Leave-One-Out Kernel Optimization for Shadow Detection

Tomás F. Yago Vicente, Minh Hoai, Dimitris Samaras
Stony Brook University, Stony Brook, NY 11794, USA
{tyagovicente, minhhoai, samaras}@cs.stonybrook.edu

Abstract

The objective of this work is to detect shadows in images. We pose this as the problem of labeling image regions, where each region corresponds to a group of superpixels. To predict the label of each region, we train a kernel Least-Squares SVM for separating shadow and non-shadow regions. The parameters of the kernel and the classifier are jointly learned to minimize the leave-one-out cross validation error. Optimizing the leave-one-out cross validation error is typically difficult, but it can be done efficiently in our framework. Experiments on two challenging shadow datasets, UCF and UIUC, show that our region classifier outperforms more complex methods. We further enhance the performance of the region classifier by embedding it in an MRF framework and adding pairwise contextual cues. This leads to a method that significantly outperforms the state-of-the-art.

1. Introduction

Shadow removal is desirable in many situations. Shadows are common in natural scenes, and they are known to wreak havoc in many computer vision tasks such as image segmentation and object detection. Therefore the ability to generate shadow-free images would benefit many computer vision algorithms. Furthermore, for aesthetic reasons, shadow removal can benefit image editing and computational photography algorithms.

Automatic shadow detection and removal from single images, however, are very challenging. A shadow is cast whenever an object occludes an illuminant of the scene; it is the outcome of complex interactions between the geometry, illumination, and reflectance present in the scene. Identifying shadows is therefore difficult because of the limited information about the scene's properties.

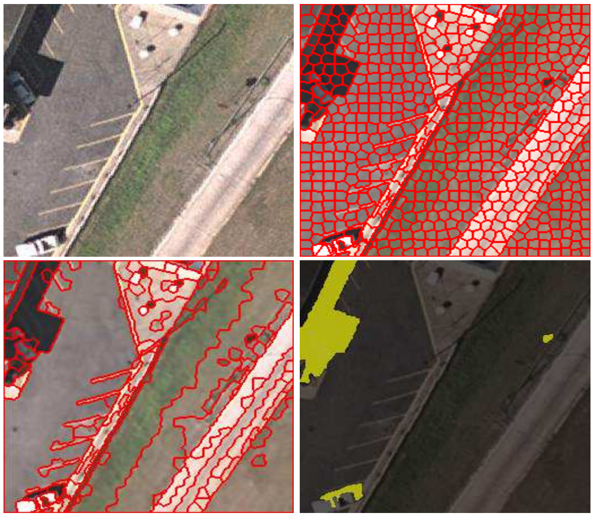

Figure 1. **Shadow detection as a region labeling problem.** top-left: input image, top-right: superpixels, bottom-left: regions obtained by merging superpixels, bottom-right: shadow prediction.

There have been a number of approaches for shadow detection. Purely physics-based methods such as the illumination invariant approaches of [5, 6] only work on high quality images. For consumer photographs and web quality images, statistical learning-based approaches [8, 9, 19, 34, 36] appear more successful, but results are still far from perfect.

In this paper, we propose a novel algorithm for shadow detection. We pose shadow detection as an image labeling problem, similar to some statistical learning-based methods [8, 10, 34]. Given an image, we first divide it into multiple regions, where each region is a group of superpixels, as illustrated in Fig. 1. We use a region classifier to estimate the shadow probability of each region based on its appearance features. Subsequently, we improve shadow detection by considering contextual cues between neighboring regions. The contextual cues are incorporated in our framework as pairwise potentials in a Markov Random Field (MRF). We solve the optimization with QPBO [18, 24] to produce the final shadow labels.

One particular novelty of our approach is the framework for training a strong shadow region classifier that can effectively integrate multiple types of local cues. In particular, we jointly learn a classifier and a discriminative kernel that combines chromatic, intensity, and texture properties for shadow detection. Unlike existing approaches for shadow detection [8, 9, 14, 34], we propose to use Least Square Support Vector Machine (LSSVM). LSSVM has been shown to perform equally well as SVM in many classification benchmarks [27]. LSSVM has a closed-form solution, which is a computational advantage over SVM. Furthermore, once the solution of LSSVM has been computed, the solution for a reduced training set obtained by removing any training data point can be found efficiently. This enables using the same training data for both learning the classifier and the kernel parameters. As will be seen, optimizing the kernel parameters is crucial for improving the discriminative power of the shadow classifier, and this can be done efficiently using our framework. Moreover, our method can be implemented in GPU, reducing the computational cost further.

As will be shown, our shadow region classifier outperforms more complex methods even without using contextual cues. Nonetheless, context is important for shadow detection as it is often difficult to discern shadows based on the local appearance of individual regions, even for human observers. We therefore enhance our method by incorporating contextual cues as pairwise potentials in an MRF framework. We introduce two types of potentials: affinity and disparity. The affinity potentials encourage similar adjacent regions to have the same label, while the disparity potentials prefer different labels for shadow/non-shadow region pairs (using the output of a classifier for region pairs).

We perform experiments on the challenging UCF [8] and UIUC [36] shadow datasets and observe that the proposed method outperforms the current state-of-the-art method [10]. On the UIUC dataset, our method reduces the false negative rate of [10] by 35.3% while maintaining a similar false positive rate. On the UCF dataset, our method reduces the false negative rate and false positive rate by 9% and 13.5%, respectively.

