ABSTRACT

Title of Thesis: SHADOW DETECTION IN VIDEOS ACQUIRED BY STATIONARY AND MOVING CAMERAS

Antonio Trias, Master of Science, 2005

Thesis directed by: Professor Rama Chellappa
Department of Electrical and Computer Engineering

Shadow Detection has become a key issue in object detection, tracking and recognition problems. Object appearances might be completely changed by the effects of shading and shadows. Finding good algorithms for shadow detection and reducing shading effects in order to segment objects from video sequences, will enhance the performance of our detection, tracking and recognition algorithms. In this thesis, we present data, physics and model-driven approaches for detecting shadows and correcting shading effects. The effectiveness of these algorithms in video sequences acquired by stationary surveillance cameras and airborne platforms is illustrated.

SHADOW DETECTION IN VIDEOS ACQUIRED BY STATIONARY AND MOVING CAMERAS

by

Antonio Trias

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2005

TABLE OF CONTENTS

Chapter 1

Introduction

The capability to extract moving objects from a video sequence is a fundamental problem in many computer vision applications such as surveillance, traffic monitoring, human-machine interface, tracking and recognition. A typical approach is to use background subtraction followed by object segmentation and tracking. Detection and tracking steps are interconnected. To have robust tracking we need a robust detector and using a good tracker we can overcome many problems in detection. Usually we do not know the shape of the moving objects. The detector output may be the only source of information about the object shape at an early stages of processing. That is why it is crucial to have a good detector of moving objects. Good detection not only leads to an improved tracker but can also improve other high level applications, like recognition or scene analysis. Frequent changes in the shape of moving object might mislead tracking or recognition algorithms. Many things can cause these changes but the ones we address in this thesis are those due to illumination variations. If we rely only on the color information of a moving object, it is sufficient to be concerned with illumination changes. But if object silhouettes are also used, dramatic changes in silhouettes due to different shadings should be address too. We are very interested in how the shape of a moving object is changed by the shadow that is attached to the object. We focus on shadow detection and delineation, so that the moving object is analyzed.

We can address shadow detection in many ways. One can retrieve the geometrical properties of shadows by considering the 3D structure of the object; we can analyze the physical properties of light itself and estimate the surface reflection properties under direct light and the shadowed surface; statistical pixel-based models can be also used to segment the foreground, shaded regions, background; etc. Finally that what we are looking for is a robust algorithm that can segment the shadow completely from the moving object so that the shape information that we need for high level analysis comes from the real object itself. For a typical static surveillance camera, the background can be easily estimated and as objects typically have enough pixels, many different model based methods can be used. On the other hand, for video acquired by an airborne platform, the number of pixels on moving vehicles and humans are typically of the order of a few tens of pixels. These detections are also quite sensitive to camera noise. Since the sensor is also moving we need to stabilize the image in order to be able to determine what is background and what is not. Typically, we use registration algorithms to stabilize the video and then perform background subtraction for moving object detection. An algorithm that is effective for video sequences acquired by a stationary camera may be completely ineffective for aerial video sequences.

In the figure 1.1, we present three image frames acquired by an aerial sensor, indoor and outdoor static cameras.

Shadow detection has other applications. Using shadows we can estimate the direction of the sun light at ground level (see figure 1.1), which can be used to

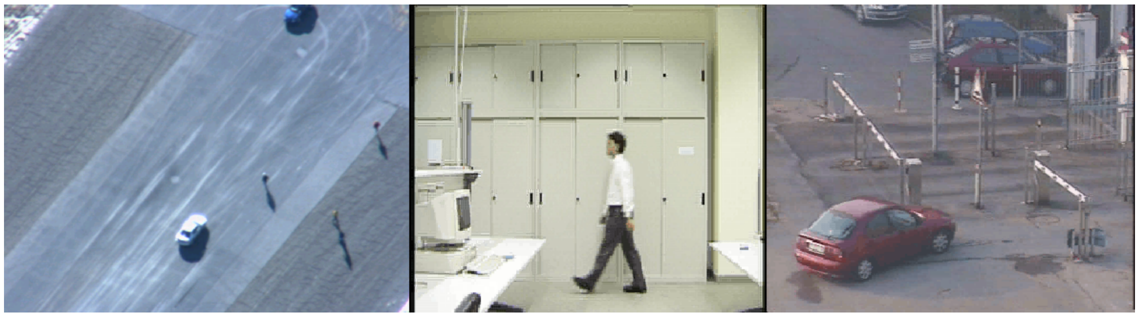

Figure 1.1: Examples

predict the shading of the object, or the other way around; we can also use the sun position to determine the likely direction of a human shadow.

The thesis is organized into three main chapters, each of which deals with a very different way of approaching the problem of shadow detection and segmentation. Whether the methods studied are effective for aerial images is also discussed in every chapter. We begin with statistical methods, as they have been studied for many years. Two types of statistical methods, parametric and non-parametric, have been studied. Parametric methods estimate the statistics of what we are trying to segment (shadows, background or foreground); the non-parametric methods use statistical properties to threshold, but does not explicitly parameterize the different classes. In chapter 3 we use the physics of light to transform the original image to an illumination invariant image. If we successfully do this, shading and shadows would no longer be issues to be taken care of. The thesis will end with a simple approach to the shadow detection problem that is effective for a specific aerial imaging scenario, to show that simple solutions can sometimes be effective.

As several approaches where taken to address the problem of shadow detection and ground truth is not available, it is very difficult to provide any quantitative performance comparison. That is why we have provided extensive experimental results in a visual way. Conclusions on the performance of all the attempted methods rely on visual evaluation and the robustness of object detection in time and space.

Chapter 2

Statistical Methods

2.1 Introduction

Finding moving objects is one of the most important tasks in video processing. For many years, the approach has been to compute the stationary background image and then identify the moving objects as those pixels that significantly differ from the background image. This is commonly called the background subtraction approach. We will classify background subtraction approaches that determine whether each pixel belongs to the background, to the foreground or to the shadow, into two groups. The first is a parametric approach, used in [1, 2], the second is a non-parametric fully described in [3].

In early works, the parametric method was used for tracking moving vehicles in freeway traffic. Moving shadows do, however, cause serious problems, since they differ from the background image as well as from the moving vehicle. The basic approach has been to classify each pixel as belonging to any one of the three classes possible, moving vehicle, shadow or background. For each class we have a probabilistic model of how the pixel looks when it belongs to that class. This probabilistic model has to be updated properly, as explained in this section. The pixel appearance is modeled as a mixture of Gaussians, one for each class, and learned by an EM algorithm. Actually recursive version is implemented. We see that this method fails to make a good classification of the pixels. Models of how the shadow is generated will have to be incorporated to make the probabilistic method converge faster and better, as proposed in [2].

The non-parametric method aims to improve the accuracy and efficiency of moving object detection by using a model for the background, shadows and highlights of the image. It introduces a new computational color model (*brightness distortion* and *chromaticity distortion*) that helps to distinguish shading background from the ordinary background or moving foreground objects. This algorithm is not only suitable for traffic scenes but is also very effective for indoor scenes as well. The non-parametric method calculates a background model and then thresholds.

2.2 Parametric Methods

Background subtraction has its roots in early photography experiments. If the exposure of the camera is bigger than the moving objects present in the scene, we could get an image of the background. We model this by using long-term averages,

$$B(x,y,t) = \frac{1}{t}\sum_{t'=1}^{t} I(x,y,t')$$

or in a recursive way as,

$$B(x,y,t) = \frac{(t-1)}{t}B(x,y,t-1) + \frac{1}{t}I(x,y,t)$$

The main problem with long-term average is that it might not capture illumination variations within the scene if the capture process has been for too long. Also a change of longer duration might be lost. A better way to capture a background sensitive to long term changes is by incorporating the exponential forgetting factor. This is the same as capturing the background using a simple filter.

$$B(x, y, y) = (1 - \alpha)B(x, y, t - 1) + \alpha I(x, y, t)$$

Here $1/\alpha$ is the exponential forgetting factor. Although methods for capturing the background work very well, as we are interested in capturing not just the background but also in capturing shadows apart from moving objects, a pixel model is proposed next.

