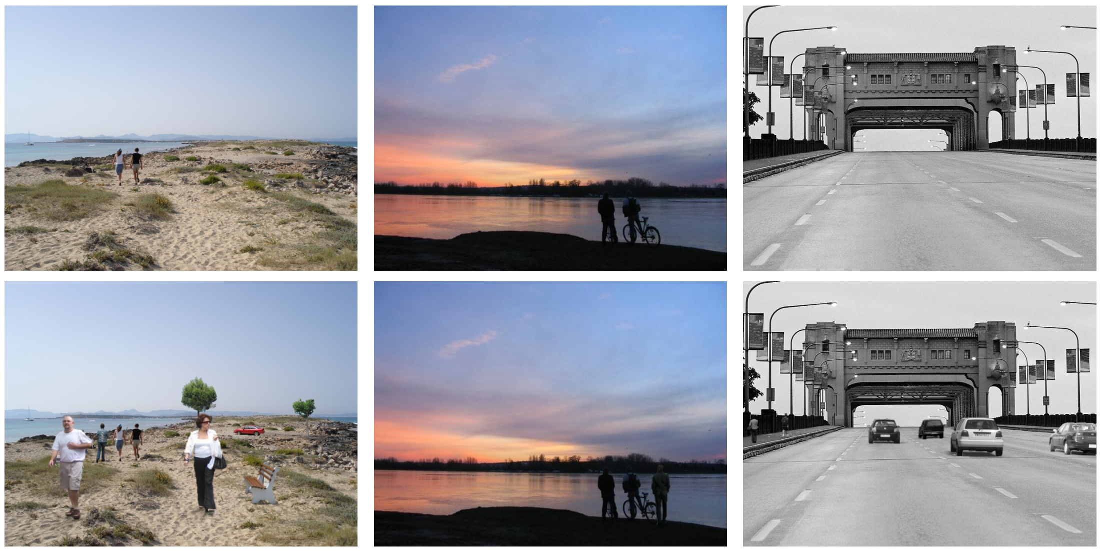


Figure 7.10: Our system can handle a variety of input lighting conditions, from midday (left) to sunset (middle), and can even work on black&white photographs (right). In the latter case the objects have been converted to grayscale after insertion.

7.6 Illuminant Transfer in Single Images

As an additional application, we now demonstrate how to use our technique to insert a 3-D object into a single photograph with realistic lighting. This requires generating a plausible environment map [Debevec, 1998] from the image. When the sky is clear, we can estimate the full illumination model from Sec. 2.4. We can then use the same approach as the one introduced in Sec. 4.5 to synthesize entire environment maps from a single image, and use

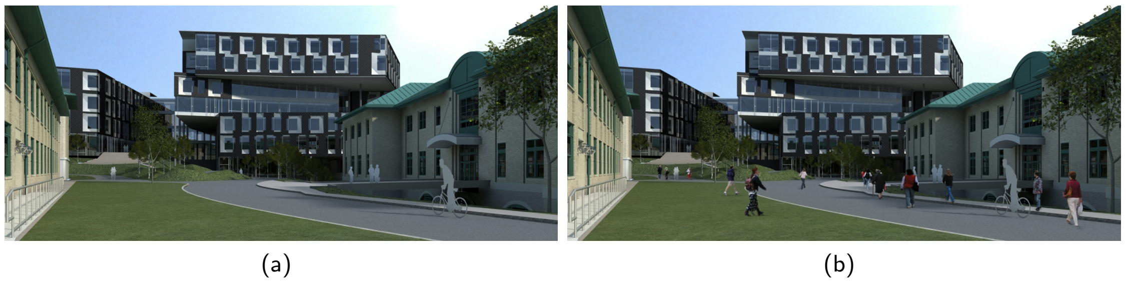
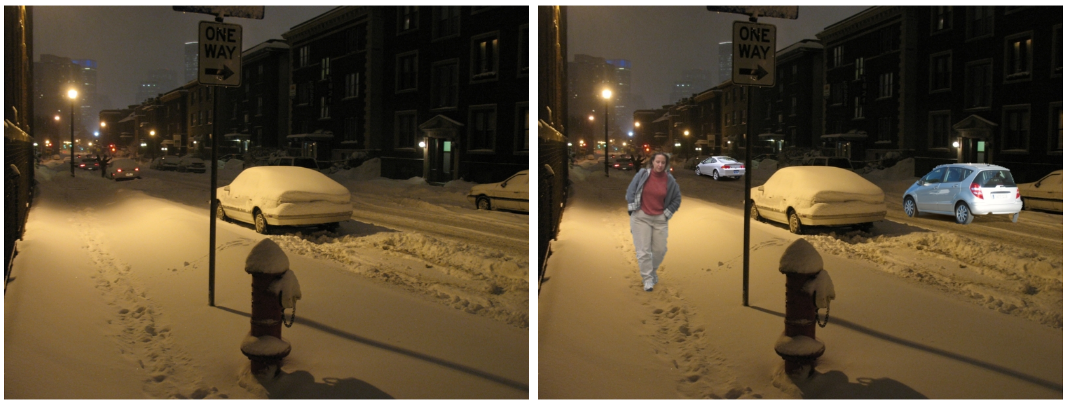
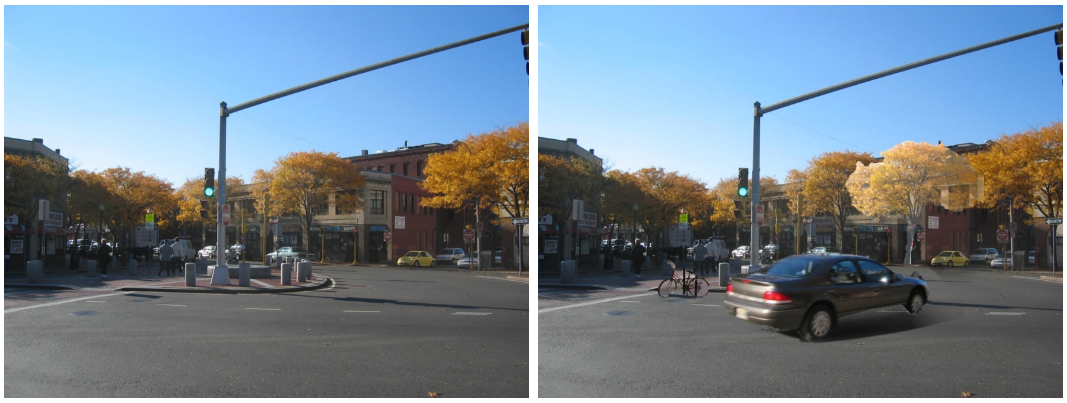


Figure 7.11: Application to architectural renderings. Here, a rendered view of the future Gates Center at CMU (a) is made to look more natural using our system (b). Note that the photo clip art people look much better than the white cutout people of the original rendering. Image used by permission from Mack Scogin Merrill Elam Architects.



(a)



(b)

Figure 7.12: Typical failure modes of our system. For input scenes with unusual illumination, the object library may not contain illumination conditions that are similar enough, making even the best-matched objects look wrong (a). Additionally, our blending algorithm may fail to blend complex and porous objects (e.g., tree and bicycle, (b)). Shadow transfer may also yield undesirable results (e.g., car, (b)).

it to relight a virtual 3-D object as shown in Fig. 7.13. Notice how the shadows on the ground, and shading and reflections on the objects are consistent with the image.

7.7 Conclusion

The chapter presents a complete system, from database preprocessing to cutting and blending to the graphical user interface. Along the way, we had to solve many difficult issues, resulting in a number of novel algorithms and techniques. Since this is a complex system, failures can occur in several stages of processing. Fig. 7.12 illustrates the three most com-

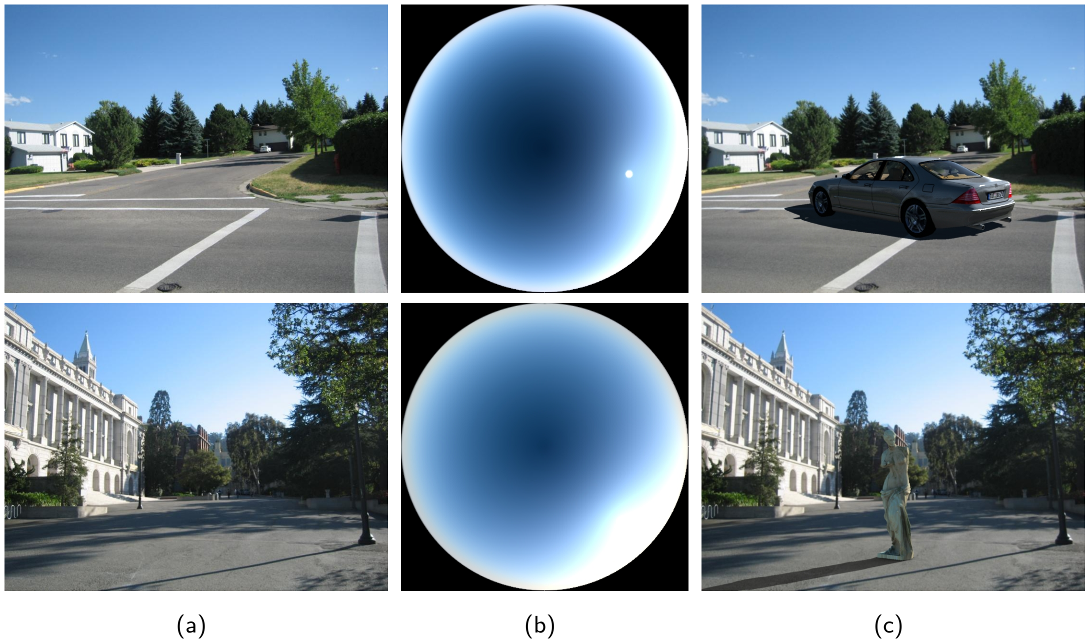


Figure 7.13: 3-D object relighting examples. From a single image (a), we render the most likely sky appearance (b) using the sun position computed with our method, and then fitting the sky parameters using the approach introduced in Sec. 3.6. We can realistically insert a 3-D object into the image (c).

mon causes of encountered failures. First, input scenes with unusual illumination are not similar to any of the images in our library. Therefore, even the best-matched objects will look unrealistic. Second, blending and automatic segmentation errors may occur when objects are porous or of complex shape (e.g., trees and bicycles). These objects contain holes through which their original background is visible. Finally, the shadow transfer algorithm may fail when the object/ground contact points are not estimated correctly, or when something else in the scene is casting a shadow onto the object. Since most of these issues are data-related, it is reasonable to believe that as the underlying image datasets improve, so will our system. This, once again, underscores one of the main themes of this thesis that the use of large-scale image databases is a promising way to tackle some of the really difficult problems in computer graphics.

Chapter 8

Determining the Realism of Image Composites

Reality is merely an illusion, albeit a very persistent one.

Albert Einstein (1879–1955)

In the previous chapter, we introduced an end-to-end system for generating image composites by transferring objects across images in an lighting-consistent manner. But how can we tell whether the resulting composite will look realistic or not? In this chapter, we present a study of the influence of *color* on the realism of images.

8.1 Introduction

Consider the images shown on Fig. 8.1. Only two of them are real. The rest are composite images—created by taking an object from one image and pasting it into a different one. The four synthetic images have been picked from a set of automatically generated composites and, as you can see, some look reasonably real while others appear quite fake. What is it, then, that makes a composite image appear real? As we saw in Ch. 7, scene semantics and geometry play a key role [Biederman, 1981]—a car floating in midair or twice as big as other cars would instantly appear out of place. In this chapter, we will assume that these

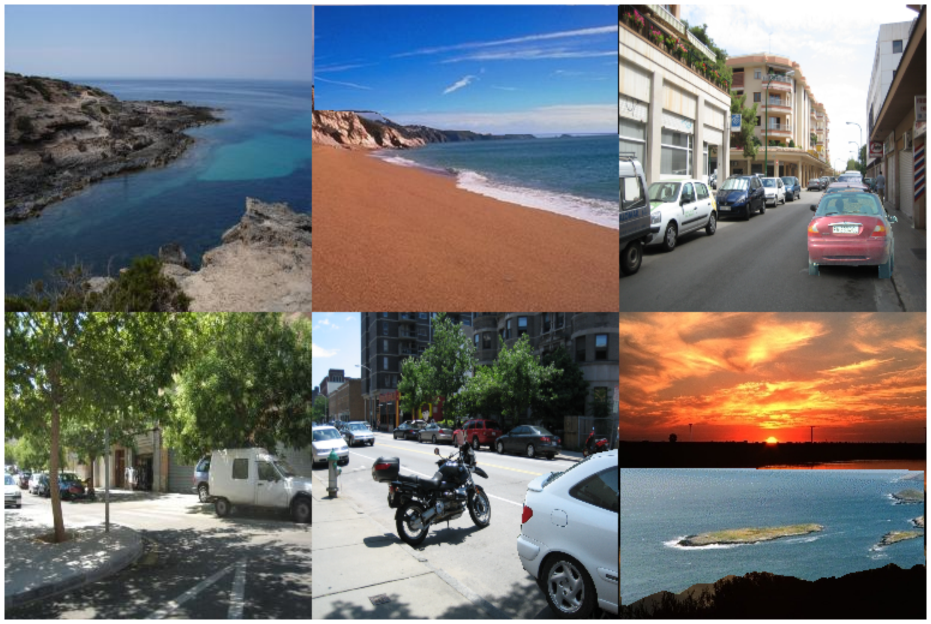


Figure 8.1: There are only two real images in this montage. Can you identify them?

high-level scene structural issues have been dealt with. We are interested in investigating the more subtle artifacts that appear even if the semantic composition of the scene is correct (e.g., right column of Fig. 8.1).

Difference in scene lighting between the source and destination images is one important consideration. The same object photographed under two different types of illumination (in a thick forest vs. a sunny beach) will usually have a strikingly different appearance. But does this mean that differently-lit objects will always appear inconsistent to a human observer when placed in the same image? Not necessarily. Cavanagh [2005] uses examples from art to demonstrate that humans are curiously unaware of the great liberties that artists often take with physics, including impossible lighting, inconsistent shadows, and incorrect reflections. Actually, this is not too surprising, considering that it is extremely difficult for both human and computers to estimate the true lighting direction from a single image, unless very strong cues like cast shadows are available (Fig. 7.4). In this chapter, we will assume that there are no such strong cues in the image, and aim for a more global effect of illumination: color. Indeed, the experience of Photoshop artists confirms that “getting color right” is one of the most important tasks for good image composites [Adelson, 2006].

Therefore, in this chapter, we concentrate on the role of color in image compositing.