2. Previous Work

2.1. Shadow detection in images

Shadow detection in images is a well studied problem. Earlier methods such as [5, 6] detect shadows by comparing the gradients of an image and its illumination invariant representations. In [22] cues from illumination invariants are further combined with the bright channel cue within an MRF framework. These methods show impressive results in high quality images, but their performance degrades significantly with consumer photographs or web quality pictures [19]. More recent methods use image datasets with annotated shadow masks to learn the appearance of shadows in images. These methods follow two main approaches: detecting shadow boundaries or detecting shadow regions. Lalonde *et al.* [19] focus on shadow boundaries on the ground. They train a shadow boundary classifier based on color and texture features and combine it with scene layout cues from [13] using a CRF to encourage boundary continuity. Huang *et al.* [14] use a set of physically inspired features to train a shadow boundary pixel classifier using an SVM. They join pixels confidently predicted as shadow boundaries with weakly predicted adjacent pixels in a Canny-like manner. Shadow boundary detection methods [14, 19] achieve good results, but struggle to segment closed shadow contours.

To detect shadow regions, Zhu *et al.* [36] propose a set of shadow variant and shadow invariant features in monochromatic images to learn a shadow region classifier, refined by a CRF. Guo *et al.* [8, 9] train two pairwise classifiers to find pairs of regions in an image that share the same material and are viewed under the same illumination conditions (both in shadow or both not in shadow), and same material but illuminated differently (only one region in shadow). They minimize an energy functional that combines the predictions of a single region classifier and the positive predictions of their pairwise classifiers. However, their single region classifier is not accurate, especially for shadow regions. They use an SVM with a $\mathcal{X}^2$ kernel that has limited discriminative power.

Yago *et al.* [34] propose a multi-kernel model to learn a shadow region SVM classifier. Their multi-kernel model is a summation of base kernels, one for each type of local feature. The main limitation of this model is the assumption of equal importance for all features. The weights and the scaling factors of base kernels are not learned. Furthermore, their approach is computationally expensive.

Most recently, Khan *et al.* [10] propose a deep learning approach to learn features for shadow detection. They train two Convolutional Neural Networks, one for detecting shadow regions and the other for shadow boundaries. They use a CRF to label pixels as shadow/non-shadow where the predictions of the two neural nets are combined into a unary potential, and the pairwise potential is an Ising prior where the pairwise penalty is determined by the similarity in intensities between adjacent pixels.

2.2. Least-Squares SVM Classifiers

We use Least-Squares Support Vector Machines (LSSVM) [26, 28] for region classification. LSSVM has a closed-form solution, which is a computational advantage over SVM. Furthermore, once the solution of LSSVM has been computed, the solution for a reduced training set obtained by removing any training data point can be found efficiently. This enables reusing training data for further calibration (e.g., [11, 12, 29]). This section reviews LSSVM and the leave-one-out formula.

Given a training set of n data points $\{\mathbf{x}_i | \mathbf{x}_i \in \Re^d\}_{i=1}^n$ and associated labels $\{y_i | y_i \in \{1, -1\}\}_{i=1}^n$, LSSVM optimizes the following:

$$\underset{\mathbf{w},b}{\text{minimize}} \ \lambda ||\mathbf{w}||^2 + \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i + b - y_i)^2. \quad (1)$$

For high dimensional data ($d \gg n$), it is more efficient to obtain the solution for $(\mathbf{w}, b)$ via the representer theorem, which states that $\mathbf{w}$ can be expressed as a linear combination of training data, i.e., $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$. Let $\mathbf{K}$ be the kernel matrix, $k_{ij} = \mathbf{x}_i^T \mathbf{x}_j$. The optimal coefficients $\{\alpha_i\}$ and the bias term b can be found using closed-form formula: $[\boldsymbol{\alpha}^T, b]^T = \mathbf{My}$. Where $\mathbf{M}$ and other auxiliary variables are defined as:

$$\mathbf{R} = \begin{bmatrix} \lambda\mathbf{K} & \mathbf{0}_n \\ \mathbf{0}_n^T & 0 \end{bmatrix}, \mathbf{Z} = \begin{bmatrix} \mathbf{K} \\ \mathbf{1}_n^T \end{bmatrix}, \quad (2)$$

$$\mathbf{C} = \mathbf{R} + \mathbf{Z}\mathbf{Z}^T, \mathbf{M} = \mathbf{C}^{-1}\mathbf{Z}, \mathbf{H} = \mathbf{Z}^T\mathbf{M}. \quad (3)$$

If $\mathbf{x}_i$ is removed from the training data, the leave-one-out decision value is given by: $\frac{\mathbf{k}_i^T \boldsymbol{\alpha} + b - y_i h_{ii}}{1 - h_{ii}}$, where $\mathbf{k}_i$ is the i^{th} column vector of $\mathbf{K}$ and h_{ii} is the i^{th} element in the diagonal of $\mathbf{H}$.

3. Shadow Detection by Region Classification

We pose shadow detection as a region classification problem. Given an image, we first segment it into regions using a two-step process [34]: 1) apply SLIC [1] superpixel segmentation to oversegment the image and obtain a set of superpixels; 2) apply Mean-shift clustering [4] and merge superpixels in the same cluster into a larger region. This segmentation process is illustrated in Fig. 1. Once the superpixels have been merged into regions, we use an LSSVM to predict the shadow probability of each region.