2.2.1 *Pixel Models*

Each pixel is modeled as a mixture of Gaussians. So for the traffic surveillance case, we have three classes of pixel, corresponding to background (road), shadow and the vehicle respectively. Thus, the distribution of each pixel is the weighted sum,

$$i_{x,y} = w_{x,y} \cdot (r_{x,y}, s_{x,y}, v_{x,y})$$

$$w_{x,y} = (w_r, w_s, w_v)$$

Here $w_{x,y}$ are the weights that correspond to prior knowledge about the contributions of each of the three classes to every pixel. Thus, $r_{x,y}, s_{x,y}, v_{x,y}$, are the three Gaussian distributions describing the pixel statistics. Therefore,

$$\{r, s, v\}_{x,y} \sim N(\mu_{\{r,s,v\}}, \Sigma_{\{r,s,v\}})$$

The dimensionality of this Gaussian distribution can vary depending on the color model being used. If we use an RGB model then these Gaussians have 3 dimensions, i.e., μ is a 3x1 and Σ is a 3x3.

Let i be the pixel value. Let L be a random variable denoting the label of the pixel in the image. Then our model defines the probability that $L = l$ and $I(x, y, t) = i$ to be:

$$P(L = l, I(x, y, t) = i|\Theta) = w_l \cdot \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(i-\mu_l)'\Sigma_l^{-1}(i-\mu_l)}$$

Given these probabilities, we choose the class l with highest posterior probability $P(L = l|I(x, y, t))$.

2.2.2 *Algorithms to infer pixel models*

Let's state the problem first. Suppose we have an observation of $1, ..., T$ pictures, we want to learn the parameters of the distributions $r_{x,y}, s_{x,y}, v_{x,y}$, as well as the relative weights $w_{x,y}$. We would like to maximize a likelihood function, formally defined as follows, for a given set of parameters $\Theta : L(\Theta) = \Pi_{t=1}^{T} P(L = l_t, I(x, y, t)|\Theta)$.

If we already know the labels, optimal parameters can be easily obtained following standard arguments from the *sufficient statistics* approach. Thus, if we know the labels we can state the next *sufficient statistics* for the mixture estimation to be N_l, M_l, and Z_l where,

N_l is the number of images for which $L_t = l$

S_l is sum of all the input vectors, $\sum_{t=1,\ldots,T,L_t=l} I(x,y,t)$

Z_l is given by $\frac{1}{T}\sum_{t=1,\ldots,T,L_t=l} I(x,y,t) \cdot I(x,y,t)'$, the sum of the outer products of all input vectors with themselves.

We then obtain the parameters we are looking for as,

$w_l = \frac{N_l}{\sum_{l'} N_{l'}}$

$\mu_l = \frac{S_l}{N_l}$

$\Sigma_l = \frac{1}{N_l} Z_l - \mu_l' \mu_l$

Obviously, we do not have the labels. We have to maximize our likelihood function with missing information. Let us define our new likelihood with respect the observable data as $L(\Theta) = \Pi_{t=1}^{T} P(I(x,y,t)|\Theta)$. The standard approach to solve this problem is by using the *expectation maximization* algorithm as in [1].

EM algorithm We iteratively explore sequences of Θ, where each setting of parameters depends on the previous values. More precisely, we have one possible set Θ^k, then we estimate the parameters to calculate the distribution of each class. After we have classified each vector $I(x,y,t)$ we redefine the new parameters Θ^{k+1}. Formally, we compute the expected value of the sufficient statistic defined before as,

$E[N_l|\ \Theta^k] = \sum_{t=1}^{T} P(L_t = l|I(x,y,t),\ \Theta^k)$

$E[S_l|\ \Theta^k] = \sum_{t=1}^{T} P(L_t = l|I(x,y,t),\ \Theta^k) I(x,y,t)$

$E[Z_l|\ \Theta^k] = \sum_{t=1}^{T} P(L_t = l|I(x,y,t),\ \Theta^k) I(x,y,t) \cdot I(x,y,t)'$

Where,

$$P(L_t = l|I(x,y,t), \Theta^k) = \frac{P(L_t = l, I(x,y,t)|\Theta^k)}{\sum_{l'} P(L_t = l', I(x,y,t)|\Theta^k)}$$

This process has two important properties. First, $L(\Theta^{k+1}) \geq L(\Theta^k)$, which means that we provide a better approximation to the distribution of the data at each iteration. Second, if we reach $\Theta^{k+1} = \Theta^k$, then Θ^k is a stationary point of the likelihood function. With these two properties, we see that this procedure will eventually converge to a stationary point, that is optimal.

Incremental EM The problem with the procedure defined above is that we have to record all the instances of vectors $I(x, y, t)$ to get the parameters. This is not real-time amenable to computations; in addition we may encounter memory problems. We prefer an approach that estimates the parameters in an incremental way.

The incremental approach can be realized by removing the previous contribution from the old parameters and add the new contribution from the new parameters, this is, to update N_l we remove $P(L_t = l|I(x, y, t), \Theta^k)$ and add $P(L_t = l|I(x, y, t), \Theta^{k+1})$. Similarly for the rest. Then S_l will be updated by removing $P(L_t = l|I(x, y, t), \Theta^k)I(x, y, t)$ and adding $P(L_t = l|I(x, y, t), \Theta^{k+1})I(x, y, t)$. Thus each time when a new frame is obtained, we add its contribution to the sufficient statistics. We are increasing our training set but yet we will not reprocessing already processed data. This does not guarantee convergence but for some cases, like the traffic scene, it is effective.

The algorithm can be summarized as follows,

```
Initialize $\Theta$
for each $l \in \{r, s, v\}$
$N_l \leftarrow w_l$
```

```
$S_l \leftarrow w_l \cdot \mu_l$
$Z_l \leftarrow w_l \cdot (\Sigma_l + \mu_l \cdot \mu_l^{'})$
where $w_l$ are weak priors
do forever
    $t \leftarrow t + 1$
    for each for each $l \in \{r, s, v\}$
        $N_l \leftarrow N_l + P(L_t = l|I(x, y, t),\, \Theta)$
        $S_l \leftarrow S_l + P(L_t = l|I(x, y, t),\, \Theta)I(x, y, t)$
        $Z_l \leftarrow Z_l + P(L_t = l|I(x, y, t),\, \Theta)I(x, y, t) \cdot I(x, y, t)'$
    Recalculate $\Theta$ from $\{N_l, S_l, Z_l\}$
```

The initialization of the iterations is an important part of the whole process. We suggest two approaches for incremental computation

2.2.3 *First approach*

We use the R, G, B color model. The pixel statistics of the three components behave similarly when the pixel is shadowed or it is illuminated, so we are not gaining much by using all three instead of a grayscale image. Let us see how the algorithm behaves after processing 261 frames with a pretty good initialization based on some heuristics derived from initial image statistics.

In figure 2.2, we see the three probability maps for the three classes, shadows, vehicles and road. And the resulting complete detection mask, where the blue pixels

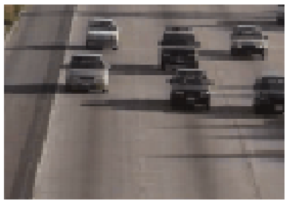

Figure 2.1: Traffic scene frame 261

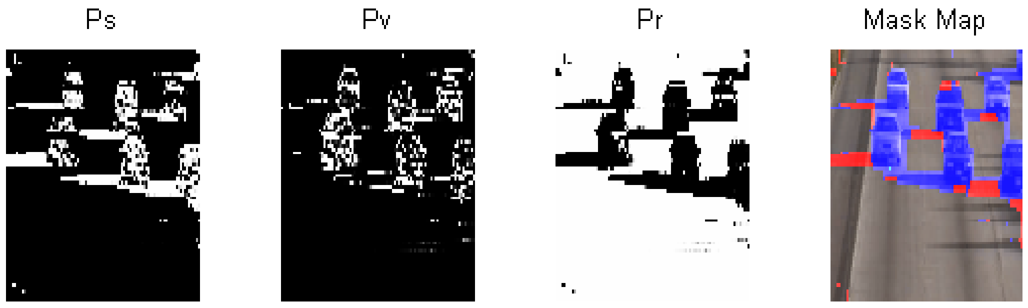

Figure 2.2: Probabilities and Masks for frame 261

denote the vehicles, and the red pixels the shadow regions. To check if the algorithm has learned the distributions, we calculate the histogram from a random pixel and see the estimated distributions. In the detection mask shown there is a small green pixel, where the different statistics were performed.

