Real-time Recognition of Shadow from Deep Edge Detection

by

Sepideh Hosseinzadeh Heydarabad

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this work we address the problem of fast shadow detection from single images of natural scenes. Different from traditional methods that employ expensive optimization methods, we propose a fast semantic-aware Convolutional Neural Network learning framework which trains on different kinds of patches, while integrating semantic shadow information. We primarily cluster pixels based on their material similarities, then in addition to considering individual regions separately, we exploit a higher level interactions between the neighbouring regions. We process the shadow edge pixels between the segments, and relate the regions together.

Keywords: fast shadow detection, deep learning, shadow semantic information.

Acknowledgements

First and foremost I would like to thank my supervisor, Prof. Hong Zhang, for his assistance and feedback during the past few months. I also would like to thank Mr. Shakeri, for his beneficial feedback during this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Lighting and Shading Concepts

In this section, we are going to review lighting and shading concepts.

In real scenes, there is a variation of shading over object surfaces caused by surface material properties,orientation of surfaces, nature and direction of light sources, view direction and shadows.

1.1.1 What is Shadow?

Shadow occurs when an object totally or partially blocks light directly from the light source. Shadows can be divided into two categories: cast and self (Figure 1.1). A cast shadow is projected by the object in the direction of the light source; a self shadow is the part of the object which is not illuminated by direct light. The part of a cast shadow where direct light is completely blocked by its object is called umbra, while the part where direct light is partially blocked is called penumbra. Self and cast shadows produce different brightness values. Self shadows usually have a higher brightness than cast shadows since they receive more secondary lighting from surrounding illuminated objects. Cast shadows can, however, cause a significant reduction in spectral variation thereby causing correlation failure.

1.1.2 Ambient Illumination

The simplest kind of shading is that from ambient illumination, that is, light that comes uniformly from all directions. The radiated light intensity I at a

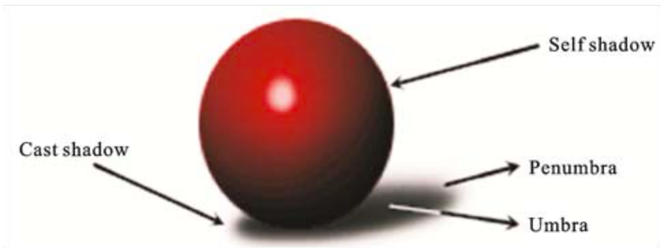

Figure 1.1: Shadow types

point on a surface depends on the intensity of the illumination I_a, and on the reflectivity k_a (or albedo) of the surfacethe fraction of the incoming light which the object reflects, near zero for black objects, near one for white objects. Thus we have $I = I_a k_a$.

Ambient illumination is mathematically an extended form of Lambertian reflection, integrating contributions from an infinite number of infinitesimal point light sources in all directions, instead of a single point light source.

1.1.3 Lambertian (Diffuse) Reflection

When a ray of light hits a surface, some fraction of it penetrates some way into the body of the object, where it is scattered (and may interact with coloured pigment particles). Eventually, some of the light is reradiated more or less uniformly in all directions. For a given surface, the brightness depends only on the angle θ between the direction $\bar{L}$ to the light source and the surface normal $\bar{N}$ (Figure 1.2). In this model, the brightness depends only on the angle θ between the direction $\bar{L}$ to the light source and the surface normal $\bar{N}$. This model is called Lambertian reflection (it is also called matte, diffuse or body reflection).

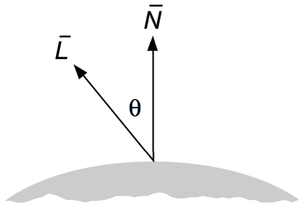

Figure 1.2: Lambertian (Diffuse) Reflection

The intensity of light re-radiated from a small patch of surface depends on the intensity I_p of the incoming light from the point light source, on how much of this light is intercepted by the surface patch, and on the reflectivity k_d (or albedo) of the surface.

If the surface patch is facing full on to the light source, then it will intercept the maximum amount of light. As the patch turns away from the light, it will intercept less of the light, following a cosine law, cos θ, where θ is the angle between the local surface normal, and the direction to the light source. Therefore, the diffuse (or Lambertian) illumination equation is:

$$I = I_p k_d cos\theta$$

This cosine can be expressed as a scalar product, thus the Lambertian contribution to the total intensity is:

$$I = I_p k_d \bar{N}.\bar{L}$$

Where $\bar{L}$ and $\bar{N}$ are unit vectors in the directions, respectively, of the light source and of the surface normal.

1.1.4 Specular Reflection

When a ray of light hits a surface, some fraction of it is also reflected immediately at the outer boundary of the surface. This is the specular reflection and leads to highlights and glossiness.

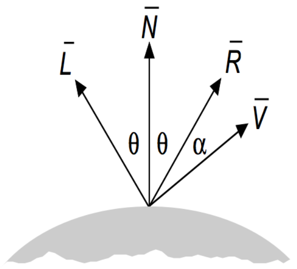

Figure 1.3: Specular Reflection

If the surface were a perfect mirror, then the reflection would follow the law of perfect reflection: For an incident ray of light from the light source, the emergent reflected ray would lie in the plane defined by the incident ray and the surface normal, and make the same angle with the surface normal as the incident ray. For most glossy surfaces, however, the reflected light is spread out (e.g. scratches in steel of texture in plastic), to a greater or lesser degree, from the direction of perfect reflection. This is caused by microscopic unevenness of the surface: there are a lot of little reflecting facets, whose normals vary from the overall surface normal. The reflection is strongest in the direction of perfect reflection, and becomes weaker for directions away from this.

This spread of reflection is modelled by looking at the angle α between the direction of perfect reflection and the viewer direction, and modify the reflected intensity by the factor $(cos\alpha)^n$ (Figure 1.3). $(cos\alpha)^n$ is at its maximum, 1, when the viewer direction coincides with the direction of perfect reflection, and becomes less for directions away from this. The exponent n is the specular reflection exponent and controls the degree of spread. High values of n lead to a rapid fall-off and sharp highlights, corresponding to a very glossy surface, almost like a mirror.

1.2 Introduction

Shadow detection is important in analyzing the nature scenes. Recently, shadow information is used in tasks related to object shape, size, movement, number of light sources and illumination conditions, etc. However, beside its useful information, shadows are a great concern in many computer vision and robotic applications such as object tracking, image segmentation, object recognition, and road detection (Figure 1.4). Moreover, being able to detect the shadow and remove it, can help augmented reality, image editing, and computational photography. Despite the fact that shadow detection is studied for a long time and its importance, it remains a challenging problem. Complex interactions of geometry, albedo, and illumination in nature scenes make the detection difficult. We cannot distinguish between a dark surface due to shading and a dark surface due to albedo. To detect the shadow region, we should consider the color, intensity, and texture of the pixels.

Since shadow is a stochastic phenomenon due to complex interplay of geometry, albedo, and illumination, finding a unified approach for shadow detection is difficult. Recent approaches motivated by this observation, and exploit learning techniques for this task [21, 61, 29, 19, 16, 25]. These methods work in a patch-wise way, and compute the likelihood that center pixel of the patch is on the edge of a shadow. One of the limitations of these approaches is that they make local prediction for each pixel independently, despite of high dependency of shadow edges in local patches. So, these predictions combined using CRF, GBP or MRF.

As most of the researches are inspired by structure of human body, we need

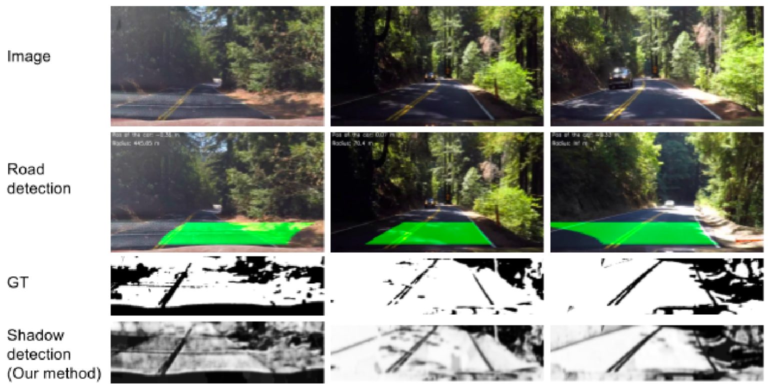

Figure 1.4: Shadows cause complications in road detection application. We can use shadow information, or detect and remove it from road images.

to equip machines with the same visual comprehension abilities. Inspired by the hierarchical architecture of the human visual cortex, recently, many tasks are using deep learning representations. We are motivated by recent success of these methods in various computer vision methods, and used a deep representation architecture [25, 51, 55].

