Taylor & Francis
Taylor & Francis Group

Spatial Cognition & Computation

An Interdisciplinary Journal

ISSN: 1387-5868 (Print) 1542-7633 (Online) Journal homepage: http://www.tandfonline.com/loi/hscc20

Shadow detection for mobile robots: Features, evaluation, and datasets

Charles C. Newey, Owain D. Jones & Hannah M. Dee

To cite this article: Charles C. Newey, Owain D. Jones & Hannah M. Dee (2017): Shadow detection for mobile robots: Features, evaluation, and datasets, Spatial Cognition & Computation, DOI: 10.1080/13875868.2017.1322088

To link to this article: http://dx.doi.org/10.1080/13875868.2017.1322088

Accepted author version posted online: 28 Apr 2017.
Published online: 28 Apr 2017.

SPATIAL COGNITION & COMPUTATION
https://doi.org/10.1080/13875868.2017.1322088

Taylor & Francis
Taylor & Francis Group

Shadow detection for mobile robots: Features, evaluation, and datasets

Charles C. Newey[a], Owain D. Jones[b], and Hannah M. Dee[c]

[a]Department of Electronics and Computer Science, University of Southampton, Southampton, UK; [b]National Oceanography Centre, MARS, University of Southampton, Southampton, UK; [c]Department of Computer Science, Aberystwyth University, Aberystwyth, UK

ABSTRACT

Shadows have long been a challenging topic for computer vision. This challenge is made even harder when we assume that the camera is moving, as many existing shadow detection techniques require the creation and maintenance of a *background model*. This article explores the problem of shadow modelling from a moving viewpoint (assumed to be a robotic platform) through comparing shadow-variant and shadow-invariant image features — primarily color, texture and edge-based features. These features are then embedded in a segmentation pipeline that provides predictions on shadow status, using minimal temporal context. We also release a public dataset of shadow-related image sequences, to help other researchers further develop shadow detection methods and to enable benchmarking of techniques.

KEYWORDS

vision and natural language; visual perception; robotics

1. Introduction

Shadows pose a problem for computer vision research. Shadows are dark but not opaque, they move (and are attached to "interesting" objects), they share shape and motion characteristics with their "parent" objects, and, because of varying illumination conditions, they don't necessarily have a consistent brightness or color. This combination of factors means that shadows can confuse object detections, they can cause false positives in motion detection, they can adversely affect illumination conditions in a scene causing errors in white-balancing or color-correction algorithms, they can cause background elimination algorithms to fail, and more. Shadows, then, are generally problematic for any vision-based computer or robotic scene analysis system. They are particularly troublesome for systems that are unable to rely upon additional 3D information to determine if an object is tangible (as shadows themselves are not).

CONTACT Hannah M. Dee hmd1@aber.ac.uk Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, United Kingdom.

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/hscc.

When we consider vision from a moving platform, such as we find in robotic agents, it becomes clear that systems which see shadows as objects could be problematic. We would not want a robotic vehicle, for example, to brake suddenly having perceived a sharp shadow on a road as a cliff. In other robotic applications we wish to recover the 3D structure of the world for manipulation or reasoning purposes, and again the existence of spurious objects caused by shadows could cause practical issues for robot performance. The problem is confounded when the viewpoint moves, as much vision-based shadow detection work relies on building pixel-based models over time to represent the color of a surface both shaded and unshaded. Without a static viewpoint, these models cannot be constructed easily as a pixel at two different times will not be guaranteed to refer to the same real-world surface.

This article is a contribution to the ongoing search for features and methods that can be used on a mobile robot and that can handle real-world shadows. The problem of shadows is one of either *shadow detection* or *shadow blindness*. With shadow detection one aims to find the shadows in the scene, which enables them to then be used in further processing. With shadow blindness the aim is to obtain a representation in which shadows effectively disappear stopping them from being mistakenly seen as objects.

In this article we provide an investigation into image features which do not change under shading, and then use brightness-based methods within image regions represented by these features to detect shadows themselves. Thus, our results are relevant to those seeking both *shadow blindness* (we seek features that are invariant to shadow) and *shadow detection* (once found, we use brightness information in regions selected by these invariant features). We work with recorded video from moving positions (some robotic) rather than a live feed from a robot, as this enables full evaluation and benchmarking of any methods produced. No public video dataset exists in this domain, and so we release our datasets to accompany this article and invite other researchers to report their results on our videos, using them as a benchmark set.

2. Background and previous work

There has been considerable work within computer vision on the problem of shadow detection, but the vast majority of early work has focused on the use of background models as shown by Sanin, Sanderson, and Lovell (2012), Dee and Santos (2011) and Leone and Distante (2007). Background models are only of use on video recorded from a static viewpoint, as they use contextual information from the temporal dimension to build up a model of what is unchanging and therefore "uninteresting" in an image. A mobile viewpoint makes the modelling of standard background appearance based upon (for example) pixel statistics very difficult indeed.

Another key strand of successful shadow detection work has concentrated on a fine-grained analysis of colors and their distributions in single images. This work is exemplified by Lalonde, Efros, and Narasimhan (2010), who combine edge features with a Conditional Random Field (CRF) to classify edges in consumer photographs. This kind of approach does not exploit temporal information, but instead looks at the fine-grained detail of edges and uses iterative smoothing to obtain edges that are consistent with the data yet continuous. A related method is presented by Huang, Hua, Tumblin, and Williams (2011) who consider a physically inspired model of simplified illumination assuming that the sun and the sky have a different effect on shading, particularly at the penumbra. Notably, this work uses edges in a similar way to Lalonde et al. (2010) who have inspired the work we present later on edge normals. Using paired regions, R. Guo, Dai, and Hoiem (2013) take high-resolution single images and use a graph-based technique to find pairs of regions that match. Their insight is that a single region is very difficult to classify as shaded or not, but given the same surface in both shaded and unshaded conditions (by matching pairs of regions), it becomes tractable to determine which are shadows.

One of the most influential and useful works in this regard is "Learning to recognize shadows in monochromatic natural images" (Zhu, Samuel, Masood, & Tappen, 2010), in which the authors investigate shadow-variant and shadow-invariant features to determine candidate shadow regions. Zhu et al. compare texton-type features, Gabor filter banks and local binary patterns, and also more crude measures such as local maxima (usually over a small pixel neighborhood). They show that these features can be combined with various machine learning methods (most notably, a CRF-based model combined with a boosted decision tree) to achieve a robust single-image shadow classifier.

These approaches are very good at dealing with high-quality images, but are not generally fast: the CRF step can take seconds with a large image. The speed of these techniques (and other related methods) preclude their application on a mobile robotics platform. However an investigation of single-image techniques such as these is vital, as single-image shadow detection techniques (i.e., techniques that can cope with a lack of temporal context) are the techniques most likely to be applicable to video captured from an active camera.

Moving away from detailed investigations of single images, the remaining work in shadow detection aims to find image or video features that are either shadow variant (that is, they change under shading), or shadow invariant. These features might be edge-based, patch-based, pixel-based, region-based, or hybrid features, built up from some combination of these. Our current article falls into this set of approaches.