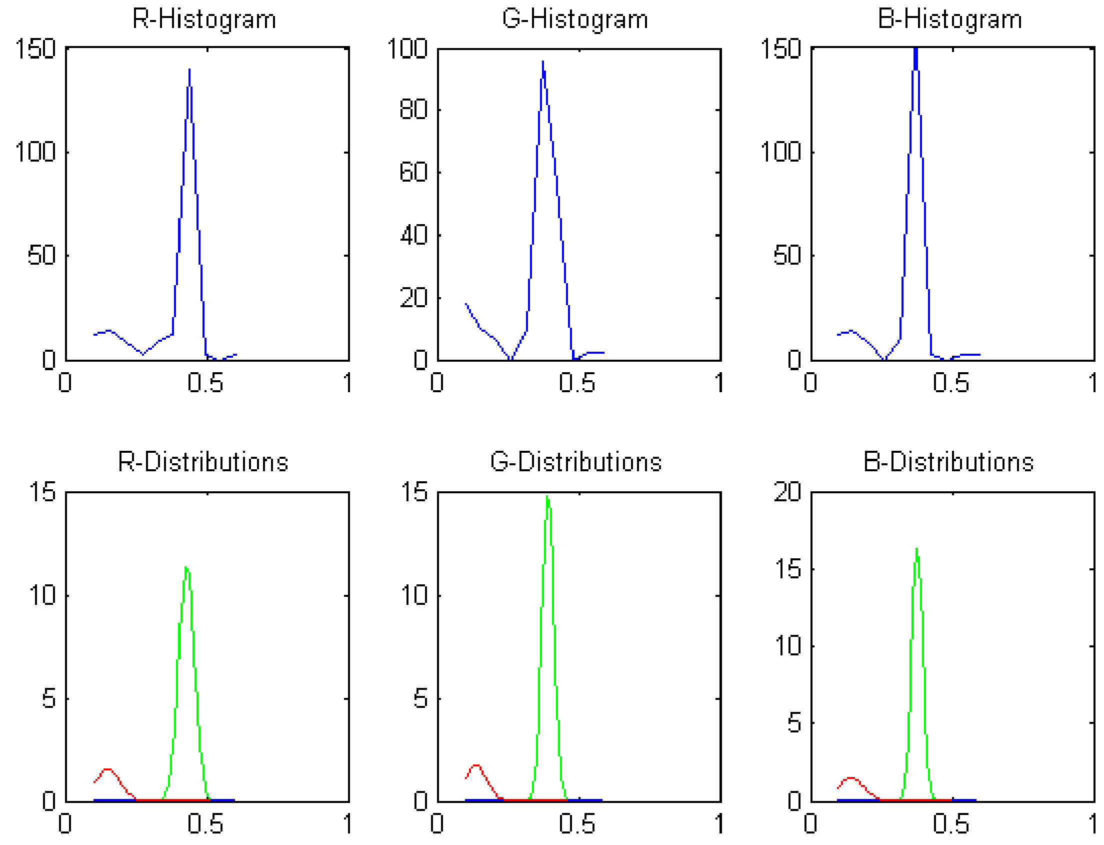

Figure 2.3: Statistics of pixel at (32,32) in frame 200

In figure 2.3, we present the statistics for each color channel of pixel x=32 and y=32 in the 200th frame. We see a very predominant Gaussian distribution due to the road (in green) which occupies a significant portion of time. Then in red, we have the distribution from the shadow region. Finally the vehicle is indicated on blue shade, almost flat and hard to see in this figure. The learning process has been very successful, as the generated distribution from our parametric model almost fits perfectly the histogram of the pixel. Yet, the detection is far from being perfect as we see in the next example.

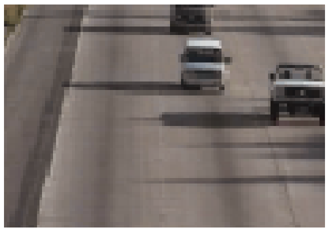

Figure 2.4: Traffic scene frame 337

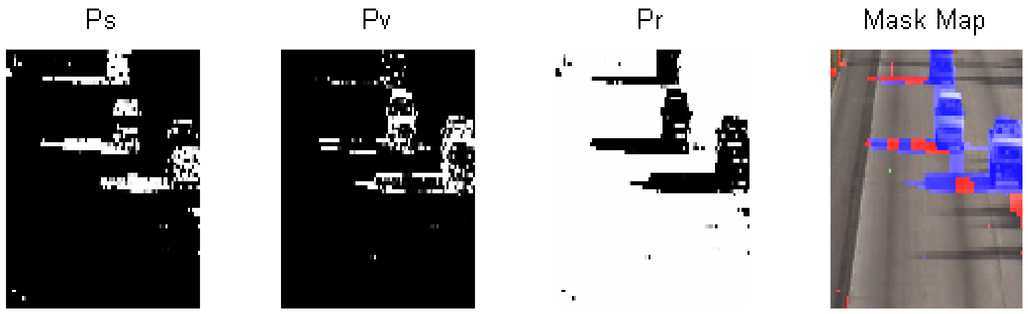

Figure 2.5: Probabilities and Mask Map

As can be seen, the detection map is far from being perfect but still we are capturing most of the shadows and the cars. Also, the computed statistics are still very good. However, we begin to see one of the major problems of this technique. That is; as much more instances we will be processing, the algorithm will be increasing the number of samples to compute its sufficient statistics so much that might end up giving unrealistic estimates if the scene changes.

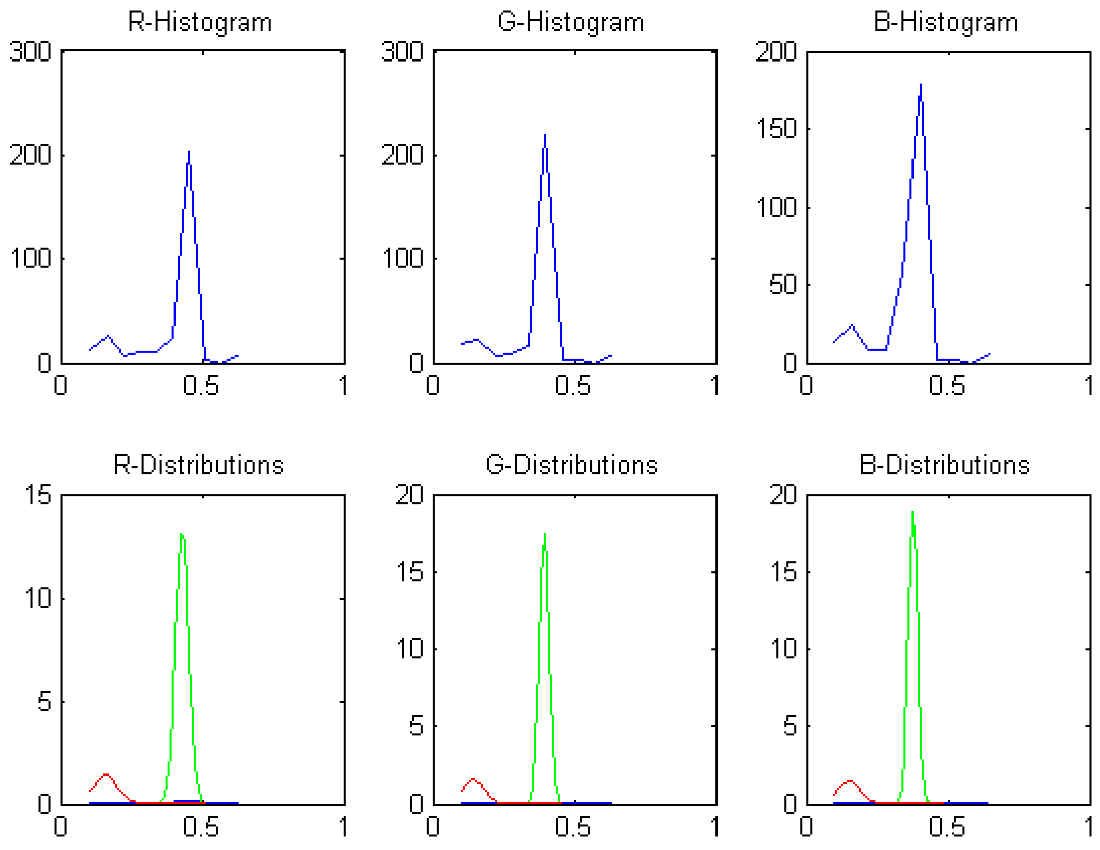

Figure 2.6: Statistics for frame 337

Despite the statistics being well behaved, as can be seen in figure 2.6, the detection result is far from being perfect. In the next sequence of frames we see how unstable the detection is.