Because shadows are the outcome of the complex interaction between scene geometry and illumination sources, it is necessary to consider multiple feature types for shadow classification. In particular, we propose to base the classification decision on the chromatic, intensity, and texture properties of the region. However, it is difficult to combine heterogeneous feature types and manually tune the importance of each type. We therefore consider this as a kernel learning problem where the kernel function has the form:

$$K(x, y) = \sum_{i=1}^{k} w_i \exp\left(-\frac{1}{\sigma_i} D_i(x, y)\right). \quad (4)$$

Here $K(x, y)$ denotes the kernel value between two regions x and y. The function $D_i(x, y)$ is the distance between x and y in some feature space (e.g., $\mathcal{X}^2$ distance between texton histograms) and it is predetermined. The function $\exp(-\frac{1}{\sigma_i} D_i(x, y))$ is called the extended Gaussian kernel [16, 31, 35], and the kernel K is the linear combination of extended Gaussian kernels. The parameters $\{w_i, \sigma_i\}$ are what needs to be learned. Additionally, we constrain the kernel weights to be non-negative and have unit sum, i.e., $\sum_{i=1}^{k} w_i = 1$.

We propose to jointly learn the kernel and the LSSVM classifier. Given a set of training regions and corresponding shadow indicator labels, our goal is to find a set of parameters $\{w_i, \sigma_i\}$ that yields the lowest leave-one-out balanced error rate. The balanced error rate is the average of false positive rate and false negative rate. For brevity, we refer to the balanced error rate simply as error rate or error. The leave-one-out error for a given kernel is defined and conceptually computed as follows. First, the leave-one-out confidence values are computed for all training examples. The leave-one-out confidence value for a particular training example is obtained by training a classifier on the remaining examples and evaluating on the left-out sample. The leave-one-out confidence values are then compared against the ground truth shadow annotation to compute the leave-one-out error rate. In general, estimating the leave-one-out error is computationally prohibitive because classifier training must be done many times, once per training example. However, as explained in Sec. 2.2, using LSSVM, the leave-one-out confidence values can be obtained efficiently without training the leave-one-out classifiers.

The leave-one-out error is a function of the kernel parameters. Even though calculating the value of this function for a particular kernel can be done efficiently, it is still unclear how to find a set of kernel parameters that yields the lowest leave-one-out error. Unfortunately, the leave-one-out error function is not convex. Even worse, this function is noncontinuous and piece-wise constant, and therefore a gradient-based optimization approach is unlikely to work well. To see this, recall that the set of possible error rates are discrete; the interval between two adjacent discrete values is inversely proportional to the number of training examples. This function is non-differentiable at many locations, and has zero gradients at the other locations.

To optimize the set of kernel parameters, we propose to use beam search with random steps. We first discretize the space of kernel parameters using a grid (details in Sec. 3.2). Starting from a random parameter vector, we perform a number of iterative updates, and in each update we:

1. Randomly choose one kernel parameter and assign a new random value. If necessary, re-normalize $\{w_i\}$ to have unit sum.
2. Train an LSSVM and compute the leave-one-out error for the new set of parameters.
3. Update the parameter set if it yields lower leave-one-out error than the current best value.

In our experiments, we perform 500 iterations. If the leave-one-out error does not decrease after 25 consecutive iterations, we randomly assign new values to all parameters.

The method proposed here has advantages over some existing kernel learning approaches. One popular approach is multiple kernel learning, e.g., [2, 15, 20, 32]. Many multiple kernel learning methods, however, can only learn a linear combination of base kernels; they cannot be used to learn other parameters such as the scaling factor of a generalized Gaussian kernel. This problem can be circumvented by creating multiple kernel instances with different parameter settings. However, this explodes the number of base kernels, so the optimization typically requires a differentiable objective function [15, 32]. Furthermore, most existing approaches learn the kernel parameters to optimize an objective function defined on the surrogate loss of training data, not the held-out data. Thus the same training data is used for both classifier training and kernel learning. The double-use of training data reduces the generalization ability of the algorithms. To avoid this problem, one can maintain a separate set of validation data and use the wrapper approach [3] for optimizing kernel parameters. This assumes we have enough labeled data for training and validation. Moreover, optimizing the kernel's parameters on a single set of validation data has the risk of overfitting to the validation data.

3.1. Feature and kernel details

In order to determine if a region is in shadow we will look at its chromatic, intensity and textural properties. For each region, we compute a 21-bin histogram for each of the components (L*,a*,b*) of the perceptually uniform color space CIELAB. To represent texture, we compute a 128-bin texton histogram. We run the full MR8 filter set [33] in the whole dataset and cluster the filter responses into 128 textons using k-means. Shadow regions tend to be less textured and darker. The CIELAB color space has been shown to perform well for shadow edge identification in outdoor scenes [17] as well as to improve reflectance segmentation [7]. The two color opponent channels behave differently under illumination changes. Especially in outdoor environments, the b* channel (yellow-blue) is more sensitive to shadows than the a* channel (red-green), which is shadow invariant to a certain degree [30]. To compare textures between regions we use the $\mathcal{X}^2$ distance between their texton histograms. For color histograms, it is more appropriate to use the Earth Mover's Distance (EMD) [25] because neighboring bins in the L*,a*,b* histograms represent proximate values and their ground distance is uniform (property of the CIELAB space), in contrast to texton histograms. Furthermore, EMD is more accurate in measuring distances between histograms of continuous entities (such as L,*a,*b), it is less sensitive to quantization error and it can be efficiently computed for 1D histograms. Since our features are normalized histograms (unit mass), both the $\mathcal{X}^2$ and EMD distances are metrics. Hence, we can use them in the form of extended Gaussian distances [16, 35]. Our kernel therefore has the form:

$$K(x,y) = \sum_{l \in \{L,a,b,t\}} w_l \exp\left(-\frac{1}{\sigma_l} D_l(x,y)\right), \quad (5)$$

where D_L, D_a, D_b are EMD distances for L*, a*, b* histograms, and D_t is $\mathcal{X}^2$ distance for texton histograms.