With robotic vision we need a system that is as fast as methods based upon background subtraction, but robust enough to camera motion to handle a perspective that moves through the visual world. In a sense, what we seek here is a "halfway house" between methods that detect shadows in static video (building strong temporal models at the pixel level), and those which aim to recover, in detail, shadow information from high-resolution images. We want the context-insensitivity of the latter, with the speed and robustness of the former. Thus we work with medium resolution videos, but do not make extensive use of temporal information at the pixel level.

3. Problem statement and hypotheses

Shadows are caused by an occlusion (the casting object, or *caster*) coming between a surface (the *screen*) and a light source. This results in less light arriving on the screen. The shadow body (*umbra*) is the region of the screen totally occluded from the light source by the caster. If the light source is a point, then this is all the shadow there is. If the light source is extended (and most real-world light sources are) some shaded screen regions are only partially occluded from the light source. These are known as the *penumbra*. Drawing on previous work we can make some observations about the visual character of shadows:

- The penumbra appears as a gradual, blurred region between umbra and unshaded screen.
- The shadow is (often, but not always) a similar color to the screen, only darker.
- The shadow has the same texture as the screen.

From these observations we form two hypotheses about the visual analysis of shadows, and shadow detection. Identifying object, screen, and shadow remains challenging. In certain situations it might be impossible without further knowledge (for example, our robot might end up in the real-world equivalent of an optical illusion, if we have a scene in which several different shades of the same textured surface are present). Nonetheless, we can begin to specify what visual features will be useful for shadow detection in general.

3.1. *Hypothesis 1: We can find some texture measure that is unchanged by shadow*

The visual appearance of a surface arises from the reflection of light from the surface; the character of the surface and the character of the light both influence this. Some statistical texture measures may be invariant to illumination change, in particular there may be some texture measure that is invariant to the illumination change due to shadows. That is, while local

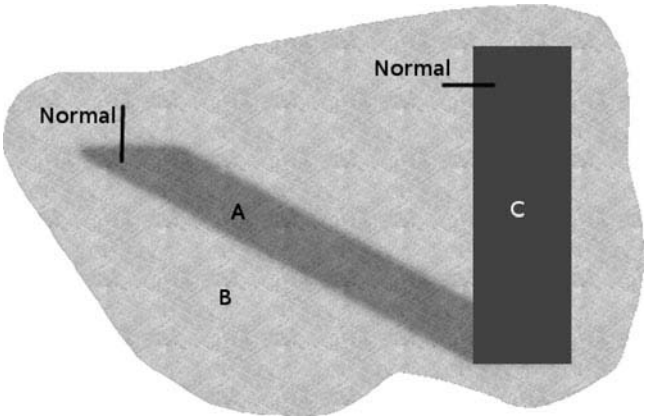


Figure 1. A visual representation of our hypotheses. Textured regions B and A retain their texture, even though other visual properties change under shading. Edge normals across shadows are more gradual than edge normals across objects, due to penumbra effects.

illumination may change (i.e., it is a *shadow-variant* feature), local texture may not (i.e., it is a *shadow-invariant* feature). Note that, in Figure 1, the textured surface retains its visual qualities even under considerable shading under the umbra and penumbra.

3.2. *Hypothesis 2: Shadow edges and object edges have different visual characteristics*

Due to penumbral effects, shadow edges are different to object edges. There will be a gradual transition between the brightness of the screen and the darkness of the shaded screen. By investigating brightness along a line normal to detected edges in an image, we hypothesize that it will be possible to distinguish shadow and non shadow edges by some measure of their "fuzziness". The width of a shadow's penumbra is directly related to the size of the light source, and so this particular hypothesis will not hold if we have point light sources.

4. Segmenting shadows using texture

In this section we present our texture-based shadow segmentation method. The key idea here is that given the right texture measures, a texture segmentation should be blind to shadows and just give us pixel sets that correspond to the visible surfaces in the scene. We can then use the brightness domain within these pixel sets (surfaces) to determine which areas of a surface are shaded and which are not. Figure 2 provides an overview of this technique.

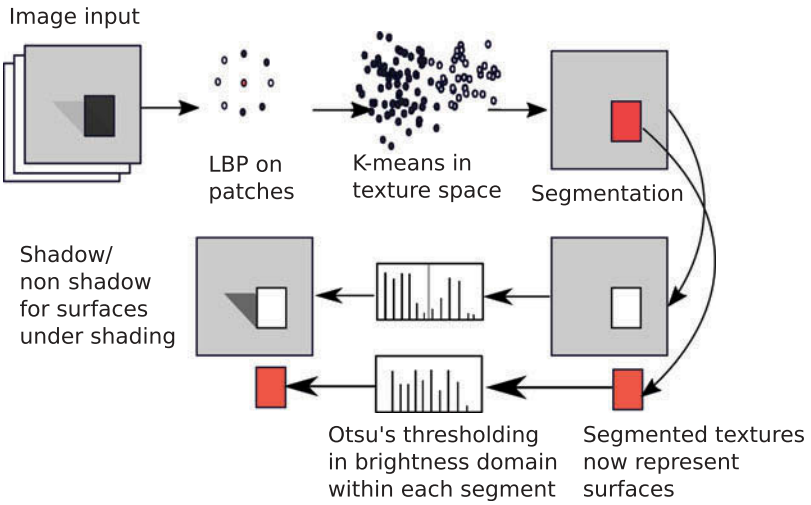


Figure 2. A graphical overview of our texture segmentation approach.

The remainder of this section considers each step in this method in turn: feature selection, clustering methods, texture segmentation, brightness thresholding and final shadow/non shadow classification.

4.1. *Texture features for shadow invariance*

Statistical and other models of surface texture enable vision researchers to classify texture independently of colour. For our purposes we seek features which show strong invariance under shading: we are looking for shadow-blind texture measures. This relates to our *Hypothesis 1*: we can find some texture measure which is unchanged by shadow. That said,but we also need to consider ease of clustering, speed of calculation and overall effectiveness. Among the features considered were Gabor filters (as used in Jain and Farrokhnia (1990) and Li and Staunton (2008)), Local Binary Patterns (LBPs) (Z. Guo, Zhang, & Zhang, 2010), and Grey-Level Co-Occurrence Matrices (GLCMs). We also considered Haralick features (Haralick, Shanmugam, & Dinstein, 1973), which are summary statistics calculated on GLCMs. Gabor filters and LBPs already have some form of pedigree in the realm of shadow research.

The easiest way of testing the shadow invariance of various texture features was to extract texture features for two sets of images of the same scene (one set without shadow, one set with shadow), and compare them statistically. In this manner, the difference between these feature sets will give a straightforward indication as to the degree of shadow-invariance of each feature. This posed a computational challenge, as not only would different features have to be tested, but different configurations of the same features

would need to be tested too (for example, different neighborhood sizes in a Local Binary Pattern, or different radii for Gaussian blur kernels). This provided a large parameter space to search.

Grey Level Co-occurrence Matrices were rejected as a possible feature early on: they provide large, sparse matrices that capture texture variation well but the features become extremely memory-intensive, and are prohibitively slow to cluster. Gabor filters are CPU-intensive, and exhibit harmonic noise that can cause interference when clustering. After eliminating Gabor filters and GLCMs as features, a search was conducted for the optimal parameter set for LBPs.[1] LBPs are defined by two crucial parameters — the neighbourhood size and the number of samples taken from that neighborhood.