Presence of shadow does not have any schema. Thus, there is no simple unified learning model that can indicate shadow. This observation brings the idea of using different representations beside RGB images, to improve the detection of shadow. Recently, Vicente et al. [55] used Fully Convolutional Neural Network (FCN) probability map beside RGB image, and train a Convolutional Neural Network (CNN) for detecting shadow patches in images. Disadvantages of this method are first, using FCN probability map which is a really poor map due to the fact that its prediction is based on each pixel's color. However, the material of the pixel such as texture is also an important factor in shadow detection [16]. Moreover, this method is processing all the pixels over and over, and the time complexity of producing the results is high and non-applicable.

1.3 Contributions of this Work

Our goal is to detect shadows of an image in a real-time manner. Previous works used optimization frameworks to refine the shadow detection results,

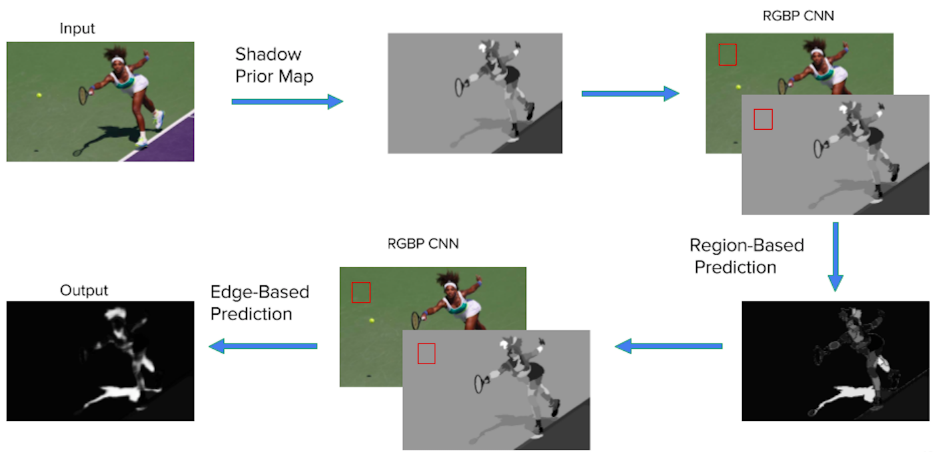

Figure 1.5: Our shadow detection method pipeline

which result in using and optimizing many parameters [51, 16, 25]. We utilize a CNN framework which is designed to capture the local structure information of shadow edges and automatically learn the most relevant features.

Shadow is a haphazard phenomenon, so learning frameworks need extra information for learning it. We employ an intelligent image level shadow prior map beside RGB image. The prior map is a segmented probability map which similar pixels clustered with respect to their color and texture differences.

The procedure for detecting shadow is as follows (Figure 1.5). First, we produced a shadow prior map and attached it to RGB image. Second, since pixels with the same material (color and texture) probably have the same prediction of shadiness, we first group the pixels with the same material together, and treat similarly each pixel in the same region. In the next step, we specify each region's confidence to be a shadow region by CNN (region-based prediction). At last, predictions made by the CNN are local and we need to utilize a higher level interactions between the neighbouring region pixels. We process the edge pixels between the segments by CNN (edge-based prediction).

Our main contributions are as follows:

1. A new real-time framework for robust shadow detection integrating both regional and across-boundary learned features.

2. Automatic learning of the most relevant feature representations from

RGB images and shadow prior map using CNN framework.

Beside our contributions, we performed a vast quantitative evaluation to prove that the proposed method is fast and applicable, and it is also robust and generalisable among various kinds of scenes.

1.4 Related Work

Related works in shadow detection can be categorized into four groups: (1) physical modelling of illumination and color, (2) statistical learning based approaches, (3) data-driven learning approaches are proposed for single-image, and (4) Deep learning frameworks.

1.4.1 Physical Modelling of Illumination and Color

Early methods proposed with a focus on physical modelling of illumination and color [30, 47, 13, 39, 14]. For example, Finlayson et al. [13] compare edges in the original image to edges in its illumination-invariant image. The problem of this approach is necessity of high-quality images [29].

1.4.2 Statistical Learning Approaches

To adapt the environment changes, statistical learning based approaches [45, 36, 21, 18] are proposed. They learn shadow model parameters at each pixel from a video. Some methods use shadow-variant and shadow-invariant texture and color related cues to capture the statistical properties of shadows [61, 29, 20, 16, 48]. The extracted features model the color, texture [61, 29, 16, 48] and illumination [20, 44]. Their weakness are requirement of a video, and their high time-complexity of optimizing parameters.

1.4.3 Data-driven Learning Approaches

To improve robustness, and get rid of need for videos, recently, some data-driven learning approaches are proposed for single-image shadow detection, learning to detect shadows based on training images. They use hand-crafted features as an input. Lalonde et al. [29] detected cast shadow edges on the ground with a classifier trained on local hand-crafted features. [61] proposed a similar approach for monochromatic images. Every pixel is classified as shadow or non-shadow. This method's constraint is that the pixel-wise outputs are noisy with poor-quality contour sequence. To address this issue, the predictions combined using CRF, GBP or MRF. Guo et al. [16] modelled long-range interaction between pairs of regions of the same material, with two types of pairwise classifiers, same/different illumination condition. Then, they incorporated the pairwise classifier and a shadow region classifier into a CRF graph-cut optimization. Similarly, Vicente et al. [56] modelled unary and pair-wise region classifier and a shadow boundary classifier, then used MRF to combine the classifiers. Regardless of good accuracy of these approaches, their limitation is requirement of extensive ground-truth annotation.

1.4.4 Deep Learning Frameworks

At last but not least, deep learning frameworks are proposed to learn the most relevant features for shadow detection automatically, and outperformed all other methods that use hand-crafted features. The first deep method for detecting shadows was proposed by Khan et al. [25]. They combined a CNN for shadow patches and a CNN for shadow boundaries with a CRF. Vicente et al. [57] optimized a multi-kernel model for shadow detection based on leave-one-out estimates. Shen et al. [51] proposed a CNN for structured shadow edge prediction. One of the constraints of these approaches is that they use optimization of shadow and illumination models which make the time complexity high, and non-applicable.

1.5 Applications

Aerial images are contaminated with shadow caused by buildings, trees and bridges, etc. As one of the features in aerial image, shadow can provide geometric and semantic clues about the shape and height of its casting object and the position of the light source. On the other hand, shadow can be treated as a special kind of image degradation. Shadow information used in tasks related to object shape [38, 43], size, movement [24], number of light sources and illumination conditions [49], and camera parameters and geo-location [23].

Shadow is one of the major problems in remotely sensed imagery which decreases the accuracy of information extraction and change detection. Hiroyuki et al [41] eliminated shadow areas in an earthquake zone for earthquake damage assessment. Cloud-free remotely sensed images acquired from earth orbiting satellites. Therefore, Song et al [54] detected clouds and shadows in order to reduce them in images. In another study, Dozier [12] discriminated snow from other materials in shadow. Shettigara et al [52] thresholded SPOT (System for Earth Observation) images to extract shadows of extended objects (i.e. buildings and trees) for determining their heights. Cheng et al [8] delimited the building heights in a city from the shadow in SPOT images. Chen et al. [7] recovered shadow information in urban areas from very high resolution satellite imagery. Zhu et al [62] presented cloud shadow detection in Landsat imagery. [59, 33] methods proposed for shadow detection of urban aerial images.

Foreground detection is an important early vision task in visual surveillance systems. The presence of moving cast shadows on the background makes it difficult to estimate shape or behavior of moving objects in surveillance videos. Csaba et al [2] detected shadow in surveillance videos. Schreer et al [50] proposed a method for shadow detection in videoconference applications. Bevilacqua [4] proposed an algorithm to detect moving shadows in the context of an outdoor traffic scene. [32, 11, 60] proposed a shadow elimination approach in video-surveillance. [26, 40, 27, 10] methods proposed for traffic flow analysis.

Shadows are a big concern in object detection application. Researchers approach this problem in two different ways. First, detecting the shadow and remove/suppress it from images [11, 17, 58]. Second, formulating the object and its casted shadow together, to localize the object [42].

In other studies, Kumar et al [28] proposed a gesture-based input interface system that utilizes shadow detection and is capable of providing robust, real-time operation in a low-cost manner. Shoaib et al. [53] presented a moving human cast shadow detection technique.

In conclusion, shadow information is used in numerous applications, on the other hand, shadows cause complications in other applications, so detecting and removing them from images can be helpful.

1.6 Outline

Our method consists of two parts, training and testing. The training part (chapter 2) consist of: (1) outline of the algorithm, (2) image level semantic information (prior probability map) we used beside the image to train the CNN, (3) structure of the utilized CNN, and (4) more information on patches for training.

The testing part (chapter 3) includes: (1) outline of algorithm, (2) region-based prediction, and (3) edge-based prediction.

At last, we show our experiments (chapter 4): (1) methods of evaluation, (2) datasets, (3) quantitative results, (4) conclusions, and (5) further work.