8.1.1 The Role of Color

The first question one must ask is whether color itself is the important cue for visual compatibility or just a manifestation of some higher-order semantic relationships? Do we prefer certain shades of green with certain shades of blue, or do we just like to see grass and sky together? This is a very difficult question. While the experience of graphic designers as well as recent work on color harmony [Cohen-Or *et al.*, 2006] suggest that humans do prefer palettes of certain colors over others, it is likely that object identity also plays a role. In this chapter, we investigate how far we can get without the use of explicit semantic information.

Our second question relates to the nature of color compatibility. Do different pairs of colors simply appear to be more or less compatible with each other? Or perhaps compatibility is evaluated over entire color distributions rather than individual colors? In this thesis, we will take a first step in trying to answer some of these questions.

8.1.2 Prior Work

In computer graphics, people have long been interested in methods for adjusting the colors of one image to make it match the "color mood" of another image. In a much-cited work, Reinhard *et al.* [2001] propose a very simple technique based on matching the means and variances of marginal color distributions between images using the $L\alpha\beta$ color space. The central assumption of the method is that the marginal color distributions of the object and its background should match. This technique was applied to compositing of synthetic objects into real videos [Reinhard *et al.*, 2004], although no quantitative evaluation of the method's effectiveness was presented. While this makes sense for some cases, often you find an object becoming very greenish due to being pasted into a forest scene. To address this problem, Chang *et al.* [2005] proposed to first assign each pixel to one of 11 "basic color categories" obtained from psycho-physical experiments and relating to universal color naming in languages. The color adjustment is then performed only within each category. This method produces much more pleasing results (on the 7 images shown) but, again, no quantitative evaluation is performed. The Color Harmonization approach [Cohen-Or *et al.*, 2006] adjusts the hue values of the image color-map according to a predefined set of templates that are thought to encode color harmony. Alternatively, Pérez *et al.* [2003] propose to copy the object *gradients* and reintegrate to get the colors. While this approach results in a seamless insertion, it often generates noticeable artifacts such as color bleeding or severe discoloration when the object and background have dissimilar color distributions (again, these works are difficult to evaluate since no quantitative results are shown).

Computer vision researchers are more interested in the task of classifying images based on various characteristics. Farid and colleagues have done extensive work on using higher-order image statistics for a variety of tasks, including distinguishing between computer renderings and photographs [Lyu and Farid, 2005], detecting digital tampering, finding art fakes [Lyu *et al.*, 2004], etc. However, their efforts are directed towards detecting differences that are not perceptible to a human observer, whereas our goals are the opposite. Cutzu *et al.* [2003] present a neat technique for distinguishing paintings from photographs based on the color distribution of the image. Their insight is that paintings are likely to exhibit more color variation than photographs because it is difficult to mix paints that have the same chromaticity but different intensity. Ke *et al.* [2006] propose a set of high-level image features for assessing the perceived artistic quality of a photograph. Their color features include a measure of histogram similarity to a set of high-quality photographs, as well as an estimate of hue variation in the image (apparently, professional photos have fewer hues).

The classic paper by Forsyth [1990] has spawned a large body of work in color constancy. However, their goals are different: retrieve the illuminant under which a given scene was lit, from a list of known illuminants. One method related to the present work has been introduced by Finlayson *et al.* [2001], in which they determine the illumination based on nearest-neighbor matching to a set of illumination palettes.

8.1.3 Overview

Our goal is to use color information to automatically predict whether a composite image such as the ones in Fig. 8.1 will look realistic or not to a human observer. In the pursuit of this endeavor, two different and complementary approaches are evaluated.

The first approach utilizes the fact that phenomena that happen in the real world are,

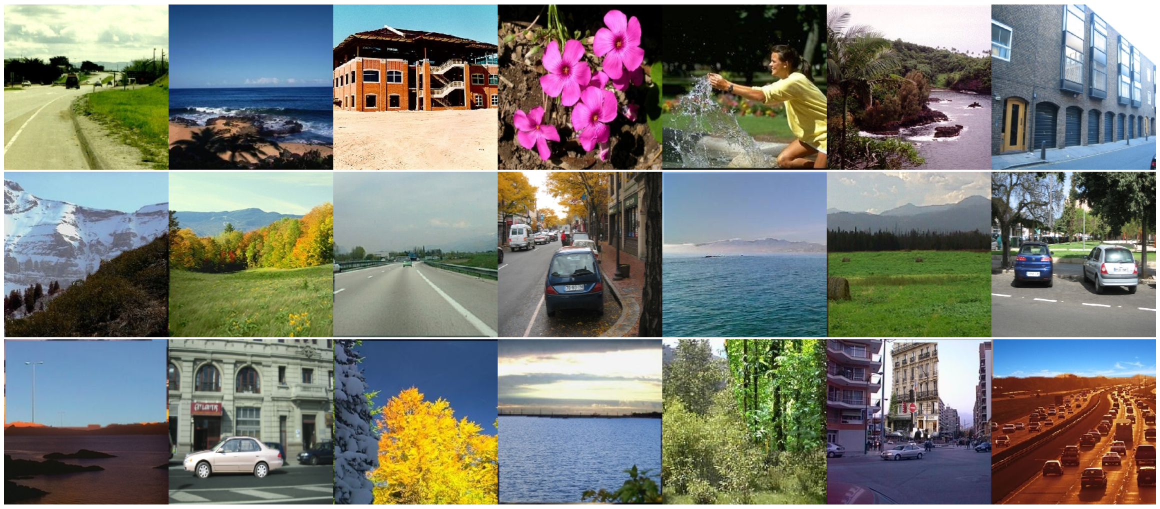


Figure 8.2: Example images randomly selected from our test set. *Top row*: Real images. *Middle row*: Realistic synthetic images. *Bottom row*: Unrealistic synthetic images. The entire test database used to produce the results presented in this chapter contains a total of 1000 images and was semi-automatically generated from images taken from the LabelMe database [Russell *et al.*, 2008].

by definition, natural. This translates to the hypothesis that colors in an image will look realistic if they appear in real images with high probability [Lotto and Purves, 2002]. We pose the problem as follows: given a set of colors (a palette), what are the other color palettes that are likely to co-occur in the same image? In Sec. 8.3, we propose several ways to estimate color palette co-occurrences.

Our second approach does not consider global color statistics and makes the assumption that a composite image will look realistic if the object and background colors have similar distributions. This idea is directly inspired by the work of Reinhard *et al.* [2004], which has never been evaluated rigorously on a large number of examples. We propose to extend the work using a better color representation and provide an extensive comparative evaluation in Sec. 8.4.

Finally, from the intuitions gathered while evaluating the global and local approaches, we suggest a way of combining them into a single classifier. Sec. 8.5 presents this combined approach and compares it to using either technique by itself. As an additional application, we show in Sec. 8.6 how to automatically shift the colors of an unrealistic object to make it look more realistic in its new scene.

8.2 Generating the Composite Image Dataset

In order to compare the different approaches proposed, a dataset of synthetic images was generated semi-automatically[1]. Since the process of manual image compositing can be long and tedious, we seek to automatically generate composite images that will look right semantically, i.e., objects should be at the appropriate locations in the resulting images. We propose a very simple algorithm to generate semantically correct images by utilizing a large segmentation dataset. We use the popular LabelMe image database [Russell *et al.*, 2008] which contains roughly 170,000 labeled objects. We first remove all incomplete objects by searching the label strings for words “part”, “occlude”, “regions” and “crop”. We then manually group objects that have similar labels, and end up with the following 15 most frequently-occurring objects in the dataset: “building”, “bush”, “car”, “field”, “foliage”, “house”, “mountain”, “person”, “road”, “rock”, “sand”, “sky”, “snow”, “tree”, and “water”.

Because segmentations are available, we can create a synthetic composite by starting with an image, and replacing one of its objects by another one of the same semantic type and shape. The algorithm only selects the objects that occupy at least 5% and at most 60% of their corresponding image area. The shape matching is done by computing the SSD over blurred and subsampled object masks, allowing for translations. Once the best matching object is found, we paste it onto the original image and apply linear feathering along the border to mask out potential seams. Even though this algorithm is very simple, it performs surprisingly well (see Figs 8.1 and 8.2), because it exploits the richness of the dataset.

Some of the automatically generated composite images happen to have matching colors and appear quite realistic, while others have color distributions that make them look unrealistic. Sometimes, however, the automatic procedure fails completely, producing results that are structurally inconsistent and obviously wrong. We label those as unsuccessful and manually remove from the test data. We asked three human observers with normal color vision to label the remaining images as either realistic or unrealistic. The realistic class is augmented with randomly selected real images from the dataset. For real images, a random object in that image is selected to be the tested inserted object. Our final test set is

[1]The dataset is publicly available at `http://graphics.cs.cmu.edu/projects/realismcolor`.

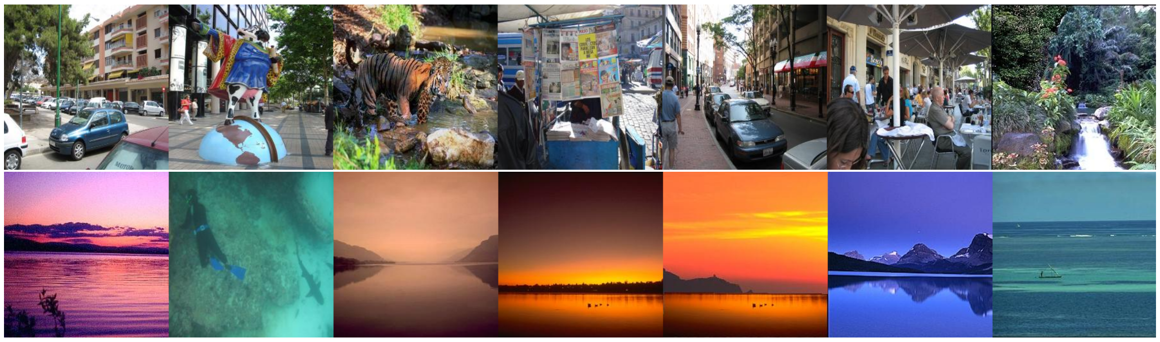


Figure 8.3: Most and least realistic images ranked by the universal palette algorithm. *Top row*: 7 most realistic images in the test database. *Bottom row*: 7 least realistic images. Note that this first-order analysis is completely unable to tell apart realistic images from unrealistic ones.

composed of 360 unrealistic, 180 real, and 180 realistic images. Fig. 8.2 shows examples of typical real and synthetic (realistic and unrealistic) images in our test set, which remained identical for all the experiments performed in this chapter. We also employ a much larger and non-overlapping part of the LabelMe dataset containing 20,000 images to compute the natural color statistics in the following experiments.

8.3 Global Natural Color Statistics

In this section, our aim is to find a way to test the naturalness of colors in a given image by computing their similarity with global color statistics accumulated over a large set of real images. We propose three ways of doing so.

8.3.1 Universal Color Palette

The simplest way of modeling the joint natural color distribution is to assume that an image is generated according to a single, global distribution, which is the same for all images. This assumes the existence of a universal palette, from which all natural images are generated. While this first-order assumption is restrictive, we can easily estimate the joint distribution of all training images by computing a global 3-dimensional joint histogram in CIE L*a*b* color space. Given a new image, we compute its color histogram, and compare it to the global model. Unless otherwise noted, all experiments in this chapter are performed in the CIE L*a*b* color space using 3-D joint histograms with 100^3 bins. Fig. 8.3 presents images that are close (top row) and far (bottom row) from this global distribution, using the χ^2-distance metric for histogram comparison. It illustrates that the universal palette has the tendency to (falsely) predict that images with a single, saturated color are less likely to be realistic. Whereas this approach is clearly not useful in our image realism classification setting, it might still have interesting applications, such as finding striking and unusual color photographs in a large dataset (see Fig. 8.3, bottom row).

8.3.2 Expected Color Palette

While the universal palette is easy to compute, it appears not to be powerful enough to model the complexity of natural images because it only models first-order color statistics. A better approach would be to consider pairs of colors that appear together in the same image. Given a 3-dimensional color space, this is a 6-dimensional function that represents the probability of observing a color distribution given a single color. Stated differently, it is modeling the palette that is likely to co-occur together with a particular color in a real image.

We represent this distribution by a 6-dimensional histogram of 16^6 bins. For every color in every object in the entire training dataset, we compute the histogram of all the colors occurring in that objet's background region. For testing: given a composite image composed of an object and its background, we sample the 6-D histogram by marginalizing over all the colors in the object and compare it with the histogram of the background.

Unfortunately, this method performs only marginally better than the first-order approximation, and still does not yield satisfying results on our test dataset. To quantitatively compare the different techniques, we use the χ^2-distance metric to assign a realism score to every image in our test dataset and construct ROC curves; the same procedure is used in the remainder of the chapter. In our experiments, the area under the ROC curve for the universal palette is 0.59, and 0.61 for the second-order. Several reasons explain this poor performance: the 6-dimensional histogram is likely too coarse and smoothes over important color shades. More importantly, this model only represents a single expected palette given a single color, which is not enough to capture the complexity of our visual world. For instance, blue sky co-occurs with grass, roads, seas, cities, etc. each of which might exhibit very different color palettes. Since this method is computing the average over all observed instances, it is not powerful enough to model each of them jointly.

8.3.3 Data-Driven Color Palette

To address the limitations of the previous method, we would ideally need to know the co-occurrences of all possible color palettes. Since this number is huge, we would require a prohibitively large number of images and computing power to compute them.

Although it might be possible to employ the method of Yang *et al.* [2007] to extract a more powerful co-occurrence feature, we note instead that because we have large amounts of real data with labelled objects, we can use a nearest-neighbor approach to approximate this distribution directly. Given the object color palette, we find a set of k most similar-looking objects based on color (k-NN), and approximate its expected co-occurring palette by the best-matching background in this k set. This method yields an area under the ROC curve of 0.74, a significant improvement over the previous techniques.

Can we improve its performance further? Let's consider an example. When determining if a tree matches a particular forest scene, it might be more important to look for similar forest images, which typically have very consistent color palettes, than for green buildings which might exhibit different shades of green and still look realistic. Clearly, incorporating object recognition could greatly help in matching similar scenes. Here, we experiment with a weak recognition cue by using texture matching between images.

First, a texton dictionary of 1000 instances is learned by clustering 32-dimensional oriented filter responses on our 20,000 training images. A texton histogram can then be computed for each object and associated background in the training data and be used in the k-NN process. To evaluate the distance between two objects, we take a linear combination of their color and texton histograms χ^2-distances. A parameter α is used to control the relative importance of color and texture, where $\alpha = 0$ indicates only texture, and $\alpha = 1$ means only color. In Table 8.1, we provide a comparative evaluation using $\alpha = \{0, 0.25, 0.5, 0.75, 1\}$, progressively increasing the influence of color. We observe that texture information improves upon results obtained with color only, but is ineffective when used alone, which seems to confirm our intuition.