With an LBP, the neighbourhood around a central pixel is sampled, then thresholded using the value of that central pixel. This gives a binary feature that encodes the pattern of light and dark in that neighbourhood. We used an automated testing framework to exhaustively search different combinations of parameters (across multiple datasets) by iteratively comparing LBPs with different parameters — each set of LBPs were generated on a set of pairs of the same scene (one image under partial shadow and image non-shadowed). The features were then analyzed to determine which feature sets were the most similar (and therefore, the most shadow-invariant). This process determined that the most shadow-invariant combination of parameters for LBPs is a neighborhood of 5 and a sample size of 15.

4.2. *Clustering and unsupervised texture segmentation*

In texture segmentation the aim is to divide an image into regions that contain coherent clusters. There are a number of techniques and methods for this, many of which involve a feature extraction phase and then a clustering phase. Clusters are then mapped back into the image domain by labelling each pixel or block with the nearest cluster center. This is often followed by some cluster "smoothing" or normalisation step to reduce noise. Previous work in this domain includes online segmentation methods such as by Manjunath and Chellappa (1991), Jain and Farrokhnia (1990) and Kim and Hong (2009).

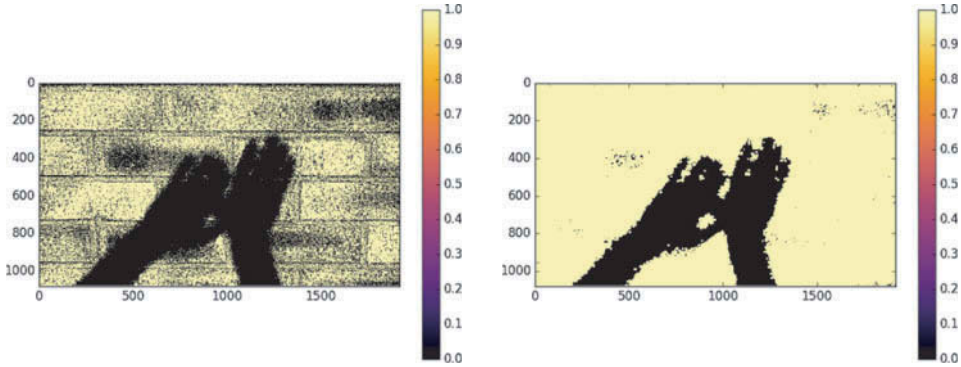
For the sake of simplicity and speed (considering that a typical image will contain several hundred thousand individual pixels), an optimized version of the K-means algorithm was used (mini-batch with K-means++) devised by Sculley (2010) and Arthur and Vassilvitskii (2007). Mini-batch K-Means is an extension of K-means for large datasets, through training on subsets of the data. The original K-means algorithm can take a long time to converge with

[1]Circular LBPs were used in these experiments.

large datasets, but by running with random subsamples minibatch K-means can converge much more quickly. We run this with a fairly large initial K value, to account for the unknown number of textures in an image. Using a large K may result in overly complex texture segmentation; for our work, we found that by keeping K below approximately 20, we obtained reasonable segmentations.

Using K-means clustering to determine texture classes was reasonably effective, and resulted in a segmentation that separated scenes into their component surfaces with a small amount of noise. The assumptions that we make here are that each surface exists in the scene in both shaded and unshaded states, and that we have been able to separate the image into the scene's component surfaces using our shadow-invariant features and subsequent clustering stage. If these assumptions hold, each texture class will have bimodal distribution of brightness values.

From there, an Otsu threshold was applied to each individual texture class — working from the assumption that different textures will have different brightness and illumination properties. This yielded results like Figure 3a, in which the shadow is well identified, but it appears that small-scale local minima were causing false positives. Thus we tidy up the final segmentation with two post-processing steps: local maxima filtering and variance filtering. The local maxima filter works on the assumption that shadows have smaller local maxima than non-shadow regions — therefore, local brightness maxima make useful features. Passing a 3×3 pixel maximum filter over the image replaces all pixel brightness values in a given area with their local maximum. The variance threshold has a similar effect on false positive shadow



(a) An example frame of the simplest K-means texture segmentation shadow detection using an Otsu threshold. Note the false positives caused by small variations on the brick texture.

(b) An example frame produced by the K-means-based shadow detection — using local maxima for the brightness threshold. Note the reduced false positive rate.

Figure 3. Two frames showing the progression of the K-Means/Otsu algorithm.

detections — candidate shadow regions below a certain variance threshold (experiments indicated that a 3×3 filter was most effective) are marked as non-shadow, thus removing false detections on "grainy" textures.

This gives us two methods to compare then: a simple K-means and Otsu-based algorithm, and a more complex K-means-based algorithm, using both Otsu and variance thresholds. As can be seen in Section 6.4 (specifically Table 3), the algorithm using the variance threshold is generally far more effective, outperforming the standard algorithm by up to 34% (for example, on the *tarmac* dataset). For example visualizations of the various algorithms, see Figure 3 and Figure 4.

In addition to experimenting with several different texture features, the optimal combinations of parameters for each feature type (and various other parts of each algorithm) had to be discovered. This was achieved by implementing a simple framework in Python to iteratively test various settings for each parameterized part of each algorithm, and quantitatively evaluate the results using various different metrics (including Adjusted Rand Index and others). This quantitative examination of each parameter combination made it simple to determine the optimal settings for numerous parameters at once — for example, k in K-means, the Gaussian blur radius, LBP neighborhood and sample rate, and so on. After running several hundred experiments with this automated testing framework, it appeared that (certainly on the datasets used for this article) the optimal parameters for LBPs were a neighborhood of 5 pixels and a sample rate of 15 (along with a Gaussian blur of radius 8). For many of the relatively simple image sequences used, it appeared that a small k was sufficient (8–10).

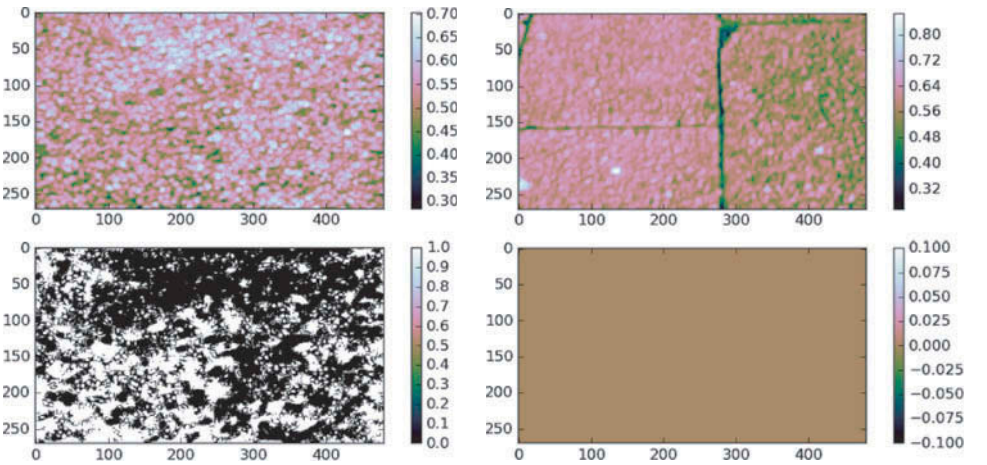


Figure 4. A comparison of the failure modes of the K-means algorithm, before and after rectification.