Chapter 2

Training a Patch-Wise CNN with 4D Channel Input

In this chapter, we will talk about: (1) how we produced the image level shadow prior map, (2) used CNN architecture, and (3) the way we selected the training patches. Each subsection is described in detail with literature review.

2.1 Training of the Patch-Wise CNN Algorithm

In this section we illustrate the steps of the training of the patch-wise CNN algorithm. The input of the algorithm is a RGB train image, and the output is a Convolutional Neural Network (CNN) trained on shadow/non-shadow patches of the image, which is capable of predicting the probability of being shadow for each pixel of the patch image. Steps of the algorithm are as follows:

1. Predict an image level shadow prior map.

2. The predicted map is attached to the RGB image as an additional channel. (shadow prior channel P).

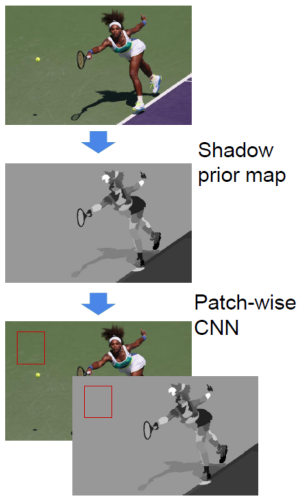

Figure 2.1: Training framework.

3. Training a CNN on RGBP patches to predict
   local shadow and non-shadow pixels (patch-wise-CNN).

2.2 Image Level Shadow Prior Map

In this section, we will explain about the way we produce the image level shadow prior map which is used beside RGB image to detect the shadows.

This section includes (1) outline of the algorithm, (2) algorithm of segmenting the image, (3) how we represent the colors, and (4) texture of the image as features to train (5) the SVM classifier, in order to predict the confidence of shadiness for each segment.

As human visual system considers not only the appearance of a local region but also that of its neighboring regions to determine whether the local region is overshadowed, we inspired from human visual system, and performed in the same way [16]. When a region is covered by shadow, it becomes darker and less textured than their surrounding regions with similar material (texture and chromaticity) [61]. In more detail, if two regions have similar material and intensity, then they probably have a similar label (shadow/non-shadow). However, if two regions have the same material and different intensity, the region with darker intensity probably should be labeled as shadow, and the other region non-shadow.

The first step to utilize this idea to detect shadows, is to group the pixels with similar material, and compare the neighboring regions together. Therefore, we segment the image using Mean Shift algorithm [9]. Then, we estimate the confidence of each region for being in shadow, using a trained classifier.

We utilize such a shadow prior map due to necessity of producing a prior map in real-time manner. Therefore, using segmentation or super-pixel (clusters of similar pixels) which a simple classification algorithm applied on each cluster, can be a fast solution.

2.2.1 Producing an Image Level Shadow Prior Map Algorithm

The goal of this section is illustrating the steps of producing an image level shadow prior map algorithm. The input of the algorithm is a RGB image, and the output is an image level shadow prior map which the value of each pixel is confidence of being shadiness of that pixel. Steps of the algorithm are as follows:

1. Segmentation of the image.
2. Representation of color with a histogram in L*a*b color space representation (Section 2.2.3), with 21 bins per channel.
3. Representation of texture with the texton histogram, with 128 textons (Section 2.2.4).
4. Training our classifier from manually labeled regions using a SVM (Section 2.2.5).
5. Definition of confidence of region i, as the output of this classifier times the pixel area of the region i.

2.2.2 Image Segmentation

The aim of this section is explaining about how we segment the image. We segment the image using Mean Shift algorithm [9]. Based on a recent research that compares different segmentation algorithms [34], Mean Shift has several properties that make it popular. First, it preserve the boundaries better than other methods. Second, its speed has been improved significantly in recent years. Third, there are fewer parameters to tune. At last, it does not require the number of clusters to be selected, but it has its own parameters that also control the number of and sizes of the regions.

2.2.2.1 Introduction to Mean Shift Algorithm

Mean Shift algorithm considers feature space as an empirical probability density function. If the input is a set of points, then Mean Shift algorithm considers them as sampled from the underlying probability density function. If dense regions (or clusters) are present in the feature space, then they correspond to the mode (or local maxima) of the probability density function. We can also identify clusters associated with the given mode using Mean Shift.

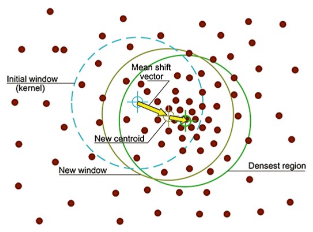

Figure 2.2: Mean Shift

For each data point, Mean Shift associates it with the nearby peak of the dataset's probability density function. For each data point, Mean Shift defines a window around it and computes the mean of the data point. Then, it shifts the center of the window to the mean and repeats the algorithm until it converges. After each iteration, we can consider that the window shifts to a more denser region of the dataset.

At the high level, we can specify Mean Shift algorithm as follows (Figure 2.2):

1. Fix a window around each data point.
2. Compute the mean of data within the window.

3. Shift the window to the mean and repeat until convergence.

2.2.2.2 Introduction to Kernel Functions

Mean Shift is a kernel based algorithm. A kernel is a function that satisfies the two requirements including $\int_{R^d} \phi(x) = 1$ and $\phi(x) \geq 0$. Some popular examples of kernels are Epanechnikov: $\phi(x) = \begin{cases} \frac{3}{4}(1-x^2) & \text{if } |x| \leq 1 \\ 0 & \text{Otherwise} \end{cases}$ and Gaussian: $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$.

2.2.2.3 Kernel Density Estimation

Kernel density estimation is a non parametric way to estimate the density function of a random variable. This is usually called as the Parzen window technique. Given a kernel K, bandwidth parameter h, kernel density estimator for a given set of d-dimensional points is

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

2.2.2.4 Gradient Ascent of Mean Shift

Mean Shift can be considered to be based on gradient ascent on the density contour. The generic formula for gradient ascent is $x_1 = x_0 + \eta f'(x_0)$. Applying it to kernel density estimator,

$$\bigtriangledown \hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K'(\frac{x - x_i}{h})$$

setting it to zero we get,

$$\sum_{i=1}^{n} K'(\frac{x - x_i}{h})\vec{x} = \sum_{i=1}^{n} K'(\frac{x - x_i}{h})\vec{x_i}$$

finally we get,

$$\vec{x} = \frac{\sum_{i=1}^{n} K'(\frac{x-x_i}{h})\vec{x_i}}{\sum_{i=1}^{n} K'(\frac{x-x_i}{h})}$$

2.2.2.5 Mean Shift

Mean Shift treats the points of the feature space as a probability density function. Dense regions in feature space correspond to local maxima or modes.

So for each data point, we perform gradient ascent on the local estimated density until convergence. The stationary points obtained via gradient ascent represent the modes of the density function. All points associated with the same stationary point belong to the same cluster. Assuming $g(x) = -K'(x)$, we have

$$m(x) = \frac{\sum_{i=1}^{n} g(\frac{x-x_i}{h})x_i}{\sum_{i=1}^{n} g(\frac{x-x_i}{h})} - x$$

The quantity $m(x)$ is called as the Mean Shift. So Mean Shift procedure can be summarized as:
For each point x_i:

1. Compute Mean Shift vector $m(x_i^t)$.
2. Move the density estimation window by $m(x_i^t)$.
3. Repeat until convergence.

For every kernel function used in Mean Shift algorithm, convergence occurs when the function satisfies the condition $m(x_i^{t+1}) \geq m(x_i^t)$.

2.2.2.6 Mean Shift Segmentation

The Mean Shift segmentation algorithm is as follows:
Let x_i and z_i, $i = 1, ..., n$, be the d-dimensional input and filtered image pixels in the joint spatial-range domain and L_i the label of ith pixel in the segmented image.

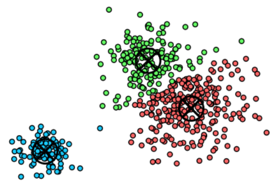
Figure 2.3: Clusters of similar points, after Mean Shift segmentation

1. Run the Mean Shift filtering procedure for the image and store all the information about the d-dimensional convergence point in z_i, i.e., $z_i = y_{i,c}$.
2. Delineate in the joint domain the clusters $\{C_p\}_{p=1,...,m}$ by grouping together all z_i which are closer than h_s in the spatial domain and h_r in the range domain, i.e., concatenate the basins of attraction of the corresponding convergence points.

3. For each $i = 1, ..., n$, assign $L_i = \{p | z_i \in C_p\}$.

4. (Optional) Eliminate spatial regions containing less than M pixels.

In this algorithm, h_s is spatial resolution parameter which affects the smoothing, connectivity of segments. It is chosen depending on the size of the image and objects. h_r is range resolution parameter which affects the number of segments. It should be kept low if contrast is low. At last, M is size of smallest segment. It should be chosen based on size of noisy patches.