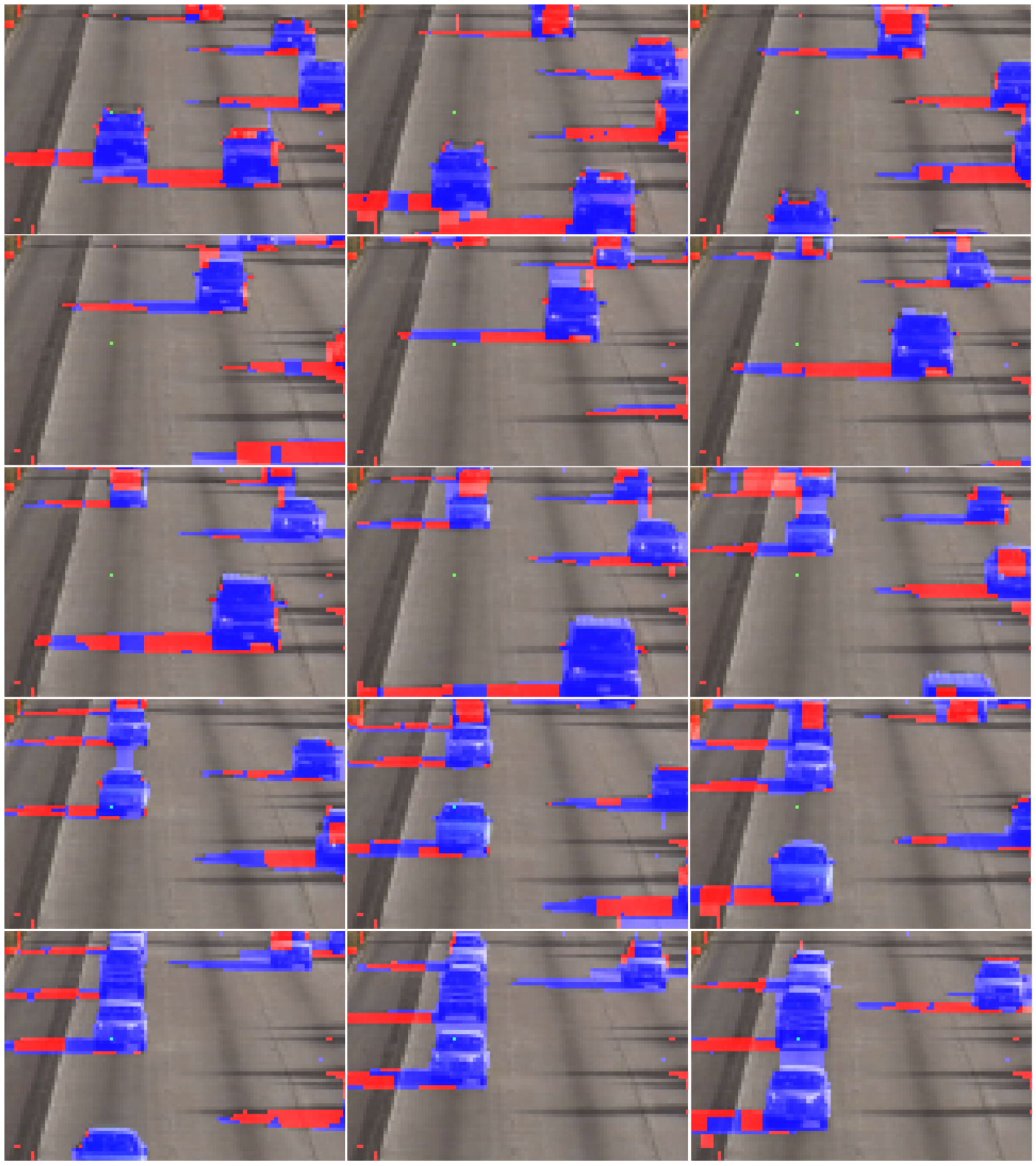

Figure 2.7: Traffic sequence 15 frames at 1/3 frames per second

With this previous color pixel model we are not taking full advantage of the color space to create well-defined differences between shadows and the road. In all the three RGB components as the shadow behaves in a similar way, not much information gained by combining these three color components. We also tried the HSV model but the result was even worse due to unreliable information from the cars and inconsistent behavior of shadows as well. Thus, a new model with some connection to the mechanism of shadow formation is needed to be able to fully exploit color information.

2.2.4 *Second approach*

Without using scene models and assumptions on location of light sources or models of the moving objects, we can identify two sources of information that can help in detecting objects and shadows. The first has to do with the pixel appearance, how the pixels gets affected in the presence of shadow. The second is spatial, as objects are compact. We integrate these two in this approach.

The idea relies on the fact that obtaining the statistical model for shadows from the road is more reliable due to the correlation of pixels from the road region.

Color change under shadow

Let us model the luminance at an image point (x, y) as:

$$l(x, y) = E(x, y)\rho(x, y)$$

where $E(x, y)$ is the irradiance. $\rho(x, y)$ is the reflectance of the object surface. Irradiance can be expressed as:

$$E(x, y) = \left\{ \begin{array}{c} c_A + c_P \cos\angle(N(x, y), L), illuminated \\ \\ c_A, shadowed \end{array} \right\}$$

where c_P and c_A are intensities of the light source and ambient light, N is the object surface normal and L the direction of the light source. Since the reflectance of the background does not change with time, the ratio of illuminance values of a point when illuminated and when covered by a shadow, $r(x, y)$ can be written as in [2],

$$r(x, y) = \frac{l_{IL}(x, y)}{l_{SH}(x, y)} = \frac{c_A + c_P \cos\angle(N(x, y), L)}{c_A}$$

If the background is almost flat, which is the case with the road, then the ratio is the same for every position and at all times with $r(x, y) = r$, we can easily derive a rule for estimating the shadow distribution from the road distribution. Thus,

$$\mu^B_{SH} = \mu^B_{IL}/r$$

$$\sigma^B_{SH} = \sigma^B_{IL}/r$$

where the upper index B is for brightness. Other color changes are more difficult to model. But it is known that the blue color increases when shadow is present and red color decreases. We do not know how the variances change but all these parameters are estimated. The mean of the blue color on the shadow will be increased by a Δb and the mean of the red color decreased by Δr. And the variances scaled by fb and fr that need to be estimated. We use normalized red and blue color components for the other two sources of information at every pixel as, $b = B/(R + G + B)$ and $r = R/(R + G + B)$.

$$\mu^b_{SH} = \mu^b_{IL} + \Delta b$$

$$\mu^r_{SH} = \mu^r_{IL} - \Delta r$$

$$\sigma^b_{SH} = \sigma^b_{IL} \cdot fb$$

$$\sigma^r_{SH} = \sigma^r_{IL} \cdot fr$$

A fading estimator calculates the background mean and variance for all pixel locations. Using previous equation, we derive statistics for the same pixels when shadowed. Gaussian distributions are assumed for background and shadow pixels and a uniform distribution is assumed for the foreground. This is similar to inferring from the statistics computed in the first approach. We see two nice Gaussians and one almost uniform distribution. Estimating the right parameters to obtain the shadow distribution is far from being perfect. What has been done is to estimate the initial parameters from histograms such that after having processed enough images with shadowed pixels, these parameters can be refined.

Algorithm overview

We follow the turbo implementation used in [2]. This method obtains the class where the pixel belongs to, from an iterative loop using the color model described in the previous section. It starts from the luminance distributions and uses the probabilities obtained from that step as priors to second step. It then uses the blue distributions and again sets the resulting probabilities as priors to the next step. This is repeated for the red channel and we get the final probabilities. By using this as the priors to the first step we close the loop. The flowchart of the algorithm is presented below.

