Shadow Detection for Improved Object Tracking in Surveillance Cameras

Katarina Berg, Jonna Hellström

Master's thesis
2013:E34

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

Shadow Detection for Improved Object Tracking in Surveillance Cameras

Katarina Berg, Jonna Hellström

14 june 2013

Abstract

Object tracking algorithms and motion triggered alarms are often disturbed by shadows. It is challenging to separate between moving objects and shadows since they have similar movement patterns and image properties. In this thesis, three different approaches to detect shadows are developed and evaluated. The identified shadows determine what parts of the image not to track and what alarms to ignore.

The first approach utilizes a mathematical model to estimate the intensity attenuation of a shadowed region. The second approach applies thresholding to identify shadows based on information about the attenuation, color change and texture preservation. The third approach makes use of probability distributions describing shadows, background and objects. An energy minimization method using discrete optimization is then used in order to classify the pixels as shadow, object or background. All three approaches were evaluated using several different image sequences with corresponding ground truth.

Deriving a shadow detection algorithm that is independent of environment and type of objects in the scene turned out to be the major challenge of this thesis. The best result, a true positive rate of 65.5% and a false positive rate of 2.2%, was achieved with the second approach applying intensity, chromaticity and texture. However, there is still a trade-off between the shadow detection and object discrimination. To further improve the performance, more features and a more extensive data set could be useful.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Aim of the Thesis . . . 3
- 1.3 Related Work . . . 3

2 Problem Formulation **4**

3 Theory **5**
- 3.1 Image Analytic Tools . . . 6
  - 3.1.1 Color Spaces . . . 6
  - 3.1.2 Morphological Operations . . . 6
  - 3.1.3 Histogram Analysis . . . 7
  - 3.1.4 Energy Minimization via Graph Cuts . . . 7
- 3.2 Shadow Detection Approaches . . . 9
  - 3.2.1 Intensity . . . 9
  - 3.2.2 Chromaticity . . . 9
  - 3.2.3 Texture . . . 9
- 3.3 Evaluation Techniques . . . 10
  - 3.3.1 Confusion Matrix . . . 10
  - 3.3.2 TPR and FPR . . . 10
  - 3.3.3 Visualizing the Results . . . 11

4 Data **11**

5 Algorithms **12**
- 5.1 Affine Intensity Relation . . . 12
- 5.2 Intensity, Chromaticity and Texture using Thresholding . . . 13
  - 5.2.1 Intensity & Chromaticity . . . 14
  - 5.2.2 Components & Texture . . . 15
- 5.3 Intensity, Chromaticity and Texture using Graph Cuts . . . 16
  - 5.3.1 Deriving Weights . . . 17
  - 5.3.2 Segmentation using Graph Cuts . . . 19

6 Results **20**
- 6.1 Affine Intensity Relation . . . 20
- 6.2 Intensity, Chromaticity and Texture using Thresholding . . . 21
  - 6.2.1 Intensity & Chromaticity . . . 21
  - 6.2.2 Components & Texture . . . 24
- 6.3 Intensity, Chromaticity and Texture using Graph Cuts . . . 25
- 6.4 Comparison . . . 28

7 Discussion **31**
- 7.1 Affine Intensity Relation . . . 31
- 7.2 Intensity, Chromaticity and Texture using Thresholding . . . 32
- 7.3 Intensity, Chromaticity and Texture using Graph Cuts . . . 33

8 Future Work **34**

1 Introduction

1.1 Background

It is very common that modern surveillance cameras contain image analytic applications. These applications can be used to automatically detect and track moving objects, making it possible to trigger events when certain rules are violated. Normally only human activity is of interest and not environmental movement such as swaying trees or shadows from nearby objects. It can be a time consuming and sometimes hopeless task to manually configure a camera to avoid such scene motion. If the detection of unwanted activity is inaccurate the operators will be overwhelmed with false alarms. Therefore there is a request for motion detection applications that are sensitive to human activity only, as opposed to shadows and environmental objects.

1.2 Aim of the Thesis

The main purpose of this master thesis is to study different strategies to separate shadows from objects when using tracking algorithms in surveillance cameras. The ulterior goal is to reduce the number of false alarms evoked by shadows and thereby improve tracking of human activity.

The thesis is performed at Axis Communications AB which is a manufacturer of network cameras for video surveillance. The shadow detection algorithm should therefore be compatible with Axis integrated systems and meet the requests imposed by Axis.

Initially it is necessary to identify and structure the different problem scenarios that can occur and in what sense they are disturbing. When understanding the complexity of the problem it is time to investigate different solutions. One solution could be to improve the current tracking algorithm so it does not react on shadows in the first place. This could possibly be achieved by detecting motion based on some feature that is insensitive to shadows. This thesis, however, approaches the problem by identifying the shadows seperately from the object tracking algorithm, and then handle the shadows in an appropriate way. By identifying the shadows, a wider range of problems can be solved.

1.3 Related Work

A number of different shadow detection algorithms have been proposed in the literature. The survey articles [2] and [12] give an overview of some of the most recent techniques for cast shadow detection. Both articles include a comparative evaluation of the most representative algorithms. According to the articles, two of the most promising techniques are presented in [1] and [11].

[1] proposes a physically-derived method for shadow detection based on intensity properties. The method assumes that the light energy received when a shadow is cast over a point is an affine transformation[1] of the light energy received at

[1] An affine transformation is a linear transformation followed by a translation, i.e. movement in a specific direction.

the same point in the absence of shadow. This assumption, together with the general condition that a shadowed region in an image has lower intensity than a non shadowed region, forms the basis for a pixel being classified as shadow or not.

Another strategy is applied in [11]. Intensity, chromaticity and large region texture is used to separate shadows from objects and background. Using three different features, they aim at reaching a better simultaneous shadow detection rate and discrimination rate. Intensity and chromaticity information is first used to extract candidate shadow regions, assuming that regions under shadow retain their chromaticity [8]. To remove object pixels that slip through the chromaticity mask, the texture in the candidate shadow regions is evaluated, assuming that regions under shadow keep most of their texture. If the gradient direction correlation between the frame and background is big enough, the region is considered a shadow. A prerequisite for this method to perform well is that the scene contains significant texture.

Some geometry based algorithms use knowledge of the illumination source, object shape and ground plane to predict the orientation, size and shape of the shadows [12]. However, scene constraints in terms of object types, assuming a unique light source, and requiring objects and shadows to have specific orientation, make geometry based shadow detection very limited in its use.

Another feature that has been empolyed to enhance the detection of shadows is temporal information [12]. Since shadows generally have the same type of continuous movement patterns as the objects that generate them, it is assumed that shadows occupy approximately the same pixel area between consecutive frames. This information can be used to apply a temporal filter to the shadows.

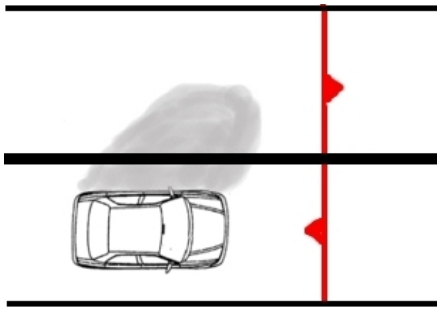
2 Problem Formulation

When detecting moving objects and shadows, a background model is used as a reference. The video frames are compared to the background model, separating the regions in motion from the static background. Figure 1 shows an example of a frame with moving object and shadow, the corresponding background model, and the desired result i.e. ground truth.

(a) Frame. (b) Background. (c) Ground truth.

Figure 1: The frame, background and ground truth from a sample image.

In current object tracking techniques, moving shadows tend to be classified as objects. This is because shadows share the same movement patterns and intensity changes relative to the background as the foreground objects. Since cast shadows can have the same size as the actual objects, their incorrect classification as foreground decreases object tracking performance significantly. See Figure 2 for an illustration of two typical situations where a shadow is classified as object and thereby triggers a false alarm.



(a) The red lines symbolize triggers which send an alarm if they are crossed in the direction of the arrows. This car is driving in the allowed direction but its shadow might be detected as another object, crossing the other line illegaly.

(b) The red area is a forbidden zone. An alarm might be triggered although the man is not traversing the forbidden area.

Figure 2: Typical false alarm situations.

Another problem scenario arises if a static shadow is not detected as foreground but is instead included in the background model. This can cause a problem later in the sequence if the shadow starts to move. Comparing the current frame to the background model, the lack of shadow can be detected as an object and thus trigger an alarm.

Regardless of problem scenario, the solution will be to identify and separate shadows from real objects and background. For the shadow detection algorithm to have practical use, certain requirements are imposed on the implementation. First, it should be flexible with regard to the scene and illumination conditions. For instance it should be independent of environment and type of objects in the scene. Second, it should have rather low complexity in order to be applicable in real-time systems. Also, it is desirable to achieve a high accuracy and precision. Specifically, we want to detect a major part of the shadows without classifying object and background as shadow.

3 Theory

In this section we will present some theory that is relevant to the work. We introduce some of the analytic tools that have been applied, the features utilized to detect shadows, and some techniques used to evaluate the results.