2.2.3 Color Representation

In this section, we describe the way we represent color of each pixel of the image, as a feature to input into SVM classifier to classify regions of the image.

We used The CIELAB (LAB or L*a*b) color space representation (Figure 2.4). Color space defined by the CIE, based on one channel for Luminance (lightness) (**L**) and two color channels (**a** and **b**). In this model, the color differences are related to colorimetric measurement. The **a** axis from green (-**a**) to red (+**a**) and the **b** axis from blue (-**b**) to yellow (+**b**) develops. The Luminance (**L**) brighten from the bottom to the top of this 3D model. After representing the color in CIELAB space, we use histogram of them with 21 bins per channel.

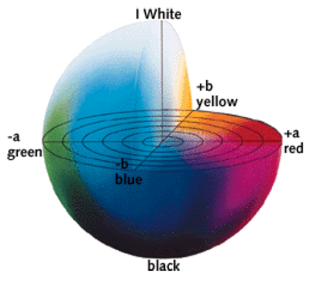

Figure 2.4: The CIELAB color space representation[2]

2.2.4 Texture Representation

In this section, we illustrate how we represent texture of the image, as a feature to input into SVM classifier to classify regions of the image.

Based on [35], the spectral histogram consists of marginal distributions of responses of a bank of filters and encodes implicitly the local structure of

[2]www.linocolor.com

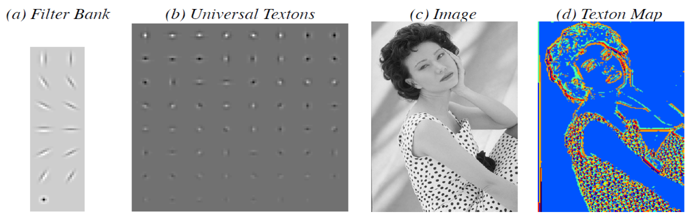

Figure 2.5: (a) The 13-element filter bank which is applied on each pixel, it is used for computing textons. (b) Instance universal textons produced from 200 images.(c) Image. (d) Texton map. For each pixel, a 13-element filter responses are produced, and they are clustered with k-means. In this example, with 200 images k=64 results in 64 universal textons. Each pixel is assigned to the nearest texton. Each texton has a different color.

images through the filtering stage and the global appearance through the histogram stage. They reveal that the spectral histogram representation provides a robust feature statistic for textures and generalizes well.

We illustrate texture with the texton histogram, with 128 universal textons provided by Martin et al. [37]. These universal textons are computed using a large and diverse image dataset.

For producing the textons, a gradient-based model is used by Martin et al. [37]. It is for indicating local changes of texture. At location (x, y), a circle with radius r is drawn, and is divided into half along the diameter with orientation θ. The gradient function $G(x, y, r, \theta)$ compares the two halves of this circle. G measures the degree to which texture of scale r differs at point (x, y) of the image in direction θ. High difference means dissimilarity along the diameter in the image. The gradient of texture is computed in 8 directions and 3 half-octave scales at each pixel.

For comparing two halves of the circle, first the pixels of each half should be convolved with different orientations of a bank of filters. Next, we compute histograms of vector quantized bank filter responses for each pixel. At last, for comparing two histograms of each half circle, with χ^2 histogram difference operator introduced by Puzicha et al. [46]:

$$\chi^2(g,h) = \frac{1}{2}\sum \frac{(g_i - h_i)^2}{g_i + h_i} \tag{2.1}$$

Figure 2.5 shows the filter bank used in producing the textons. It includes six pairs of oriented filters, and a center-surround filter. The oriented filters are Gaussian second derivative which is an even symmetric filter, and Hilbert Transform which is an odd symmetric filter. The center-surround filter is a difference of Gaussian (DoG). Each filter is applied to each pixel, and it produces a 13 elements length vector. Therefore, each pixel has a 13 elements length vector associated with it.

Each half of the circle contains several filter response vectors. These vectors are clustered using k-means algorithm. Cluster centers are *Textons*, which are linear combination of the filters.

After defining the textons, each pixel is assigned to its nearest texton. The texture differences are computed by difference of histograms of two halves of the circle using formula 2.1.

2.2.5 Support Vector Machine (SVM) Classifier

The reason why we use SVM to classify segments of the image is the requirement for classifying regions in real-time manner. SVM is the simplest known classifier that meets our requirement.

We train our classifier from manually labeled regions using an SVM with a χ^2 kernel and slack parameter $C = 1$ [6]. We define the confidence of being shadow of each region, as the log likelihood output of this trained classifier times the pixel area of the region.

2.3 Convolutional Neural Network (CNN)

In this section, we describe the CNN architecture. The reason why we use CNN framework is the need for a basic structure for training that is capable of learning shadow/non-shadow pixels in image patches. Our aim is being as fast as possible to detect shadow in the images. We used ConvNet patch wise on the patches of edges in the image, because we need to refine information on the edges of the image. We used a basic ConvNet structure which has six Convolutional layers, two Pooling layers, and one Fully connected layer (Figure 2.6). The size of the input (patch size) is 32×32 which is an optimum

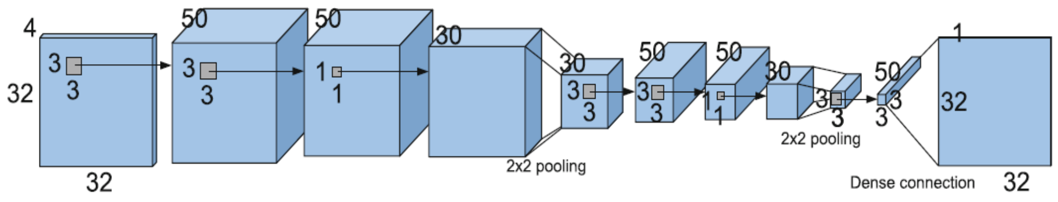

Figure 2.6: Patch-wise CNN with structured output. The input is a 32 × 32 RGBP (RGB image and Image level shadow prior map) image, and the output is a 32 × 32 shadow probability map.

size for learning information from image.

Convolutional neural networks are similar to ordinary neural networks. They are made of neurons that have learnable weights and biases. Each neuron gets some inputs, performs a dot product and optionally non-linearities. The whole network performs as a differentiable score function, such that the input is the raw image pixels, and the output is class scores. They have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer.

In ConvNet architectures, it is assumed that the inputs are images, which allow to encode certain attributes into its structure, to make the forward function more efficient to implement, and significantly reduce the number of parameters.

LeNet was one of the very first convolutional neural networks which helped introducing the field of Deep Learning. This pioneering work by Yann LeCun was named LeNet5 after many previous successful iterations since the year 1988 [31]. At that time the LeNet architecture was used mainly for character recognition tasks.

2.3.1 Layers to Build a CNN

There are three main types of layers to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer**.

2.3.2 Convolutional Layer

The convolutional layer is the core element of CNN structure. The layer's parameters include a set of learnable filters (or kernels), which have a small receptive field extended through the whole depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input, and producing a 2-dimensional activation map of that filter. Therefore, the network learns filters that are activated when they detect some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume, is an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

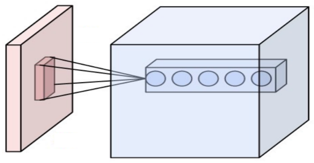

Figure 2.7: Convolutional layer architecture. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, to the full depth. There are multiple neurons along the depth, all looking at the same region in the input.

2.3.3 Local Connectivity of CNN

When dealing with high-dimensional inputs such as images, it is practical to connect the neurons to the neurons in the previous volume that have spatial local patterns, because the network needs to learn specific important information, not all the information.

Spatially local correlations in CNN structure, employing a local connectivity patterns between neurons of adjacent layers. Each neuron is connected to only a small region of the input volume. The extent of this connectivity is a hyper-

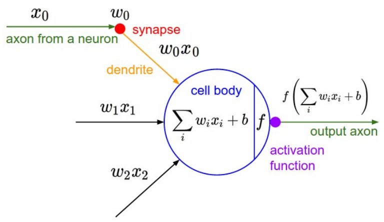

Figure 2.8: Convolutional layer's neuron model. The neurons compute a dot product of their weights with the input followed by a non-linearity. Their connectivity is now restricted to be local spatially.

parameter called the receptive field of the neu-
ron. Thus, learnt filters are sensitive to spatially local input patterns, and response strongly to these kind of patterns. Stacking many such layers leads to have non-linear filters that are more global and responsive to a larger region of pixels. This allows CNN to first build proper representations of small regions of the input. Then, assemble representations of larger areas from them. This characteristic allows CNNs to achieve a better generalization.

2.3.4 Spatial Arrangement in CNN

Three hyperparameters control the size of the output volume of the convolutional layer: the depth, stride and zero-padding.

- **Depth** of the output volume controls the number of neurons in the layer that connect to the same region of the input volume. All of these neurons will learn to activate for different features in the input. For example, if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color.