α	0	0.25	0.5	0.75	1
ROC AUC	0.59	0.69	0.74	**0.79**	0.74

Table 8.1: Influence of color and texture on the nearest-neighbor retrieval. Using texton only ($\alpha = 0$) fails to return good nearest neighbors, and maximum performance is obtained when $\alpha = 0.75$ (in bold). Scores are obtained by computing the ROC area under the curve (AUC) on our test set.

The limitations of such an approach is that it wholly depends on the training data. Given the huge dimensionality of the space of all scenes, we cannot expect to find a matching scene for any given image, even with a dataset of 20,000 images. We will now investigate a different, more local class of techniques, which can hopefully compensate when the training data cannot help.

8.4 Local Color Statistics

Reinhard *et al.* [2004] have demonstrated a very simple way of making an object match its background by shifting its colors to make them closer to the background colors. The intuition behind this idea is that this shift appears to be increasing the correlation between the object and background illuminants. For example, the reddish hue of a sunset sky should appear on all objects in the scene. An object taken from a bright day scene can be made to look better in the new sunset background by shifting its colors towards red. While their application is in image recoloring, it can also be used in our context by computing the distance between the object and its new background colors.

The color description used in [Reinhard *et al.*, 2004] is a simple marginal histogram in $L\alpha\beta$ color space. A straightforward improvement is to use the full 3-D joint histograms instead of marginals because color components are still quite correlated even in $L\alpha\beta$. Interestingly, this yields substantial improvement, going from an area under the ROC curve of 0.66 for marginals to 0.76 for joint on our test data, at the cost of a higher-dimensional representation. The same intuition of using texture as mentioned in the previous section also applies here, and improves performance as well, as shown in Table 8.2.

technique	marginals	joint	joint with texture
ROC AUC	0.66	0.76	**0.78**

Table 8.2: Summary of local techniques. The best performance (bold) is obtained by combining a joint histogram representation with texture matching using texton histograms. Scores are obtained by computing the ROC area under the curve (AUC) on our test set.

8.5 Combining Global and Local Statistics

Let us consider for a moment the two techniques introduced in the previous sections. When the global method yields a high realism score, we can be confident that it is correct because it relies on matches to actual real scenes. However, when it is uncertain, it means that no good match was found, and the results are not reliable. We can then only rely on the local approach. This suggests that we can combine both global and local ideas into one coherent classifier.

We propose a two-stage cascade. First, the algorithm computes the distance to the nearest-neighbor according to the best global measure from Sec. 8.3. If the match is good enough (as determined by a threshold τ), it classifies this image as realistic. Otherwise, it uses the local method from Sec. 8.4 to assign a realism score. We use 10-fold cross-validation on our labeled dataset to determine the best parameter τ (0.35 in our case) which will maximize the area under the ROC curve.

Fig. 8.4 shows the ROC curves for the best techniques presented here. We compare our techniques against the baseline proposed by Reinhard *et al.* [2004]. The results clearly show that combining the global and local techniques results in performance superior to any single one. In another paper [Reinhard *et al.*, 2001], they make a point of using $L\alpha\beta$ color space. We performed each experiment by using the $L\alpha\beta$, CIE L*a*b*, HSV and RGB color spaces and found that the CIE L*a*b* color space performs the best, closely followed by $L\alpha\beta$.

In Fig. 8.5, we show a visual representation of the rankings produced by our combined classifier on our test set. Images are shown ranging from least (top) to most (bottom) realistic. Ground truth labels are illustrated by color borders: a red border indicates an unrealistic image, green and blue indicates realistic and real respectively. It is interesting to observe the output of the algorithm at mid-range where confusion is higher: even for

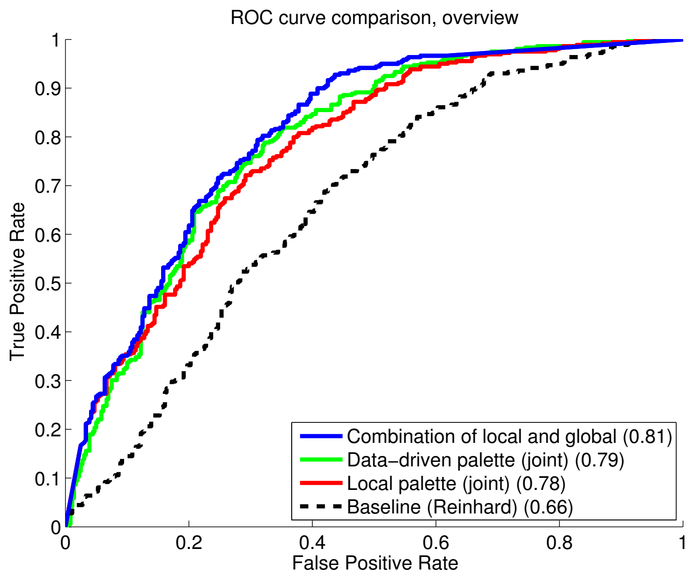


Figure 8.4: The ROC curves comparing the best approaches of the global and local categories, as well as the combination of both, shown against the baseline method, suggested by Reinhard *et al.* [2004]. The combination of both local and global methods outperforms any single method taken independently, and is significantly better than the baseline approach.

humans, assessing image realism is much less obvious and requires a more careful inspection than for images at both ends of the spectrum.

8.6 Application to Automatic Image Recoloring

An interesting application as well as a visual evaluation of this technique is the recoloring of an image to make it appear more realistic. The idea is to first classify the image using our proposed method and retrieve either a nearest scene (if the global method is used), or the determination that no matching global scene is available. In the first case, we need to recolor the object to match the colors of similar objects in that nearest scene. In the second case, we can only try to make the object more similar to its surroundings, as in [Reinhard *et al.*, 2004].

The goal of recoloring is to modify a source color distribution D_s in order to match a target color distribution D_t. In our setting, D_s represents the object colors, and D_t is the nearest neighbor object if the global method is used, or the background otherwise. Our

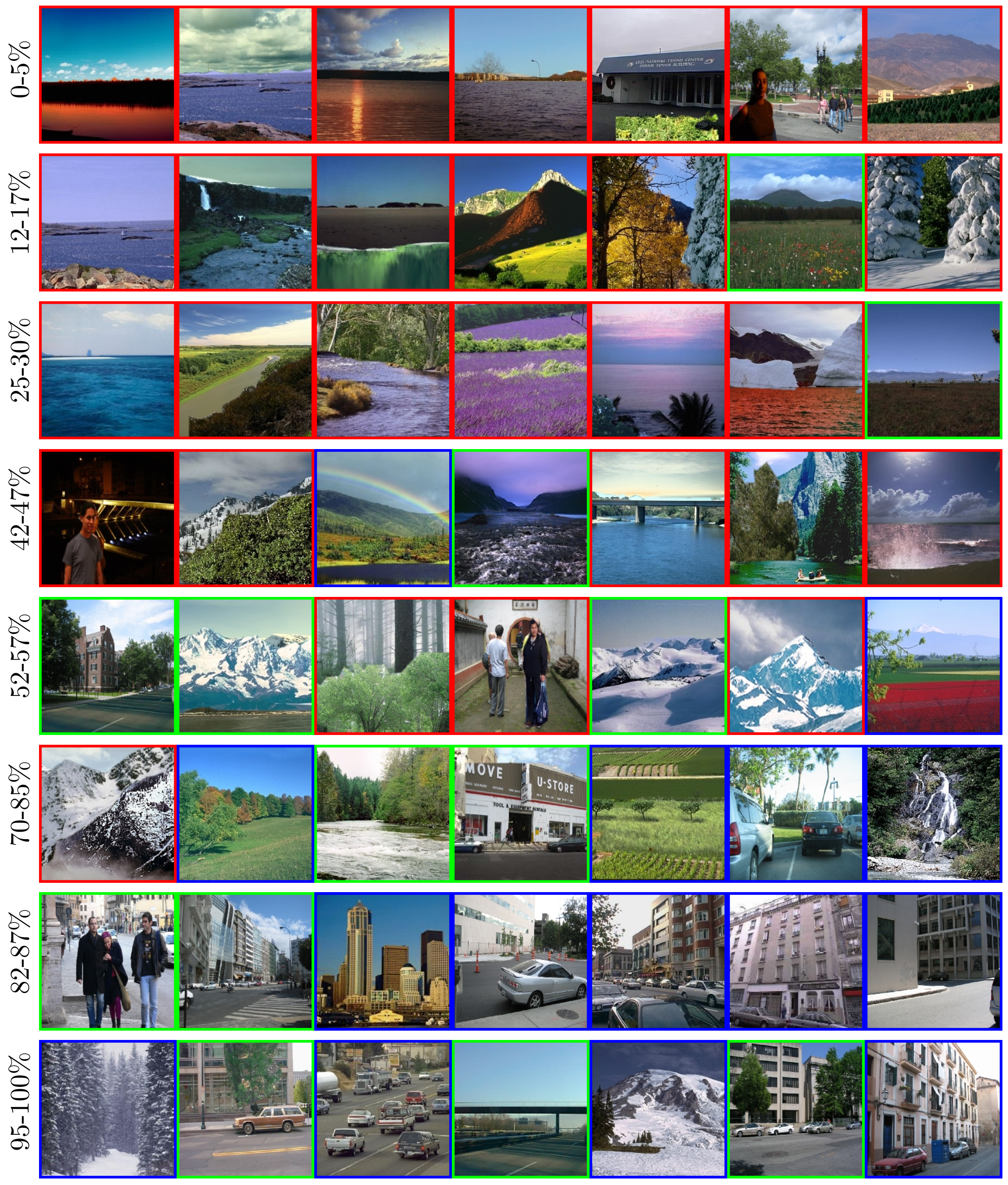


Figure 8.5: Images ranked according to their realism score, determined by combining the best global and local methods. The border color indicates the labeled class for each image (red: unrealistic synthetic; green: realistic synthetic; blue: real). Each row corresponds to the percentile interval shown on the left, and images are randomly selected within each interval. *Row 1*: 0-5% interval (most unrealistic), *2*: 12-17%, *3*: 25-30% , *4*: 42-47%, *5*: 52-57%, *6*: 70-75%, *7*: 82-87%, *8*: 95-100% (most realistic).

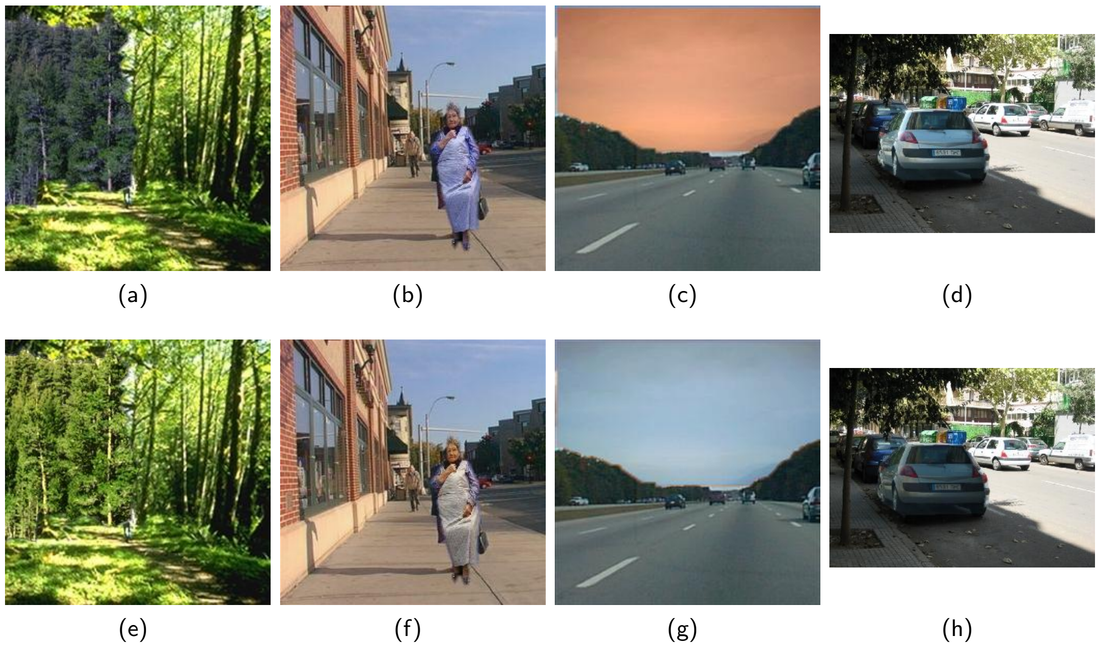


Figure 8.6: Automatic image recoloring. The input images (a)–(d) are recolored by using the local (e)–(f) and the global statistics (g)–(h). Recoloring these unrealistic input images increases their realism.

color matching procedure is an automatic extension of the interactive recoloring approach from [Reinhard *et al.*, 2001], where they propose to represent both D_s and D_t by k color clusters, with k being manually chosen to be the number of major colors in the scene. Each cluster in D_s is matched to a cluster in D_t by comparing their means and variances.

Instead, we propose an entirely automatic algorithm. Each color distribution is represented by a mixture of k spherical gaussians ($k = 100$ and remains constant for all images), and the distributions are matched in a soft way using the solution to the well-known transportation problem. The algorithm is divided in three steps. First, we use the Earth Mover's Distance algorithm [Rubner *et al.*, 2000] to compute the best assignment between the clusters in D_s and D_t. Second, color shift vectors for each cluster in D_s are computed as a weighted average of its distance in color space to each of its assigned clusters in D_t. Finally, every pixel in D_s can be recolored by computing a weighted average of clusters shifts, with weights inversely proportional to the pixel-cluster distance. These three steps are performed in the CIE L*a*b* color space, and the results are converted back to RGB for visualization.

Examples of recoloring using the global and local models are illustrated in Fig. 8.6, and show that we can automatically improve the realism of composite images by using the same general approach.

8.7 Conclusion

In this chapter, we study the problem of understanding color compatibility using image composites as a useful application domain. We propose two measures for assessing the naturalness of an image: 1) a global, data-driven approach that exploits a large database of natural images, and 2) a local model that depends only on colors within the image. We show that while both techniques provide substantial improvement over previous work, the best approach needs to use both techniques for different types of images.

We evaluate our approach on a large test dataset of synthetic images, generated by a novel semi-automatic technique. We are the first work in this field to provide a quantitative evaluation and we demonstrate performance superior to the state of the art method [Reinhard *et al.*, 2004]. We also qualitatively validate our approach using a novel image recoloring method that makes composite images look more realistic.

A number of issues, such as position of objects in the image, object semantics, material properties, and more explicit illumination parameters still need to be addressed. For instance, it will be important to evaluate the influence of correctly matching the sun position estimated in Ch. 6 on image realism. We are planning to explore these issues in future work.

While the ideas presented in this chapter are only a first step, lessons learned from these experiments should be extremely useful to steer this area towards a better understanding of natural color statistics and color perception.

Chapter 9

Discussion and Future Work

And now for something completely different.