3.1 Image Analytic Tools

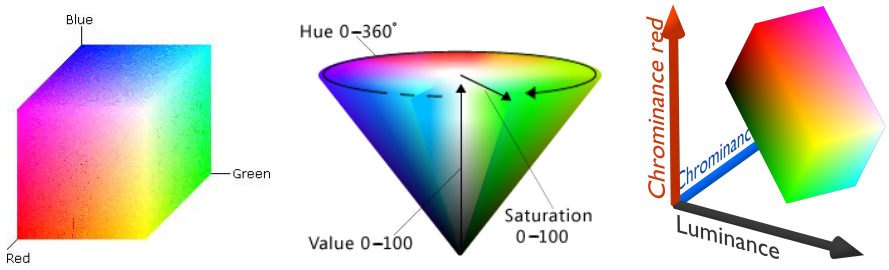
3.1.1 Color Spaces

Color spaces represent colors with different vector values. One of the most common examples is the RGB space where a pixel has three values which represent the amount of red, green and blue. These three values span a color space that can represent most of the colors that can be detected with the human eye.

HSV is another color space that might be more intuitive and is often used in computer graphics. HSV stands for Hue, Saturation and Value, and was designed to approximate the way humans perceive and interpret color. Hue describes the shade of color and is normally the first thing we notice about a color. Saturation describes how pure the hue is and is determined by a combination of light intensity and how much it is distributed across the spectrum of different wavelengths. Finally, the value describes the brigthness of a color [6].

The color space YCbCr is commonly used in video streams and digital photography. Y represents the luminance (brightness) and Cb and Cr are the blue-difference and red-difference chrominance components, conveying color information.

The color spaces described above are visualized in Figure 3.

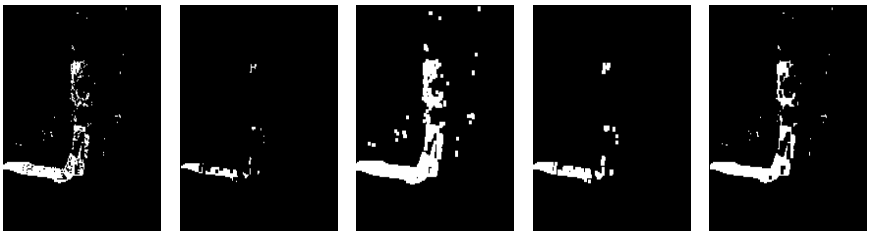


(a) RGB color space. (b) HSV color space. (c) YCbCr color space.

Figure 3: Color spaces.

3.1.2 Morphological Operations

Morphological operations are used on images to change their shape and structure. Most commonly, the morphological operations are performed on binary images. The basic idea is to examine the image with a simple shape, a so called structuring element, to gain knowledge on how the shape fits the structures in the image. The structuring element could be a disk, square, line or some other pre-defined shape. The convex hull, skeleton or contours are examples of features that can be extracted by different morphological operations. Some of the most common morphological operations are demonstrated in Figure 4.



(a) Binary image. (b) Erode operation. (c) Dilate operation. (d) Opening: erosion followed by dilation. (e) Closing: dilation followed by erosion.

Figure 4: Morphological operations using a square structuring element of size 3x3 pixels.

3.1.3 Histogram Analysis

Histogram analysis is used to gain knowledge about for example the intensity distribution, contrast, or exposure[2] of an image. A histogram simply visualizes how many pixels that have a specific value as bars. One example is illustrated below in Figure 5.

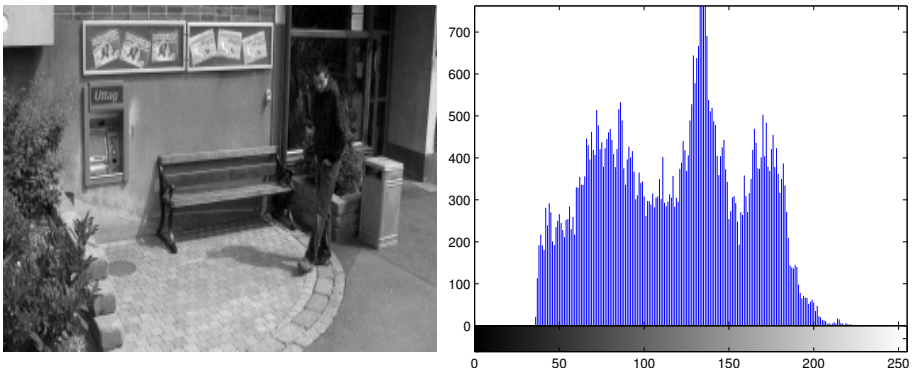


Figure 5: An image and its intensity value histogram with grayscale intensity on the x-axis and number of pixels on the y axis.

3.1.4 Energy Minimization via Graph Cuts

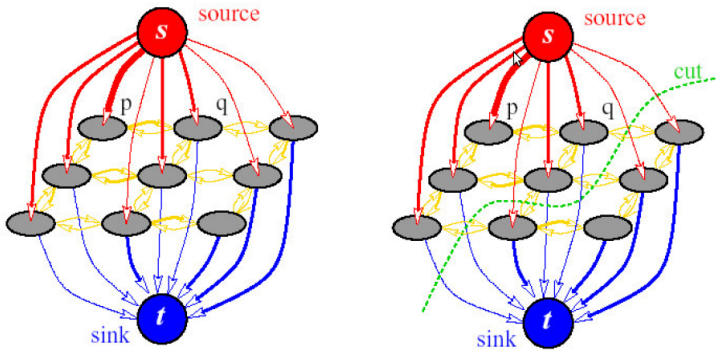
Energy minimization can be used to solve many computer vision problems, including image segmentation. Segmentation problems consist in finding a labeling g that assigns each pixel a label where g is both piecewise smooth and consistent with the observed data. Such a problem can be formulated in terms of energy minimization [5]. We seek the labeling g that minimizes the energy

$$E(g) = E_{data}(g) + E_{smooth}(g) \ . \tag{1}$$

[2]Exposure is the amount of light captured on each area unit of the camera's sensor while taking a photograph.

E_{data} measures the disagreement between g and the observed data, while E_{smooth} measures the extent to which g is not piecewise smooth. The smaller energy, the better g segments the image. The major difficulty with energy minimization lies in the enormous computational costs. The number of possible labelings is $nbrOfLabels^{nbrOfPixels}$, which can be huge. In addition, the energy functions typically have many local minima, complicating the optimization problem further.

There have been numerous attempts to design fast algorithms for energy minimization [5]. The technique applied in this thesis is based on graph cuts. A graph $G = (v, \varepsilon)$ consists of a set of nodes v, corresponding to pixels, and a set of edges ε that connect them. The graph also contains some additional nodes called terminals that correspond to the set of labels that can be assigned to pixels. For a simple example of a graph with only two terminals, see Figure 6.



(a) Graph with two terminals (source and sink). The terminals represents the labels that the pixels can be assigned to.

(b) A graph cut minimizing the cost as determined by the sum of the edge weights. Here, the thickness of the edges represents their weight.

Figure 6: Graph cuts.

All edges in the graph are assigned some weight. Normally, there are two types of edges in the graph: n-links and t-links [4]. n-links connect pairs of neighboring pixels (nodes), thus representing a neighborhood system in the image. The weights of n-links correspond to a penalty for discontinuity between pixels. These weights represent the smoothing term E_{smooth} in (1). t-links connect pixels with the terminals. The weights of a t-link correspond to a penalty for assigning the corresponding label to the pixel. These weights represent the data term E_{data} in (1).

A graph cut is a set of edges such that the terminals become completely separated, partitioning the nodes between the terminals (labels). See Figure 6. Thus, any cut corresponds to some partitioning of the underlying image into as many segments as there are labels. The cost of the cut equals the sum of its edge weights. The minimum cut problem is to find the cut with smallest cost. If weights are set based on parameters of the energy in (1), a minimum cut will correspond to a labeling with the minimum value of this energy.

3.2 Shadow Detection Approaches

Shadow detection is based on finding features that are characteristic to shadows. These features are chosen to separate shadows from surrounding background and objects. The features that have been evaluated in this thesis are described below.

3.2.1 Intensity

The most basic approach to detect shadows is based on the presumption that regions under shadow become darker as they are blocked from the illumination source. However, since the region still is exposed to ambient illumination, there is a limit on how much darker a shadow region can become. These assumptions can be used to set a presumed range of intensity reduction of a region covered by shadow. This intensity information is often used as a first step to separate between shadows and background.

3.2.2 Chromaticity

Another common shadow detection method is based on the assumption that regions under shadow retain their chromaticity. Chromaticity is a measure of the quality of a color, determined by its hue and saturation but independent of intensity. Algorithms based on chromaticity are generally simple to implement and have a low computational complexity. Normally, the HSV color space is used since it offers a good separation between intensity and chromaticity.

A drawback with the chromaticity based methods lies in the assumption that the illumination source produces white light. This is generally a rough simplification of the reality. In outdoor scenes, the main illumination source is approximately white light from the sun scattered in the atmosphere. When a region is covered by shadow, excluding direct sunlight, ambient light will dominate and the chromaticity will be shifted towards blue [9]. In this case, the assumption on constant chromaticity will no longer be satisfied. In indoor scenes it can be even more complicated since the chromaticity is dependent on the light sources and their spectral profiles which might vary.