3.2. Optimization grid details

Our task is to optimize the leave-one-out error over eight kernel parameters, which are the kernel weights $\{w_L, w_a, w_b, w_t\}$ and the scaling factors $\{\sigma_L, \sigma_a, \sigma_b, \sigma_t\}$. We define an 8-dimensional grid; one dimension per kernel parameter. The discrete values for each scaling factor σ_l form a set of multiples of the mean distance. This is inspired by a common heuristic: using the mean of the pairwise distances as the scaling factor [35]. If $\{x_1, \ldots, x_n\}$ is the set of training examples, the mean distance is computed as $\mu_l = \frac{1}{(n-1)n}\sum_{i \neq j} D_l(x_i, x_j)$. The possible discrete values of σ_l are $\{s\mu_l | s \in \{\frac{1}{8}, \frac{1}{6}, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 6, 8\}\}$. For the weight w_l of a base kernel, we use $\{s/40 | s \in \{1, \cdots, 10\}\}$ as the set of possible values.

3.3. Error rate computation details

Our optimization criterion is the leave-one-out balanced error rate. This requires having a threshold for separating between positive and negative predictions. While the LSSVM classifier also has the default threshold of 0, this setting is optimized for the total error rate instead of the balanced error rate. To resolve this issue, we first use Platt scaling [23] to map the decision values of LSSVM to probabilities, and then use the probability threshold of 0.5.

More specifically, suppose f_i is the leave-one-out score for the i^{th} training example. We map f_i to a probability using a sigmoid function: $P_{a,b}(f_i) = 1/(1 + \exp(af_i + b))$, where a, b are the parameters of the function. Let N_+ and N_- be the number of positive and negative training examples, and y_i the label of the i^{th} training example. Assuming a uniform uninformative prior over the probabilities of the correct labels, the MAP estimates for the target probabilities are: $t_i = \frac{N_+ + 1}{N_+ + 2}$ if $y_i = 1$ and $t_i = \frac{1}{N_- + 2}$ otherwise. The parameters a, b are set by solving, using Newton's method with backtracking line search [21], the regularized maximum likelihood problem:

$$\max_{a,b}. \sum_i \left(t_i \log(P_{a,b}(f_i)) + (1 - t_i)\log(1 - P_{a,b}(f_i))\right).$$

3.4. GPU acceleration and running time

Efficient GPU implementation is an advantage of our kernel learning algorithm. Each of the many iterations, involves computing a kernel, solving an LSSVM, and calculating the leave-one-out balanced error rate. Computing a kernel for a set of kernel weights requires several element-wise matrix exponentiation and additions, efficiently performed by GPU. Furthermore, the distance matrices between training regions $D_l(\cdot, \cdot)$ need to be computed just once. Solving an LSSVM involves a series of matrix operations (Sec. 2.2), that have GPU implementation. Leave-one-out error computation is efficient, even on a CPU.

In our experiments, the number of training examples is around 12K, corresponding to a kernel matrix of size $12K \times 12K$. For each iteration, our Matlab GPU implementation takes 1.25s to load the distance matrices to the GPU, 0.25s to compute the kernel, 7.5s to train an LSSVM, and 0.2s to compute the leave-one-out error (including Platt's scaling). The total is less than 10s per iteration. Without GPU, each iteration takes about 30s. We run our algorithm for 500 iterations, and the training procedure typically terminates within 1.5 hours. Notably, the random grid optimization procedure can also be parallelized to further reduce the training time.

4. Incorporating Context in Shadow Detection

We enhance shadow detection by embedding the shadow region classifier in an MRF framework. As before, we first segment an image into regions $\{R_i\}$. We construct a graph where each node corresponds to a region and each edge corresponds to a pair of neighboring regions. We associate a binary label x_i for each graph node to indicate whether the corresponding region is in shadow or not ($x_i = 1$ if R_i is shadow and $x_i = -1$ otherwise). Shadow detection is then posed as the minimization of the following energy function:

$$\sum_i \phi(x_i) + \overbrace{\sum_{i,j \in \Omega^a} \psi_a(x_i, x_j)}^{affinity} + \overbrace{\sum_{i,j \in \Omega^d} \psi_d(x_i, x_j)}^{disparity}. \quad (6)$$

In the above $\{x_i\}$ is the set of variables that need to be optimized. The first term of the above energy is the sum of unary potentials. The unary potential $\phi(x_i)$ is based on the probability that region R_i is in shadow, and this depends on the decision value of the shadow region classifier (Sec. 3). The last two terms are the sums of pairwise potentials. They correspond to contextual cues between neighboring regions.

4.1. Unary potentials

We define the unary potential $\phi(x_i)$ in terms of the predictions of the single region classifier (LSSVM with probabilistic output): $\phi(x_i) = -\omega_i P(x_i|R_i)$, where ω_i is the area in pixels of the region R_i, and $P(x_i|R_i)$ is the Platt's scaling probability. The unary potential encourages agreement between the label of a region and the prediction based on the appearance of the region.

4.2. Affinity pairwise potentials

We model relationships between neighboring regions with two types of pairwise potentials. The affinity potential ψ_a penalizes assigning different labels to similar regions. The similarity metric between two regions R_i, R_j is based on the kernel used for the single region classifier. For R_i, R_j where $K(R_i, R_j) > 0.5$, the affinity potential term is:

$$\psi_s(x_i, x_j) = \begin{cases} \omega_{ij}\, K(R_i, R_j) & \text{if } x_i \neq x_j, \\ 0 & \text{otherwise.} \end{cases}$$

The penalty for having different shadow labels is the similarity between the two regions weighted by the geometric mean of the areas of the regions, i.e., $\omega_{ij} = \sqrt{\omega_i \omega_j}$, where ω_i and ω_j are the areas of R_i and R_j, respectively.