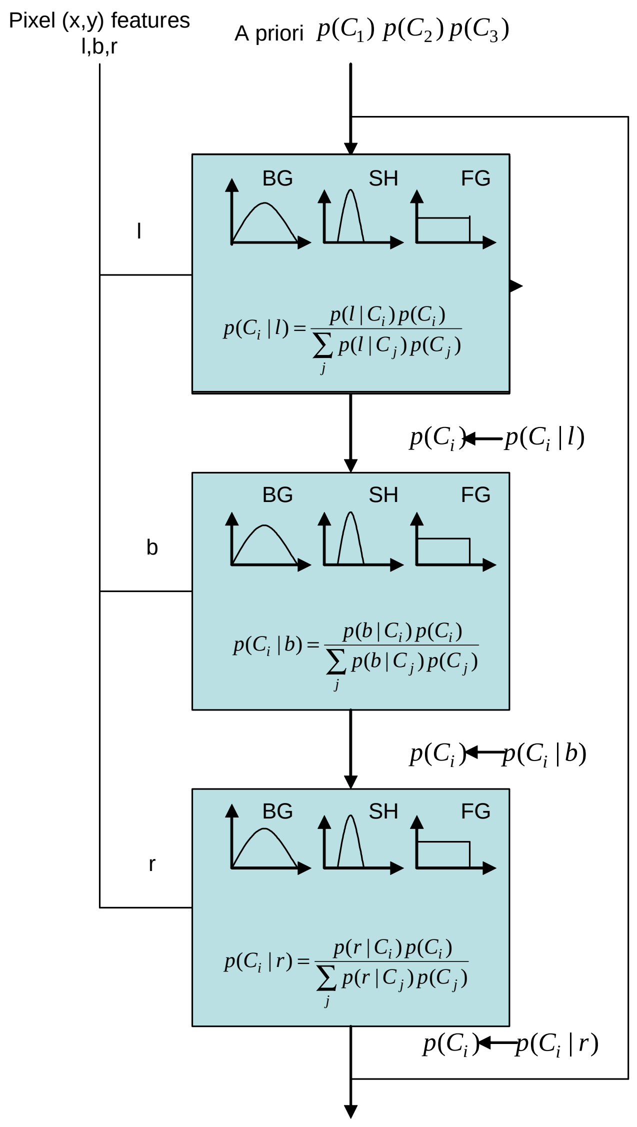
Pixel (x,y) features l,b,r
A priori $p(C_1)\ p(C_2)\ p(C_3)$
l
b
r
BG
SH
FG
$p(C_i|l)=\frac{p(l|C_i)p(C_i)}{\sum_j p(l|C_j)p(C_j)}$
$p(C_i|b)=\frac{p(b|C_i)p(C_i)}{\sum_j p(b|C_j)p(C_j)}$
$p(C_i|r)=\frac{p(r|C_i)p(C_i)}{\sum_j p(r|C_j)p(C_j)}$
$p(C_i) \leftarrow p(C_i|l)$
$p(C_i) \leftarrow p(C_i|b)$
$p(C_i) \leftarrow p(C_i|r)$

Figure 2.8: Algorithm flowchart

It only takes up to ten iterations to get one of the three priors to be bigger than 0.9. We terminate the iterations then. Thus, after only ten iterations, we threshold the final priors to get either foreground pixels or shadowed pixels. When both priors are less than 0.9 we classify them as background. After each classification the priors are updated for the next frame and this is continued.

With this implementation, we get really fast and more stable segmentation of the frames, though our shadow model is not the perfect model. In figure 2.9, we see how different our estimated shadow and the real one are, by where real we mean the average of the pixels classified as shadow.

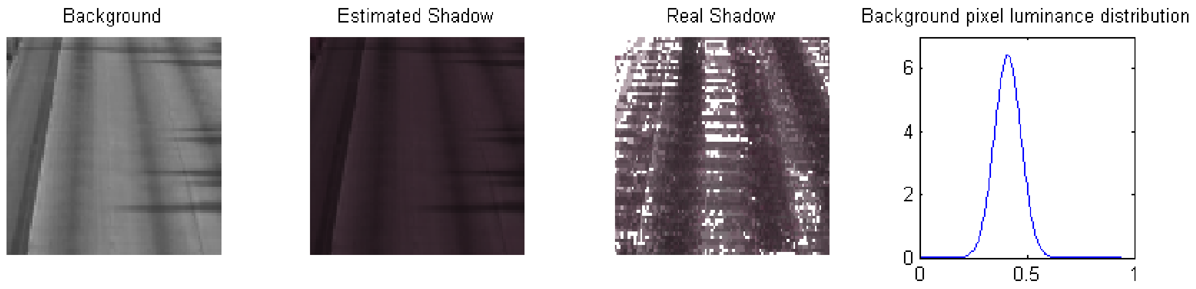

Figure 2.9: Background and shadow estimation on frame 99

We see in figure 2.10 that the detection result is not that good. This is because the initial parameters do not lead to a reasonable shadow distribution. That is why we recalculate the parameters every 100 frames. The process is really simple. Through the real distributions of the background and the shadow, we get the parameters to make the estimation as close as possible to the real one.

We see in figure 2.11, just before doing another recalculation of the parameters, how close the estimated shadow is to the real one. Also, we see how the background distribution variance is reduced.

In figure 2.12, the estimation result is almost the same, and the parameters barely change from the ones obtained in frame 200. So these are the best parameters we could get, based on the assumption that the real shadow distribution is the closest

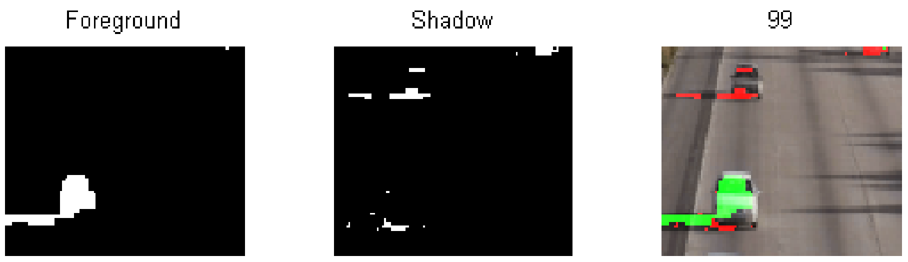

Figure 2.10: Detection of frame 99

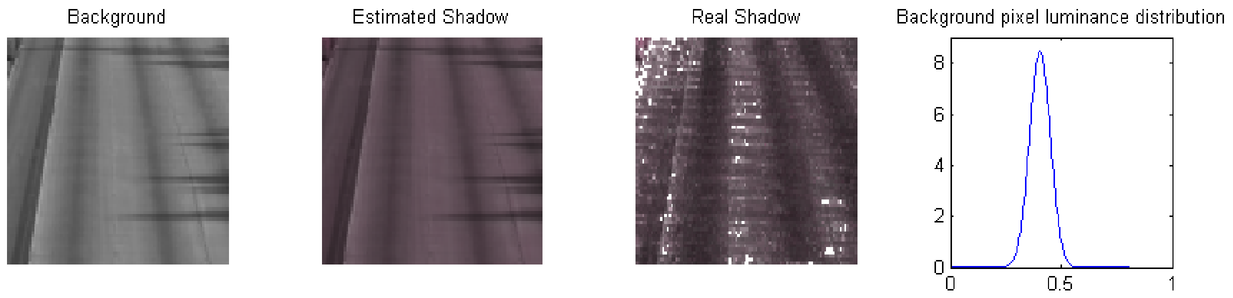

Figure 2.11: Shadow estimation at frame 199

distribution to the shadow one.

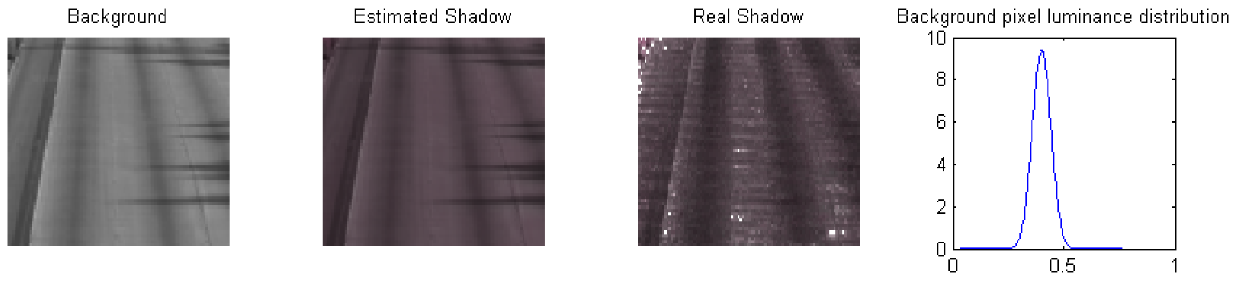

Figure 2.12: Shadow estimation in frame 299

Once the shadow is captured we see that the detection of each class becomes more stable, but it does not guarantee that the detection will be good. In figure 2.13 we show how that detection is made.