Monty Python (1969–1983)

Our work represents only a small step towards providing a full understanding of natural illumination in image sequences and single images. In order to reach that goal, we believe there is still much to be done to improve upon and extend the proposed framework, and to move beyond. In this final chapter, we discuss the combination of data-driven techniques with physical models, talk about the challenges of dealing with image sequences and single images, explore interactions between lighting and objects, present future research directions, and finally consider further steps towards illumination-aware scene understanding.

9.1 Physical Models vs. Data-driven Techniques

Throughout this thesis, we have proposed to use both physically-based methods—physical model of the sky appearance, color ratios for shadow detection, geometric matching of the sun position—and data-driven techniques—nearest-neighbor classifier for image synthesis and realism assessment, SVM classifier for shadow detection—with the goal of understanding and exploiting natural illumination in images. We now discuss some challenges related to these two broad categories of techniques, and the benefit of using them jointly.

Physical Models Physically-based models provide a deep, mathematical understanding of the variables that come into play in creating a physical phenomenon, and how they interact together to produce the results observed in images. Their main advantage is that they can predict complex appearance effects, such as the appearance of the sky, from a limited number of parameters. They can also be useful to predict cases that have never been seen a priori. However, their main drawbacks are that they assume high-quality images with very little distortions due to noise and non-linearities, and their parameters can be very hard—sometimes even mathematically impossible—to estimate from single images. How far can we go using only physical models? Unfortunately, we cannot go very far in single images unless we assume some of the variables (e.g., geometry, lighting, or camera parameters) to be known.

Data-driven Techniques In contrast, data-driven techniques have many more parameters to estimate, and these parameters do not typically correspond to physical interpretations of the phenomena at hand. Because of that, it may be hard to obtain an intuitive explanation of their results. Their main advantage, however, is that the parameters can be automatically learned from labelled—and, in some cases, unlabeled—data, provided the training database is sufficiently large and varied. This makes such techniques ideal for dealing with the inherent uncertainty of single images, but their expressiveness is limited by the examples in the training data. How far can we go using purely data-driven techniques? Unfortunately, we can only go as far as the data will go. So before we can capture the entire world, we need to keep working on extracting trends from limited, representative datasets.

We have shown in Ch. 6 that combining physical models with data-driven techniques in a probabilistic manner results in a framework which overcomes the limitations of both methods, and takes advantage of their strengths. We believe that by striving to combine physical models with data-driven techniques, we should be able to achieve better performance than by considering them independently.

9.2 Image Sequences vs. Single Images

This thesis has considered the tasks of estimating the illumination conditions and synthesizing novel, lighting-consistent appearance in two situations: 1) in image sequences (Part I), and 2) in single images (Part II). We briefly discuss here important insights learned from considering these two sources of data.

Image Sequences Capturing a sequence of images over time, from a static camera, constrains several variables to a fixed value so it becomes easier to estimate the others. Part I showed how the temporal information available in such image sequences can be used to estimate the illumination conditions. In particular, doing so has allowed us to discover that the sky, when observed over time, is a trove of information that had yet to be tapped into.

Single Images In the end however, our ultimate goal is to be able to reason about single images, in which temporal information is not available (Part II). One of the key ideas of this thesis has been to show how we the insights learned from image sequences can be applied to this challenging case. In the case of the sky, we have shown that by using the same physical model in a probabilistic framework, it can also be exploited in single images. But we believe we can go much further by exploiting the novel dataset of webcam sequences presented in Ch. 4. Using the temporal information available, we can robustly and automatically extract the illumination conditions at each frame. If we ignore temporal information, this effectively results in a database of millions of *single* images where the lighting conditions are known. This should prove to be an invaluable tool for learning how the appearance of real-world images vary as a function of illumination, and we hope that the public availability of such a dataset[1] will spur research in this area.

9.3 Illumination-Aware Pedestrian Detection

One pressing research question that rises from our work is: can our explicit knowledge of illumination be used to improve existing computer vision tasks? We have obtained some

[1]Freely available at `http://graphics.cs.cmu.edu/projects/webcamdataset`.

promising preliminary results in the context of outdoor pedestrian detection, and report them here.

9.3.1 Overview

It has been shown that it is sometimes beneficial to separate an object class into subclasses, and train separate detectors for them independently. The main reason explaining an increase in performance is that when there is too much variation in appearance within a class, it puts a large burden on the classifier which must somehow group together objects that have very different features. By splitting the data into more homogeneous groups, the classification task becomes easier, and recognition performance increases. The critical point here is to split the data into coherent subgroups of similar appearance. For instance, Hoiem *et al.* [2006] have observed increased performance in pedestrian detection when their training set was split into subgroups according to scale: large and small people should be trained on (and detected) independently. Another common way to split classes is by orientation: cars appear widely different whether they are seen from the side or from the front [Savarese and Fei-Fei, 2007].

We propose the hypothesis that strong changes produced by varying illumination could be harming the performance of an object detector. By exploiting illumination as an explicit variable, it might be possible to improve detection performance. This intuition has been explored in the context of faces. Faces have the advantage that the main variation in their appearance is due to lighting, and not to intra-class variations [Moses *et al.*, 1994]. Indeed, PCA analysis on registered faces has shown that the first few principal components encode specific lighting directions [Ramamoorthi, 2002] (assuming a lambertian reflectance model). Even with such properties though, recognizing faces under varying illumination is still a very hard problem [Nishino *et al.*, 2005; Han *et al.*, 2010], especially when there is a single dominant light source, as is the case outdoors with the sun [Lee *et al.*, 2001].

On the other hand, for many other objects such as cars or people, the largest source of variation lies in the within-class deformations (e.g., brands of cars), so lighting becomes much harder to capture. However, now that we can directly observe the usually hidden lighting variable, we can bypass lighting invariance, essential in recognition, and hopefully increase performance for many different types of objects observed in natural settings. This is akin to the work of Nishino *et al.* [2005], in which they explicitly estimate the illumination conditions based on the reflections on the eyes. This results in much improved face recognition performance.

9.3.2 Our Approach

In our work, we experimented with a manual split of the training and test set into four bins, based on the sun direction: back-left $[-180°, -90°]$, front-left $[-90°, 0°]$, front-right $[0°, 90°]$ and back-right $[90°, 180°]$. We use the same training and test set as in Secs 6.3.4 and 6.5.2 respectively, which contain 2,000 training and 239 test images, all taken from the LabelMe dataset [Russell *et al.*, 2008].

For the detector, we use the approach of Felzenszwalb *et al.* [2008], with a single component, and with the parts model turned off. This effectively results in an SVM classifier trained on standard Histogram of Gradients (HOG) features [Dalal and Triggs, 2005], where the root location of the bounding boxes is treated as a latent boundary, and where the hard negative data mining step is activated during training. We train four independent classifiers on the four splits of the training set, and apply the sigmoid normalization step of Platt [1999] to have each one generate probabilistic outputs.

9.3.3 Initial Experiments

At test time, we *manually* select which detector to run based on the *ground truth* sun position in the 239 test images, and compute the average precision (AP) curve for the resulting detections. Overall, the improvement is modest: using the 4 illumination splits results in an increase of 2% in AP in our test sets. A few caveats may explain such a limited improvement in performance. First, the HOG features employ a contrast-normalization scheme which strives to achieve illumination invariance. Since this is contrary to our goal of capturing variations due to illumination, new or complementary features should instead be used. Interestingly, the HOG feature weights recovered for each classifier do differ based on illumination direction, even despite this normalization scheme. As shown in Fig. 9.1, differences arise where hard shadows (e.g., around the feet) or hard contrast with the back-

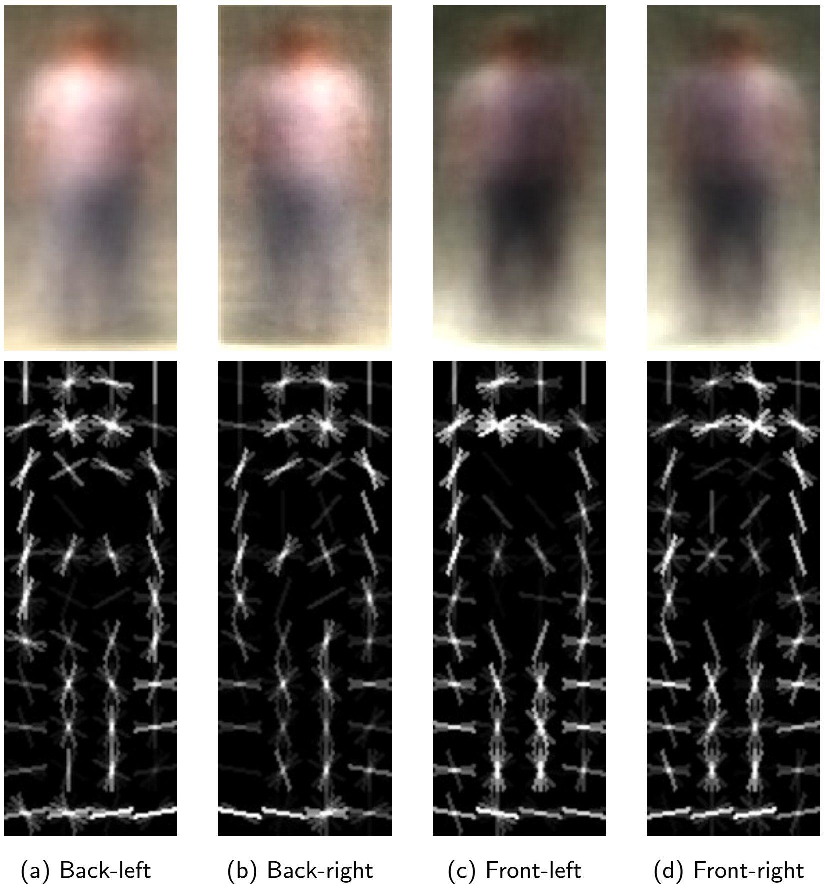


Figure 9.1: Mean images (*top row*) and HOG [Dalal and Triggs, 2005] weights learned (*bottom row*) for pedestrians with the sun coming from four different directions: (a) back-left, (b) back-right, (c) front-left, and (d) front-right. Differences in HOG weights are visible around the feet and shoulders regions.

ground (around shoulders or sides) are likely to be. This is encouraging, but it suggests that further research in this area should focus on designing features that explicitly aim to capture variations caused by lighting. The second caveat is that the training and test set contain many very small pedestrians, where the limited resolution obfuscates illumination effects. A better, higher-resolution dataset should be used instead [Dollár *et al.*, 2009]. Overall, these experiments suggest that knowledge about illumination might indeed be used to improve object detector, but further research in this direction is required to show additional improvement.

9.4 Future Directions

We now discuss possible extensions to our current framework.

9.4.1 Integrating Other Illumination Cues

Since our framework is probabilistic in nature, additional cues can readily be added without any significant changes. We review a few possibilities of existing or new illumination cues that, although not as commonly-available in images as the ones we have used in our work (see Ch. 6), could also be integrated within our framework.

Human faces Front-facing human faces can easily be detected in images using off-the-shelf face detectors as in [Bitouk *et al.*, 2008]. From them, it should be possible to also estimate the illumination conditions [Huang *et al.*, 2008; Bitouk *et al.*, 2008; Han *et al.*, 2010].

Occluding contours If occluding contours can be detected in a single image [Hoiem *et al.*, 2010], then the illumination estimation technique of Johnson and Farid [2007] could readily be integrated in our framework.

9.4.2 Higher-Level Shadow Reasoning

This thesis has argued in Ch. 5 and 6 that detecting shadow boundaries is a useful tool for understanding illumination in single images. A possible extension to this framework is to reason about shadow *regions* instead [Zhu *et al.*, 2010]. Regions might provide additional robustness to noise and to uncertainty in boundary classification, since it requires that a closed surface area be in shadows. In addition, shadow regions would provide the spatial support required for reasoning about the object which cast the shadow in the first place: if elongated, then the caster is likely a vertical object such as a pedestrian, or a pole; if scattered with holes, then it likely comes from foliage or vegetation.

9.5 Towards Illumination-Aware Scene Understanding

Throughout this document, we have argued that illumination interacts with the world in complex ways. Objects cast shadows on their surroundings, the clear sky exhibits a gradient that is a function of the sun position, surfaces facing the sun are brightly lit, etc. In this thesis, we have proposed techniques to analyze these effects in order to reason about illumination, and exploit them in graphics applications, in both image sequences and single images. Fig. 9.2 illustrates this graphically, with the red arrows corresponding to the contributions of this thesis. Going forward, we propose that we should also move beyond the task of solely estimating illumination, and aim at reasoning about the interplay between scene and illumination as a whole (blue arrows in Fig. 9.2).

The interaction between illumination and objects (Fig. 9.2, center) has already been discussed in Sec. 9.3. Let us now elaborate on the relationship between scene geometry and lighting. The goal of scene layout estimation, as introduced by Hoiem *et al.* [2007], is to segment the image into broad geometric classes: sky, vertical surfaces, and support (ground). The vertical surfaces are further subdivided into planar orientation (facing left, facing right, facing forward), porous (e.g., vegetation) or solid (with no definable orientation, like pedestrians or cars for example). We have already discussed at length in Ch. 6 the interplay between scene geometry and illumination: surfaces facing the sun should be brighter than those facing away from the sun[2]. While we already know that bright surfaces inform us about the sun position (see Sec. 6.3.3), knowledge of the sun position also tells us something about surfaces: if the sun comes from the right, then left-facing surfaces should be dark. Can information about the sun position help improve scene layout estimation? This is an interesting future research direction.

Another interesting example of scene-illumination interactions is in scene boundary reasoning [Canny, 1986; Martin *et al.*, 2004; Hoiem *et al.*, 2010]. Several phenomena are responsible for creating boundaries in an image: reflectance (material) changes, surface normal discontinuities (e.g., corner of a building), occlusions (e.g., edge of an object), and cast shadows. However, since cast shadows do not coincide with object boundaries in the

[2]Surface reflectance (albedo) also has a role to play here, but illumination changes typically dominate reflectance changes in brightly-lit scenes.