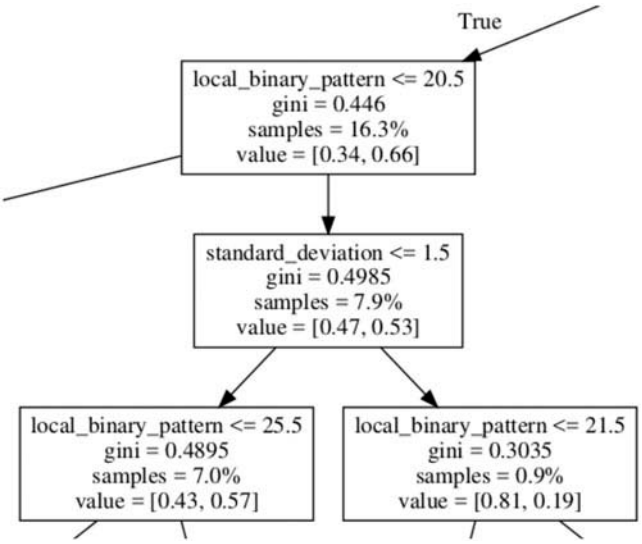


Figure 5. A small sample of a generated set of decision tree rules. This method still yields a complex set of learned rules (even after applying complexity restrictions), but performance is rather good on unseen testing data.

In addition to the preceding unsupervised K-means methods for texture and shadow segmentation, supervised learning techniques were trialled using the same texture feature set. Decision tree techniques are well known for being tolerant to noisy data, they can cope to an extent with high-dimensional features, and they perform reasonably well on large datasets — an ideal property as in this case, millions of individual pixels will have to be classified in each image sequence.

Decision trees are also straightforward to understand, which is a useful property, enabling inspection of the rules learned post-training. Using the default decision tree in Scikit-Learn (Pedregosa et al., 2011) yielded some positive results, but the trees learned were extremely complex — in fact, attempting to visualise them was impractical, as the trees were often several hundred nodes deep — a very clear indication of overfitting. After applying some complexity restrictions and post pruning the trees, however, it became possible to visualize and understand the rules (as in Figure 5).

5. Edges of shadows and objects

Work by Lalonde et al. (2010) has shown that edges can be used to determine shadow boundaries in images. Their method works well for the specific case of detecting ground shadows in outdoor photographs, but does not generalize well for other environments or shadow types, due to their use of a scene model in which all shadows not belonging to the ground plane are discarded.

It is also too computationally expensive to implement in a real-time image processing pipeline such that it could be used on a mobile robotics platform.

In this section we investigate methods that take Lalonde et al.'s central idea of training a collection of decision trees to classify edges as shadow or background (edges detected in non shaded image regions), in a simplified manner which should be possible to run in real-time and does not rely on a scene layout model. We can see that pixels across a shadow boundary may have a distinctive profile. Often they have a greater but more gradual difference in luminance than the harder edges of objects. An example of this is shown in Figure 6, which compares the changes in luminance across all edges on shadow boundaries to edges entirely within shadowed or non shadowed region of the sample image. Although subtle, there is still a perceptible difference in how the luminance signal changes across the shadow edges. Our hypothesis is that by measuring the differences in values across an edge within several different color spaces, we can train a Random Forest of decision tree learners to classify shadow edges.

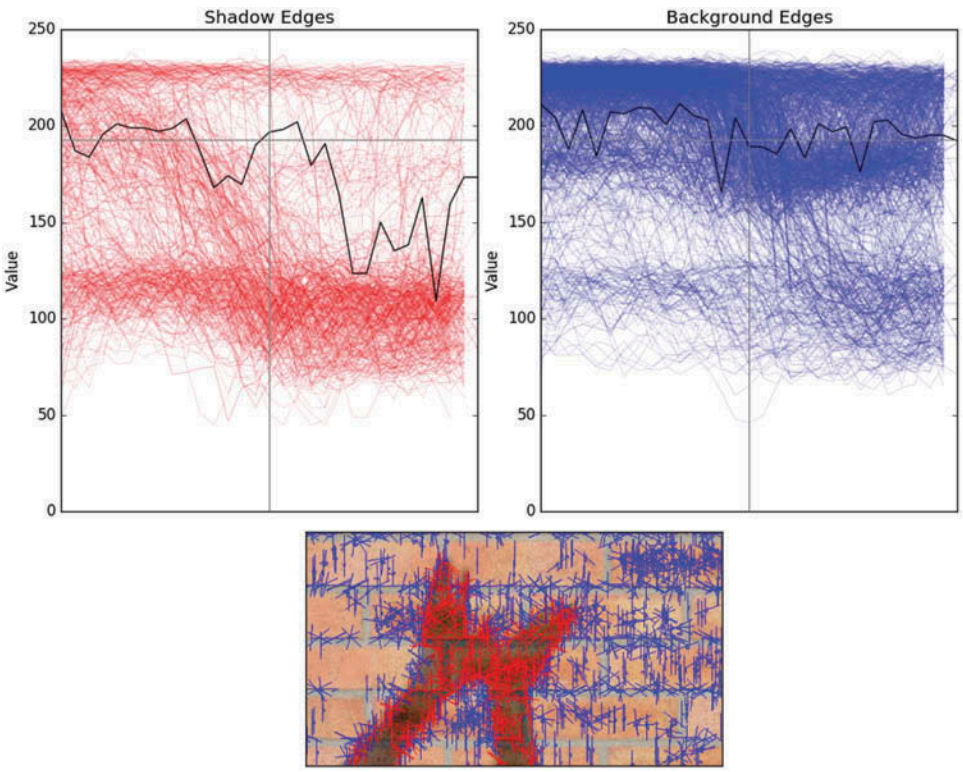


Figure 6. Top left: The brightness profile across lines normal to a shadow edge; Top right: The brightness profile of non shadow edges. Both of these plots have the mean brightness profile superimposed in black. Bottom: the image from which edge normals were drawn, with normals superimposed in blue. These show that the profiles of shadow edges are demonstrably different to edges detected in non shadow "background" regions.

5.1 *A detection method based upon edges*

An outline of the proposed shadow versus background edge classification method is shown in Figure 7. Essentially, we detect all edges in the scene, which gives us a set of edges containing object edges, spurious edges and shadow edges. We investigate the particular character of edges generated as a result of cast shadows, classifying them with a Random Forest technique (Breiman, 2001).

First, to detect strong edges in the image, Canny edge detection (Canny, 1986) is applied to a grey-scale copy of the input image with a lower hysteresis threshold of 0.3 and an upper threshold of 0.6. These thresholds have been selected empirically for our image sets, as they were shown to filter weak edges caused by textured surfaces, while preserving the edges of shadow boundaries in the candidate images. Afterwards, a list of of edge normals is generated: To generate a list of normals to inspect, a contour-finding algorithm (Suzuki & Be, 1985) is applied to the edges image to create a list of contour points. Normals are selected at the center of two points. To ensure complex contours aren't over-sampled, a minimum distance of 5 pixels between candidate normals is enforced. Pixel values alongare then sampled across each normal provide us withto get the profile of the edge to which that normal belongs.