2.2.5 *Evaluation*

After experiencing with the two approaches, several conclusions about the pros and cons of implementing parametric statistical techniques can be drawn. It is preferable to have three Gaussian distributions for the three classes of pixel models we are dealing with. Even if that were true, we will still have detection errors due to unrealistic statistical assumptions. But that is only the case for the background and maybe for the shadows. As vehicles are of completely random colors, almost uniform distribution seems to work.

The quality of background detection is very good. We are always segmenting well if there is a moving object or the pixel belongs to the background. The problem is when discriminating the shadow pixels from the car pixels, specially separating

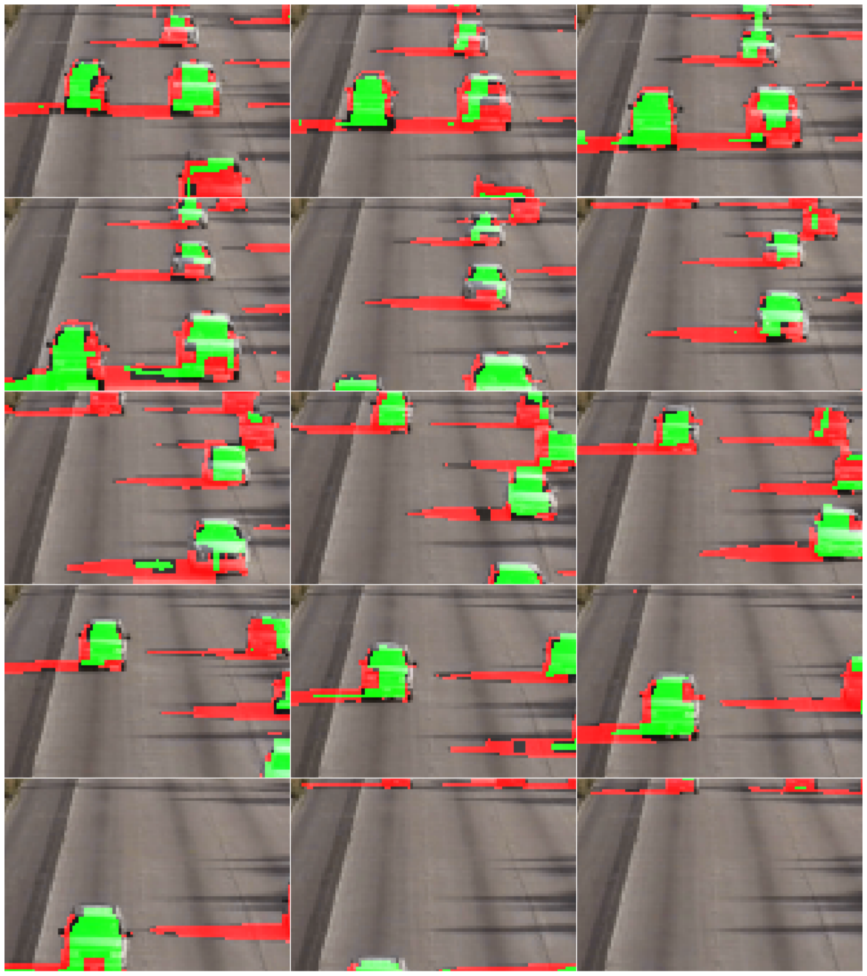

Figure 2.13: Traffic sequence at 4 frames per second

the shadow pixels from the pixels of dark cars. The algorithm works well when bright cars are present in the sequence.

In the second approach, the Gaussian distributions were learnt faster but if the initializations were not very good, the learning process failed, leading to wrong Gaussian distributions. At the end, the second approach could not do much better than the first approach because of the nature of the problem as stated earlier, learning Gaussian distributions for classes that were difficult to classify. Even if a training stage were introduced and Gaussian distributions fitted to the training samples, the second approach tend to fail quite often.

2.3 Non-parametric

One fundamental ability in human vision is color consistency, as discussed in [3]. Humans tend to assign constant color to an object with illumination changing over time or space. As has already been explained, the color of an object depends on many factors like physical properties of the point on the surface of the object. For a Lambertian model this physical property of the surface called spectral reflectance determines, along with illumination, the color properties of every point. This is explained in [3] for separating brightness and chromaticity, as shown in figure 2.14 .

This color model is based on the assumption that there is some background in which we can compare our new color pixels and obtain some measures. This background is represented by $E_i = [E_R(i), E_G(i), E_B(i)]$, the expected value of the pixel i. The OE_i is called the expected chromaticity line, fundamental to separate brightness

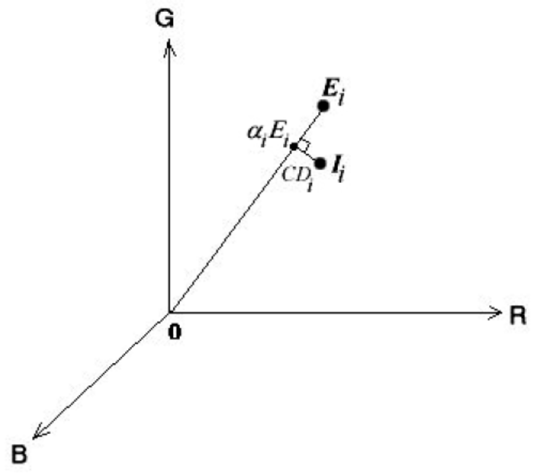

Figure 2.14: 3D Color model

and chromaticity. Also we have the current pixel color as $I_i = [I_R(i), I_G(i), I_B(i)]$. We want to obtain the distortion between I_i and E_i. This is done by decomposing the distortion into two components, *brightness distortion* and *chromaticity distortion*.

Brightness Distortion (α)

Brightness distortion is a scalar value that brings the observed color closer to the chromaticity line. We can obtain it by minimizing

$$\phi(i) = (I_i - \alpha_i E_i)^2$$

where α_i represents the pixel's strength with respect to the expected value. If it is greater than 1, it means that the pixel is brighter than the background, if it is less than 1 the opposite is true.

Color Distortion (CD)

It is defined as the orthogonal distance between the observed color and the chromaticity line. The color distortion is defined as

$$CD_i = ||I_i - \alpha_i E_i||$$

We have to take into account the non-linear properties of the CCD cameras as well as the specifics of the capturing process that produce the output images that are processed. In [3], the authors considered the following relevant factors.

Color variation: The colors captured vary along time due to noise or even illumination changes.

Band unbalancing: Depending on each spectral band that we will be working with to capture the RGB, we have different standard deviation, due to sensitivities of the CCD. Thus, in order to balance the weights, we normalize the pixel color by its standard deviation by,

$$s_i = [\sigma_R(i), \sigma_G(i), \sigma_B(i)]$$

computed using N number of frames.

Clipping: Due to the limited dynamic range of responsiveness, the saturated colors, the ones outside that range are clipped either to the first value of the range, this is 0, or to the last one, on a 8 bit resolution, this is 255. As the standard deviation for these colors will be almost zero we will have to restrict the minimal standard deviation to a default value.