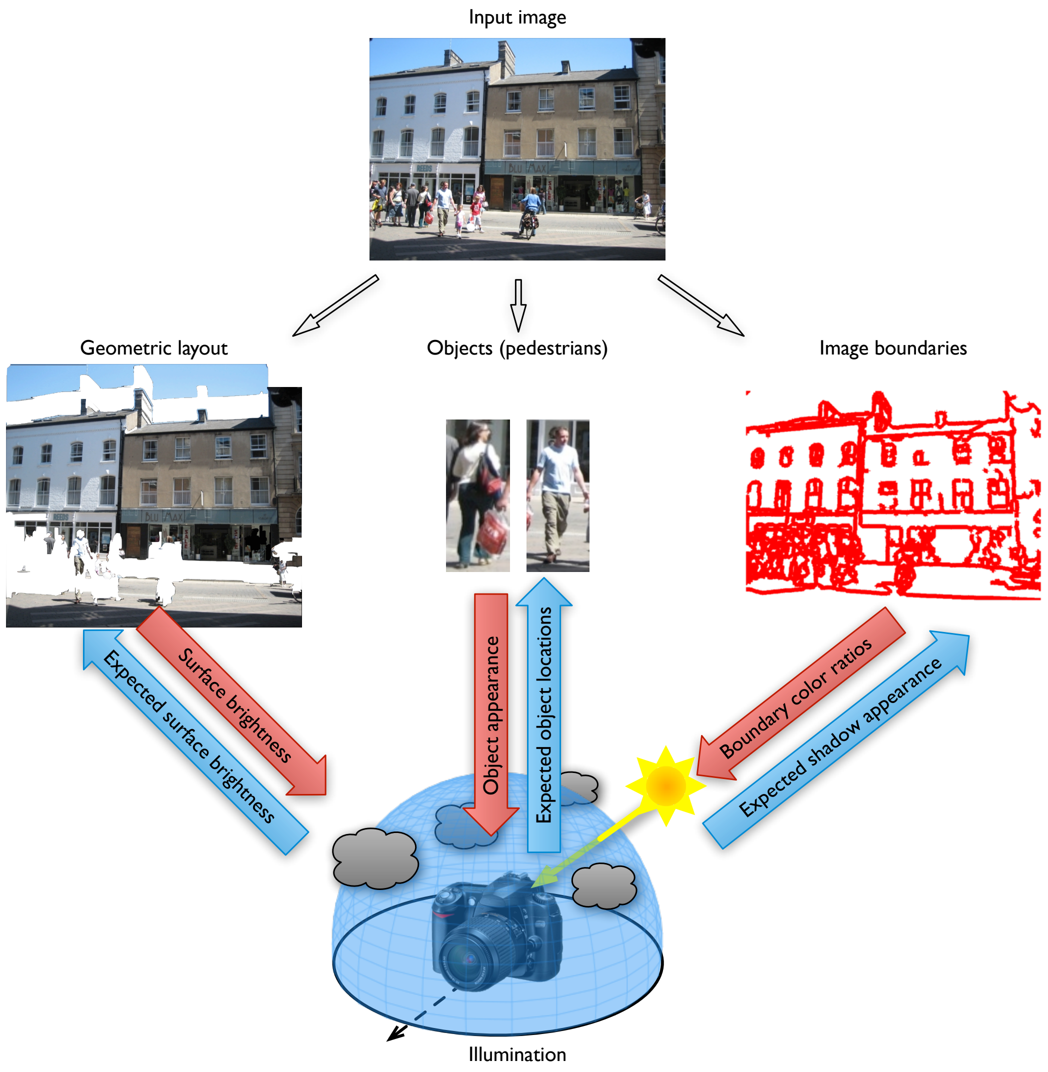


Figure 9.2: Towards illumination-aware scene understanding. Starting from an input image (top), this thesis has shown how to exploit geometric surfaces [Hoiem *et al.*, 2007], objects [Felzenszwalb *et al.*, 2010], and image boundaries [Canny, 1986] to recover the illumination conditions in an image (red arrows). An interesting future research direction consists in understanding how this knowledge can be used to improve these important computer vision tasks (blue arrows).

image, existing algorithms treat them as nuisances and strive to become invariant to them. Instead, we argue that we should explicitly include knowledge of shadows in boundary reasoning algorithms, for example using specific detectors like the one presented in Ch. 5.

In all the cases discussed here and presented in Fig. 9.2, there is one important caveat that anyone attempting to solve this problem will have to carefully examine and handle:

knowledge of the illumination conditions (especially the sun position) cannot *always* help reasoning about the scene in a given image. Because the sun seldom shines on the *entire* scene all at once (unless it is an open field devoid of objects), its light is bound to be occluded by some objects and thereby creates large shaded areas. Objects, boundaries, or surfaces in these areas suddenly become unpredictive of the sun direction: they simply are dark. So we are facing a tricky ambiguity here: is the region of the image dark because the sun comes from a different direction (e.g., not facing the surface), or is it because some other scene element blocks the sunlight, which would otherwise shine straight on? Solving this ambiguity might require specific tools like the "sunlit pedestrian" classifier, from Sec. 6.3.4, which is explicitly trained to distinguished between sunlit and shaded pedestrians.

Conclusion

What we call the beginning is often the end.
And to make an end is to make a beginning.
The end is where we start from.

T. S. Eliot (1888–1965)

In conclusion, this dissertation proposes a radical shift from the way natural illumination is considered in computer vision: instead of striving to become invariant to illumination, we should *embrace* it by actively seeking ways to analyze and exploit its effects on images. It has also shown how this novel understanding of illumination in images can be used in computer graphics to generate novel photo-realistic content by *borrowing* appearance in similarly-lit images. It is our hope that this thesis will inspire others, and provide them with the insights and tools required to move towards illumination-aware scene understanding and synthesis.

Appendix A

Estimating Illumination in Image Sequences: Additional Material

A.1 Geolocating the Camera Using the Sun and the Sky

The techniques presented so far have been considering either the sun position or the sky appearance as two independent sources of information that could be used to recover the camera parameters. We now demonstrate that, by combining them, we can avoid the requirement of having to know the GPS location of the camera, and also estimate it.

Recovery of the GPS location, or geolocation, has been explored in a variety of scientific fields. In biology, scientists are tracking many marine animals using light sensors. The sunset and sunrise times are found by analyzing the light intensity profiles captured by these sensors, which are then used to geolocate the animals and track them [Hill, 1994]. In robotics, altimeters are used on outdoor mobile robots to accurately detect the sun position, and compute the GPS location from several observations [Cozman and Krotkov, 1995]. In computer vision, Jacobs *et al.* [2007b] determine the position of webcams by correlating their intensity variations computed over several months with sunlight satellite images of the same period.

A.1.1 Algorithm

In our work, we show how we can estimate the latitude and longitude of the camera, as well as its geometric parameters, from an image sequence in which the sun and sky are visible. We introduce the following algorithm:

```
Algorithm 3: Camera localization from the sun and the sky
    Input: Sun position in images: $(u_s, v_s)$;
    Input: Clear sky images;
    Input: Date and time of capture of each image.
  1 Find set $\mathcal{I}$ of images where the sun is far away from the field of view;
  2 Solve the non-linear minimization (3.16) to recover $f_c$ and $\theta_c$;
  3 Compute the sun angles $(\theta_{si}, \phi_{si})$ from $(f_c, \theta_c)$ and the sun labels $(u_s, v_s)$ using (A.3);
  4 Solve the non-linear minimization (A.4) to estimate the latitude $l$ and longitude $L$;
  5 Solve the non-linear minimization (3.17) to recover the camera azimuth $\phi_c$.
    Output: Camera parameters: $(f_c, \theta_c, \phi_c)$;
    Output: Camera latitude and longitude: $(l, L)$.
```

We now detail lines 3 and 4 of Algorithm 3: how to recover the latitude l and longitude L of the camera given the date and time of capture, the camera zenith angle θ_c and focal length f_c, as well as the sun position in images (u_s, v_s). Our approach relies on an equation expressing the sun zenith and azimuth angles θ_{sg} and ϕ_{sg}, as a function of time, date, latitude and longitude on Earth [Preetham *et al.*, 1999]:

$$\begin{aligned}\theta_{sg} &= \frac{\pi}{2} - \arcsin\left(\sin l \sin\delta - \cos l \cos\delta \cos\frac{\pi t}{12}\right)\\ \phi_{sg} &= \arctan\left(\frac{-\cos\delta \sin\frac{\pi t}{12}}{\cos l \sin\delta - \sin l \cos\delta \cos\frac{\pi t}{12}}\right) \quad ,\end{aligned} \tag{A.1}$$

where δ is the solar declination in radians, and t is solar time in decimal hours. δ and t are given by:

$$\begin{aligned}\delta &= 0.4093 \sin\left(\frac{2\pi(J-81)}{368}\right) \quad ,\\ t &= t_s + 0.17\sin\left(\frac{4\pi(J-80)}{373}\right) - 0.129\sin\left(\frac{2\pi(J-8)}{355}\right) + \frac{12L}{\pi} \quad ,\end{aligned} \tag{A.2}$$

where t_s is standard time in UTC coordinates in decimal hours, and J is the julian date (the day of the year as an integer in the range 1 to 365).

The relationship between the sun pixel coordinates (u_s, v_s) in the images and its zenith and azimuth angles θ_{si} and ϕ_{si} is the following:

$$\begin{aligned} \theta_{si} &= \arccos\left(\frac{v_s \sin\theta_c + f_c \cos\theta_c}{\sqrt{f_c^2 + u_s^2 + v_s^2}}\right) \\ \phi_{si} &= \arctan\left(\frac{f_c \sin\phi_c \sin\theta_c - u_s \cos\phi_c - v_s \sin\phi_c \cos\theta_c}{f_c \cos\phi_c \sin\theta_c + u_s \sin\phi_c - v_s \cos\phi_c \cos\theta_c}\right) \quad , \end{aligned} \tag{A.3}$$

see Appendix A.3 for the derivation.

Note that at the ground truth GPS location, $\theta_{sg} = \theta_{si}$ and $\phi_{sg} = \phi_{si}$. We can therefore recover the GPS location by solving the following least-squares minimization problem:

$$\min_{l,L} \sum_{k=1}^{N} \angle(\overrightarrow{s}(\theta_{sg}^{(k)}, \phi_{sg}^{(k)}), \overrightarrow{s}(\theta_{si}^{(k)}, \phi_{si}^{(k)}))^2 \ , \tag{A.4}$$

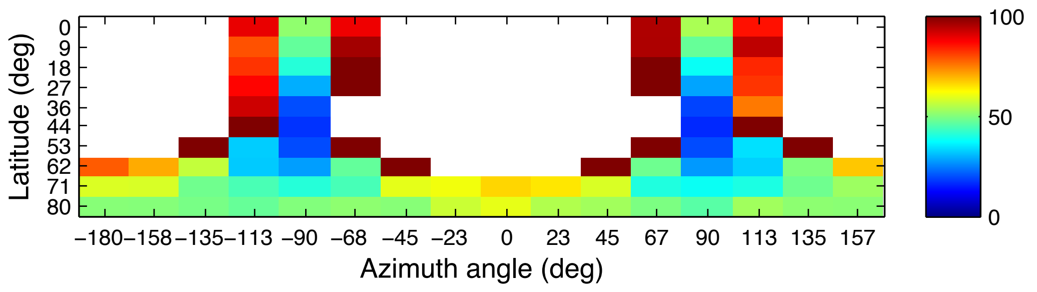
where N is the number of images where the sun has been labeled, $\angle(\cdot)$ denotes the angular difference, and $\overrightarrow{s}(\theta, \phi)$ is the vector obtained by expressing the angles (θ, ϕ) in Cartesian coordinates. A solution in l and L can be recovered using a non-linear least-squares optimizer. We have experimentally found that first minimizing the error on zenith angles $\angle(\theta_{sg}, \theta_{si})$ and using its solution to initialize (A.4) resulted in greater stability. This entire process is summarized in Algorithm 3.

A.1.2 Camera Localization Results

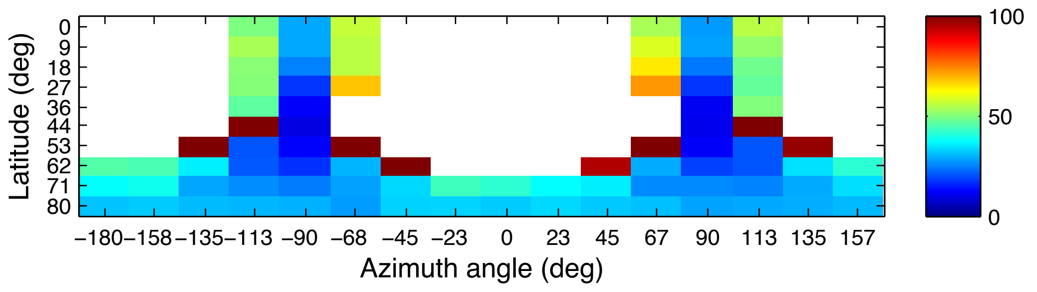
To evaluate the precision of our algorithm, we tested it over a large set of conditions using synthetic data, as well as on our ground truth sequence.

A.1.2.1 Synthetic Data

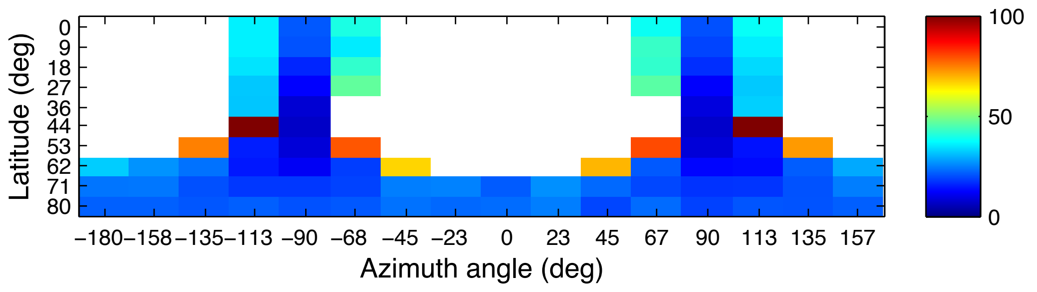
We evaluated the performance of our algorithm on synthetic data, obtained by varying the latitude l, longitude L, camera azimuth angle ϕ_c, and number of available images n. The resulting 4-dimensional parameter space was discretized the following way: 9° increments for both l and L, 23° increments for ϕ_c, and $n = 20, 50, 100$. For each point in the that parameter space, n sun positions are randomly generated according to (A.1) and (A.3) with



(a) $n = 20$



(b) $n = 50$



(c) $n = 100$

Figure A.1: Error in GPS coordinates estimation, in km, for different values of n: (a) 20, (b) 50, and (c). Since longitude does not seem to affect the precision of the results, the errors shown here are averaged over all longitudes. The white cells indicate configurations where the sun is never visible.

gaussian noise of variance $\sigma^2 = 5px$. Each experiment is repeated 15 times to account for randomness, and the mean over all these tries are reported. Fig. A.1 shows the errors (in km) obtained by our algorithm at every point in this parameter space. Since longitude does not seem to affect the precision of the results, the errors shown are averaged over all values of longitude. The white cells indicate configurations where the sun is never visible, so the GPS position cannot be recovered. When $n = 100$, the mean error is 25km.