A fixed number of pixels (20) across each normal in the input image are sampled in RGB and HSV color spaces – the sample size of 20 was empirically derived and found to best capture the difference in pixel intensity

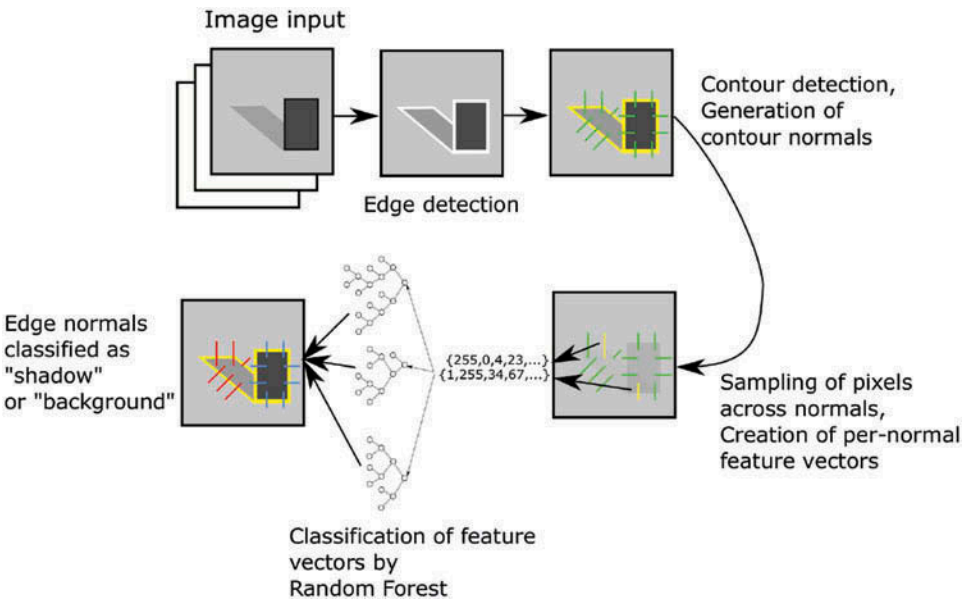


Figure 7. A graphical overview of our edge classification approach.

profiles between shadow and non shadow regions of all the images within the kondo image set. It should be noted that the pixel values from the input images interpolates values when the sample line overlaps two or more pixels. Color ratios and differences in luminosity onfor either side of the edge, as well as the color ratios and differences between the two sides of the edge, are calculated for both HSV and RGB color spaces and stored as a feature vector per normal sampled. Thirty different features are generated in total.

These feature vectors are then given to a Random Forest classifier, which classifies them either as background or shadow. Labels for training the classifier are taken from the ground truths of the image sets – edge normals detected in the input image are labelled as shadow if they are contained within a shadow region in the ground truth. While generating the feature vectors for training, a 1:4 ratio of shadow (positive) to background (negative) examples was maintained, such that the classifier would not over- or under fit the training data.

6. Results

6.1. *Datasets for shadow modelling*

A number of existing datasets exist for the problem of shadow characterisation, particularly with regard to texture analysis and shadow detection. Among these were the CDNet Change Detection datasets by Goyette (see Figure 8), Jodoin, Porikli, Konrad, and Ishwar (2012) and Wang et al. (2014),, which include several datasets with shadow ground-truth. Better yet, some of the datasets also included artificially-applied jitter, which can be a useful surrogate for camera motion. However, for our purposes these datasets were lacking: after performing various texture analyses on the input video sets, it became clear that the spatial resolution of much of the input data was poor. This is mainly due to the footage being captured from a surveillance camera and therefore being a large distance from the target object. This poor spatial resolution meant that any texture features extracted were at a very coarse scale.

Other key shadow datasets have the opposite problem: they exist as high resolution single frames, which provide an excellent level of spatial resolution but no temporal consistency. Some of these single image datasets come with edge-level ground truth, such as the dataset released by Lalonde et al. (2010). Although these could be of use for the work we describe later on shadow edge detection, we would like to have the whole shadow (and not just the edge).

Our eventual aim is shadow detection from a mobile robot, so we have created a set of datasets of increasing "difficulty," which conclude with actual robotic footage. These datasets share some key features: moving shadows

Figure 8. A sample frame from the CDNet "shadow" dataset. Note the particularly poor spatial resolution (which precludes the use of texture analysis techniques).

and/or cameras, reasonable resolution both spatially and temporally, and pixel-level "ground truth". These are summarized in Table 1.

The first dataset we present is simulated, thus not of use in all shadow work due to the lack of texture. It has some key characteristics (strangely shaped casting objects, unusual viewpoints) and pixel perfect ground truth. We call this dataset "Blender," as seen in Figure 9.

We present two high resolution sets of videos with a range of textured surfaces: one with a static camera (for the development of online texture segmentation techniques), and another with an active camera (for the testing of shadow detection techniques in a "realistic" situation). The data comprised of several short video clips (10–15 seconds each) at a high resolution (1920×1081), which was then downscaled to 480×270 so that the dataset could be processed in a reasonable time frame by the various algorithms.

Our final dataset is exceedingly challenging, using a low-quality web camera fixed to a Kondo bipedal walking robot. The poor resolution and low frame rate (and low illumination conditions) pose an extremely difficult set of circumstances under which to attempt to detect shadows. The Kondo KHR-2HV is a toy bipedal robot with a rolling gait. This tabletop robot does not come with a camera or

Table 1. A summary of the datasets we use in this article.

Dataset	Frames	Image Size	Ground truth
Blender	250	720 × 480	3D simulation
Static: Bobbly Slabs	238	480 × 270	Threshold/manual
Static: Bricks	343	480 × 270	Threshold/manual
Static: Smooth Slabs	361	480 × 270	Threshold/manual
Static: Tarmac	340	480 × 270	Threshold/manual
Active: Grass Path	286	480 × 270	Threshold/manual
Active: Seafront Gravel	229	480 × 270	Threshold/manual
Active: Seafront Path	215	480 × 270	Threshold/manual
Kondo	32	960 × 720	Manual