4.3. Disparity pairwise potentials

For disparity potentials, we classify shadow/non-shadow transitions between two regions of the same material. We train an LSSVM that takes a pair of neighboring regions and predicts if the input is a shadow/non-shadow pair. We use an RBF kernel with the following features:

- The $\mathcal{X}^2$ distance between the texton histograms.
- The EMD between corresponding L*, a* and b* histograms of the two regions.
- The average RGB ratios. Given two regions i and j, compute the ratios of region average intensity for each R, G and B channels: $\rho_R = R_i/R_j, \rho_G = G_i/G_j, \rho_B = B_i/B_j$, and the feature vector is: $((\rho_R + \rho_G + \rho_B)/3, \rho_R/\rho_B, \rho_G/\rho_B)$.

We penalize same shadow labeling for the pairs of regions that are classified as positive by the learned classifier. The penalty is the prediction confidence weighted by the geometric mean of the regions' areas:

$$\psi_d(x_i, x_j) = \begin{cases} 0 & \text{if } x_i \neq x_j, \\ \omega_{ij}\, P^d(1|R_i, R_j) & \text{otherwise.} \end{cases}$$

The energy function (6) requires optimizing the node labels of a sparse graph. This energy function has submodular pairwise interactions $\psi_a(x_i, x_j)$ and supermodular interactions $\psi_d(x_i, x_j)$. We optimize it using QPBO [18, 24].

5. Experiments

5.1. Experimental setup

We perform experiments on the UCF Shadow dataset [36] and the UIUC Shadow dataset [8]. Both datasets come with shadow masks for performance evaluation. For the UCF dataset, shadow masks are provided by human annotators. In contrast, UIUC shadow masks are obtained automatically. Each shadow image of the UIUC dataset has a corresponding non-shadow version, without the shadow-casting objects. The shadow masks for the UIUC dataset are based on the difference between the shadow and non-shadow images, as illustrated in Fig. 2. It typically leads to a good shadow mask, but not always.

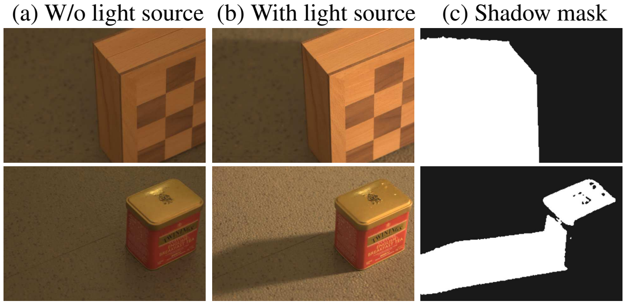

Figure 2. **Generation of 'ground truth' shadow masks on UIUC dataset.** Two images of the same scene are taken, with and without blocking a light source. The shadow mask is obtained by considering the difference between two images. This process typically yields a good shadow mask, but not always. Top: good shadow mask. Bottom: bad shadow mask; the top of the tea box is not in shadow, and should have not been a part of the shadow mask.

Method	Shadow	Non Shad.	BER
UnarySVM [9]	45.7	8.9	27.3
MK-SVM [34]	20.5	4.3	12.4
ConvNet [10]	16.4	5.3	10.6
LooKOP (this paper)	**14.9**	**4.2**	**9.5**

Table 1. **Performance of several region classifiers on the UIUC shadow dataset.** This table shows the error rates for shadow area (second column), non-shadow area (third column), and the balanced error rate (last column). For error rates, a lower number indicates better performance. Best results are printed in bold.

Dataset	Iter 1	Iter 50	Iter 200	Iter 500
UCF	12.6	11.9	11.6	11.5
UIUC	7.3	5.8	5.6	5.5

Table 2. **Leave-one-out balanced error rate as a function of iterations.** This table shows the average error over five trials. The optimization effectively reduces the error rate as the number of iteration increases, converging after about 200 iterations.

For quantitative evaluation, we compare the shadow masks produced by our method to the provided shadow masks. We compute the classification error rates at the pixel level on shadow and non-shadow areas separately. We also report the Balanced Error Rate (BER).

We experimented with different settings of our method. First, we evaluate the single region classifier without the contextual cues from pairwise potentials. We refer to this method as Leave-one-out Kernel Optimization (LooKOP). Second, we evaluate the fully-developed shadow detection framework, embedding the LooKOP in the MRF framework; this will be called LooKOP+MRF. We also experiment with a variant of LooKOP+MRF where the Disparity Pairwise potentials are removed.

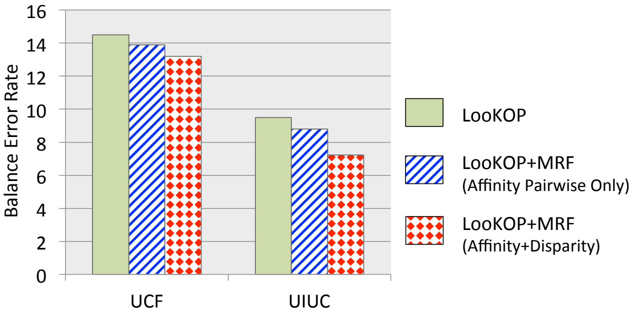

Figure 3. **Benefit of pairwise potentials.** This figure shows the balanced error rates for three methods: LooKOP, LooKOP with Affinity pairwise potential, and LooKOP with both Affinity and Disparity pairwise potentials.

Methods	Shadow	Non Shad.	BER
UnarySVM+Pairwise [9]	28.4	4.8	16.6
ConvNet+CRF [10]	15.3	4.5	9.9
LooKOP+MRF (this paper)	**9.9**	**4.4**	**7.2**

Table 3. **Performance on the UIUC dataset of several complete shadow detection pipelines.** All the methods incorporate pairwise potentials between neighboring regions.