Sequence name	Ground truth Latitude (°)	Ground truth Longitude (°)	Estimated Latitude (°)	Estimated Longitude (°)	Error (km)
257	38.97	-76.61	38.47	-76.40	58.44
279	43.32	-84.60	42.06	-84.13	145.82
513	40.20	-79.61	39.42	-80.61	122.25
524	40.41	-77.13	40.09	-77.66	56.72
569	44.00	-103.24	45.21	-103.54	136.65
601	32.69	-96.62	31.72	-95.50	150.78
630	28.98	-98.50	28.58	-97.79	81.53
695	36.60	-79.38	37.77	-79.25	129.84

Table A.1: Detail of localization results for 8 sequences taken from the AMOS dataset. On average, our method makes a localization error of 110km.

A.1.2.2 Ground Truth Results

We also applied our technique on 8 sequences from the AMOS database [Jacobs *et al.*, 2007a], where the ground truth GPS positions are known. We obtain a mean localization error of 110km (straight-line distance on the surface of the Earth), and the results for each individual sequence are shown in Table A.1. On average, each sequence was localized by using 48 images as input.

A.2 Calibrating the Camera From the Sun Position: Derivation of the Linear System of Equations

In Sec. 3.2, we presented an overview of the method employed to find an initial estimate of the camera parameters from the sun position gathered over several frames. For completeness, we now present all the details of the derivation.

Recall the following goal: we wish to recover the projection matrix $\mathbf{M}$, which we constrain to be of the form $\mathbf{M} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$, where $\mathbf{R}$ and $\mathbf{K}$ are defined in (3.3) and (3.4) respectively. Written explicitely, we have:

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} = \begin{bmatrix} f_c \sin\phi_c & -f_c \cos\phi_c & 0 & 0 \\ -f_c \cos\phi_c \cos\theta_c & -f_c \sin\phi_c \cos\theta_c & f_c \sin\theta_c & 0 \\ \cos\phi_c \sin\theta_c & \sin\phi_c \sin\theta_c & \cos\theta_c & 0 \end{bmatrix} . \tag{A.5}$$

Each observation is a pair of sun 3-D coordinates $\mathbf{p}$, and its corresponding location in an image (u_s, v_s). If $\mathbf{m}_i$ is the ith row of $\mathbf{M}$, then each observation defines two equations (following [Forsyth and Ponce, 2003]):

$$\begin{aligned} (\mathbf{m}_1 - u_s \mathbf{m}_3) \cdot \mathbf{s} = 0 \ \ , \\ (\mathbf{m}_2 - v_s \mathbf{m}_3) \cdot \mathbf{s} = 0 \ \ . \end{aligned} \tag{A.6}$$

If we have N such observations, we can write the linear system of equations from (A.6) directly in matrix notation with the form $\mathbf{Pm} = \mathbf{0}$:

$$\begin{bmatrix} x_w^{(1)} & y_w^{(1)} & 0 & 0 & 0 & -u_s^{(1)} x_w^{(1)} & -u_s^{(1)} z_w^{(1)} & -u_s^{(1)} z_w^{(1)} \\ 0 & 0 & x_w^{(1)} & y_w^{(1)} & z_w^{(1)} & -v_s^{(1)} x_w^{(1)} & -v_s^{(1)} z_w^{(1)} & -v_s^{(1)} z_w^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_w^{(N)} & y_w^{(N)} & 0 & 0 & 0 & -u_s^{(N)} x_w^{(N)} & -u_s^{(N)} z_w^{(N)} & -u_s^{(N)} z_w^{(N)} \\ 0 & 0 & x_w^{(N)} & y_w^{(N)} & z_w^{(N)} & -v_s^{(N)} x_w^{(N)} & -v_s^{(N)} z_w^{(N)} & -v_s^{(N)} z_w^{(N)} \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = \mathbf{0}. \tag{A.7}$$

This is a system of $2N$ equations and 8 unknowns. When $N \geq 4$, homogeneous linear least-squares can be used to compute the value of the vector $\mathbf{m}$ that minimizes $|\mathbf{Pm}|^2$ as the solution of an eigenvalue problem.

Because $\mathbf{M}$ is rank-deficient (rank = 2), it can only be recovered up to an unknown scale factor. However, observe that the third row in $\mathbf{M}$ must have unit length [Forsyth and Ponce, 2003], so we normalize $\mathbf{m}$ by $\epsilon = \pm\sqrt{m_{31}^2 + m_{32}^2 + m_{33}^3}$ before applying (A.8). After normalization, the camera parameters (f_c, θ_c, ϕ_c) can be recovered by:

$$\begin{aligned}
\theta_c &= \arctan\left(\frac{\sqrt{m_{31}^2 + m_{32}^2}}{m_{33}}\right) \\
f_c &= \sqrt{m_{11}^2 + m_{12}^2} \\
\phi_c &= \arctan\left(\frac{m_{11}}{-m_{12}}\right) \quad .
\end{aligned} \tag{A.8}$$

The sign of ϵ is chosen such that the points $\mathbf{s}$ lie in front of the camera (i.e., have positive x coordinates).

A.3 Expressing the Sky Model as a Function of Camera Parameters: Full Derivation

In Sec. 3.3.1.3, we presented a way to express the sky model as a function of camera parameters, which made the assumption that the camera zenith and azimuth angles were independent in order to come up with a simpler model. In this appendix, we derive the exact expressions for θ_p and ϕ_p, the zenith and azimuth angles corresponding to a pixel at coordinates (u_p, v_p) in the image, as illustrated in Fig. 3.5.

We first convert the (u_p, v_p) coordinates to a point $\mathbf{s}'$ in the 3-D camera reference frame $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$, and then rotate it to align it with the global reference frame $(\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w)$. The coordinates of the point in the camera reference frame are

$$\mathbf{s}' = \begin{bmatrix} x'_s \\ y'_s \\ z'_s \end{bmatrix} = \begin{bmatrix} f_c \\ -u_p \\ v_p \end{bmatrix} \quad . \tag{A.9}$$

The rotation that maps the point $\mathbf{s}'$ in the camera reference frame to a point $\mathbf{s}$ in the world reference frame is given by R^{-1}, where R has already been defined in (3.3). We apply the rotation to express the point in the world reference frame:

$$\mathbf{s} = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} = R^{-1}\mathbf{s}' \ . \tag{A.10}$$

Finally, the angles are obtained by converting into spherical coordinates:

$$\theta_p = \arccos\left(\frac{z_s}{\sqrt{x_s^2 + y_s^2 + z_s^2}}\right) , \ \ \phi_p = \arctan\left(\frac{y_s}{x_s}\right) \ . \tag{A.11}$$

We obtain the final, exact equations for θ_p and ϕ_p by substituting (A.9) into (A.10), and the resulting expression into (A.11):

$$\theta_p = \arccos\left(\frac{v_p \sin\theta_c + f_c \cos\theta_c}{\sqrt{f_c^2 + u_p^2 + v_p^2}}\right) \tag{A.12}$$

$$\phi_p = \arctan\left(\frac{f_c \sin\phi_c \sin\theta_c - u_p \cos\phi_c - v_p \sin\phi_c \cos\theta_c}{f_c \cos\phi_c \sin\theta_c + u_p \sin\phi_c - v_p \cos\phi_c \cos\theta_c}\right) \ . \tag{A.13}$$

A.4 Determination of Minimum Angular Difference for the Azimuth-independent Sky Model

In this appendix, we elaborate on the synthetic experiments that are performed in order to evaluate the conditions in which our azimuth-independent sky model (3.9) introduced in Sec. 3.3.1.2 is valid. As in Sec. 3.3, we consider only clear skies where turbidity $t = 2.17$ (see the first row of Fig. 3.6 for a visualization of the Perez sky model (3.7) at that particular turbidity).

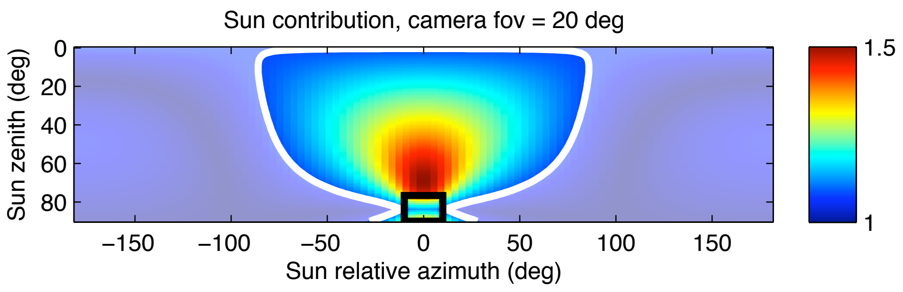
We proceed to evaluate the influence of the azimuth-dependent component of the Perez sky model (second factor in (3.7)). Our goal is to determine sun-camera configurations where that influence is mostly constant over the image. Given the field of view of the camera, we generate synthetic sky images over all possible sun positions. More precisely, we generate images that cover the sun zenith angle $\theta_s \in [0, 90^\circ]$, and the sun relative azimuth angle $\Delta\phi_s = \phi_s - \phi_c$ with respect to the camera $\Delta\phi_s \in [-180°, 180°]$. For each synthetic image, we then compute:

$$r = \frac{\max(c^*)}{\min(c^*)} \ , \tag{A.14}$$

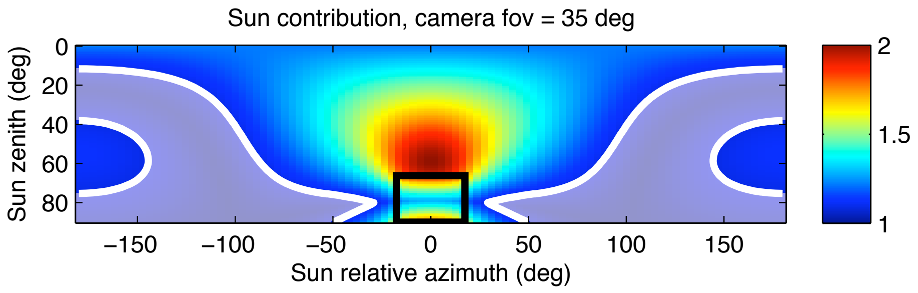
where c^* is the mean column in the image, computed over the visible sky region only. In other words, we summarize the effect of the sun on an image by a single value r, which captures how the sky columns are affected. When $r = 1$, the sun has no effect on the sky. Unfortunately, this is never the case, as the sun will *always* have some effect on the sky when it is clear. Therefore, we approximate that the sun has little effect when $r \leq 1.1$. In Fig. A.2, we plot r over the entire $(\theta_s, \Delta\phi_s)$ space. The white lines are the $r = 1.1$ isocontours, and the shaded regions indicate configurations where $r < 1.1$.

For the typical case of 35° field of view shown in Fig. A.2b, we can safely affirm that when the sun is at least 100° away from the camera field of view, then $r \leq 1.1$, except in a region located immediately behind the camera where it rises up to $r = 1.2$. When the field of view diminishes to 20° as in Fig. A.2a, then the number of sun-camera configurations where $r \leq 1.1$ is much larger, as the sun has to be closer to the camera to induce a noticeable influence on the sky appearance. The opposite effect is observed in the case of a larger field of view, as shown in Fig. A.2c.

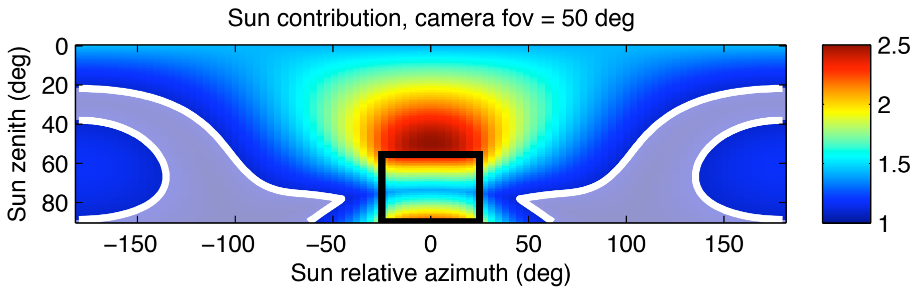
In conclusion, we used synthetic experiments to explore the validity of our azimuth-independent sky model (3.9), and we showed that for standard cameras, the sun has little influence on the sky when it is at least 100° away from the camera field of view.



(a)



(b)



(c)

Figure A.2: Synthetic experiments to evaluate the influence of the sun on the sky gradient, generated for different camera fields of view: (a) $20°$, (b) $35°$, and (c) $50°$. Each value is obtained by first synthesizing the sun-dependent component of the Perez sky model (second term in (3.7)) at the corresponding sun zenith θ_s and relative azimuth $(\Delta\phi_s)$ angles. The white lines are the isocontours corresponding to $r = 1.1$, and the shaded areas indicate that $r < 1.1$, where r is defined in (A.14). The black rectangles indicate the camera field of view.

Bibliography

[Adelson, 2006] Edward H. Adelson. personal communication, 2006.

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

[Arnold *et al.*, 2003] D. Arnold, A. Chalmers, F. Niccolucci, Jessi Stumpfel, Christopher Tchou, Nathan Yun, Timothy Hawkins, Andrew Jones, Brian Emerson, and Paul Debevec. Digital reunification of the parthenon and its sculptures. In *4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, 2003.

[Barnard and Finlayson, 2000] Kobus Barnard and Graham D. Finlayson. Shadow identification using colour ratios. In *Proc. IS&T/SID 8th Color Imaging Conf. Color Science, Systems and Applications*, 2000.

[Basri *et al.*, 2007] R. Basri, D.W. Jacobs, and I. Kemelmacher. Photometric stereo with general, unknown lighting. *International Journal on Computer Vision*, 72(3):239–257, May 2007.

[Berg *et al.*, 2004] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, Michael Maire, Ryan White, Yee-Whye Teh, Erik Learned-Miller, and D. A. Forsyth. Names and faces in the news. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[Biederman, 1981] Irving Biederman. On the semantics of a glance at a scene. In M. Kubovy and J. R. Pomerantz, editors, *Perceptual Organization*, chapter 8. Lawrence Erlbaum, 1981.

[Bird, 1984] R. E. Bird. A simple spectral model for direct normal and diffuse horizontal irradiance. *Solar Energy*, 32:461–471, 1984.

[Bitouk *et al.*, 2008] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter N. Belhumeur, and Shree K. Nayar. Face swapping: automatically replacing faces in photographs. *ACM Transactions on Graphics (SIGGRAPH 2008)*, 27(3), 2008.

[Blinn and Newell, 1976a] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. In *Proceedings of ACM SIGGRAPH 1976*, 1976.

[Blinn and Newell, 1976b] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10), 1976.

[Bousseau *et al.*, 2009] Adrien Bousseau, Sylvain Paris, and Frédo Durand. User-assisted intrinsic images. *ACM Transactions on Graphics (SIGGRAPH Asia 2009)*, 28(5), 2009.

[Boykov and Jolly, 2001] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *IEEE International Conference on Computer Vision*, 2001.

[Boykov and Kolmogorov, 2004] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 26(9), September 2004.

[Boykov *et al.*, 2001] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11), 2001.