3.2.3 Texture

Texture based methods rely on the fact that regions under shadow keep most of their texture. These algorithms vary a lot in implementation but in general they follow the same basic steps: selection of candidate shadow regions and classification of these regions as shadow or foreground based on the texture correlation between frame and background. However, various implementations use different sizes of regions to correlate textures and also different correlation techniques. Texture based methods offer great potential since textures are very characteristic, robust to illumination changes, and independent of colors. However, they are typically computationally heavy.

3.3 Evaluation Techniques

3.3.1 Confusion Matrix

When identifying shadows it is of interest how often we make an incorrect classification, but also what kind of error we are doing. The confusion matrix in Table 1 has been used when evaluating different shadow detection algorithms. The confusion matrix can be calculated for a specific frame, an entire video sequence, or various video sequences from different scenes.

Classification \ True	Shadow	Object	Background
Shadow	How big part of the shadow is detected as shadow	How big part of the object is detected as shadow	How big part of the background is detected as shadow
Non-shadow	How big part of the shadow is detected as non-shadow	How big part of the object is detected as non-shadow	How big part of the background is detected as non-shadow

Table 1: Confusion matrix.

Depending on what application the algorithm should be used for, the different fields of the confusion matrix is in focus. For example when suppressing false alarms it is very important that objects are not classified as shadows and thereby suppresses a true alarm.

3.3.2 TPR and FPR

The confusion matrix can be summoned up in two commonly used measures; the true positive rate (TPR) and the false positive rate (FPR). These measures are defined as follows:

$$TPR = \frac{TP}{TP + FN} \quad \text{and} \quad FPR = \frac{FP}{FP + TN} \, . \tag{2}$$

The relation between the confusion matrix and the four quantities TP (true positive), FN (false negative), FP (false positive), and TN (true negative) is illustrated in Table 2.

Classification \ True	Shadow	Object | Background
Shadow	TP	FP
Non-shadow	FN	TN

Table 2: The fields of the confusion matrix in terms of TP (true positive), FN (false negative), FP (false positive) and TN (true negative).

In other words, TPR is the proportion of the shadow pixels detected as shadows and FPR is the proportion of the object and background pixels detected

as shadows.

Ideally, the algorithm should produce a TPR equal to 1 and a FPR equal to 0. There is, however, a trade-off between the hit rate and the fall-out and the goal will be to reach a high TPR and a low FPR.

These evaluation measures are more general than the confusion matrix in the sense that the false positive rate does not tell whether the false shadow detections originate from objects or background. Also, the false positive rate can be somewhat misleading since the background area generally is considerably greater than the object area. This means that FPR is dominated by the classification of background pixels and less sensitive to the classification of objects.

3.3.3 Visualizing the Results

It is also relevant where in the image a certain misclassification occurs. For example, if 50% of the shadow pixels are identified correctly, it is very important to know how these 50% are located in the image. It could be a severe problem if the algorithm is detecting about half of the different moving shadows. On the other hand, if every other pixel within the same shadow represents these 50% correct classifications, the relatively low detection rate can be easily and successfully helped by a morphology operation.

To understand where in the image misclassifications are made, many of the steps in the algorithm have been visualized as images. These steps can in many ways tell you more than just numbers. It is important though to remember that the eye is not consistent in its evaluation. The human brain draws conclusions based on experience and imagination which a computer is unable to.

4 Data

While developing the algorithms, two data sources were used. In the early work we used image sets from [12]. Since the final algorithm should be compatible with Axis surveillance cameras, these downloaded images were eventually extended with representative video sequences from Axis database.

When evaluating our algorithms, eight different video sequences were used with a total of 259 frames. Five of them are sequences from Axis (ATM, basement, burglar, lobby and parklot) while the remaining three (corridor, parking and traffic) were taken from [12]. The pixel resolution of the video sequences varied from 135×240 low resolution to 540×720 high resolution.

To be able to evaluate the algorithms we needed ground truth images for the various video sequences, indicating where the shadows, objects, and background are located in each frame. The image sets downloaded from [12] were already provided with a reference image with labeled objects and shadows. The same kind of ground truth was obtained for the video clips from Axis by first subjecting them to Axis object tracking algorithm [3], producing foreground images with labeled objects. This foreground includes object and background, and in

some cases shadows incorrectly identified as object. A correct ground truth for these image sets was made manually by marking the regions with shadows in MATLAB.

5 Algorithms

Three principal algorithms have been implemented in this thesis. To begin with, a rather simple, one featured algorithm based on the affine intensity relation described in [1] was implemented. A somewhat more complex algorithm, making use of intensity, chromaticity, and texture, was then developed in order to increase the precision. This algorithm rely on thresholding to separate shadows from objects and background. At last, to obtain a more dynamic shadow detection, additionally one algorithm was invented. This was inspired by the earlier algorithm, but extended with a classification based on probability functions instead of thresholding. All implementation was performed in MATLAB (R2012b).

5.1 Affine Intensity Relation

One of the most promising and simple algorithms is described in [1]. An implementation was made in order to see how the algorithm would perform in the context of this master thesis.

The algorithm relies on the assumption that the light energy L received at a point r in a neighborhood n_q in the absence of shadow is approximately affinely related to the light energy L^* received when the neighborhood is covered by shadow. Together with the general presumption that less light energy is received under shadow, this relationship forms the basis for the algorithm. Mathematically, the conditions are

$$L^*(r) \approx \lambda L(r) + \mu \tag{3}$$

and

$$L^*(r) < L(r) \ , \tag{4}$$

for some constants λ and μ.

The affine relation (3) holds for all points within a local neighborhood. It is therefore natural to implement the algorithm in a block-wise manner, checking the luminance conditions in block pairs corresponding to the frame and background.

The received light energy, L, is represented as one intensity value in each pixel. A block of gray scale pixel values is denoted P. If P^{fr} is a block in the frame and P^{bg} is a block in the background, equation (3) yields

$$\bar{P}^{fr} \approx \lambda \bar{P}^{bg} + \mu \ . \tag{5}$$

In words; the mean intensity value in the frame block is affinely related to the mean intensity value in the background block.

In equation (5) there are two unknown parameters, λ and μ, which should indicate whether P^{fr} is shadowed or not. In order to isolate these parameters and calculate their values, one more equation is needed. The rules of standard deviation σ give the following equation:

$$\sigma(P^{fr}) = \lambda\sigma(P^{bg}) \ . \tag{6}$$

From this information, the algorithm can be explained in a stepwise manner:

(a) Create blocks, P^{fr} and P^{bg}, of size $startSize$, for example 16×16 pixels.

(b) Calculate the mean intensity, $\bar{P}^{fr}$ and $\bar{P}^{bg}$, in all blocks.

(c) Set $\lambda = \frac{\sigma(P^{fr})}{\sigma(P^{bg})}$.

(d) Set $\mu = \bar{P}^{fr} - \lambda\bar{P}^{bg}$.

(e) Calculate $ratio1 = \frac{\bar{P}^{fr}}{\bar{P}^{bg}}$, which represents the intensity change between background and frame.

(f) Calculate $ratio2 = \frac{\|P^{fr} - (\lambda P^{bg} + \mu)\|_2}{\|P^{bg}\|_2}$, which represents how far the frame is from being an affine transformation of the background.

(g) Determine if the block is shadowed or not:

  (i) If $ratio1 < 1$ and $ratio2 \approx 0$, define the block as a shadow.

  (ii) If not, continue to search for smaller shadows within the block. Divide the block in four and repeat from step (b) until $blockSize = stopSize$.

Probability Map In step (g) the algorithm is able to return a probability map instead of a binary *isShadow*-matrix. The probability for a block being a shadow is assumed to consist of two normal distributions with expected values approximated by the values of $ratio1$ and $ratio2$ in a typical shadow block. The shadow probability in a block of size $stopSize$ represents the accumulated probabilities from all the levels.

5.2 Intensity, Chromaticity and Texture using Thresholding

After implementing the algorithm above it was of interest to see what could be achieved if more features were used. A promising method described in [11] uses intensity, chromaticity and texture properties to identify shadows. Inspired by this method, a new algorithm was implemented.

In order to cut down on memory and complexity, this algorithm is implemented stepwise where the finding of shadows in one step is dependent on the shadows found in the previous step. The decisions are made by thresholding the feature values into shadow or non-shadow.

5.2.1 Intensity & Chromaticity

In the first part of this algorithm, intensity and chromaticity information is used to identify shadows. The pixels classified as shadows constitute a candidate shadow mask that can later be passed on to the texture algorithm in Section 5.2.2 in order to be more discriminating towards objects.

The algorithm is based on the assumption that shadow pixels have similar chromaticity in the frame and background while the intensity is lower in the shadow. The intensity condition here is a more basic one than the affine condition in Section 5.1. The intensity change is simply represented as a ratio between the intensity in the frame and the intensity in the background.

When measuring the chromaticity change, the most natural approach utilizes the HSV color space since the change in hue (H) is a direct measure of the change in chromaticity. This HSV approach, however, is not appropriate in this thesis since the color space normally used in video handling is YCbCr rather than HSV. The translation between the two spaces is computationally expensive and therefore not an alternative.