- **Stride** controls how depth columns around the spatial dimensions (width and height) are allocated. When the stride is s then we move the filters s pixel at a time. This leads to heavily overlapping receptive fields between the columns, and also to large output volumes. When stride gets larger, the receptive fields will overlap less and the resulting output volume will have smaller dimensions spatially.

- Size of **zero-padding** is the third hyperparameter. Sometimes it is convenient to pad the input with zeros on the border of the input volume. Zero padding provides control of spatial size of the output volume. In particular, sometimes it is desirable to exactly preserve the spatial size of the input volume.

The spatial size of the output volume can be computed by the formula $(W - K + 2P)/S + 1$. In this formula, W is input volume size, K is the kernel field size of the convolutional layer neurons, S is the stride, and P is the amount of zero padding used on the border. If this number is not an integer, then the strides are set incorrectly, and the neurons cannot be tiled to fit across the input volume in a symmetric way. In general, setting zero padding to be $P = (K-1)/2$ when the stride $S = 1$, ensures that the input volume and output volume will have the same size spatially. Although, it is generally not necessary to use all of the neurons of the previous layer.

2.3.5 Sharing Parameters in CNN

Sharing parameters is used in convolutional layers to control the number of parameters being learnt. It significantly reduces the number of them, thus, decreasing the memory usage, and therefore training of a larger and more powerful network.

In CNN, each filter is replicated across the entire architecture, because when one patch feature is useful to compute at a specific spatial position, then it should also be useful to compute at a different position. These replicated units share the same parameters (weights and bias) to build a feature map. In other words, all the neurons in a given convolutional layer detect exactly the same feature. Replicating units allows the CNN to detect features regardless of the position in the visual field, thus the structure is *translation invariance*.

Since all neurons in a single depth slice are sharing the same parameters, then the forward pass in each depth slice of the convolutional layer can be computed as a convolution of the neuron's weights with the input volume. Therefore, it is common to refer to the sets of weights as a filter (or a kernel).

It is noteworthy that sometimes the parameter sharing assumption may not be reasonable. One case is when the input images have specific centered structures, which we should learn different features at different spatial locations. In this case it is common to relax the parameter sharing scheme, and instead simply call the layer a locally connected layer.

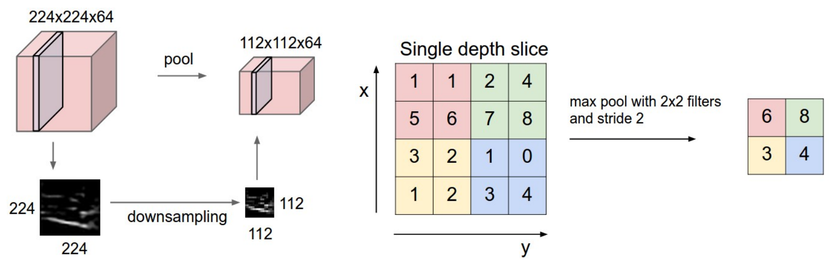

Figure 2.9: Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. In the left side, the input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$. The volume depth is preserved. In the right side, max pooling is shown with a stride of 2. Each max is taken over 4 numbers (2×2 squares).

2.3.6 Pooling Layer

One other important structural element of CNN is pooling layers, which is a form of non-linear down-sampling. There are various kinds of pooling such as average pooling or L2-norm pooling, and max pooling. Among them max pooling is the most common. It partitions the input image into some non-overlapping rectangles, for each one, outputs the maximum amount of the region. The assumption is that the exact location of the feature is less important than its rough location, compare to other features. Thus, it provides a form of translation invariance to CNN. It is common to insert sometimes a pooling layer between successive convolutional layers in a CNN structure to reduce the spatial size of the representation, number of parameters, and amount of computation in CNNs, and therefore to control overfitting.

Due to the aggressive reduction in the size of the representation, the current trend is towards using smaller filters or discarding the pooling layer altogether.

Region of Interest pooling (RoI pooling) is a variant of max pooling layer, in which output size is fixed and input rectangle is a parameter. This layer is an important component of convolutional neural networks for object detection based on Fast R-CNN architecture [15].

2.3.7 ReLU Layer

ReLU (Rectified Linear Units) is a layer of neurons that applies the non-saturating activation function $f(x) = \max(0, x)$. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for instance, the saturating hyperbolic tangent $f(x) = \tanh(x)$, $f(x) = |\tanh(x)|$, and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. Among them, ReLU is more common to use, because it results the training be faster, without making any significant difference to generalisation accuracy.

2.3.8 Fully Connected Layer

At last, after several convolutional and max pooling layers, Fully Connected Layers are employed to reason about the input in a high-level and globally. Neurons in a Fully Connected Layer have full connections to all activations in the previous layer.

2.3.9 Loss Layer

The Loss Layer apears in the last layer of CNN. It specifies how the network training assess the error between the predicted and true labels. Different loss functions proper for different tasks may be used. Softmax loss is used for predicting a single class of several mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting several independent probability values in $[0, 1]$. Euclidean loss is used for regressing to real-valued labels $(-\infty, \infty)$.

2.4 Selection of Training Patches

In this section, we describe the choices of training patches of the images for training the patch-wise CNN.

We should select patches in a way that the structured CNN learns to detect shadows on the edges of the image. The reason is that after performing region-based shadow detection (Section 3.2), predictions made by the CNN are

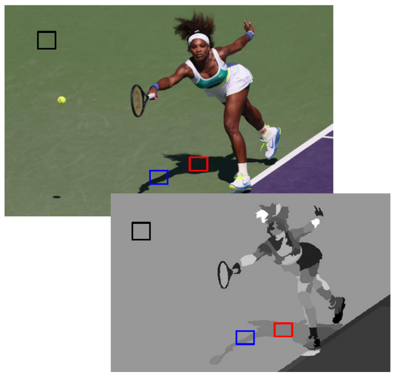
Figure 2.10: The training patches: (1) black patch: on random non-shadow location, (2) blue patch: on Canny-edges between shadow and non-shadow regions, (3) red patch: shadow locations.

local and we therefore need to exploit a higher level interactions between the neighbouring region pixels, therefore we process the edge pixels between the segments by CNN. Additionally, the way of selection should guarantee that CNN learn various types of material. Thus, we selected the training patches in the following ways: (1) On random non-shadow location, to include patches of various textures and colors. (2) On Canny-edges between shadow and non-shadow regions, to include hard-to-classify boundaries. (3) Shadow locations, to guarantee a minimum number of positive instances (Figure 2.10).

2.4.1 Canny-Edge Detection

We include patches on edges between shadow and non-shadow regions to train the CNN. For detecting edges, we utilize Canny-edge detector algorithm. Therefore, in this section we explain about Canny-edge detector method.

Based on a recent research, Juneja et al. [22] compared performance evaluation of edge detection techniques for images in spatial domain, and showed that on visual perception, it can be shown clearly that the Sobel, Prewitt, and Roberts edge detectors provide low quality edge maps relative to the others. A representation of the image can be obtained through the Canny and Laplacian of Gaussian methods. Among the various methods investigated, the Canny method is able to detect both strong and weak edges, and seems to be more suitable than the Laplacian of Gaussian [22].

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986 [5].

2.4.1.1 Canny-Edge Detection Algorithm

The steps are as follows:

1. Apply Gaussian filter to smooth the image in order to remove the noise.
2. Find the intensity gradients of the image.
3. Apply non-maximum suppression to get rid of spurious response to edge detection.
4. Apply double threshold to determine potential edges.
5. Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

2.4.1.2 Gaussian Filter

In order to reduce noise and unwanted details and textures, we should smooth the image with a Gaussian filter.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

where g is the smooth version of f the image, and G_σ is:

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{m^2+n^2}{2\sigma^2})$$

2.4.1.3 Intensity Gradient of an Image

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator (such as Roberts, Prewitt, or Sobel) returns a value for the first derivative in the horizontal direction (G_x) and the vertical direction (G_y). From this the edge gradient and direction can be determined:

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \tan^{-1}[\frac{G_y}{G_x}]$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0°, 45°, 90°, and 135°).

2.4.1.4 Non-Maximum Suppression Technique

Non-maximum suppression is an edge thinning technique. It is applied to “thin” the edge. In other words, it can help to suppress all the gradient values to 0, except the local maximum value which indicates the location of the sharpest change of intensity value. The algorithm for each pixel in the gradient image is:

1. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.

2. If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (i.e., the pixel that is pointing in the y direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

2.4.1.5 Double Thresholding in Canny-Edge Method

Canny does use two thresholds upper and lower (Hysteresis): (a) If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge. (b) If a pixel gradient value is below the lower threshold, then it is rejected. (c) If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a 8-connected neighborhood pixel that is above the upper threshold.

2.5 Summary of the Chapter

In this chapter, we described about how we produce image level shadow prior map, and used it beside image to train patch-wise CNN. As prerequisites to understand completely the algorithm of training the CNN, we explained about

Mean Shift segmentation algorithm, color and texture representations, SVM classifier, CNN architecture, and Canny-edge detector algorithm.