5.2. Comparison between single region classifiers

Table 1 compares the performance of several region classification methods, that predict shadow/non-shadow labels for each region separately (i.e., no pairwise potentials are used). UnarySVM [9] uses a predefined kernel SVM. MK-SVM [34] combines multiple kernels, but kernel weights are not learned. ConvNet [10] combines the predictions of two convolutional neural networks: one for shadow regions and the other for shadow boundaries. This method is referred to as ConvNets (Region+Boundary) in [10]. LooKOP, proposed in this paper, optimizes the kernel parameters to minimize the leave-one-out balanced error rate. As can be seen, LooKOP outperforms the other methods in all evaluation categories. Comparing the balanced error rate of LooKOP and MK-SVM, we note more than 20% error reduction. MK-SVM has two main differences from LooKOP: (i) MK-SVM uses SVM instead of LSSVM; (ii) MK-SVM uses predefined kernel weights and scaling factors, instead of learning them. This demonstrates the importance of learning the kernel parameters. The advantage of LooKOP over other methods is more significant for shadow regions. All methods have higher error rates for shadow regions, commensurate to the difficulty of detecting shadows.

Table 2 shows the leave-one-out balanced error rate (on the training data) of LooKOP as the number of iteration varies. As can be seen, the error rate decreases as the number of iterations increases. The optimization procedure converges and the error stabilizes after around 200 iterations.

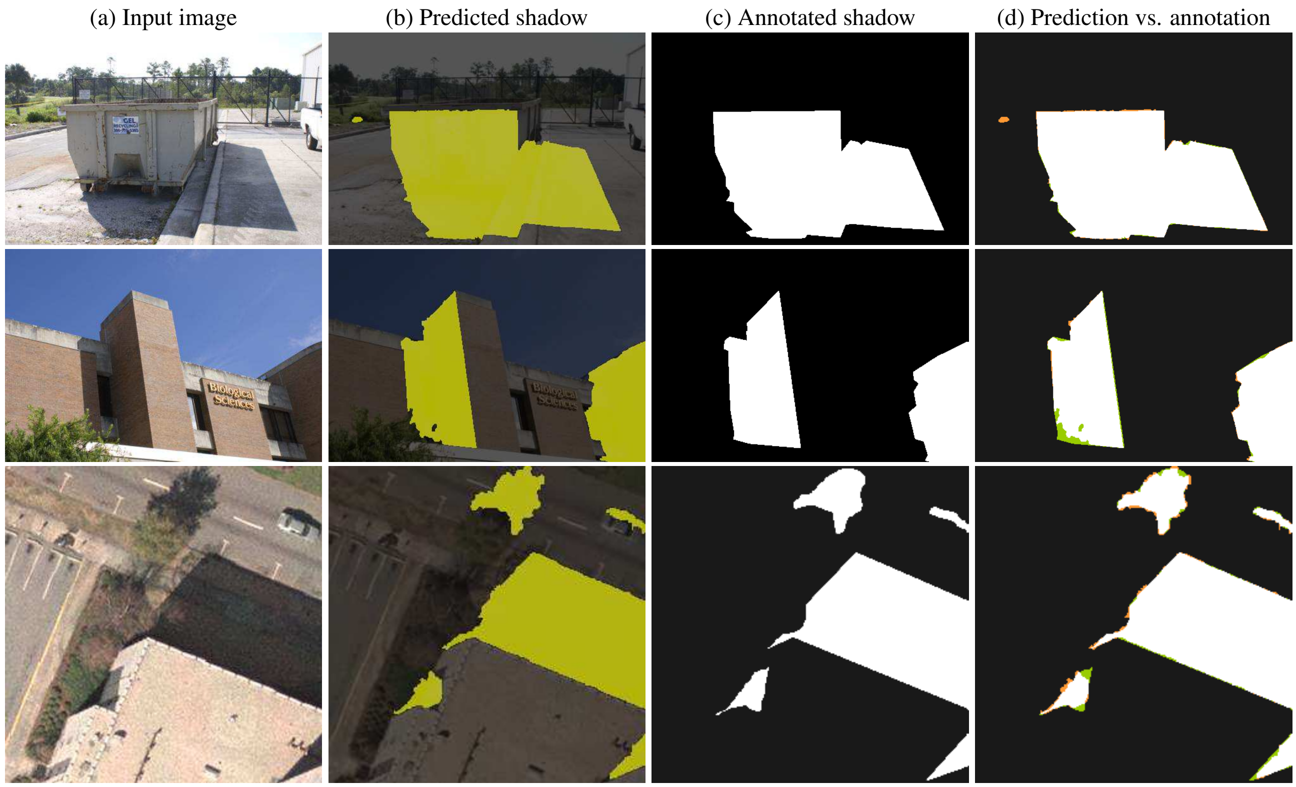

Figure 4. **Shadow detection examples.** The last column compares the predicted shadow and the provided annotation; false positive is shown in orange, false negative in green. This figure is best seen in color. Most of the errors occur at the shadow boundaries.

5.3. Incorporating pairwise potentials

Table 3 compares the performance of several fully-developed shadow detection methods that enhance the single region classifiers by incorporating pairwise potentials between neighboring regions. LooKOP+MRF, proposed in this paper, combines the benefits of a learned kernel and the contextual cues from affinity and disparity pairwise potentials. ConvNet+CRF [10] achieved the prior state-of-the-art result on this dataset. Considering the balanced error rate, we see that LooKOP+MRF outperforms ConvNet+CRF by a wide relative margin of 27.3%. The performance gap between these two methods is even wider in shadow regions.