[Boykov *et al.*, 2006] Yuri Boykov, Vladimir Kolmogorov, Daniel Cremers, and Andrew Delong. An integral solution to surface evolution PDEs via Geo-Cuts. In *European Conference on Computer Vision*, 2006.

[Buluswar and Draper, 2002] Shashi D. Buluswar and Bruce A. Draper. Color models for outdoor machine vision. *Computer Vision and Image Understanding*, 85(2):71–99, 2002.

[Campbell and Robson, 1968] F. W. Campbell and J. G. Robson. Application of Fourier analysis to the visibility of gratings. *Journal of Physiology*, 197:551–566, 1968.

[Canny, 1986] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[Cavanagh, 2005] Patrick Cavanagh. The artist as neuroscientist. *Nature*, 434:301–307, March 2005.

[Chakrabarti *et al.*, 2009] Ayan Chakrabarti, Daniel Scharstein, and Todd Zickler. An empirical camera model for internet color vision. In *British Machine Vision Conference*, 2009.

[Chang and Lin, 2001] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[Chang *et al.*, 2005] Youngha Chang, Suguru Saito, Keiji Uchikawa, and Masayuki Nakajima. Example-based color stylization of images. *ACM Transactions on Applied Perception*, 2(3):322–345, 2005.

[Chen *et al.*, 2000] H.F. Chen, P.N. Belhumeur, and D.W. Jacobs. In search of illumination invariants. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[Chen *et al.*, 2009] Tao Chen, Ming-Ming Chen, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics (SIGGRAPH Asia 2008)*, 28(5), 2009.

[Chong *et al.*, 2008] Hamilton Y. Chong, Steven J. Gortler, and Todd Zickler. A perception-based color space for illumination-invariant image processing. *ACM Transactions on Graphics (SIGGRAPH 2008)*, 2008.

[Chuang *et al.*, 2003] Yung-Yu Chuang, Dan B. Goldman, Brian Curless, David H. Salesin, and Richard Szeliski. Shadow matting and compositing. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3):494–500, July 2003.

[CIE, 1994] CIE. Spatial distribution of daylight – luminance distributions of various reference skies. Technical Report CIE-110-1994, International Commission on Illumination, 1994.

[Cohen-Or *et al.*, 2006] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. Color harmonization. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):624–630, 2006.

[Collins *et al.*, 2002] M. Collins, R. Shapire, and Yorav Singer. Logistic regression, adaboost and Bregman distances. *Machine Learning*, 48(1), 2002.

[Cozman and Krotkov, 1995] Fabio Cozman and Erik Krotkov. Robot localization using a computer vision sextant. In *IEEE International Conference on Robotics and Automation*, 1995.

[Criminisi *et al.*, 2000] Antonio Criminisi, I. Reid, and Andrew Zisserman. Single view metrology. *International Journal on Computer Vision*, 40(2):123–148, 2000.

[Cutzu *et al.*, 2003] Florin Cutzu, Riad Hammoud, and Alex Leykin. Estimating the photorealism of images: Distinguishing paintings from photographs. In *Proceedings of Computer Vision and Pattern Recognition*, 2003.

[Dalal and Triggs, 2005] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[Dale *et al.*, 2009] Kevin Dale, Micah K. Johnson, Kalyan Sunkavalli, Wojciech Matusik, and Hanspeter Pfister. Image restoration using online photo collections. In *International Conference on Computer Vision*, 2009.

[Debevec and Malik, 1997] Paul Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of ACM SIGGRAPH 1997*, August 1997.

[Debevec *et al.*, 1996] Paul Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH 1996*, 1996.

[Debevec *et al.*, 2002] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. A lighting reproduction approach to live-action compositing. *ACM Transactions on Graphics (SIGGRAPH 2002)*, 2002.

[Debevec *et al.*, 2004] Paul Debevec, Chris Tchou, Andrew Gardner, Tim Hawkins, Charis Poullis, Jessi Stumpfel, Andrew Jones, Nathaniel Yun, Per Einarsson, Therese Lundgren, Marcos Fajardo, and Philippe Martinez. Estimating surface reflectance properties of a complex scene under captured natural illumination. Technical Report ICT-TR-06.2004, USC ICT, 2004.

[Debevec, 1998] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH 1998*, pages 189–198, 1998.

[Diakopoulos *et al.*, 2004] Nicholas Diakopoulos, Irfan Essa, and Ramesh Jain. Content based image synthesis. In *Conference on Image and Video Retrieval*, 2004.

[Dobashi *et al.*, 1995] Y. Dobashi, T. Nishita, K. Kaneda, and H. Yamashita. Fast display method of sky color using basis functions. In *Pacific Conference on Computer Graphics and Applications*, 1995.

[Dollár *et al.*, 2009] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: a benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[Dror *et al.*, 2004] Ron O. Dror, Alan S. Willsky, and Edward H. Adelson. Statistical characterization of real-world illumination. *Journal of Vision*, 4:821–837, 2004.

[Efros and Freeman, 2001] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, August 2001.

[Elder *et al.*, 2004] James H. Elder, Sherry Trithart, Gregor Pintilie, and Donald MacLean. Rapid processing of cast and attached shadows. *Perception*, 33:1319–1338, 2004.

[Everingham *et al.*, 2006] Mark Everingham, Andrew Zisserman, Chris Williams, and Luc Van Gool. The pascal visual object classes challenge 2006 results. Technical report, Oxford University, 2006.

[Fei-Fei *et al.*, 2004] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision*, 2004.

[Felzenszwalb *et al.*, 2008] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[Felzenszwalb *et al.*, 2010] Pedro Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[Figov *et al.*, 2004] Zvi Figov, Yoram Tal, and Moshe Koppel. Detecting and removing shadows. In *Computer Graphics and Imaging*, 2004.

[Finlayson *et al.*, 1993] Graham D. Finlayson, Mark S. Drew, and Brian V. Funt. Diagonal transforms suffice for color constancy. In *International Conference on Computer Vision*, 1993.

[Finlayson *et al.*, 2001] Graham D. Finlayson, Steven D. Hordley, and Paul M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1209–1221, 2001.

[Finlayson *et al.*, 2002] Graham D. Finlayson, Steven D. Hordley, and Mark S. Drew. Removing shadows from images. In *European Conference on Computer Vision*, 2002.

[Finlayson *et al.*, 2004] Graham D. Finlayson, Mark S. Drew, and Cheng Lu. Intrinsic images by entropy minimization. In *European Conference on Computer Vision*, 2004.

[Finlayson *et al.*, 2006] Graham D. Finlayson, Steven D. Hordley, Cheng Lu, and Mark S. Drew. On the removal of shadows from images. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 28(1):59–68, 2006.

[Finlayson *et al.*, 2007] Graham D. Finlayson, Clément Fredembach, and Mark S. Drew. Detecting illumination in images. In *IEEE International Conference on Computer Vision*, 2007.

[Finlayson *et al.*, 2009] Graham D. Finlayson, Mark S. Drew, and Cheng Lu. Entropy minimization for shadow removal. *International Journal of Computer Vision*, 85, 2009.

[Forsyth and Ponce, 2003] David A. Forsyth and Jean Ponce. *Computer vision a modern approach.* Prentice Hall, 2003.

[Forsyth, 1990] David A. Forsyth. A novel algorithm for colour constancy. *International Journal of Computer Vision*, 5(1):5–36, July 1990.

[Fredembach and Finlayson, 2006] Clément Fredembach and Graham D. Finlayson. Simple shadow removal. In *International Conference on Pattern Recognition*, 2006.

[Freeman *et al.*, 2000] William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1), 2000.

[Goesele *et al.*, 2007] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *IEEE International Conference on Computer Vision*, 2007.

[Grossberg and Nayar, 2004] Michael D. Grossberg and Shree K. Nayar. Modeling the space of camera response functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1272–1282, Oct 2004.

[Haber *et al.*, 2009] Tom Haber, Christian Fuchs, Philippe Bekaert, Hans-Peter Seidel, Michael Goesele, and Hendrik P. A. Lensch. Relighting objects from image collections. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[Han *et al.*, 2010] Hu Han, Shiguang Shan, Laiyun Qing, Xilin Chen, and Wen Gao. Lighting aware preprocessing for face recognition across varying illumination. In *European Conference on Computer Vision*, 2010.

[Hasinoff and Kutulakos, 2008] Samuel W. Hasinoff and Kiriakos N. Kutulakos. Light-efficient photography. In *European Conference on Computer Vision*, 2008.

[Hasinoff *et al.*, 2010] Samuel W. Hasinoff, Frédo Durand, and William T. Freeman. Noise-optimal capture for high dynamic range photography. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[Hays and Efros, 2007] James Hays and Alexei A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.

[Hays and Efros, 2008] James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[Healey and Slater, 1999] Glenn Healey and David Slater. Models and methods for automated material identification in hyperspectral imagery acquired under unknown illumination and atmospheric conditions. *IEEE Transactions on Geoscience and Remote Sensing*, 37(6):2706–2717, November 1999.

[Hill, 1994] R. Hill. Theory of geolocation by light levels. In B. J. LeBouef and R. M. Laws, editors, *Elephant Seals: Population Ecology, Behavior, and Physiology*, chapter 12, pages 227–236. University of California Press, 1994.

[Hoiem *et al.*, 2005] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *IEEE International Conference on Computer Vision*, 2005.

[Hoiem *et al.*, 2006] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[Hoiem *et al.*, 2007] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, October 2007.

[Hoiem *et al.*, 2010] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering occlusion boundaries from an image. *International Journal of Computer Vision (to appear)*, 2010.

[Hoiem, 2007] Derek Hoiem. *Seeing the world behind the image: Spatial layout for 3D scene understanding*. PhD thesis, School of Computer Science, Carnegie Mellon University, August 2007.

[Horry *et al.*, 1997] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of ACM SIGGRAPH 1997*, 1997.

[Hsu *et al.*, 2008] Eugene Hsu, Tom Mertens, Sylvain Paris, Shai Avidan, and Frédo Durand. Light mixture estimation for spatially varying white balance. *ACM Transactions on Graphics (SIGGRAPH 2008)*, 27(3), 2008.

[Huang *et al.*, 2008] Xinyu Huang, Xianwang Wang, Jizhou Gao, and Ruigang Yang. Estimating pose and illumination direction for frontal face synthesis. In *Computer Vision and Pattern Recognition Workshops*, 2008.

[Huerta *et al.*, 2009] I. Huerta, M. Holte, T. Moeslund, and J. Gonzàlez. Detection and removal of chromatic moving shadows in surveillance scenarios. In *IEEE International Conference on Computer Vision*, 2009.

[Ineichen *et al.*, 1994] Pierre Ineichen, Benoît Molineaux, and Richard Perez. Sky luminance data validation: comparison of seven models with four data banks. *Solar Energy*, 52(4):337–346, 1994.

[Jacobs *et al.*, 2007a] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[Jacobs *et al.*, 2007b] Nathan Jacobs, Scott Satkin, Nathaniel Roman, Richard Speyer, and Robert Pless. Geolocating static cameras. In *IEEE International Conference on Computer Vision*, 2007.

[Jacobs *et al.*, 2008] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Toward fully automatic geo-location and geo-orientation of static outdoor cameras. In *Workshop on applications of computer vision*, 2008.

[Jacobs *et al.*, 2010] Nathan Jacobs, Brian Bies, and Robert Pless. Using cloud shadows to infer scene structure and camera calibration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[Jia *et al.*, 2006] Jiaya Jia, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Drag-and-drop pasting. *ACM Transactions on Graphics (SIGGRAPH 2006)*, 25(3):631–637, July 2006.

[Johnson and Farid, 2007] Micah K. Johnson and Hany Farid. Exposing digital forgeries in complex lighting environments. *IEEE Transactions on Information Forensics and Security*, 2(3):45–461, 2007.

[Johnson *et al.*, 2006] Matthew Johnson, Gabriel J. Brostow, Jamie Shotton, Ognjen Arandjelović, Vivek Kwatra, and Roberto Cipolla. Semantic photo synthesis. *Computer Graphics Forum (Proc. Eurographics)*, 25(3):407–413, 2006.

[Judd *et al.*, 1964] Deane B. Judd, David L. Macadam, Gütnter Wyszecki, H. W. Budde, H. R. Condit, S. T. Henderson, and J. L. Simonds. Spectral distribution of typical daylight as a function of correlated color temperature. *Journal of the Optical Society of America*, 54(8):1031–1036, 1964.

[Junejo and Foroosh, 2008] Imran N. Junejo and Hassan Foroosh. Estimating geo-temporal location of stationary cameras using shadow trajectories. In *European Conference on Computer Vision*, 2008.

[Kasten and Young, 1989] F. Kasten and A. T. Young. Revised optical air mass tables and approximation formula. *Applied optics*, 28, 1989.

[Kawakami *et al.*, 2005] Rei Kawakami, Katsushi Ikeuchi, and Robby T. Tan. Consistent surface color for texturing large objects in outdoor scenes. In *IEEE International Conference on Computer Vision*, 2005.

[Ke *et al.*, 2006] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[Kersten *et al.*, 1996] Daniel Kersten, David Knill, Pascal Mamassian, and I. Bulthoff. Illusory motion from shadows. *Nature*, 379(6560):31–31, 1996.

[Khan and Reinhard, 2005] Erum Arif Khan and Erik Reinhard. Evaluation of color spaces for edge classification in outdoor scenes. In *IEEE International Conference on Image Processing*, September 2005.

[Khan *et al.*, 2006] Erum Arif Khan, Erik Reinhard, Roland Fleming, and Heinrich Büelthoff. Image-based material editing. *ACM Transactions on Graphics (SIGGRAPH 2006)*, 25(3):654–663, August 2006.

[Kim and Hong, 2005] Taeone Kim and Ki-Sang Hong. A practical single image based approach for estimating illumination distribution from shadows. In *IEEE International Conference on Computer Vision*, 2005.

[Kim and Polleyfeys, 2008] Seon Joo Kim and Marc Polleyfeys. Robust radiometric calibration and vignetting correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4), April 2008.

[Kim *et al.*, 2008] Seon Joo Kim, Jan-Michael Frahm, and Marc Polleyfeys. Radiometric calibration with illumination change for outdoor scene analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[Kittler, 1967] R. Kittler. Standardisation of outdoor conditions for the calculation of daylight factor with clear skies. In *CIE Conference on Sunlight*, 1967.

[Koenderink *et al.*, 2004] J. J. Koenderink, A. J. van Doorn, and S. C. Pont. Light direction from shad(ow)ed random gaussian surfaces. *Perception*, 33(12):1405–1420, 2004.

[Kolmogorov and Boykov, 2005] Vladimir Kolmogorov and Yuri Boykov. What metrics can be approximated by Geo-Cuts, or global optimization of length/area and flux. In *IEEE International Conference on Computer Vision*, 2005.