In order to produce an image level shadow prior map, we performed the following steps: (1) segmented the image using Mean Shift algorithm, (2) represented the color of the image with a histogram in L*a*b space, with 21 bins per channel, (3) represented the texture of the image with the texton histogram, with 128 textons, (4) trained our classifier from manually labeled regions using a SVM, (5) defined confidence of each region (segment), as the output of this classifier times the pixel area of the region.

We selected 3 kinds of patches from images to train the CNN: (1) on random non-shadow locations, to include patches of various textures and colors, (2) on Canny-edges between shadow and non-shadow regions, to include hard-to-classify boundaries, and (3) shadow locations, to guarantee minimum number of positive instances.

Chapter 3

Recognition of Shadow from Deep Edge Detection

In this chapter, we will talk about the way we detect the shadows and produce shadow confidence for each pixel of the test image. There are three subsections: (1) the algorithm outline, (2) description of region-based detection of shadows, (3) explaining about how we relate these regions together, and refine the detection results in the edge pixels.

3.1 Shadow Prediction Algorithm

In this section we illustrate the steps of the shadow detection algorithm. Our main goal is processing as less pixels as possible, to have a fast algorithm. The input of the algorithm is a RGB image, and the output is shadow prediction for each pixel of the image.

The algorithm steps are as follows (Figure 3.1):

1. Computing the image level shadow prior map (explained in Section 2.2).
2. Shadow Region Detection: predicting for each region the confidence to be a shadow region.
3. Deep Shadow Edge Detection: relating each region to its neighbor regions, and predicting higher level of interaction among regions.

3.2 Shadow Region Detection

In this section, we explain about the second step of the shadow detection algorithm which is a region based shadow detection.

Primarily, in order to use the trained patch-wise CNN, we generate an image level shadow prior map, and attach it beside the RGB image. Thus, we have the segmented image, and know the clusters of similar pixels (explained in Section 2.2.2).

Secondly, we made a reasonable assumption that pixels in a same region probably have the same prediction of shadiness. We specify each region's confidence to be a shadow region using CNN. Considering the middle point c of each segment, we pass a 32×32 window patch with center c to CNN, and obtain the predictions for the patch pixels. Then, we compute the mean of this predictions. At last, we set the value of each pixel of the segment to be this mean value.

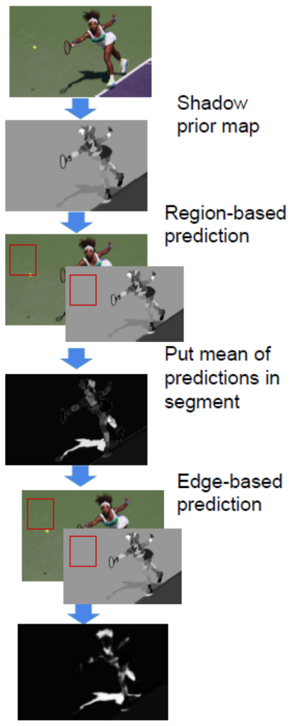

Figure 3.1: Testing framework.

The reason why we use segmentation of the image is that it produces clusters of similar pixels that can be treated in a same way. Thus, we selected one pixel (middle pixel of each segment), and decide the confidence of being shadow for the whole segment. Therefore, number of processed pixels is one pixel for each segment.

3.3 Deep Shadow Edge Detection

The aim of this section is explaining about the third step of the shadow detection algorithm which is deep shadow edge detection.

Predictions made by the CNN are local and we need to utilize higher level interactions between the neighbouring region pixels. So, we process the edge pixels between the neighboring segments by CNN.

We start from the first row and column, and continue to the last one. We select a row and a column at a time, and refine the edge pixels by CNN, then continue to perform the same to all other rows and columns (if we have n rows and m columns, we run the CNN for edge patches at most minimum of n and m times). Because all the pixels in a segment have the same value, edge pixels are the ones that their value are different from their right or below neighbors.

We refine the edge pixels and their neighbors as follows. When an edge pixel is recognized, a window patch with size 32×32 around it is passed to the CNN to get predictions, then we set the edge pixel and its 8 neighbors' values to be mean value of these 9 pixels.

It is noteworthy that we process each pixel on the edge at most once. In addition, we know that shadow pixels are much less than non-shadow pixels, therefore we can assume that if the maximum confidence of being shadow is δ, we should process the pixel of the edge, if its confidence is more than at least 20% of δ.

It should be mentioned that we could extract all the edge patches at once, then input all to get predictions and refine edges at once. But, this way of refinement producing new edge pixels. The reason why processing edge pixels in our way (row-column way) does not produce new edge pixels is that we have three predictions for each pixels and mean of them is final prediction.

Chapter 4

Experiments

In this chapter, we illustrate our experiments. We evaluate our method both quantitatively and qualitatively on main publicly available datasets for single image shadow detection. Our approach assessed on various types of scenes such as aerial images, road scenes, buildings, forests, and beaches. The databases include shadows under significant illumination changes such as sunny, dark, and cloudy weather. It also contain scenes occupied with objects that make the shadow detection more challenging.

Training detail: We employ data augmentation for training images. We downsample the images by six factors 1 to 0.5 with step 0.1, and apply left to right flip. Rotation and flip on patches is performed randomly. We implement the CNN using Theano [1, 3].

4.1 Methods of Evaluation

In this section, we show methods of evaluation that we used to examine our method.

- Accuracy of shadow pixels = $\frac{\text{TP}}{\text{all shadow pixels}}$
  Rate of pixels that are shadows, and correctly detected.

- Accuracy of non-shadow pixels = $\frac{\text{TN}}{\text{all non-shadow pixels}}$
  Rate of pixels that are non-shadows, and correctly detected.

- Accuracy of pixels = $\frac{\text{TP+TN}}{\text{all pixels}}$
  Rate of pixels that are correctly detected.

4.2 Datasets

UCF Shadow Dataset: This dataset contains 355 images with manually labeled region-based ground truth. Only 245 out of 355 images were used in [16, 61]. The split of the train/test data is according to the software package provided by [16] as the original authors did not disclose the split.

UIUC Shadow Dataset: This dataset contains 108 images (32 train images and 76 test images) with region-based ground truth.

SBU Shadow Dataset: This new dataset contains 4,727 images (4,089 train images and 638 test images) with region-based ground truth.

Combined Dataset: It is the combination of above datasets. It includes 5,078 images. We randomly selected 25% of the images for testing, and the rest for training. It includes 3,808 training images, and 1,270 testing images.

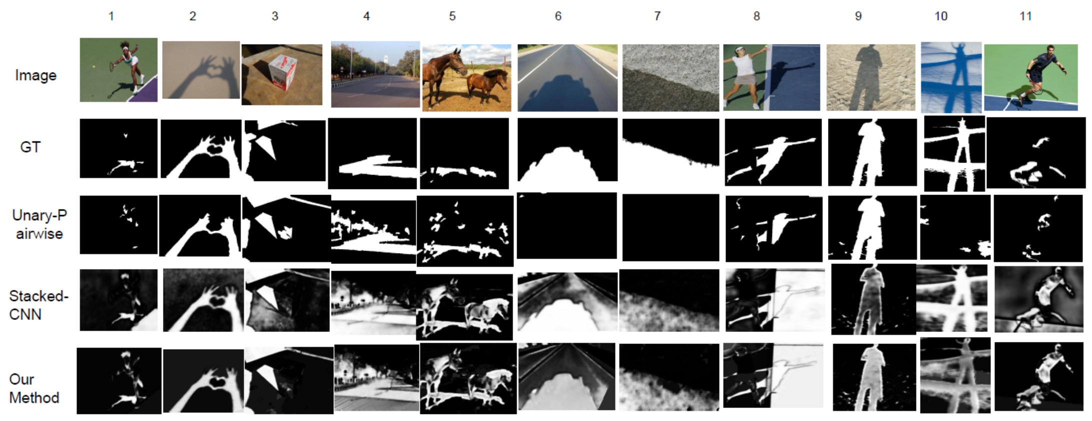

Figure 4.1: Qualitative results on combined dataset.

4.3 Quantitative and Qualitative Results

In this section, we evaluate our method quantitatively and qualitatively, and compare it with state-of-the-art methods. We also compare their time complexities.