The hypothesis adapted to YCbCr is instead that a region under shadow does not change its color neither in red nor in blue, i.e. Cb and Cr are both constant. This however does not apply to outdoor scenes subjected to strong shadows. In this case, the chromaticity change more towards blue components. To account for both phenomenas, a combination of the two hypotheses was implemented.

The steps of the algorithm:

(a) Calculate $yRatio = \frac{Y^{fr}}{Y^{bg}}$ for each pixel. Note, this ratio describes the intensity reduction just like $ratio1$, but pixelwise instead of blockwise.

(b) Define $yIsShadow = \{yRatio > \alpha \ \& \ yRatio < 1\}$. This mask consists of pixels that have become somewhat darker (as determined by α), i.e pixels that are shadow or moderately dark objects.

(c) Calculate $cbDiff = Cb^{fr} - Cb^{bg}$ and $crDiff = Cr^{fr} - Cr^{bg}$ for each pixel.

(d) Define $chromeIsShadow = \{|cbDiff| < \gamma\} \cap \{|crDiff| < \delta\}$. This mask should contain all pixels where the chromaticity remains unchanged, i.e all pixels that are background or weak shadow.

(e) Create $weakShadows = chromeIsShadow \cap yIsShadow$. This mask is supposed to only consist of weak shadow pixels.

(f) Define $strongShadows = \{yRatio < \alpha\} \cap \{CbDiff > 0\} \cap \{CrDiff < 0\} \cap \{CbDiff + CrDiff \approx 0\}$, picking up strong bluish shadows.

(g) Create $shadows = weakShadows \cup strongShadows$.

(h) Transact a morphological operation on *shadows* in order to get rid of scattered shadow pixels.

Adaptive Parameter Setting In order for the algorithm to be more general to weak and sharp shadows, histograms are used to set adaptive threshold values instead of using hard coded thresholds. This also allows for the background model not to be perfect.

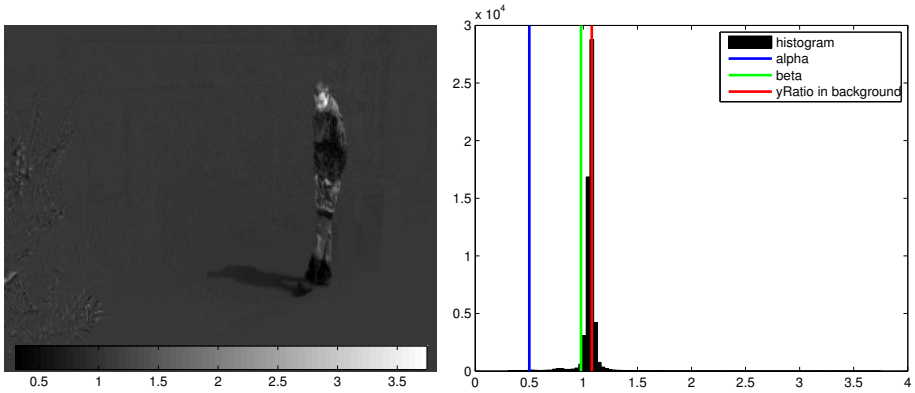


Figure 7: $yRatio$ values depicted as an image and a histogram.

The thresholding in step (b) assumes that $yRatio$ in a background pixel is 1. In order to be more persistent to variations in exposure between the current frame and the background model, a refined version of the algorithm applies instead the following condition: $yIsShadow = \{yRatio > \alpha \ \& \ yRatio < \beta\}$, where β is set dynamically.

By investigating the histogram of $yRatio$, see Figure 7, the value of $yRatio$ in a background pixel is assumed to be the value of the peak. This is because of the assumption that the background pixels dominate the image. $yRatio$ for the background pixels is distributed around the peak value, so to leave room for some noise, the following is set: $\beta = yRatio^{peak} - \epsilon$. α remains static.

5.2.2 Components & Texture

To remove object regions that were included in the candidate shadows, one extra condition is applied. This condition relies on the assumption that regions under shadow keep most of their texture. There are two important parts of this algorithm; dividing the candidate shadows into components and measuring the texture correlation.

The intensity and chromaticity features are evaluated pixelwise. However, the texture is better evaluated within larger regions. Before investigating the texture, the pixels are therefore gathered in components. A component specifies a continuos area in the image and the corresponding area in the background. It is highly important to divide shadows and objects in different components as the subsequent algorithm will compare the texture in each connected component and classify the entire component as either shadow or non-shadow. Ideally a component consists of an entire shadow or object.

In order to separate regions of shadows and objects, edges between these regions

need to be extracted and removed from the candidate shadow mask. Having removed these edges, the regions of objects and shadows should be unconnected and therefore divided into different components.

The texture is measured by calculating the gradient of the intensity in each pixel of the frame and background. The gradients with large enough magnitude are selected to represent the texture within a component. Having selected pixels with large gradient magnitude, the difference in gradient direction between the frame and the background is calculated for each selected pixel.

The gradient direction correlation between the frame and the background is estimated for each component, based on the total difference in gradient direction within the component. Finally, components with high correlation are considered shadow regions. The steps of the algorithm are presented below.

(a) Extract edges that exist in the frame but not in the background using Canny edge detection [7]: $edgediff = canny^{fr} - canny^{bg}$. Extrapolate to get continuous edges.

(b) Remove the edges from the candidate shadows and apply morphological operations.

(c) Divide the candidate shadow mask into components with the MATLAB function `bwlabel`.

(d) For each component:

  (i) Extract pixels with high gradient magnitude. For those pixels calculate $gradDirDiff = arctan(\frac{\delta Y}{\delta y}/\frac{\delta Y}{\delta x})^{fr} - arctan(\frac{\delta Y}{\delta y}/\frac{\delta Y}{\delta x})^{bg}$.

  (ii) Calculate the mean value of $gradDirDiff$ within the component.

  (iii) Determine whether the component is a shadow or an object by thresholding. If $mean(gradDirDiff) \leq \tau$, for some threshold τ, the component is considered a shadow.

5.3 Intensity, Chromaticity and Texture using Graph Cuts

Instead of thresholding, a probabilistic view is now introduced. This algorithm employs graph cuts to make the classification in terms of energy minimization. The algorithm is divided into two parts.

First, a training set is used to model how pixels in the different classes behave regarding intensity, chromaticity, texture and spatial dependency. These models are used to produce weights for each class, punishing uncommon behavior. When training the algorithm, the ground truth is used in order to separate the classes.

Secondly, the estimated weights can be used to classify pixels in an unseen video frame. The optimal classification of the image is calculated with graph cuts, minimizing the total cost as determined by the weights (see Section 3.1 for more details on graph cuts).

5.3.1 Deriving Weights

The algorithm stepwise:

(a) Calculate feature weights from the training data.
For each $feature\ measure$ ($yRatio$, $chromeCriterion$ and $gradDirDiff$) :

(i) Create a histogram over the feature values in each of the three classes shadow, background, and object.

(ii) Estimate the probability distribution for each class based on the histograms:

$$P(feature\ value) = \frac{sum(pixels == feature\ value)}{sum(pixels)} .$$

Examples of probability distributions can be seen in the left column in Figure 8.

(iii) Convert the probability distributions into integer weights, punishing feature values with low probability by higher weights.

$$weight(feature\ value) = round[k(-log(P(feature\ value))] .$$

k is a number set automatically to obtain integer weights. Examples of derived weights are shown in the right column in Figure 8.

(b) Calculate spatial weights from the training data:

(i) Estimate the probability that a pixel has at least one neighbor of the same class by simply counting the number of pixels that belong to the same class as a pixel next to it. The neighborhood connectivity applied is the commonly used 4-connected, where each pixel is considered to have four neighbors (vertically and horizontally).

(ii) Convert this probability into a weight, by taking the negative logarithm of the probability as in a) iii).

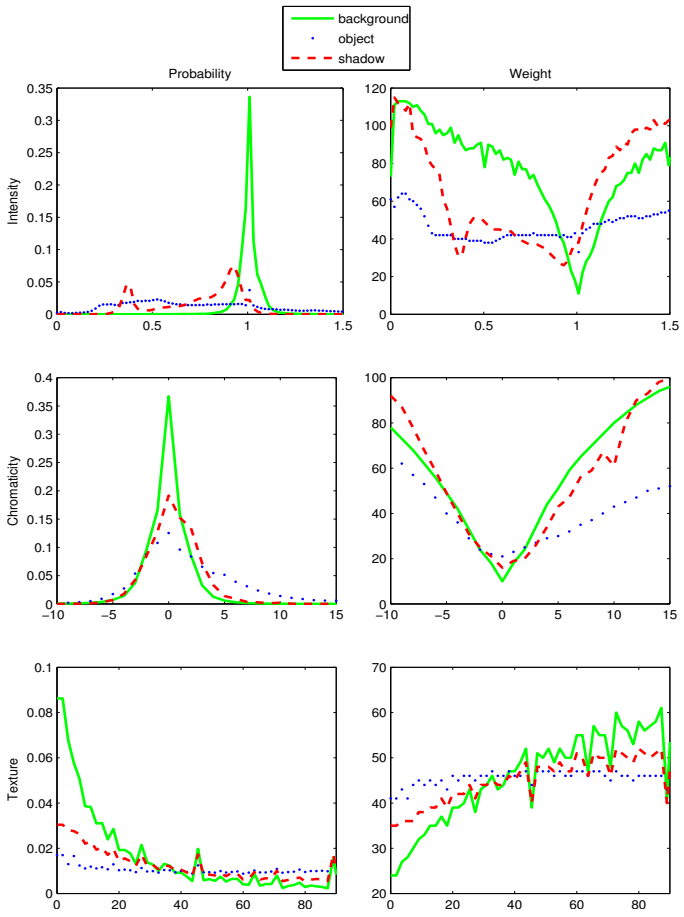


Figure 8: Probability distribution and weights for the three features intensity, chromaticity and texture. These probabilities and weights are based on all eight video sequences and a total of 259 frames. Intensity is represented by $yRatio$, chromaticity by $cbDiff + crDiff$, and texture by $gradDirDiff$.