2.3.1 *Algorithm*

Background Modeling

The first thing to do is to make a background model over several frames. We calculate the average of each pixel's RGB colors and the standard deviation as well, over N number training frames. Thus, the background will be modeled as

$$E_i = [\mu_R(i), \mu_G(i), \mu_B(i)]$$

Now that we have E_i and s_i calculated, we proceed to obtain the brightness and color distortions as follows

$$\alpha_i = \min \left[\left(\frac{I_R(i) - \alpha_i \mu_R(i)}{\sigma_R(i)} \right)^2 + \left(\frac{I_G(i) - \alpha_i \mu_G(i)}{\sigma_G(i)} \right)^2 + \left(\frac{I_B(i) - \alpha_i \mu_B(i)}{\sigma_B(i)} \right)^2 \right]$$

which is the same as the projection of I_i weighted by s_i and the normalized E_i also weighted by s_i.

$$\alpha_i = \frac{\left(\frac{I_R(i)\mu_R(i)}{\sigma_R^2(i)} + \frac{I_G(i)\mu_G(i)}{\sigma_G^2(i)} + \frac{I_B(i)\mu_B(i)}{\sigma_B^2(i)} \right)}{\left(\left[\frac{\mu_R(i)}{\sigma_r(i)} \right]^2 + \left[\frac{\mu_G(i)}{\sigma_G(i)} \right]^2 + \left[\frac{\mu_B(i)}{\sigma_B(i)} \right]^2 \right)}$$

Once we got the brightness distortion, obtaining the color distortion (CD) is pretty straightforward as,

$$CD_i = \sqrt{\left(\frac{I_R(i) - \alpha_i \mu_R(i)}{\sigma_R(i)} \right)^2 + \left(\frac{I_G(i) - \alpha_i \mu_G(i)}{\sigma_G(i)} \right)^2 + \left(\frac{I_B(i) - \alpha_i \mu_B(i)}{\sigma_B(i)} \right)^2}$$

Next, we consider the variation of brightness and chromaticity distortions over space and time of the training background images. In [3], the authors find that different pixels yield different distributions of α and CD. These variations will be related to the scaling factors a$_i$ and b_i. Thus, we can use same thresholds for different pixels over time. The scaling factors are calculated as follows

a_i represents the variation of the brightness distortion of the i^{th} pixel, which is given by

$$a_i = RMS(\alpha_i) = \sqrt{\frac{\sum_{i=0}^{N}(\alpha_i - 1)^2}{N}}$$

b_i represents the variation of chromaticity and is given by

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{i=0}^{N} CD_i^2}{N}}$$

One conclusion made in [3] is that the value of the scaling factors make sense due to the fact that the brightness distortion scaling factor is greater than the chromaticity distortion factor. This takes into account the fact that the brightness has more variation than chromaticity, thus confirming that this computational color model is similar to human vision which is more sensitive to changes in luminosity than to changes in color. Finally we note that every background pixel has been modeled by the 4-tuple $< E_i, s_i, a_i, b_i >$

Pixel Classification

In this step we apply some thresholds on both brightness distortion and chromaticity distortion of a pixel to obtain an object mask $M(i)$ which indicates the type of the pixel. This method classifies pixels into four categories. These categories are

1. *Original Background* (B) When both brightness and chromaticity are similar to the background pixel.

2. *Shaded Background or shadow* (S) It has similar chromaticity but lower brightness. This is based on the Lambertian formation of colors in the image from surface objects.

3. *Highlighted Background* (H) On the other hand when the brightness is higher and the chromaticity remains pretty much the same.

4. *Moving Foreground* (F) When the pixel chromaticity differs from the background one.

   The first step to implement before classifying a new pixel is to normalize it according to the scaling factors defined before. Then

$$\widehat{\alpha_i} = \frac{\alpha_i - 1}{a_i}$$

$$\widehat{CD}_i = \frac{CD_i}{b_i}$$

Now all the needed thresholds are set to delineate the object mask. First we need to determine a minimum chromaticity threshold τ_{CD}, if the chromaticity is greater than τ_{CD}, then it is too far from the background chromaticity and therefore may belong to a moving object. Once we have all pixels with similar chromaticity, we focus on the background pixels. For that, we use thresholds $\tau_{\alpha 1}$ and $\tau_{\alpha 2}$ to cluster pixels within a small range of brightness distortion from our background. Finally, we have to distinguish between highlighted or shadowed background as the remaining pixels with negative brightness distortion will be shadows and the rest will be highlighted background. When the pixel color is very dark then α_i will be close to zero and therefore CD_i, meaning that has a similar chromaticity as the background. This will make us classify the pixel as a shadow pixel. To avoid this we add a threshold $\tau_{\alpha 0}$ to classify as a moving object, pixels with really low brightness distortion. Finally, we summarize the classification process through the $M(i)$ mask as follows

$$M(i) = \left\{ \begin{array}{l} F : \widehat{CD}_i > \tau_{CD} \quad or \quad \widehat{\alpha_i} < \tau_{\alpha 0} \ \ , \quad else \\ B : \widehat{\alpha_i} < \tau_{\alpha 1} \quad and \quad \widehat{\alpha_i} > \tau_{\alpha 2} \ \ , \quad else \\ S : \widehat{\alpha_i} < 0 \ \ , \quad else \\ H : \ otherwise \end{array} \right\}$$

Threshold Selection

It is desirable to automatically determine the thresholds. In order to accomplish this we adjust the false alarm rate r for the training background frames. For all background frames in the training set, we calculate the histogram of the *brightness distortion* and the *chromaticity distortion.* We observe that the *brightness distortion* is quite Gaussian like but that does not hold for the *chromaticity distortion.* The whole point here is to set the τ_{CD} and both $\tau_{\alpha 1}, \tau_{\alpha 2}$, so as to $P(\tau_{\alpha 2} < \widehat{\alpha_i} < \tau_{\alpha 1}) = 1-r$ and $P(\widehat{CD}_i > \tau_{CD}) = r$, thus automatically obtaining all the three sets of coefficients. For $\tau_{\alpha 0}$, we don not have an automatic way, thus set it experimentally.

Clustering Detection Elimination

In [3], the authors observed, and a so did I, the accumulation of false alarms in a contained region. In [3] the authors found that this often occurred with pixels corresponding to those with very low chromaticity distortion, i.e., a very small b_i. When such a small b_i appeared it made the $\widehat{CD}_i$ to likely exceed the foreground threshold τ_{CD}. Thus we should establish a *minimum* b_i, to avoid such problems. In order to find *minimum* b_i an optimization procedure is needed. That is the trickiest part of the training process but still not too difficult. We increase that *minimum* b_i until we balance the error. This is achieved when the number of pixels that have failed just once on the training frames and those that failed more than one are balanced.

2.3.2 *Indoor examples*

This first example shows how the algorithm performs when dealing in an indoor environment. We see a man walking from left to right. In red we see what has been classified as foreground and in blue what has been detected as a shadow. It is interesting to notice how fine our thresholds are because the shadow is very similar to the background and also the sweater the man is wearing is quite similar to the closet.

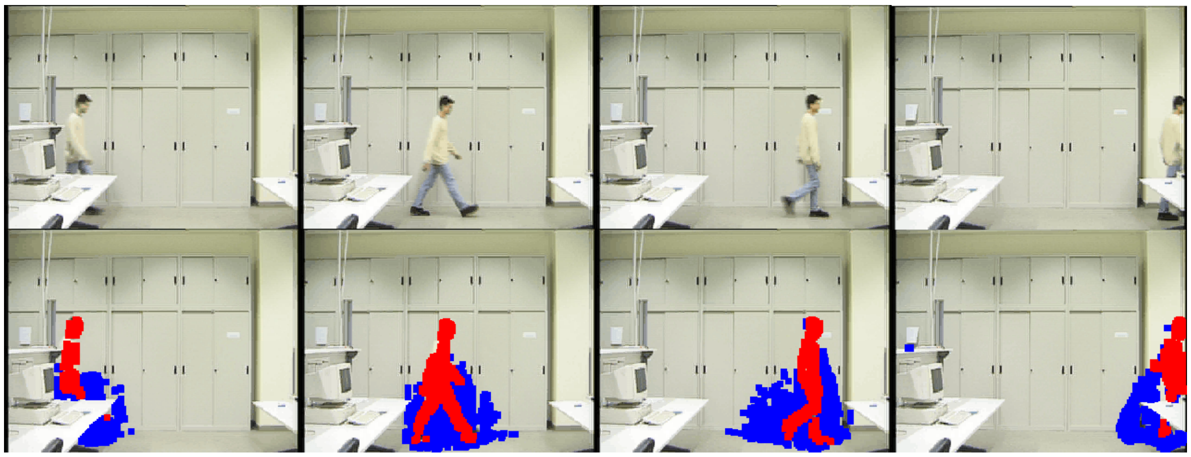

Figure 2.15: Indoor example 1

The fact that the man is walking near the closet makes the detection better.

In this next example we see how the same man gets confused with the background as he is walking from the camera towards the closet. Other subjects men that walked through the scenes were dressed more different than the closet, thus, the classification algorithm worked much better.