Table 4 reports the performance of these methods on the UCF dataset. The first three methods only use the unary potentials, while the last three combine both unary and pairwise potentials. Incorporating pairwise potentials improves the performance of all methods, judging by the balanced error rate. Our proposed method, LooKOP+MRF, yields lower error rates than ConvNet+CRF on all evaluation categories, shadow or non-shadow. Interestingly, even LooKOP, the proposed method that does not use pairwise potentials, outperforms ConvNet+CRF on the balanced error rate.

Fig. 3 shows the benefits for embedding LooKOP in an MRF framework, adding pairwise potentials to incorporate contextual cues. These pairwise potentials reduce the balanced error rates on both datasets. This figure also illustrates the importance of the disparity pairwise potentials.

Methods	Shadow	Non Shad.	BER
UnarySVM [9]	63.3	**2.7**	33.0
ConvNet [10]	27.5	7.9	17.7
LooKOP (this paper)	22.9	6.2	14.5
UnarySVM+Pairwise [9]	26.7	6.3	16.5
ConvNet+CRF [10]	22.0	7.4	14.7
LooKOP+MRF (this paper)	**20.0**	6.4	**13.2**

Table 4. **Performance of various methods on UCF dataset.** UnarySVM, ConvNet, and LooKOP are the methods that predict the shadow label of each region individually. The others are fully-developed methods, incorporating contextual cues in terms of pairwise potentials. For each evaluation category, the best performance is printed in bold.

5.4. Qualitative evaluation

Fig. 4 shows some examples of shadow detection using LooKOP+MRF. Overall, this method works well, achieving high precision and recall detection results. Most of the errors occur at the boundaries between shadow and non-shadow areas. These errors are possibly propagated from

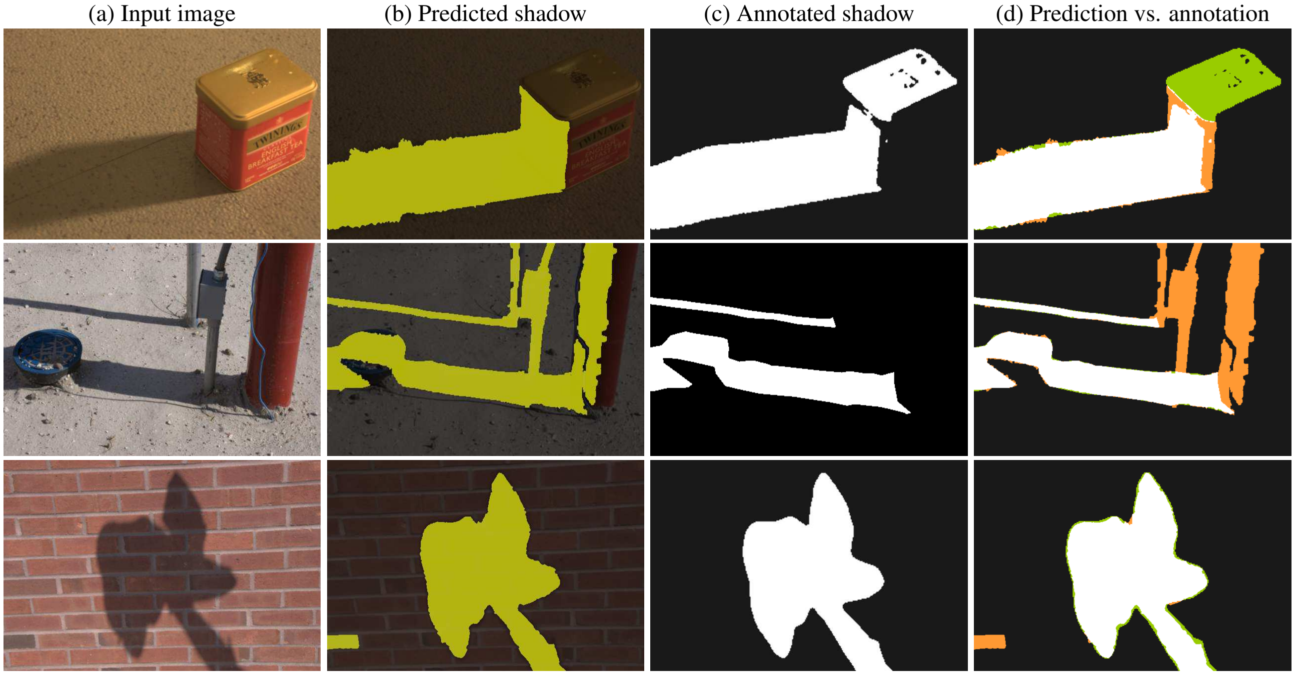

Figure 5. **Examples of significant mismatches between predicted and annotated shadow regions.** The last column compares predicted shadow and provided annotation; false positives in orange, false negatives in green. Rows (1,2): imperfect shadow masks cause mismatches. Row (3): limitation of appearance-based approaches that ignore scene geometry.

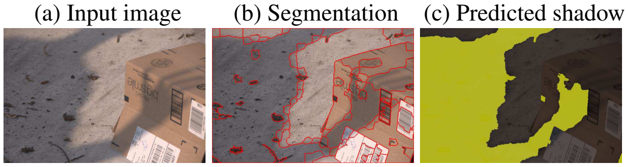

Figure 6. **Failure due to segmentation.** Soft and elongated shadow regions are hard to detect, partly due to the error during superpixel segmentation and grouping. This process may produce regions that contain both shadow and non-shadow pixels.

the process of superpixel segmentation and grouping.