Designing Weights The histograms, probability distributions and resulting weights for the three features display characteristics of the scenes from which they are generated. Since the number of frames used to produce the weights are rather limited, the probability distributions and consequently the weights are somewhat noisy. Using a larger training set, these curves would presumably be more continuous.

Having studied the diagrams in Figure 8, the next approach was to design the probability distributions ourselves. This design was based on the derived distributions but adjusted to remove noise and irregularities to better approximate the presumed behavior of the features. The designed probability distributions and corresponding weights are displayed in Figure 9.

Apart from smoothing the distributions to get more continuous weights, the major intervention lies in the probability distribution of the intensity, i.e. $yRatio$,

in shadow. The shadow distribution of $yRatio$ is assumed to be rather constant in the interval between 0.4 and 0.9 in order to represent both weak and strong shadows. Compare the red graph in upper left corner of Figure 8 and Figure 9. Another adjustment is the distribution of the chromaticity of objects. This curve is supposed to be constant as objects in general should have an arbitrary chromaticity.

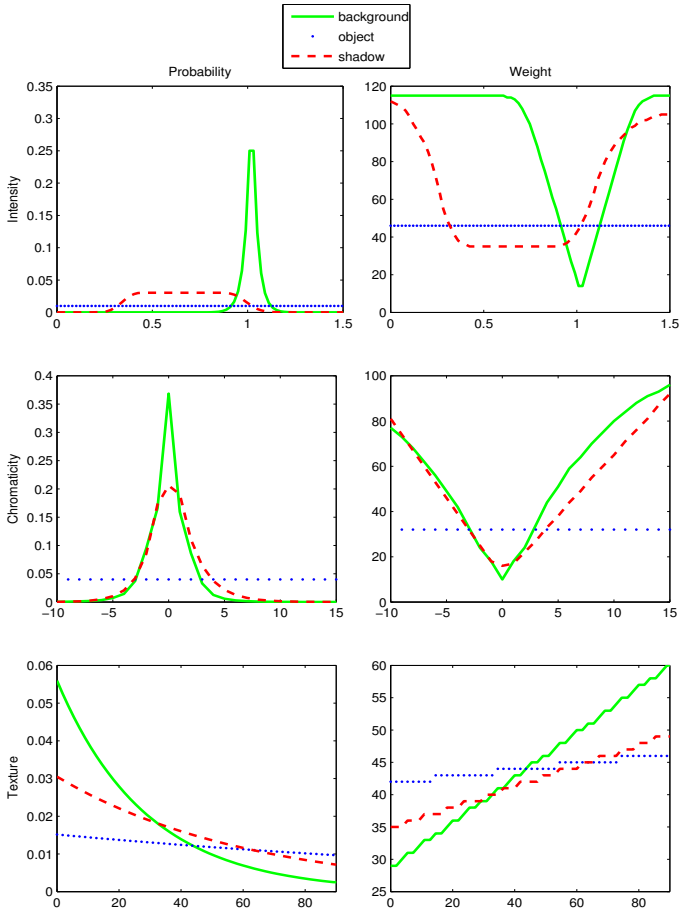


Figure 9: Designed probability distribution and weights for the three features intensity, chromaticity and texture. Intensity is represented by $yRatio$, chromaticity by $cbDiff + crDiff$, and texture by $gradDirDiff$.

5.3.2 Segmentation using Graph Cuts

Using the weights derived in the previous section, an optimal classification is calculated. The classification can be based on one, two or all three of the features intensity, chromaticity and texture. The algorithm stepwise:

(a) For each pixel and selected feature, get the shadow, object, and background weights corresponding to the pixel's feature value.

(b) Add the weights for all selected features and construct a data cost matrix of dimension $3 \times nbrOfPixels$ with three costs for each pixel; one for shadow, one for object, and one for background.

(c) Define a (sparse) matrix of dimension $nbrOfPixels \times nbrOfPixels$ with neighborhood costs. The 4-connected sites are assigned with costs equal to the spatial weights.

(d) Make the most optimal classification based on the data costs and neighborhood costs. An open source implementation of this multi-label optimization step is described in [4] [5] [10].

6 Results

In this section, results from the various algorithms are presented for the example image displayed in Figure 10. This image is taken from Axis ATM video sequence. A comparison betweem the algorithms is presented in Section 6.4.

(a) Example frame.

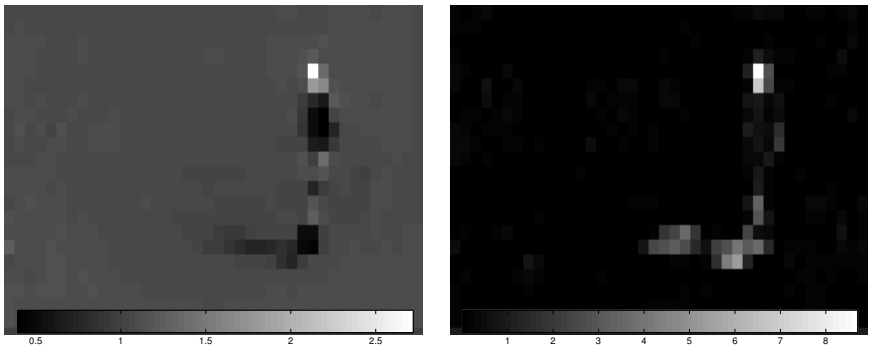
(b) Ground truth. Gray represents shadows, white objects, and black static background.

Figure 10: This image from the ATM video sequence and the corresponding ground truth will be used to show stepwise results.

6.1 Affine Intensity Relation

The two ratios, $ratio1$ and $ratio2$, from the affine intensity algorithm are visualized in Figure 11. As mentioned earlier, $ratio1$ is supposed to be less than one and $ratio2$ near zero for shadow pixels. As seen in Figure 11 (b) the assumption on $ratio2$ does not hold very well.

In Figure 12 the final result is shown, using $startSize = 8$ and $stopSize = 4$. The binary shadow mask is displayed in Figure 12 (a) and the corresponding probability map in Figure 12 (b). In Table 3, the confusion matrix for the binary shadow results is displayed. These numbers are an average of the confusion matrices for all eight scenes. The binary shadow mask in Figure 12 includes most shadow pixels but also some object blocks. However, as seen in the confusion matrix representing a wider range of scenes, the general result reveals the algorithm's difficulty in separating between objects and shadows.



(a) $ratio1$. (b) $ratio2$.

Figure 11: Ratios of the affine intensity algorithm described in Section 5.1.

(a) Binary shadow mask. (b) Probability map for shadow, bright is high probability.

Figure 12: Shadows detected by the affine intensity algorithm.

Classification \ True	Shadow	Object	Background
Shadow	52.0621 %	34.8585 %	3.8738 %
Non-shadow	47.937 %	65.1415 %	96.1262 %

Table 3: Confusion matrix using the affine intensity algorithm. These numbers are an average of the confusion matrices for all eight scenes.

6.2 Intensity, Chromaticity and Texture using Thresholding

6.2.1 Intensity & Chromaticity

In Figure 13 (a), $yRatio = \frac{Y^{fr}}{Y^{bg}}$ is visualized. $yRatio$ should be equal to one in background pixels since $Y^{fr} = Y^{bg}$ there. It is also expected that shadow pixels have a $yRatio$ value less than one and that object pixels are distributed over the entire range. These assumptions are confirmed in Figure 13 (a).

The pixels that satisfy the shadow condition on $yRatio$ are shown in Figure

13 (b) and (c). In Figure 13 (b), the upper threshold is static, while in Figure 13 (c), $yIsShadow$ is generated using adaptive parameter setting decreasing the background noise substantially. The corresponding confusion matrices based on an average of our eight scenes are shown in Table 4 and Table 5. The desired reduction of background noise in Table 5 was achieved with the adaptive parameter setting. The simultaneous reduced shadow detection percent originates mainly from the loss of weak shadow boundaries, compare Figure 13 (b) and (c).

(a) $yRatio$.

(b) $yIsShadow = \{yRatio > \alpha \ \& \ yRatio < 1\}$.

(c) $yIsShadow = \{yRatio > \alpha \ \& \ yRatio < \beta\}$, where β is adaptive.

Figure 13: $yRatio$ and shadows defined by the intensity criterion, before and after the improvement of adaptive parameter setting.