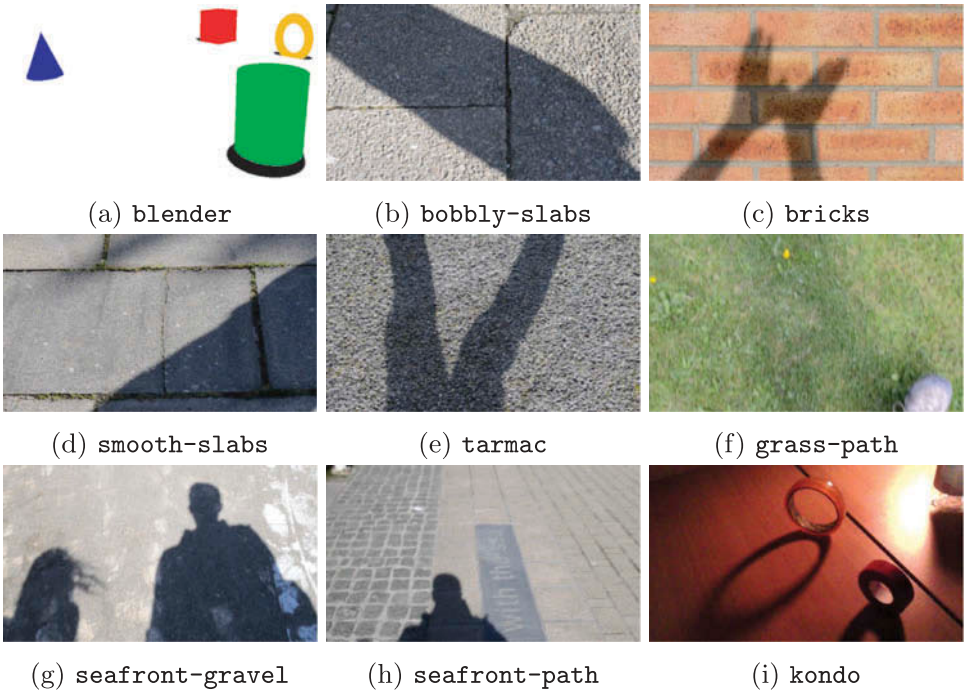


Figure 9. Some example frames from the static and active datasets captured for this article. Note that we have a range of difficulties present in our static and active datasets, too, e.g. in the grass image, the shadow is barely discernible.

indeed any vision capability. In order to capture a realistic bipedal robot dataset we mounted a consumer grade webcam (HP Webcam HD 2300) to the Kondo's "head" and recorded the output.

For *Static*, *Active* and *Kondo* our ground truth was obtained by interactively applying a threshold to the video, and then refined using several other techniques. With data at this scale (thousands of individual frames), it was impractical to hand-annotate the input video in a pixel-wise fashion but working in a frame-wise fashion we were able to separate shadow from background in most cases, and tidy up holes using morphological operations in postprocessing. Each frame has been checked by eye and hand-edited to make the classification more robust if necessary. The "Blender" video ground truth comes from within the Blender software itself and is pixel perfect. It is worth noting here that all datasets come with shadow/background/object ground truth; we have additional ground truths that identify the penumbra for Kondo and for Blender.

All of these datasets are used in this article for offline shadow processing, and are intended for use as a testbed for shadow detection systems to be introduced into robots at a later date. By providing video files and hand-crafted ground truth we lay the foundations for comparison between methods.

Our datasets are all available for public use at http://dx.doi.org/10.5281/zenodo.59019. We ask that if researchers choose to use these, they cite this article.

6.2. Quantifying classification effectiveness

There are two classification metrics that have been predominantly used in the analysis of shadow segmentation methods. The first of these is known as the Jaccard index, and it is a function of true positive, false positive, and false negative classifications — as given by Equation (1). The second, the Rand index (Equation (2)), considers true negatives (as well as true positive, false positive, and false negatives). This produces a slightly different metric that is perhaps more effective for describing datasets where large numbers of true negatives are likely (such as correct classifications of a large non-shadow region in an image). We report both in this article for completeness; however, the Rand index is probably the more robust measure in a situation where classifications are likely to result in a large number of true negatives. For both measures, a score of 1 is perfect and a score of 0 is the worst that can be achieved.

$$J = \frac{TP}{TP + FP + FN} \tag{1}$$

$$R = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

6.3. A baseline method

Before investigating texture based methods, it is worth trying some simple techniques to call *baseline*. For this we present global threshold-based methods, by frame and by sequence. Although these might be considered very basic, we present these partly because they show the effect of some early shadow detection methods, but mainlyand partly because they serve to illustrate the comparative difficulty of our presented datasets. We use two thresholding techniques: a simple static threshold over the complete sequence that classifies every pixel below the threshold as shadow, and an Otsu's threshold (Otsu, 1979) calculated per-frame. Note that Otsu's method assumes that the grey level histogram has two peaks, and chooses a threshold to maximize the difference between these; as mentioned earlier, many authors use some variation on Otsu's method in shadow detection.

From Table 2 we can see that there are questions about the variability between the datasets - for some of the static datasets, a simple fixed threshold works surprisingly well. Note the divergence between Jaccard and Rand indices on some of the datasets. This illustrates the way that the Jaccard index tends to neglect true negatives. In the majority of datasets used here, the predominant part of the image will be non shadow — meaning that a

Table 2. Baseline results of naïve thresholding and frame-based Otsu's thresholding.

	Brightness threshold							
	50		100		200		Otsu threshold	
Image Set	J	R	J	R	J	R	J	R
Blender	0.38	**0.99**	0.17	0.97	0.08	0.93	0.09	0.93
Static: Bobbly	0.17	0.87	0.68	**0.93**	0.17	0.27	0.44	0.80
Static: Bricks	0.03	0.90	0.79	**0.98**	0.10	0.10	0.50	0.90
Static: Smooth	0.08	0.86	0.63	**0.94**	0.16	0.21	0.66	0.92
Static: Tarmac	0.19	**0.95**	0.19	0.82	0.05	0.07	0.12	0.66
Active: Grass	0.00	0.80	0.13	**0.82**	0.20	0.20	0.36	0.69
Active: Gravel	0.26	0.63	0.84	**0.92**	0.60	0.67	0.88	0.94
Active: Path	0.13	0.70	0.79	**0.93**	0.52	0.68	0.88	0.96
Kondo	0.10	**0.39**	0.10	0.24	0.09	0.13	0.11	0.37
Combined	0.14	0.84	0.46	0.88	0.20	0.42	0.44	0.83

Underlined text indicates the best static threshold for a dataset according to Jaccard and bold text indicates the best according to the Rand index.

large part of (correct) classifications will be largely ignored by the Jaccard index, distorting the overall scores.

6.4. *Results for texture-based shadow segmentation*

Considering the texture-based methods, our first methods used a regime in which each machine learning model was trained on each image sequence separately — this is due to different illumination conditions, different textures, and other features being different between each sequence. This meant that there was a potential for overfitting the learning model to each image sequence (which obviously improved performance on the image sequence the model was trained on), but it also meant that the resultant model may not generalize to other image sequences under different circumstances. For example, a model trained on one of the *tarmac* sequences mightwould not necessarily generalize to the *grass* sequences and so forth.

The models were trained with a 30%/70% training/testing split on each sequence, which yielded both a reasonably large amount of (randomly selected) training data, while still remaining small enough to complete processing within reasonable time. As can be seen in Table 3 below, the algorithms generally perform well in terms of accuracy — particularly the decision tree approach; in some cases, reaching up to 90% accuracy.

6.4.1. *Cross-validated results*

Further experiments used a 90%/10% training/testing split on each sequence, using a K-fold learning model (K=10) — see Table 4. This training-testing regime shows that the selection of training data from within a dataset does not unduly affect the results. We also experimented with training on the start of the dataset, and testing on the end of the dataset (taking 30% of frames

Table 3. Results for texture-based shadow segmentation, using LBP, K-means, and decision trees.