Table 4.1: Evaluation of shadow detection methods on combined dataset

Method	Accuracy/std	Shadow-Accuracy/std	Non-shadow-Accuracy/std
Stacked-CNN	0.9044 / 0.12	**0.8614 / 0.18**	0.9140 / 0.13
Unary-Pairwise	0.8835 / 0.13	0.6374 / 0.32	**0.9366 / 0.11**
Our method	**0.9103 / 0.11**	0.8527 / 0.20	0.9248 / **0.11**

Table 4.2: Time complexity of shadow detection methods on combined dataset, using GPU[2]

Method	Testing (hours)	Testing (sec/image)	Training (hours)
Stacked-CNN	39.38	111.62	5.4+Train FCN
Unary-Pairwise	20.77	58.87	-
Our method	**1.45**	**4.11**	**2.18**

[2]Note that in our experiments, Unary-Pairwise method does not use GPU

Table 4.3: Time complexity of shadow detection methods on combined dataset, using CPU

Method	Testing (hours)	Testing (sec/image)	Training (hours)
Stacked-CNN	1693.33	4800.5	10+Train FCN
Unary-Pairwise	20.77	58.87	-
Our method	**11.4**	**32.5**	**4**

Table 4.4: Evaluation of shadow detection methods on UCF dataset

Method	Accuracy	Shadow-Accuracy	Non-shadow-Accuracy
Stacked-CNN	0.8649	**0.8532**	0.8681
Unary-Pairwise	**0.9020**	0.7330	**0.9370**
Our method	0.8682	0.8312	0.8719

Table 4.5: Time complexity of shadow detection methods on UCF dataset, using GPU

Method	Testing (hours)	Testing (sec/image)	Training (hours)
Stacked-CNN	2.5	39.08	0.15+Train FCN
Unary-Pairwise	1.34	40.29	-
Our method	**0.11**	**3.3**	**0.08**

Table 4.6: Evaluation of shadow detection methods on UIUC dataset

Method	Accuracy/std	Shadow-Accuracy/std	Non-shadow-Accuracy/std
Stacked-CNN	0.8121 / 0.20	0.4305 / 0.39	0.9399 / 0.19
Unary-Pairwise	**0.8748 / 0.12**	**0.5735** / 0.39	**0.9746 / 0.07**
Our method	0.7949 / 0.20	0.3771 / **0.38**	0.9394 / 0.18

Our aim is comparing our shadow detection algorithm with not only learning-based methods, but also statistical approaches. Therefore, we select two different methods to compare our approach to them: (1) Stacked-CNN [55], and (2) Unary-Pairwise [16]. The reason of our choices are that Stacked-CNN is

Table 4.7: Time complexity of shadow detection methods on UIUC dataset, using GPU

Method	Testing (hours)	Testing (sec/image)	Training (hours)
Stacked-CNN	2.94	139.29	0.05+Train FCN
Unary-Pairwise	0.32	15.15	-
Our method	**0.07**	**3.62**	**0.03**

Table 4.8: Evaluation of shadow detection methods on SBU dataset

Method	Accuracy/std	Shadow-Accuracy/std	Non-shadow-Accuracy/std
Stacked-CNN	**0.8850 / 0.13**	0.8609 / 0.23	0.9059 / 0.15
Unary-Pairwise	0.8639 / 0.14	0.5636 / 0.35	**0.9357 / 0.12**
Our method	0.8664 / 0.14	**0.8987 / 0.20**	0.8773 / 0.15

Table 4.9: Time complexity of shadow detection methods on SBU dataset, using GPU

Method	Testing (hours)	Testing (sec/image)	Training (hours)
Stacked-CNN	35.56	200.65	68.9+Train FCN
Unary-Pairwise	9.13	51.56	-
Our method	**1.02**	**5.8**	**8.7**

state-of-the-art method that uses a shadow prior map beside RGB images like our method, and is a deep learning based method. Unary-Pairwise is state-of-the-art approach that is a statistical method.

Results of evaluation for selected methods and our approach on Combined dataset, are illustrated in Table 4.1. The accuracy of our method is higher than others, which shows that our method generally performs better than others. The standard deviation of our method is lower than others, which clear the fact that our method is more robust. Our shadow accuracy and non-shadow accuracy is about 1% lower than state-of-the-art methods, which is a trade-off for lower time complexity. Time complexity results of the methods are shown in Table 4.2 and Table 4.3. These tables show that our method is much faster than other methods. Our method not only is extremely fast and real-time using GPU memory, it is also fast using only CPU memory. Other methods are not applicable using only CPU memory.

Moreover, we evaluate our method and compare it to other methods on UCF, UIUC, and SBU shadow datasets (Tables 4.4 to 4.9). The results show that with an accuracy close to the state-of-the-art methods, our method is extremely faster. In Table 4.4, our method's general accuracy is almost similar to Stacked-CNN method which is a deep learning based approach. Unary-pairwise method's accuracy is higher than other methods. In Table 4.6, Unary-pairwise method outperforms other methods. Because, UCF dataset's training images are only 120 images, and UIUC dataset's training images are only 32 images, and learning based methods' performance depends on number of training images. The more the training images are, the better the performance is. In Table 4.8, our shadow accuracy is higher, and the standard deviation is lower than other methods, which shows the stability and robustness of our method.

Please note that we adjusted threshold to maximize accuracy in binary segmentation. In all experiments, we used one threshold value for binarizing the results.

Qualitative results are shown in Figure 4.1. As you can see in this figure, in image number 1, our method can detect the ground as non-shadow region successfully, in contrast of Stacked-CNN method. The reason is not only using color features, but also using texture features in shadow prior map. Unary-Pairwise approach can not even detect the shadow correctly. In image number 6, Stacked-CNN detect the road as shadow wrongly, but our method did not. Unary-Pairwise can not detect the shadow on the road in image number 6 and 7, because their color is similar to shadow color. In image number 8, our method and Stacked-CNN detect the ground as shadow falsely, but Unary-Pairwise approach detect it truly as non-shadow.

We also try a deeper CNN, in such a way that we replicate the CNN structure twice. First CNN's input is shadow prior map attached beside the

RGB image, the output is shadow probability map (P1). The second CNN's input is P1 attached to the RGB image, the output is final shadow probability map result.

By applying this change, our accuracy decreases from 0.9103 to 0.9011, shadow accuracy increases from 0.8527 to 0.8965, and non-shadow accuracy decreases from 0.9248 to 0.9042. Thus, we can say that making the CNN deeper in this way, make the general accuracy worse.

We also quantify how much the final step of our shadow detection approach Deep Shadow Edge Detection (Section 3.3) helps, and how much the time complexity would be without this step. We did experiment on Combined dataset. The accuracy decreases from 0.9103 to 0.8894, shadow accuracy decreases from 0.8527 to 0.7157, and non-shadow accuracy decreases from 0.9248 to 0.9163. Time of processing for the whole dataset decreases from 1.45 hours (4.11 sec/image) to 0.42 hours (1.2 sec/image).

4.4 Conclusions

We proposed a real-time and novel method for shadow detection from single images. Our deep learning architecture performs at the local patch level, and it can make use of image level semantic information beside RGB images. This method is real-time, and applicable for many robotic applications such as road detection for autonomous cars.

In this work, we utilize a CNN framework which is designed to capture the local structure information of shadow edges and automatically learn the most relevant features.

Shadow is a haphazard phenomenon, so learning frameworks need extra information for learning it. We employ an intelligent image level shadow prior map beside RGB image. The prior map is a segmented probability map which similar pixels clustered with respect to their color features and texture differences. The procedure of producing shadow prior map, and training the CNN patch-wise, is described in Chapter 2.

The procedure for detecting shadow is explained in Chapter 3. In nutshell, the procedure is as follows. First, since pixels with the same material (color and texture) probably have the same prediction of shadiness, we first group the pixels with the same material together, and treat similarly each pixel in the same region. In the next step, we specify each region's confidence to be a shadow region by CNN (Section 3.2). At last, predictions made by the CNN are local and we need to utilize higher level interactions between the neighbouring region pixels. We process the edge pixels between the segments by CNN (Section 3.3).

Our main contributions are as follows:

1. A new real-time framework for robust shadow detection integrating both regional and across-boundary learned features.

2. Automatic learning of the most relevant feature representations from RGB images and shadow prior map using CNN framework.

4.5 Further Works

One question that needs more research is how we can use a deeper framework instead of CNN to gain accuracy while keeping the time complexity low. We tried a deeper CNN, but the results got worse. We need to use a different deep structure other than CNN, or make the CNN deeper in a different way, to make the method more accurate.

Bibliography

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[2] Csaba Benedek and Tamás Szirányi. Bayesian foreground and shadow detection in uncertain frame rate surveillance videos. *IEEE Transactions on Image Processing*, 17(4):608–621, 2008.

[3] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

[4] Alessandro Bevilacqua. Effective shadow detection in traffic monitoring applications. 2003.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[6] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[7] Y Chen, D Wen, L Jing, and P Shi. Shadow information recovery in urban areas from very high resolution satellite imagery. *International Journal of Remote Sensing*, 28(15):3249–3254, 2007.

[8] F Cheng and K-H Thiel. Delimiting the building heights in a city from the shadow in a panchromatic spot-imagepart 1. test of forty-two buildings. *Remote Sensing*, 16(3):409–415, 1995.