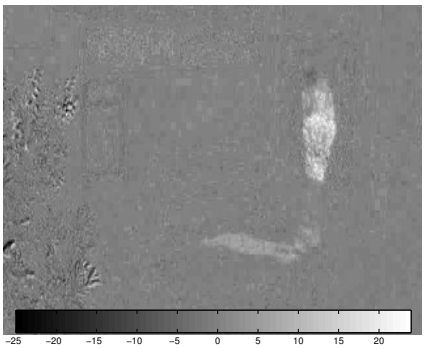
Classification \ True	Shadow	Object	Background
Shadow	77.8818 %	45.3013 %	11.5267 %
Non-shadow	22.1182 %	54.6987 %	88.4733 %

Table 4: Confusion matrix using only intensity feature.

Classification \ True	Shadow	Object	Background
Shadow	70.8966 %	43.2192 %	4.5830 %
Non-shadow	29.1034 %	56.7808 %	95.4170 %

Table 5: Confusion matrix using the intensity feature with adaptive parameter setting.

Applying the hypothesis that both Cb and Cr should be rather constant in shadow pixels gives an expected value of $CbDiff$ and $CrDiff$ near zero, compare these assumptions with Figure 14 (a) and (b). The corresponding $chromeIsShadow$ mask is shown in Figure 15 (a). In this frame, a rather big part of the object is successfully extracted and can thus be subtracted from the intensity shadow mask. The resulting shadows, shown in Figure 15 (b), contains less object compared to the shadow mask in Figure 13 (c) based on intensity only. Despite discriminating objects, information about chromaticity has also been used to find strong shadows that were not included in the intensity mask due to a too large attenuation. The increased shadow detection percentage adding the strong shadows can be seen comparing Table 6 and 7.



(a) $cbDiff$ = difference in Cb between the frame and background.

(b) $crDiff$ = difference in Cr between the frame and background.

Figure 14: Chromaticity measures.

(a) The chromaticity mask *chromeIsShadow*. The black areas do not satisfy the chromaticity condition and should thus represent objects.

(b) The resulting *shadows*, representing the union between *weakShadows* and *strongShadows*.

Figure 15: The chromaticity mask and shadows defined by intensity and chromaticity.

Classification \ True	Shadow	Object	Background
Shadow	66.9966 %	27.0768 %	4.4890 %
Non-shadow	33.0034 %	72.9232 %	95.5110 %

Table 6: Confusion matrix for *weakShadows* based on an average of all scenes.

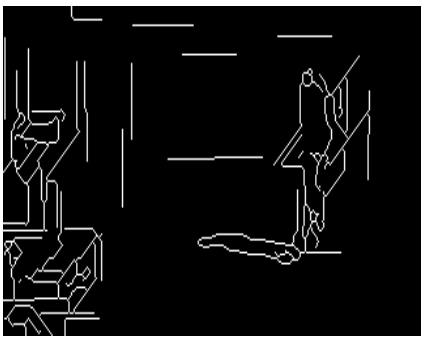
Classification \ True	Shadow	Object	Background
Shadow	75.6671 %	31.9542 %	4.4968 %
Non-shadow	24.3329 %	68.0458 %	95.5032 %

Table 7: Confusion matrix after adding *strongShadows* to *weakShadows* based on an average of all scenes.

6.2.2 Components & Texture

Below, the steps performed to divide the candidate shadows from Figure 15 (b) into components are illustrated. The frame edges extracted with Canny edge detection are visualized in Figure 16 (a) and the corresponding extrapolated edges in Figure 16 (b). Removing the extrapolated edges from a morphed version of the candidate shadow mask, results in the component partitioning shown in Figure 17 (a). This plot also visualizes the texture correlation for each component. A dark component has a low correlation between background and frame, and is thus probably an object. The resulting shadow mask using intensity, chromaticity, and texture is displayed in Figure 17 (b). In this frame, the algorithm is successful in separating shadows from objects and background.

(a) $edgediff = canny^{fr} - canny^{bg}$. $edgediff$ represents edges that exist in the frame but not in the background.



(b) Extrapolated $edgediff$ to get continuous edges.

Figure 16: Edges extracted in order to divide the candidate shadows into connected components.

(a) The resulting components and their texture correlation. Brighter color means stronger correlation.

(b) The final result using intensity, chromaticity and texture.

Figure 17: The components, colored according to their texture correlation value, and the resulting shadows.

The confusion matrix using intensity, chromaticity and texture is shown in Table 8 below. Compared to the confusion matrix in Table 7, the number of misclassified object and background pixels has been reduced substantially.

Classification \ True	Shadow	Object	Background
Shadow	65.4869 %	14.637 %	1.6363 %
Non-shadow	34.5131 %	85.3630 %	98.3637 %

Table 8: Confusion matrix using intensity, chromaticity and texture.

6.3 Intensity, Chromaticity and Texture using Graph Cuts

The segmentation using graph cuts is shown below. The previous algorithms classify each video frame into shadow or non-shadow. This algorithm, however, produces a full segmentation that separates between object and background as well. The results are therefore comparable to the ground truth in Figure 10 (b) with shadows in gray and objects in white.

In Figure 18 we show training results where one frame from the ATM sequence has been labeled based on weights derived from the entire ATM sequence. These results give an indication ot the potential of the graph cut algorithm. In Figure 19 the same ATM frame has been labeled based on weights derived from a training set consisting of all sequences but ATM.

(a) Training result using only intensity.

(b) Training result using intensity and chromaticity.

(c) Training result using intensity, chromaticity and texture.

Figure 18: Segmentation with graph cuts after training the algorithm on the ATM sequence. Gray represents shadow, white represents object, and black represents background.

Confusion matrices for the test results can be found in Table 9, 10 and 11. The numbers are based on an average of the confusion matrices for all eight sequences. For each sequence, the weights are derived from every sequence but the one being classified. Adding more features give a higher rate of detected shadows but also more misclassified objects.

(a) Test result using only intensity.

(b) Test result using intensity and chromaticity.

(c) Test result using intensity, chromaticity and texture.

Figure 19: Segmentation with graph cuts after training the algorithm on all sequences but ATM. Gray represents shadow, white represents object, and black represents background.

Classification \ True	Shadow	Object	Background
Shadow	59.9708 %	28.584 %	3.8633 %
Non-shadow	40.0292 %	71.416 %	96.1367 %

Table 9: Confusion matrix applying graph cuts with weights based on intensity only.

Classification \ True	Shadow	Object	Background
Shadow	60.8658 %	34.7542 %	3.2118 %
Non-shadow	39.1342 %	65.2458 %	96.7882 %

Table 10: Confusion matrix applying graph cuts with weights based on intensity and chromaticity.

Classification \ True	Shadow	Object	Background
Shadow	64.6401 %	44.3151 %	3.006 %
Non-shadow	35.3599 %	55.6849 %	96.994 %

Table 11: Confusion matrix applying graph cuts with weights based on intensity, chromaticity and texture.

The segmentation using the manually designed weights in Figure 9 is shown in Figure 20. These results should be compared with the segmentation in Figure 19 based on the derived weights. The major difference is the increased number of object pixels being misclassified as shadows. This is also reflected in the confusion matrices in Table 12, 13 and 14. The confusion matrices also reveals a higher shadow detection rate using designed weights.

(a) Segmentation using weights designed for intensity.

(b) Segmentation using weights designed for intensity and chromaticity.

(c) Segmentation using weights designed for intensity, chromaticity and texture.

Figure 20: Segmentation with graph cuts and designed weights. Gray represents shadow, white represents object, and black represents background.

Classification \ True	Shadow	Object	Background
Shadow	82.0428 %	56.9410 %	2.9145 %
Non-shadow	17.9572 %	43.0590 %	97.0855 %

Table 12: Confusion matrix applying graph cuts with weights designed for intensity.

Classification \ True	Shadow	Object	Background
Shadow	81.8453 %	62.2612 %	2.6362 %
Non-shadow	18.1547 %	37.7388 %	97.3638 %

Table 13: Confusion matrix applying graph cuts with weights designed for intensity and chromaticity.

Classification \ True	Shadow	Object	Background
Shadow	80.4103 %	63.4737 %	2.5148 %
Non-shadow	19.5897 %	36.5263 %	97.4852 %

Table 14: Confusion matrix applying graph cuts with weights designed for intensity, chromaticity and texture.