	K-Means and LBP		K-Means, LBP and Variance		Decision Tree	
Image Set	J	R	J	R	J	R
Blender	0.000	0.946	0.000	**0.948**	0.531	0.516
Static: Bobbly	0.415	0.823	0.415	0.547	0.938	**0.880**
Static: Bricks	0.698	**0.915**	0.688	0.524	0.856	0.774
Static: Smooth	0.435	**0.901**	0.469	0.546	0.946	0.900
Static: Tarmac	0.205	0.664	0.189	0.513	0.851	**0.769**
Active: Grass	0.389	0.718	0.068	**0.797**	0.793	0.694
Active: Gravel	0.893	0.943	0.893	**0.943**	0.932	0.874
Active: Path	0.865	0.971	0.878	**0.991**	0.949	0.906
Kondo	0.000	0.297	0.000	0.500	0.549	**0.525**
Mean	0.433	**0.798**	0.400	0.701	0.816	0.760

Underlined text indicates best performing method according to the Jaccard index, bold text the best performing method according to the Rand.

Table 4. 10-fold cross-validated Jaccard and Rand scores for the most successful algorithm (naïve decision trees), tested on each dataset.

	10-Fold Cross-Validation Score		Train-start test-end	
Image Set	J	R	J	R
Blender	0.963	0.717	0.970	0.753
Static: Bobbly	0.934	0.696	0.776	0.009
Static: Bricks	0.952	0.710	0.951	0.693
Static: Smooth	0.958	0.791	0.934	0.705
Static: Tarmac	0.975	0.700	0.937	0.174
Active: Grass	0.872	0.482	0.879	0.493
Active: Gravel	0.935	0.760	0.930	0.742
Active: Path	0.975	0.902	0.966	0.871
Kondo	0.859	0.135	0.875	0.143

training data, as before). These results are also shown in Table 4. By training on the start of a sequence and testing on the end, we simulate the idea of a robot being trained on some marked up data, and then being left to classify the rest of a stream.

6.4.2. *Performance on unseen data*

Finally, to test the algorithm's performance on unseen data, a naïve decision tree classifier was trained on a mixture of 180 images from six different datasets, and then tested on the seventh, held out dataset. In this way we test the ability of the classifier to adapt to unseen data, while training on a broader range of shadow images. This is shown in Table 5. The training regimen shown here is an attempt to simulate the training of a robot on labelled data, then testing in a new environment.

6.4.3. *Comments on speed and performance*

Generally, processing time was reasonably slow (averaging approximately 0.2 seconds per 270×480 image), but this is may have been due to the choice of

Table 5. Results from a decision-tree classifier trained on the first 30 frames from 6 different datasets, and then tested on the entirety of the 7th dataset.

	Jaccard Score	Rand Score
Held–Out Dataset	J	R
Static: Bobbly	0.929	0.693
Static: Bricks	0.952	0.714
Static: Smooth	0.953	0.776
Static: Tarmac	0.967	0.702
Active: Grass	0.864	0.450
Active: Gravel	0.932	0.748
Active: Path	0.946	0.794

development language — Python. While the algorithms were generally developed using Scikit-Learn and Numpy, they were developed with an exploratory attitude, rather than with an eye on performance. Properly designed, optimized Python code may well perform considerably better.

In terms of throughput and CPU performance, the training and testing processes using the various machine learning techniques took broadly the same amount of time as the K-means methods—approximately 0.21 seconds per image, on average (including both training and testing) on a single 3.40GHz Intel core.

Overall, results were surprisingly good—indeed, it appears that texture is a potentially valuable additional heuristic for identifying shadow regions in images.

6.5. *Results from the edge-based method*

The Weka Data Mining and Machine Learning software version 3.6 (various, 2016) provided the Random Forest classifier, as well as the tools necessary to train and test such a classifier. Weka's Random Forest classifier has several parameters to alter its behavior; these were kept at their defaults: no limit on decision tree depth or number of features used per tree, with 100 trees generated per forest, and a seed value of 1.

Edge feature performance was tested on each sequence individually (within sequences). A more generalized classifier was also trained on 50 images taken at random from each image set (across sequences; 432 images in total), to test how well this method performed across the differing environments of each image set. All of the tests used 10-fold cross-validation. Results are presented in Table 6.

The Jaccard and Rand indexes in the results show that the implementation of this method performed poorly, with scores lower than our baseline thresholding methods. However, the generalized classifier performed surprisingly well on the seafront-gravel image set specifically. The performance is shown to be highly variable, dependent on the characteristics of the image set. This

Table 6. Random Forest "Edges" classifier tested on each image sequence individually; and across all sequences.

	Within sequence		Across sequences	
Image Set	**J**	**R**	**J**	**R**
Blender	0.009	**0.688**	0.061	0.683
Static: Bobbly	0.000	**0.747**	0.103	0.675
Static: Bricks	0.000	0.705	0.002	**0.705**
Static: Grass	0.144	**0.624**	0.169	0.616
Kondo	0.037	0.729	0.282	0.648
Active: Gravel	0.182	0.308	0.631	**0.644**
Active: Path	0.383	**0.655**	0.419	0.615
Active: Smooth	0.350	**0.667**	0.259	0.640
Active: Tarmac	0.011	**0.658**	0.067	0.636

These results are after 10-fold cross-validation. Underlined text indicates best performing method according to the Jaccard index, italic text is the best performing method according to Rand.

could be due to the features used – indicating that chrominance and luminosity ratios do not capture the differences between shadow and object edges. How accurately shadow edges are sampled is also very dependent on the performance of Canny edge detection, which is susceptible to the sensor noise and high-frequency textures within the images.

The benefit of this approach is that, after an initial model has been trained, the random forest classifier is very quick to run taking several seconds to classify the normals collected from an entire image sequence (ranging from several thousand feature instances to several hundred thousand). Canny edge detection, contour finding and normal sampling are all computationally cheap. Future work could improve the classification accuracy of this method by improving the edge detection and using a feature space that better encodes the different profiles of shadow versus non-shadow edges. This method could then be used to accelerate our texture segmentation based method, providing regions of interest within the input images.

7. Conclusions and future directions

This article has investigated edge and feature representations for shadow detection, within a machine learning/classification framework. We have concentrated on edge features and texture features, with lower level image statistics (variance and so on) used to reduce noise. We hypothesized that there would be shadow invariant texture classes, which we could then use as part of a shadow detector; this article has shown that that is possible. We also hypothesized that the edges of shadows would have different visual characteristics to object edges. This has been less convincingly shown, as we have not found a reliable classifier to distinguish between object and shadow edges across datasets. In short: Texture features work more reliably than edge features; however, both show promise.

We have not exploited any temporal consistency in this work. By concentrating on a thorough evaluation of input features, limiting our search to those

which could work near real-time, we restrict ourselves to features that will be of use in active and/or robotic shadow vision. In future work we plan to experiment with iterative clustering methods, seeding the clustering stage at the start of each frame with the cluster centres from the current frame. This should speed up the texture segmentation step. It also has a clear motivation: assuming that we are dealing with video from a moving camera, we can expect the surfaces in the scene to change location between frames but not change entirely.