[9] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[10] Rita Cucchiara, C Grana, Metal Piccardi, and A Prati. Statistic and knowledge-based moving object detection in traffic scenes. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 27–32. IEEE, 2000.

[11] Rita Cucchiara, Costantino Grana, Massimo Piccardi, Andrea Prati, and Stefano Sirotti. Improving shadow suppression in moving object detection with hsv color information. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 334–339. IEEE, 2001.

[12] Jeff Dozier. Spectral signature of alpine snow cover from the landsat thematic mapper. *Remote sensing of environment*, 28:9–22, 1989.

[13] GD Finlayson, SD Hordley, and Drew Cheng Lu. Ms, on the removal of shadows from images, pattern analysis and machine intelligence. *IEEE Transactions on*, 2006.

[14] Graham D Finlayson, Mark S Drew, and Cheng Lu. Entropy minimization for shadow removal. *International Journal of Computer Vision*, 85(1):35–57, 2009.

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[16] Ruiqi Guo, Qieyun Dai, and Derek Hoiem. Single-image shadow detection and removal using paired regions. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2033–2040. IEEE, 2011.

[17] Jun-Wei Hsieh, Wen-Fong Hu, Chia-Jung Chang, and Yung-Sheng Chen. Shadow elimination for effective moving object detection by gaussian shadow modeling. *Image and Vision Computing*, 21(6):505–516, 2003.

[18] Jia-Bin Huang and Chu-Song Chen. Moving cast shadow detection using physics-based features. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2310–2317. IEEE, 2009.

[19] Xiang Huang, Gang Hua, Jack Tumblin, and Lance Williams. What characterizes a shadow boundary under the sun and sky? In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 898–905. IEEE, 2011.

[20] Xiaoyue Jiang, Andrew J Schofield, and Jeremy L Wyatt. Shadow detection based on colour segmentation and estimated illumination. In *BMVC*, pages 1–11, 2011.

[21] Ajay J Joshi and Nikos P Papanikolopoulos. Learning to detect moving shadows in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2055–2063, 2008.

[22] Mamta Juneja and Parvinder Singh Sandhu. Performance evaluation of edge detection techniques for images in spatial domain. *international journal of computer theory and Engineering*, 1(5):614, 2009.

[23] Imran Junejo and Hassan Foroosh. Estimating geo-temporal location of stationary cameras using shadow trajectories. *Computer Vision–ECCV 2008*, pages 318–331, 2008.

[24] Daniel Kersten, David C Knill, Pascal Mamassian, and Isabelle Bülthoff. Illusory motion from shadows. *Nature*, 379(6560):31–31, 1996.

[25] Salman Hameed Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Automatic feature learning for robust shadow detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1939–1946. IEEE, 2014.

[26] Michael Kilger. A shadow handler in a video-based real-time traffic monitoring system. In *Applications of Computer Vision, Proceedings, 1992., IEEE Workshop on*, pages 11–18. IEEE, 1992.

[27] Dieter Koller, Kostas Daniilidis, and Hans-Hellmut Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer 11263on*, 10(3):257–281, 1993.

[28] Senthil Kumar and Jakub Segen. Gesture-based input interface system with shadow detection, September 23 2003. US Patent 6,624,833.

[29] Jean-François Lalonde, Alexei Efros, and Srinivasa Narasimhan. Detecting ground shadows in outdoor consumer photographs. *Computer Vision–ECCV 2010*, pages 322–335, 2010.

[30] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 61(1):1–11, 1971.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] Alessandro Leone, Cosimo Distante, and Francesco Buccolieri. A shadow elimination approach in video-surveillance context. *Pattern Recognition Letters*, 27(5):345–355, 2006.

[33] Yan Li, Tadashi Sasagawa, and Peng Gong. A system of the shadow detection and shadow removal for high resolution city aerial photo. *Proc. ISPRS Congr, Comm*, 35:802–807, 2004.

[34] Dingding Liu, Bilge Soran, Gregg Petrie, and Linda Shapiro. A review of computer vision segmentation algorithms. *Lecture notes*, 53, 2012.

[35] Xiuwen Liu and DeLiang Wang. Texture classification using spectral histograms. *IEEE transactions on image processing*, 12(6):661–670, 2003.

[36] Zhou Liu, Kaiqi Huang, Tieniu Tan, and Liangsheng Wang. Cast shadow removal combining local and global features. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[37] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

[38] Yasuyuki Matsushita, Ko Nishino, Katsushi Ikeuchi, and Masao Sakauchi. Illumination normalization with time-dependent intrinsic images for video surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1336–1347, 2004.

[39] Bruce A Maxwell, Richard M Friedhoff, and Casey A Smith. A bi-illuminant dichromatic reflection model for understanding images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[40] Ivana Mikic, Pamela C Cosman, Greg T Kogut, and Mohan M Trivedi. Moving shadow and object detection in traffic scenes. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 321–324. IEEE, 2000.

[41] Hiroyuki Miura and Saburoh Midorikawa. Updating gis building inventory data using high-resolution satellite images for earthquake damage assessment: Application to metro manila, philippines. *Earthquake spectra*, 22(1):151–168, 2006.

[42] Sohail Nadimi and Bir Bhanu. Physical models for moving shadow and object detection in video. *IEEE transactions on pattern analysis and machine intelligence*, 26(8):1079–1087, 2004.

[43] Takahiro Okabe, Imari Sato, and Yoichi Sato. Attached shadow coding: Estimating surface normals from shadows under unknown reflectance and lighting conditions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1693–1700. IEEE, 2009.

[44] Alexandros Panagopoulos, Chaohui Wang, Dimitris Samaras, and Nikos Paragios. Estimating shadows with the bright channel cue. In *European Conference on Computer Vision*, pages 1–12. Springer, 2010.

[45] Fatih Porikli and Jay Thornton. Shadow flow: A recursive method to learn moving cast shadows. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 891–898. IEEE, 2005.

[46] Jan Puzicha, Joachim M Buhmann, Yossi Rubner, and Carlo Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1165–1172. IEEE, 1999.

[47] Visvanathan Ramesh et al. A class of photometric invariants: Separating material from shape and illumination. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1387–1394. IEEE, 2003.

[48] Elena Salvador, Andrea Cavallaro, and Touradj Ebrahimi. Cast shadow segmentation using invariant color features. *Computer vision and image understanding*, 95(2):238–259, 2004.

[49] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):290–300, 2003.

[50] Oliver Schreer, Ingo Feldmann, Ulrich Golz, and Peter Kauff. Fast and robust shadow detection in videoconference applications. In *Video/Image Processing and Multimedia Communications 4th EURASIP-IEEE Region 8 International Symposium on VIPromCom*, pages 371–375. IEEE, 2002.

[51] Li Shen, Teck Wee Chua, and Karianto Leman. Shadow optimization from structured deep edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2067–2074, 2015.

[52] VK Shettigara and GM Sumerling. Height determination of extended objects using shadows in spot images. *Photogrammetric Engineering and Remote Sensing*, 64(1):35–43, 1998.

[53] Muhammad Shoaib, Ralf Dragon, and Jorn Ostermann. Shadow detection for moving humans using gradient-based background subtraction. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 773–776. IEEE, 2009.

[54] Mingjun Song and Daniel L Civco. A knowledge-based approach for reducing cloud and shadow. In *Proc. of*, pages 22–26, 2002.

[55] Tomás F Yago Vicente, Le Hou, Chen-Ping Yu, Minh Hoai, and Dimitris Samaras. Large-scale training of shadow detectors with noisily-annotated shadow examples. In *European Conference on Computer Vision*, pages 816–832. Springer, 2016.

[56] Tomás F Yago Vicente, Chen-Ping Yu, and Dimitris Samaras. Single image shadow detection using multiple cues in a supermodular mrf. In *BMVC*, 2013.

[57] Yago Vicente, F Tomas, Minh Hoai, and Dimitris Samaras. Leave-one-out kernel optimization for shadow detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3388–3396, 2015.

[58] Yang Wang. Real-time moving vehicle detection with cast shadow removal in video based on conditional random field. *IEEE transactions on circuits and systems for video technology*, 19(3):437–441, 2009.

[59] Yue Wang and Shugen Wang. Shadow detection of urban color aerial images based on partial differential equations. *The international archives of the photogrammetry, remote sensing and spatial information sciences*, 37:B2, 2008.

[60] Mei Xiao, Chong-Zhao Han, and Lei Zhang. Moving shadow detection and removal for traffic sequences. *International Journal of Automation and Computing*, 4(1):38–46, 2007.

[61] Jiejie Zhu, Kegan GG Samuel, Syed Z Masood, and Marshall F Tappen. Learning to recognize shadows in monochromatic natural images. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 223–230. IEEE, 2010.

[62] Zhe Zhu and Curtis E Woodcock. Object-based cloud and cloud shadow detection in landsat imagery. *Remote Sensing of Environment*, 118:83–94, 2012.