6.4 Comparison

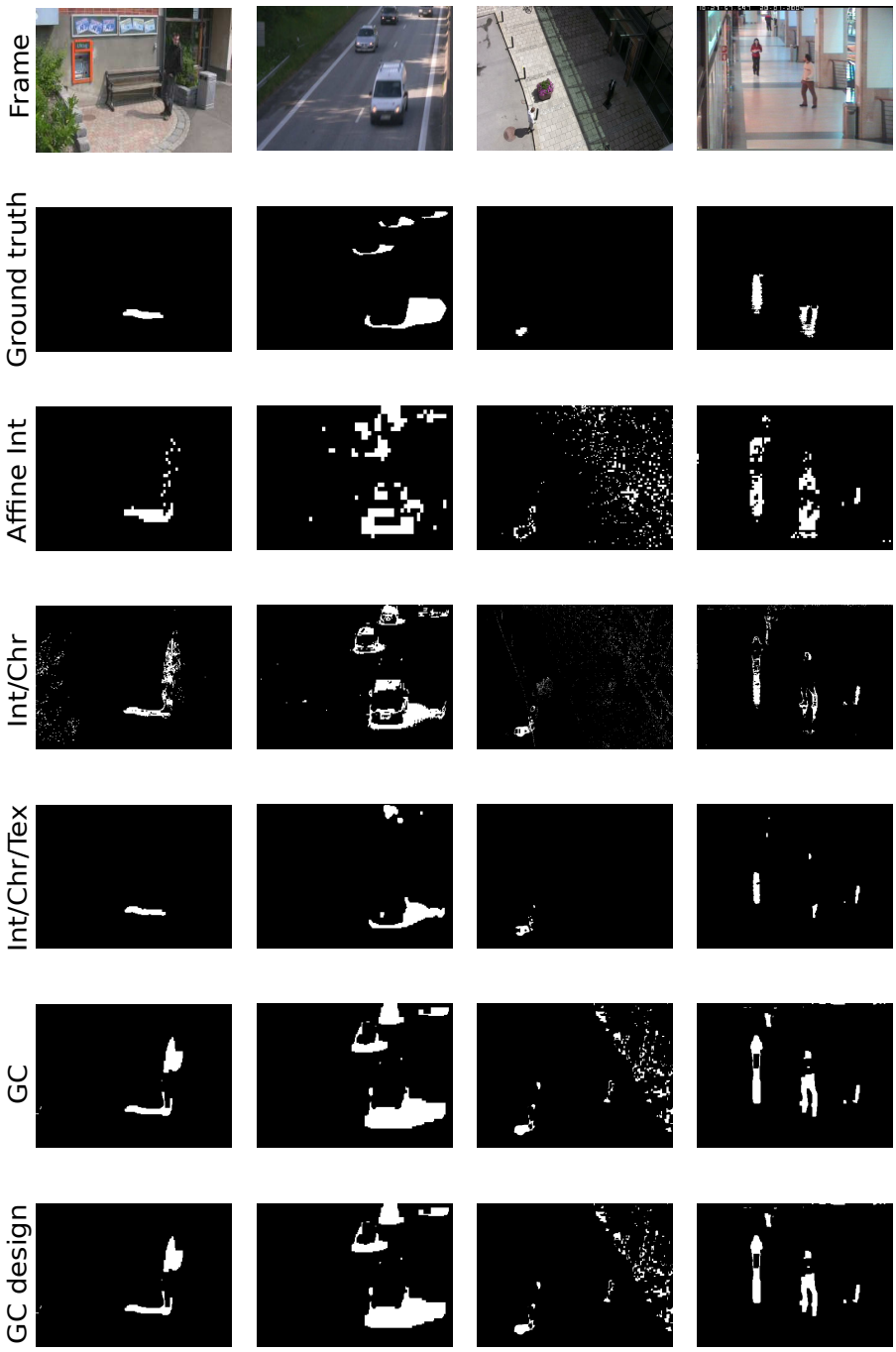
In Figure 21 and 22, a comparison between the different algorithms is displayed for the eight different scenes. The corresponding true positive rate (TPR) and false positive rate (FPR) for all evaluated methods are displayed in Table 15. These values are calculated as an average of the TPR and FPR for the eight scenes.

In this section we refer to the various algorithms using the following abbreviations: Int (intensity), Chr (chromaticity), Tex (texture), GC (graph cuts).

Algorithm	**TPR** (%)	**FPR** (%)
Affine Int	52.0621	5.1719
Int	70.8966	6.2576
Int/Chr	75.6671	5.5464
Int/Chr/Tex	65.4869	2.1951
GC Int	59.9708	4.8108
GC Int/Chr	60.8658	4.4191
GC Int/Chr/Tex	64.6401	4.5053
GC design Int	82.0428	5.2021
GC design Int/Chr	81.8453	5.1439
GC design Int/Chr/Tex	80.4103	5.0380

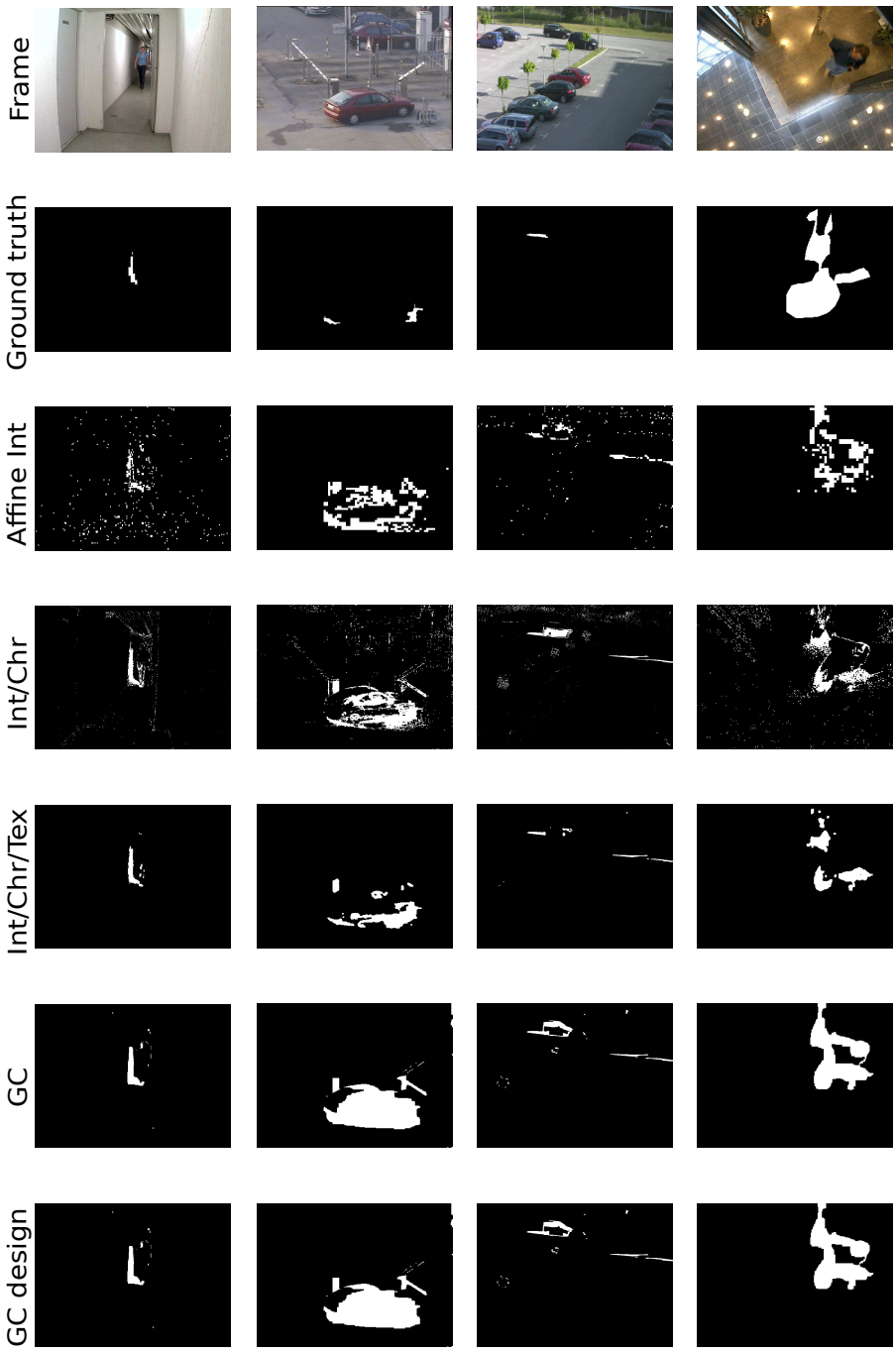
Table 15: TPR and FPR for the evaluated methods. The numbers are based on an average of the TPR/FPR for all eight scenes.

Prioritizing a low FPR in order to not suppress real motion triggered alarms, the algorithm Int/Chr/Tex from Section 5.2.2 is considered the best. Although the graph cuts algorithm with designed weights has significantly higher TPR, the small increase in FPR is more severe than it might seem. Since FPR is the proportion of the object and background pixels incorrectly classified as shadows, and the number of background pixels totally dominates the frame, the large number of misclassified object pixels using graph cuts is concealed in the false positive rate.



(a) ATM (b) Traffic (c) Burglar (d) Corridor

Figure 21: Shadow detection results in four different scenes for five of the evaluated algorithms. The following abbreviations are used: Int (intensity), Chr (chromaticity), Tex (texture), and GC (graph cuts).



(a) Basement (b) Parking (c) Parklot (d) Lobby

Figure 22: Shadow detection results in four different scenes for five of the evaluated algorithms. The following abbreviations are used: Int (intensity), Chr (chromaticity), Tex (texture), and GC (graph cuts).

7 Discussion

The main challenge in this master thesis has been to design a shadow detection algorithm that is able to extract shadows in every possible scene. In a specific scene and lighting, the shadows introduce similar changes with respect to the intensity and chromaticity. With specific values of these features, the shadows are easy to extract by thresholding. However, when adjusting the algorithm to work for other sequences and scenes, the range of the feature values gets much wider. Extracting only shadows, without also including objects or background, is then a very hard problem when applying thresholding.

As an example, the intensity change of a pixel covered by shadow, varies a lot depending on the lighting of the scene. A sunny day, the intensity change under shadow can be quite substantial while it might be barely visible a cloudy day or in a poorly illuminated indoor scene.