The systems and techniques presented here use spatial coherence and texture measures to distinguish between regions and edges that include or abut shadows. These measures and the resulting segmentations could form part of a reasoning system which deals with shadows. To date, reasoning systems that have shadows as part of their logic either assume the vision is completed, or use a cut-down environment so that Otsu's method alone is good enough to obtain a shadow segmentation, as noted by Fenelon, Santos, Dee, and Cozman (2013). Although this article does not provide a complete pixels-to-predicates solution for shadow reasoning, we provide a thorough evaluation of the opening stages of such a process and hopefully bring the prospect of fully automated shadow detection and reasoning a little closer.

By introducing a new dataset for shadow detection and releasing this for other authors to use in their experimentation, we hope to encourage more work in this domain, and also to enable authors to benchmark against our public dataset.

8. Resources and code

As mentioned earlier in the paper, we release the dataset upon which this work is based for other researchers to exploit. We also release the associated code, to enable replication of our results.

- Shadow detection dataset; https://zenodo.org/record/59019
- Texture based shadow exploration; https://github.com/charlienewey/sdmr-paper-code
- Edge-based shadow exploration; https://github.com/erinaceous/shadows-edges

We welcome reuse, but request that if any of these resources are used authors cite the current article.

References

Arthur, D., & Vassilvitskii, S. (2007). K-means++: The Advantages of Careful Seeding. In H. Gabow (Ed.), *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1027–1035). Philadelphia, PA, USA: Society for Industrial and Applied

Mathematics. Retrieved 2016-04-25, from http://dl.acm.org/citation.cfm?id=1283383.1283494

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Canny, J. (1986). A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, *8*(6), 679698.

Dee, H. M., & Santos, P. E. (2011, August). The Perception and Content of Cast Shadows: An Interdisciplinary Review. *Spatial Cognition & Computation*, *11*(3), 226–253. Retrieved 2016-04-22, from http://dx.doi.org/10.1080/13875868.2011.565396 doi: 10.1080/13875868.2011.565396

Fenelon, V., Santos, P. E., Dee, H. M., & Cozman, F. G. (2013). Reasoning about shadows in a mobile robot environment. *Applied Intelligence*, *38*(4), 553–565. doi: 10.1007/s10489-012-0385-5

Frank, E., Hall, M. A., & Witten, I. H. (2016). "The WEKA Workbench." Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques" (Fourth Edition). Morgan Kaufmann.

Goyette, N., Jodoin, P. M., Porikli, F., Konrad, J., & Ishwar, P. (2012, June). Changedetection.net: A new change detection benchmark dataset. In R. Chellappa (Ed.), *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 1–8). doi: 10.1109/CVPRW.2012.6238919

Guo, R., Dai, Q., & Hoiem, D. (2013, December). Paired regions for shadow detection and removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(12), 2956–2967. Retrieved from http://dx.doi.org/10.1109/tpami.2012.214 doi: 10.1109/tpami.2012.214

Guo, Z., Zhang, L., & Zhang, D. (2010, March). Rotation invariant texture classification using LBP variance (LBPV) with global matching. *Pattern Recognition*, *43*(3), 706–719. Retrieved 2016-02-12, from http://www.sciencedirect.com/science/article/pii/S0031320309003331 doi: 10.1016/j.patcog.2009.08.017

Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973, November). Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-3* (6), 610–621. doi: 10.1109/TSMC.1973.4309314

Huang, X., Hua, G., Tumblin, J., & Williams, L. (2011, November). What characterizes a shadow boundary under the sun and sky? In P. Sturm, B. Schiele, S. Lin, & S. Soatto (Eds.), *2011 International Conference On Computer Vision* (pp. 898–905). IEEE. Retrieved from http://dx.doi.org/10.1109/iccv.2011.6126331 doi: 10.1109/iccv.2011.6126331

Jain, A., & Farrokhnia, F. (1990, November). Unsupervised texture segmentation using Gabor filters. In, *IEEE International Conference on Systems, Man and Cybernetics, 1990. Conference Proceedings* (pp. 14–19). doi: 10.1109/ICSMC.1990.142050

Kim, J.-S., & Hong, K.-S. (2009). Color–texture segmentation using unsupervised graph cuts. *Pattern Recognition*, *42*(5), 735–750.

Lalonde, J.-F., Efros, A., & Narasimhan, S. G. (2010). Detecting ground shadows in outdoor consumer photographs. In K. Daniilidis, P. Maragos, & N. Paragios (Eds.), *Computer Vision ECCV 2010 Lecture Notes In Computer Science* (Vol. 6312). Retrieved from http://repository.cmu.edu/robotics/787/ doi: 10.1007/978-3-642-15552-924

Leone, A., & Distante, C. (2007, April). Shadow detection for moving objects based on texture analysis. *Pattern Recognition*, *40*(4), 1222–1233. Retrieved 2016-02-03, from http://www.sciencedirect.com/science/article/pii/S0031320306004146 doi: 10.1016/j.patcog.2006.09.017

Li, M., & Staunton, R. C. (2008, April). Optimum Gabor filter design and local binary patterns for texture segmentation. *Pattern Recognition Letters*, *29*(5), 664–672. Retrieved 2016-02-12, from http://www.sciencedirect.com/science/article/pii/S016786550700390X doi: 10.1016/j.patrec.2007.12.001

Manjunath, B. S., & Chellappa, R. (1991). Unsupervised Texture Segmentation using markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(5), 478–482. doi: 10.1109/34.134046

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics, 9*(1), 62–66.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O.,. . . Duchesnay,. (2011, November). Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. Retrieved 2016-04-14, from http://dl.acm.org/citation.cfm?id=1953048.2078195

Sanin, A., Sanderson, C., & Lovell, B. C. (2012, April). Shadow detection: A survey and comparative evaluation of recent methods. *Pattern Recognition*, *45*(4), 1684–1695. Retrieved 2016-01-26, from http://www.sciencedirect.com/science/article/pii/S0031320311004043 doi: 10.1016/j.patcog.2011.10.001

Sculley, D. (2010). Web-scale K-means Clustering. In M. Rappa, P. Jones, J. Freire, & S. Chakrabarti (Eds.), *Proceedings of the 19th International Conference on World Wide Web* (pp. 1177–1178). New York, NY, USA: ACM. Retrieved 2016-04-26, from http://doi.acm.org/10.1145/1772690.1772862 doi: 10.1145/1772690.1772862

Suzuki, S., & Be, K. (1985, April). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46. Retrieved from http://dx.doi.org/10.1016/0734-189x(85)90016-7 doi: 10.1016/0734-189x(85)90016-7

Wang, Y., Jodoin, P. M., Porikli, F., Konrad, J., Benezeth, Y., & Ishwar, P. (2014, June). CDnet 2014: An expanded change detection benchmark Dataset. In M. Betke & J. Davis (Eds.), *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 393–400). doi: 10.1109/CVPRW.2014.126

Zhu, J., Samuel, K., Masood, S., & Tappen, M. (2010, June). Learning to recognize shadows in monochromatic natural images. In T. Darrell, D. Hogg, & D. Jacobs (Eds.), *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 223–230). doi: 10.1109/CVPR.2010.5540209