[Kolmogorov and Zabih, 2004] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), February 2004.

[Koppal and Narasimhan, 2006] Sanjeev J. Koppal and Srinivasa G. Narasimhan. Clustering appearance for scene analysis. In *IEEE International Conference on Computer Vision*, 2006.

[Košecká and Zhang, 2002] Jana Košecká and Wei Zhang. Video compass. In *European Conference on Computer Vision*, 2002.

[Kuthirummal *et al.*, 2008] Sujit Kuthirummal, Aseem Agarwala, Dan B. Goldman, and Shree K. Nayar. Priors for large photo collections and what they reveal about cameras. In *European Conference on Computer Vision*, 2008.

[Lalonde and Efros, 2007] Jean-François Lalonde and Alexei A. Efros. Using color compatibility for assessing image realism. In *IEEE International Conference on Computer Vision*, October 2007.

[Lalonde and Efros, 2010] Jean-François Lalonde and Alexei A. Efros. Synthesizing environment maps from a single image. Technical Report CMU-RI-TR-10-24, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2010.

[Lalonde *et al.*, 2007] Jean-François Lalonde, Derek Hoiem, Alexei A. Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo Clip Art. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.

[Lalonde *et al.*, 2008a] Jean-François Lalonde, Srinivasa G. Narasimhan, and Alexei A. Efros. Camera parameters estimation from hand-labelled sun positions in image sequences. Technical Report CMU-RI-TR-08-32, Robotics Institute, Carnegie Mellon University, June 2008.

[Lalonde *et al.*, 2008b] Jean-François Lalonde, Srinivasa G. Narasimhan, and Alexei A. Efros. What does the sky tell us about the camera? In *European Conference on Computer Vision*, 2008.

[Lalonde *et al.*, 2009a] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Estimating natural illumination from a single outdoor image. In *IEEE International Conference on Computer Vision*, 2009.

[Lalonde *et al.*, 2009b] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Webcam clip art: Appearance and illuminant transfer from time-lapse sequences. *ACM Transactions on Graphics (SIGGRAPH Asia 2009)*, 28(5), December 2009.

[Lalonde *et al.*, 2009c] Jean-François Lalonde, Srinivasa G. Narasimhan, and Alexei A. Efros. The webcam clip art dataset. `http://graphics.cs.cmu.edu/projects/webcamdataset`, 2009.

[Lalonde *et al.*, 2010a] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Detecting ground shadows in outdoor consumer photographs. In *European Conference on Computer Vision*, 2010.

[Lalonde *et al.*, 2010b] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Ground shadow boundary dataset. `http://graphics.cs.cmu.edu/projects/shadows`, September 2010.

[Lalonde *et al.*, 2010c] Jean-François Lalonde, Srinivasa G. Narasimhan, and Alexei A. Efros. What do the sun and the sky tell us about the camera? *International Journal on Computer Vision*, 88(1):24–51, May 2010.

[Langer and Büelthoff, 2001] Michael S. Langer and Heinrich H. Büelthoff. A prior for global convexity in local shape-from-shading. *Perception*, 30(4):403–410, 2001.

[Lee *et al.*, 2001] Kuang-Chih Lee, Jeffrey Ho, and David Kriegman. Nine points of light: acquiring subspaces for face recognition under variable lighting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[Levin *et al.*, 2006] Anat Levin, Dani Lischinski, and Yair Weiss. A closed form solution to natural image matting. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2006.

[Li *et al.*, 2004] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3):303–308, August 2004.

[Lin *et al.*, 2004] Stephen Lin, Jinwei Gu, Shuntaro Yamazaki, and Heung-Yeung Shum. Radiometric calibration from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[Lotto and Purves, 2002] R. Beau Lotto and Dale Purves. The empirical basis of color perception. *Consciousness and Cognition*, 11(4):609–629, December 2002.

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

[Lyu and Farid, 2005] S. Lyu and H. Farid. How realistic is photorealistic? *IEEE Transactions on Signal Processing*, 53(2):845–850, 2005.

[Lyu *et al.*, 2004] S. Lyu, D. Rockmore, and H. Farid. A digital technique for art authentication. *Proceedings of the National Academy of Sciences*, 101(49):17006–17010, 2004.

[Manduchi, 2006] Roberto Manduchi. Learning outdoor color classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1713–1723, November 2006.

[Martin *et al.*, 2004] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5), May 2004.

[Matsushita *et al.*, 2004] Y. Matsushita, Ko Nishino, Katsushi Ikeuchi, and M. Sakauchi. Illumination normalization with time-dependent intrinsic images for video surveillance. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 26(10):1336–1347, October 2004.

[Maxwell *et al.*, 2008] Bruce A. Maxwell, Richard M. Friedhoff, and Casey A. Smith. A bi-illuminant dichromatic reflection model for understanding images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[Mills, 2004] D. Mills. Advances in solar thermal electricity and technology. *Solar Energy*, 76:19–31, January 2004.

[Mitsunaga and Nayar, 1999] Tomoo Mitsunaga and Shree K. Nayar. Radiometric self calibration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[Moses *et al.*, 1994] Yael Moses, Yael Adini, and Shimon Ullman. Face recognition: the problem of compensating for changes in illumination direction. In *European Conference on Computer Vision*, 1994.

[Narasimhan *et al.*, 2002] Srinivasa G. Narasimhan, C. Wang, and Shree K. Nayar. All images of an outdoor scene. In *European Conference on Computer Vision*, pages 148–162, May 2002.

[Narasimhan *et al.*, 2005] Srinivasa G. Narasimhan, Visvanathan Ramesh, and Shree K. Nayar. A class of photometric invariants: Separating material from shape and illumination. In *IEEE International Conference on Computer Vision*, 2005.

[Nimeroff *et al.*, 1996] Jeffry Nimeroff, Julie Dorsey, and Holly Rushmeier. Implementation and analysis of an image-based global illumination framework for animated environments. *IEEE Transactions on Visualization and Computer Graphics*, 2(4), December 1996.

[Nishino *et al.*, 2005] Ko Nishino, Peter N. Belhumeur, and Shree K. Nayar. Using eye reflections for face recognition under varying illumination. In *IEEE International Conference on Computer Vision*, 2005.

[Opperman *et al.*, 2009] Ingo Opperman, Simon Ferndriger, and Jörg Eugster. Touristic webcams worldwide. Accessed: 04/01/09, `http://www.webcams.travel`, 2009.

[Park *et al.*, 2010] Dennis Park, Deva Ramanan, and Charless C. Fowlkes. Multiresolution models for object detection. In *European Conference on Computer Vision*, 2010.

[Perez *et al.*, 1993] Richard Perez, Robert Seals, and Joseph Michalsky. All-weather model for sky luminance distribution – preliminary configuration and validation. *Solar Energy*, 50(3):235–245, March 1993.

[Pérez *et al.*, 2003] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3):313–318, 2003.

[Platt, 1999] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 1999.

[Pokrowski, 1929] G. I. Pokrowski. Über die Helligkeitsverteilung am Himmel. *Journal of Physics*, 30, 1929.

[Porter and Duff, 1984] Thomas Porter and Tom Duff. Compositing digital images. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, pages 253–259, July 1984.

[Preetham *et al.*, 1999] A. J. Preetham, Peter Shirley, and Brian Smits. A practical analytic model for daylight. In *Proceedings of ACM SIGGRAPH 1999*, August 1999.

[Ramamoorthi and Hanrahan, 2001] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of ACM SIGGRAPH 2001*, 2001.

[Ramamoorthi, 2002] Ravi Ramamoorthi. Analytic PCA construction for theoretical analysis of lighting variability in images of a lambertian object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1–12, October 2002.

[Reda and Andreas, 2005] Ibrahim Reda and Afshin Andreas. Solar position algorithm for solar radiation applications. Technical Report NREL/TP-560-34302, National Renewable Energy Laboratory, November 2005.

[Reinhard *et al.*, 2001] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications, special issue on Applied Perception*, 21(5):34–41, September - October 2001.

[Reinhard *et al.*, 2004] Erik Reinhard, Ahmet Oguz Akuyz, Mark Colbert, Charles E Hughes, and Matthew O'Connor. Real-time color blending of rendered and captured video. In *Interservice/Industry Training, Simulation and Education Conference*, December 2004.

[Reinhart *et al.*, 2006] C. F. Reinhart, J. Mardaljevic, and Z. Rogers. Dynamic daylight performance metrics for sustainable building design. *Leukos*, 3(1):1–25, 2006.

[Rensink and Cavanagh, 2004] Ronald A. Rensink and Patrick Cavanagh. The influence of cast shadows on visual search. *Perception*, 33:1339–1358, 2004.

[Romeiro and Zickler, 2010] Fabiano Romeiro and Todd Zickler. Blind reflectometry. In *European Conference on Computer Vision*, 2010.

[Rother *et al.*, 2004] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3):309–314, August 2004.

[Rother, 2007] Carsten Rother. Cut-and-paste for photo clip art. Technical Report MSR-TR-2007-45, Microsoft Research, 2007.

[Rubner *et al.*, 2000] Yossi Rubner, Carlo Tomasi, and Leonidas Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision and Image Understanding*, 2000.

[Russell *et al.*, 2006] Bryan C. Russell, Alexei A. Efros, Josef Sivic, William T. Freeman, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[Russell *et al.*, 2008] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

[Sato and Ikeuchi, 1995] Yoichi Sato and Katsushi Ikeuchi. Reflectance analysis under solar illumination. In *Proceedings of the IEEE Workshop on Physics-Based Modeling and Computer Vision*, pages 180–187, 1995.

[Sato *et al.*, 2003] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 25(3):290–300, March 2003.

[Savarese and Fei-Fei, 2007] Silvio Savarese and Li Fei-Fei. 3d generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision*, 2007.

[Schöld *et al.*, 2000] Arno Schöld, Richard Szeliski, David Salesin, and Irfan Essa. Video textures. In *ACM SIGGRAPH*, 2000.

[Shor and Lischinski, 2008] Yael Shor and Dani Lischinski. The shadow meets the mask: pyramid-based shadow removal. *Computer Graphics Forum Journal (Eurographics 2008)*, 27(2), 2008.

[Shotton *et al.*, 2006] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, May 2006.

[Sinha and Adelson, 1993] Pawan Sinha and Edward H. Adelson. Recovering reflectance and illumination in a world of painted polyhedra. In *IEEE International Conference on Computer Vision*, 1993.

[Slater and Healey, 1998] David Slater and Glenn Healey. Analyzing the spectral dimensionality of outdoor visible and near-infrared illumination functions. *Journal of the Optical Society of America*, 15(11):2913–2920, November 1998.

[Snavely *et al.*, 2006] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics (SIGGRAPH 2006)*, 25(3), 2006.

[Snavely *et al.*, 2008] Noah Snavely, Rahul Garg, Steven M. Seitz, and Richard Szeliski. Finding paths through the world's photos. *ACM Transactions on Graphics (SIGGRAPH 2008)*, 27(3):11–21, 2008.

[Stauffer, 1999] Chris Stauffer. Adaptive background mixture models for real-time tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[Stumpfel *et al.*, 2004] Jessi Stumpfel, Andrew Jones, Andreas Wenger, Chris Tchou, Tim Hawkins, and Paul Debevec. Direct HDR capture of the sun and sky. In *Proceedings of AFRIGRAPH*, 2004.

[Sun *et al.*, 2009] Mingxuan Sun, Grant Schindler, Greg Turk, and Frank Dellaert. Color matching and illumination estimation for urban scenes. In *IEEE International Workshop on 3-D Digital Imaging and Modeling*, 2009.

[Sunkavalli *et al.*, 2007] Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Factored time-lapse video. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.

[Sunkavalli *et al.*, 2008] Kalyan Sunkavalli, Fabiano Romeiro, Wojciech Matusik, Todd Zickler, and Hanspeter Pfister. What do color changes reveal about an outdoor scene? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[Tao *et al.*, 2009] Litian Tao, Lu Yuan, and Jian Sun. SkyFinder: attribute-based sky image search. *ACM Transactions on Graphics (SIGGRAPH 2009)*, 28(3), 2009.

[Tappen *et al.*, 2005] Marshall F. Tappen, William T. Freeman, and Edward H. Adelson. Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1459–1472, September 2005.

[Tian *et al.*, 2009] Jiandong Tian, Jing Sun, and Yandong Tang. Tricolor attenuation model for shadow detection. *IEEE Transactions on Image Processing*, 18(10), October 2009.

[Tomasi and Manduchi, 1998] Carlo Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the 6th International Conference on Computer Vision*, 1998.

[Torralba and Oliva, 2003] Antonio Torralba and Aude Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems*, 14(3):391–412, August 2003.

[Trebi-Ollennu *et al.*, 2001] Ashitey Trebi-Ollennu, Terry Huntsberger, Yang Cheng, E. T. Baumgartner, Brett Kennedy, and Paul Schenker. Design and analysis of a sun sensor for planetary rover absolute heading detection. *IEEE Transactions on Robotics and Automation*, 17(6):939–947, December 2001.

[Tsin *et al.*, 2001] Yanghai Tsin, Robert T. Collins, Visvanathan Ramesh, and Takeo Kanade. Bayesian color constancy for outdoor object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[von Ahn *et al.*, 2006] Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: A game for locating objects in images. In *ACM Conference on Human Factors in Computing Systems*, 2006.

[Wang and Cohen, 2006] Jue Wang and Michael Cohen. Simultaneous matting and compositing. Technical Report MSR-TR-2006-63, Microsoft Research, May 2006.

[Ward, 1994] Greg Ward. The RADIANCE lighting simulation and rendering system. In *Proceedings of ACM SIGGRAPH 1994*, 1994.

[Weiss, 2001] Yair Weiss. Deriving intrinsic images from image sequences. In *IEEE International Conference on Computer Vision*, 2001.

[Wu and Tang, 2005] Tai-Pang Wu and Chi-Keung Tang. A bayesian approach for shadow extraction from a single image. In *IEEE International Conference on Computer Vision*, 2005.

[Yang *et al.*, 2007] Liu Yang, Rong Jin, Caroline Pantofaru, and Rahul Sukthankar. Discriminative cluster refinement: Improving object category recognition given limited training data. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[Yu and Malik, 1998] Yizhou Yu and Jitendra Malik. Recovering photometric properties of architectural scenes from photographs. In *Proceedings of ACM SIGGRAPH 1998*, July 1998.

[Zheng *et al.*, 2006] Yuanjie Zheng, Stephen Lin, and Sing Bing Kang. Single-image vignetting correction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[Zhu *et al.*, 2010] Jiejie Zhu, Kegan G. G. Samuel, Syed Z. Masood, and Marshall F. Tappen. Learning to recognize shadows in monochromatic natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[Zickler *et al.*, 2006] Todd Zickler, Satya P. Mallick, David J. Kriegman, and Peter N. Belhumeur. Color subspaces as photometric invariants. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.