The problem with objects slipping through as shadows was expected to be helped by applying more features discriminating in different directions. The added features, however, proved to be more complex than expected. Overall, the impression is that intensity gives the most valuable information when trying to detect shadows, especially considering the low complexity of this feature compared to the other features.

7.1 Affine Intensity Relation

The algorithm based on the affine intensity relation was implemented as a first attempt to see how far we could get with a rather basic method. Although, since the algorithm detects shadows recursively in progressively smaller blocks, the computational load varies a lot depending on the chosen $startSize$ and $stopSize$.

The results in Figure 12 and Table 3 were not as satisfying as expected. Clearly, this algorithm has difficulties separating between shadows and objects. The trouble consists in finding suitable thresholds for $ratio1$ and $ratio2$. According to the theory, $ratio1$ should be below one and $ratio2$ near zero for a block to be defined as a shadow. Learning more about the effects of the ratios, we discovered that $ratio1$ could use a lower bound as well. This lower bound prevented some of the objects to slip through as shadows, but at the same time we lost some of the stronger shadows.

$ratio1$, representing the intensity reduction in shadows, seems to give more valuable information than $ratio2$, representing the accuracy of the affine condition in Equation (5). Visualizing $ratio2$ as in Figure 11 (b) revealed weaknesses in the affine intensity hypothesis. There was a bigger variation of the ratio within a shadow than between shadows and objects. Therefore it was impossible to find a threshold for $ratio2$, separating the two classes.

7.2 Intensity, Chromaticity and Texture using Thresholding

Modifying the intensity hypothesis and adding conditions on chromaticity, we aimed at solving some of the problems above. Measuring the intensity attenuation pixelwise instead of blockwise proved to be a big improvement. However, applying chromaticity as an indicator for shadow was a lot harder than expected.

It was really difficult to find a good measure of the chromaticity change in YCbCr since the hue in HSV had no direct counterpart. None of the tested conditions for preserving the chromaticity gave satisfying results in all possible scenes. However, we believe that this is a consequence of false assumptions rather than the measure itself.

One weakness, applying the theory in practice, is the need for a good background model since the chromaticity is very sensitive to small changes in color and white balance. We use as background model a frame taken from the beginning of the video sequence, before any objects or shadows has emerged in the scene. Since the camera changes exposure between frames frequently depending on variations in the scene, the background model is rarely completely up to date relative to the current frame. As a consequence, the hypothesis on constant chromaticity in shadows does not necessarily hold when comparing the frame to the background. This can be seen comparing Table 5 to Table 6 where the shadow detection rate has decreased from 70.9% to 67.0% due to the shadows not fulfilling the chromaticity condition.

Another weakness was, as expected, that the chromaticity behaves differently in outdoor and indoor scenes. Particularly, the light shifts towards blue under shadows in outdoor scenes. If the algorithm was to be applied only indoors or outdoors, the measure could have been adjusted to fit the specific conditions of that scene.

Adding texture as further indication for shadow, some of the previously misclassified object pixels were excluded from the shadow mask. Compare Figure 15 (c) and Figure 17 (c) to see the striking improvement. The advantage of evaluating the texture in components rather than in blocks is that the evaluated region then consists of only one class. This, however, relies on a skilled component partitioning which was a challenging mission. The task is delicate as it ideally should extract the borders which separate shadows from objects, whereas this in some sense is the goal with the algorithm as a whole.

The drawback with the texture as a feature is that it demands a somewhat high resolution in order for the video frames to preserve the texture information. It also demands that the shadow attenuation is not too strong, such that the texture gets lost due to the decreased contrast. For example, a relatively dark texture containing values between 0 and 50 might be attenuated to values between 0 and 5. This small intensity change is almost not notable and the texture is lost.

This texture loss is not the only problem with strong shadows. In fact, none

of the feature conditions are fully satisfied in strong shadows. The intensity change in strong shadows is hard to distinguish from an object, since it is so pronounced. Furthermore the chromaticity is generally not preserved in strong shadows since a low intensity gives more color neutrality.

7.3 Intensity, Chromaticity and Texture using Graph Cuts

Having spent a lot of time evaluating and trying to improve the application of the affine intensity relation in Section 5.1 and the different features of the algorithm in Section 5.2, the rigidity of thresholding was exposed. To make better use of these features, the probabilistic view in Section 5.3 was introduced. This approach gave us valuable knowledge of the behavior of the different features. The probability distributions in Figure 8 reveal whether the features behave as expected and give some indication of the potential of the features.

As expected, *yRatio* performs quite well as a measure of the intensity change in the frame compared to the background, see upper left corner in Figure 8. The probability distributions of *yRatio* look like expected; shadow has a high probability somewhere between 0.4 and 1, background has a peak near 1, and the probability for objects are spread out over a larger range. The peak in the shadow distribution around 0.4 originate from a dominant scene with strong shadows while the peak around 0.9 originate from indoor and outdoor scenes with more weak shadows. With a larger test data, reflecting all possible scenes and lighting, this shadow distribution would presumably be more constant between the two peaks.

The purpose with the chromaticity and texture features is that they should discriminate objects in order to achieve a low FPR. For the algorithm to be able to separate objects from shadows and background, it is necessary that there is a clear difference in the feature measure betweeen objects and the other classes. As seen in Figure 8, the chromaticity and texture distributions for objects are quite similar to the distributions for shadows and background. Thus, the chromaticity and texture as calculated here is not discriminative for shadows. Apparently, the texture feature is less informative using graph cuts than in the algorithm applying thresholding. One possible reason could be that the texture is evaluated pixelwise instead of in components, making the algorithm more sensitive to noise.

Theoretically the chromaticity distribution for objects should be more or less rectangular since the chromaticity of an object is rather arbitrary. This condition would also make the object distribution more unique and easy to separate from the other distributions. Although this was accounted for when designing weights, the expected improvement did not occur. Compare Table 9-11 with Table 12-14. Instead of discriminating objects, more of both object and shadow was found. A possible explanation for this could be that the training set used to derive weights resembles the evaluated scene and thereby fits the scene better than the general assumption accounted for when designing the weights.

8 Future Work

To further improve the results, a better data set could be valuable. A bigger variation of the scenes and a larger amount of frames would give more information and more trustworthy results. Better ground truth and background models would also make the results more trustworthy.

To increase the precision, more features could be added to the algorithms. Perhaps an estimate of the shape of objects and their attached shadows would make it easier to separate them. The challenge here would be to make the estimate of the shape general to all types of objects. Since shadows normally move continuously like their corresponding objects, another improvement could be to apply a temporal filter in order to avoid shadows rapidly emerging and disappearing from the shadow mask.

Another approach could be to divide shadows into different groups depending on their properties, such as weak versus strong shadows, and customize the algorithm to the characteristics of the scene. Allowing the algorithms to be less general simplifies both thresholding and estimating the probability distributions. The challenge lies in estimating the type of shadows present in the scene. A possible solution could be to separate between indoor and outdoor environment.

To reduce the number of false alarms evoked by shadows, another approach could be to make the classification using the object tracking mask. In that case the algorithm would only have to search for shadows within the mask.

Acknowledgements

This master thesis was done in collaboration between the R&D department Core Technologies Analytics & Systems at Axis Communications AB and the mathematical imaging group at the Faculty of Engineering at Lund University. Special thanks to our supervisors Gustav Träff and Martin Ljungqvist at Axis for their valuable opinions and guiding. Thanks also to Kalle Åström and Yubin Kuang from the imaging group who have contributed with lots of enthusiasm and great ideas. Finally, we would like to thank Axis for giving us the opportunity to do our master thesis with them in such an inspiring environment.

References

[1] N. AI-Najdawi, H. Bez, and E. Edirisinghe. A novel approach for cast shadow modelling and detection. In *Visual Information Engineering, 2006. VIE 2006. IET International Conference on*, pages 553 –558, sept. 2006.

[2] N. Al-Najdawi, H. E. Bez, J. Singhai, and E. Edirisinghe. A survey of cast shadow detection algorithms. *Pattern Recognition Letters*, 33(6):752 – 764, 2012.

[3] J. Almbladh and K. Netzell. Real-time illumination-invariant motion detection in spatio-temporal image volumes. *Proceedings - Applications of Computer Vision (WACV), IEEE Workshop on*, pages 691–696, 2011.

[4] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1124–1137, 2004.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:2001, 2001.

[6] W. Burger and M. Burge. *Digital image processing*. Texts in Computer Science. Springer London, Limited, 2008.

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

[8] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003.

[9] L. Gu and A. Robles-Kelly. Shadow detection via rayleigh scattering and mie theory. In *International Conference on Pattern Recognition*, page 4, Tsukuba /JAPAN, November 2012.

[10] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81, 2004.

[11] A. Sanin, C. Sanderson, and B. Lovell. Improved shadow removal for robust person tracking in surveillance scenarios. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 141 –144, aug. 2010.

[12] A. Sanin, C. Sanderson, and B. C. Lovell. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern Recognition*, 45(4):1684 – 1695, 2012.

Master's Theses in Mathematical Sciences 2013:E34
ISSN 1404-6342

LUTFMA-3250-2013

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/