Fig. 5 shows several cases where there are significant differences between the predicted shadow mask and the annotated shadow mask. Interestingly, not all mismatches correspond to a bad result, due to annotation inaccuracies. The shadow mask in the first row of this figure should not have contained the box-top, (per Fig. 2). In the second row, the self-shadow regions should have been part of the shadow mask. The third row is a challenging case. Our method correctly classifies almost all regions, but for a small brick. This is a limitation of appearance-based approaches that ignore scene geometry; they cannot distinguish a dark brick from a brick in shadow. Unfortunately, the Markovian assumptions and pairwise potentials between neighboring regions do not help in this case. Fig. 6 illustrates another failure mode. Our algorithm fails to detect elongated soft shadows. This is partly due to the propagated error from the process of superpixel segmentation and grouping.

6. Summary

We have proposed a framework for shadow detection. To detect shadows in an image, we first divide it into multiple disjoint regions and use a Least-Square SVM to compute the shadow probability of each region. In an MRF framework, we jointly optimize the labels of the regions, taking into account contextual influences of neighboring regions. We have performed experiments on two challenging datasets, and observed that our method achieves lower error rate than the prior state-of-the-art; the reduction in balanced error rate is as high as 27.3% on the UIUC dataset. Qualitatively, we observe minor errors at the boundaries between shadow and non-shadow areas. Moderate errors can be attributed to the inability to reason about scene geometry and the propagation of error from the segmentation process. We also find multiple cases where there is significant difference between the predicted shadow mask and the annotated mask, but those correspond to imperfect annotation.

Acknowledgments. Partially supported by NSF IIS-1161876, IIS-1111047, FRA DTFR5315C00011, the Subsample project from DIGITEO Institute, France, and a gift from Adobe Corp.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE PAMI*, 34(11):2274–2281, 2012. 3

[2] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple

kernel learning, conic duality, and the smo algorithm. In *Proc. ICML*, 2004. 3

[3] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1–3):131–159, 2002. 4

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24(5):603–619, 2002. 3

[5] G. Finlayson, M. Drew, and C. Lu. Entropy minimization for shadow removal. *IJCV*, 85:35–57, 2009. 1, 2

[6] G. Finlayson, S. Hordley, C. Lu, and M. Drew. On the removal of shadows from images. *IEEE PAMI*, 28(1):59–68, 2006. 1, 2

[7] E. Garces, D. Gutierrez, and J. Lopez-Moreno. Graph-based reflectance segmentation. In *SIACG*, 2011. 4

[8] R. Guo, Q. Dai, and D. Hoiem. Single-image shadow detection and removal using paired regions. In *Proc. CVPR*, 2011. 1, 2, 5

[9] R. Guo, Q. Dai, and D. Hoiem. Paired regions for shadow detection and removal. *IEEE PAMI*, 35(12):2956–2967, 2012. 1, 2, 6, 7

[10] S. Hameed Khan, M. Bennamoun, F. Sohel, and R. Togneri. Automatic feature learning for robust shadow detection. In *Proc. CVPR*, 2014. 1, 2, 6, 7

[11] M. Hoai. Regularized max pooling for image categorization. In *Proc. BMVC*, 2014. 2

[12] M. Hoai and A. Zisserman. Improving human action recognition using score distribution and ranking. In *Proc. ACCV*, 2014. 2

[13] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007. 2

[14] X. Huang, G. Hua, J. Tumblin, and L. Williams. What characterizes a shadow boundary under the sun and sky? In *Proc. ICCV*, 2011. 1, 2

[15] A. Jain, S. V. N. Vishwanathan, and M. Varma. Spg-gmkl: Generalized multiple kernel learning with a million kernels. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2012. 3, 4

[16] F. Jing, M. Li, H. jiang Zhang, and B. Zhang. Support vector machines for region-based image retrieval. In *Proc.ICME*, 2003. 3, 4

[17] E. Khan and E. Reinhard. Evaluation of color spaces for edge classification in outdoor scenes. In *Proc. ICIP.*, 2005. 4

[18] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts - a review. 2007. 1, 5

[19] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan. Detecting ground shadows in outdoor consumer photographs. In *Proc. ECCV*, 2010. 1, 2

[20] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004. 3

[21] H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on platt's probabilistic outputs for support vector machines. *ML*, 68(3):267–276, 2007. 4

[22] A. Panagopoulos, C. Wang, D. Samaras, and N. Paragios. Estimating shadows with the bright channel cue. In *CRICV*, 2010. 2

[23] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, 1999. 4

[24] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *Proc. CVPR*, 2007. 1, 5

[25] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proc. ICCV*, 1998. 4

[26] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proc. ICML*, 1998. 2

[27] J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. DeMoor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002. 2

[28] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999. 2

[29] T. Tommasi and B. Caputo. The more you know, the less you learn: From knowledge transfer to one-shot learning of object categories. In *Proc. BMVC*, 2009. 2

[30] T. Troscianko, R. Baddeley, C. A. Parraga, U. Leonards, and J. Troscianko. Visual encoding of green leaves in primate vision. *JoV*, 3:137–, 2003. 4

[31] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. 3

[32] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Proc. ICML*, 2009. 3, 4

[33] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *Proc. ECCV*, 2002. 4

[34] T. F. Yago Vicente, C.-P. Yu, and D. Samaras. Single image shadow detection using multiple cues in a supermodular MRF. In *Proc. BMVC*, 2013. 1, 2, 3, 6

[35] J. Zhang, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *IJCV*, 73:2007, 2007. 3, 4

[36] J. Zhu, K. Samuel, S. Masood, and M. Tappen. Learning to recognize shadows in monochromatic natural images. In *Proc. CVPR*, 2010. 1, 2, 5