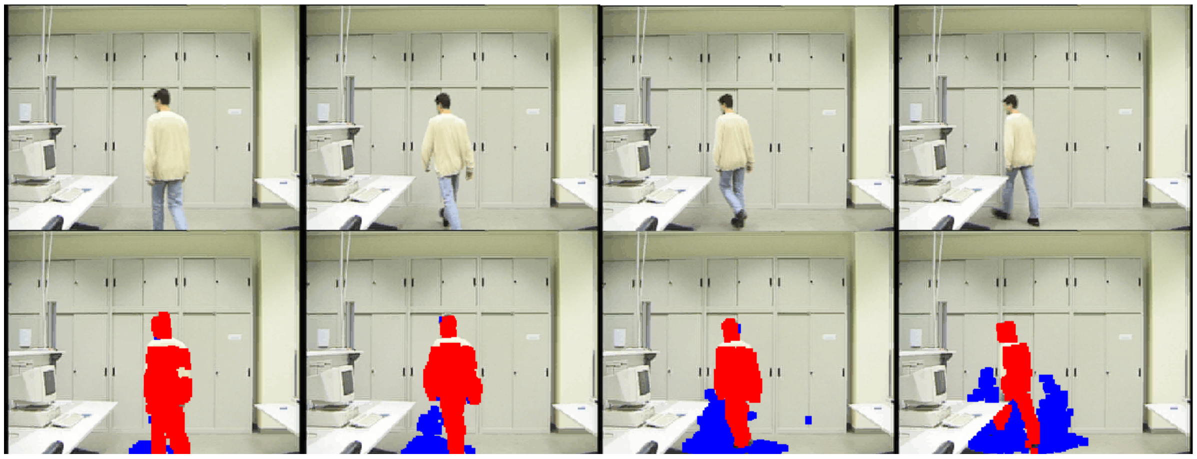
Figure 2.16: Indoor example 2

2.3.3 *Outdoor example*

We expect inferior performance when we go to an outdoor scene where the videos are more sensitive to illumination changes and scene variations. In the next video slide we see how the algorithm performs for these situations. We see that when the object is very big, like in the case of the an entering car, our classification result is not that good. On the other hand, for the pedestrians, the performance is quite impressive.

We see in figure 2.17 that the stopping bar produces false detections, as we are not updating the background. In order to fix the problems due to changes in

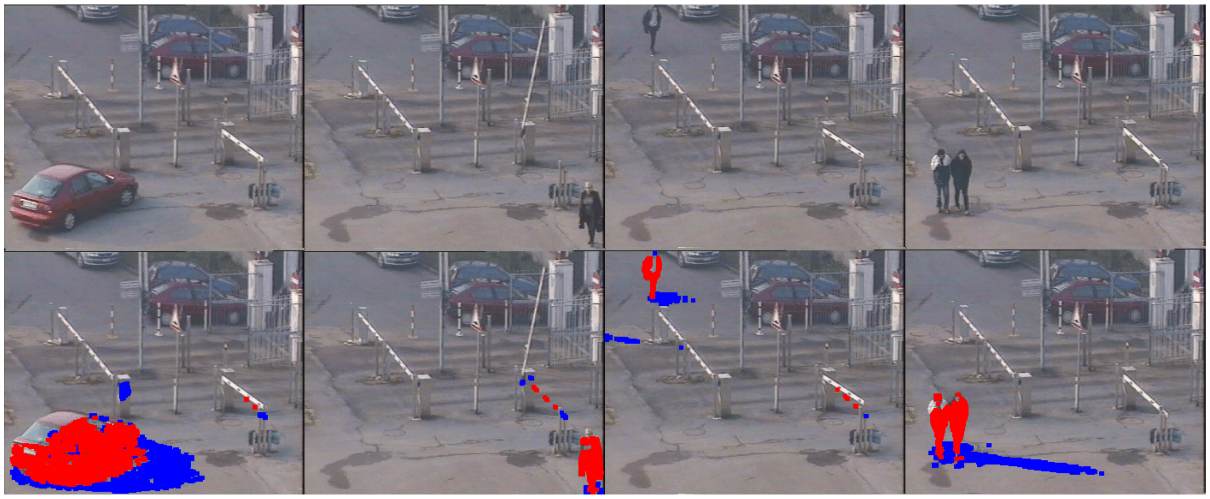

Figure 2.17: Parking entrance

the background, we should have an adaptive background estimation algorithm.

2.3.4 *Highway example*

For comparison purposes we show how the algorithm performs for the same highway video used for demonstrating the parametric techniques. In this case the background is obtained by averaging as before, but we need to let more time elapse to have a good estimate of it due to the interference of the foreground cars and shadows.

The performance is not that bad but, as can be checked, the statistical algorithms did slightly better.

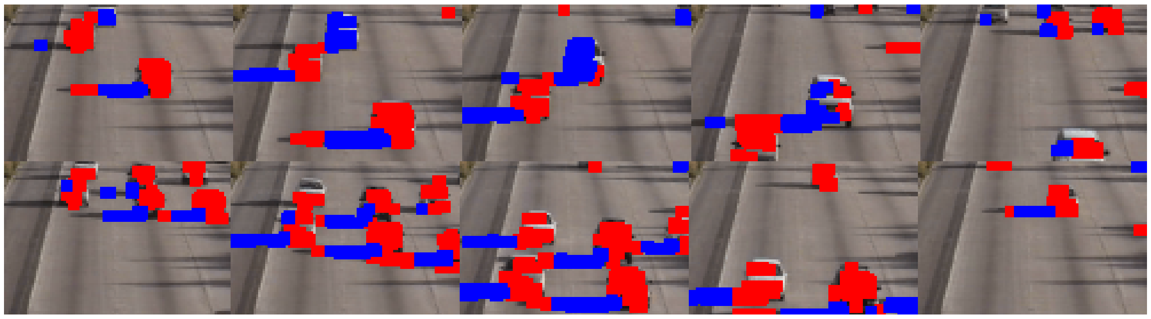

Figure 2.18: Highway example

2.3.5 *Evaluation*

This non-parametric approach seems to perform quite decently in many situations. The fact that we are indeed using an illumination model makes our algorithm more robust. If we let everything to be learned as in parametric methods, we may be more prone to erroneous solutions. In this algorithm we are modeling how the chromaticity and brightness behave when shading is present and by doing so our classification is pretty good. The fact that we are not updating the background at every frame makes it vulnerable to event changes. A good feature of our approach is speed. We only need to estimate the background parameters for a first few frames. This makes our frame-by-frame detection method only need to do thresholding, which is a lot more faster than reestimating the model parameters at every instant.

2.4 Conclusions

The two methods introduced in this chapter worked reasonably well for some particular situations. Both rely on obtaining a good statistical background model.

The parametric approaches not only estimate a background model but also does so for the foreground and the shadows. For the highway video sequences, where cars run almost all the time in the same lanes, having pixel models of shadow, foreground and background seemed to be the right choice. Priors make sense in that case as shadow pixels tend to be always at same positions and the same is true for foreground pixels. But when trajectories are more random, it is more difficult to obtain good models for every class. In these situations we can only get a good background model and, based on that, try to estimate the rest that is happening in the scene. That's where the second approach, the non-parametric one, performed best.

We have not implemented these algorithms to aerial images as we did not have any background information. We have obtained acceptable non-parametric background subtraction algorithm of [3] for videos acquired by stationary cameras. We get well-defined human shapes out of video sequences and completely segment the shadow. Also, for the moving objects in the traffic scene, we obtained quite decent results for localizing the shadows they produced.

Chapter 3

Illumination Invariance

3.1 Introduction

Illumination conditions can confound many algorithms in vision. As an example, different illumination conditions can result in completely different objects and therefore mislead the detection process. Also, an illumination source can contribute to the presence or absence of shadows and so completely change the image of the object. Understanding how shadows are produced due to illumination is of great interest in image understanding. Also for digital photography applications like color correction it is of great interest to understand how illumination conditions affects the image.

Getting those shadow free images is the aim of this chapter and it is shown to be not simple at all. Once we get images with all the salient image information, but without the shadows, the detection problem is less confounding. Several methods were tried to accomplish this task. This chapter presents brief descriptions of all of them and concludes with an overview of their strengths and weaknesses.

The first method studied is the one proposed in [4], which uses a very simple but useful space color transformation for achieving illumination invariance. Once in the new space, edge detection is performed to obtain invariance to illumination edges. Combining these edges with the original edges in the image, the algorithm is able to obtain shadow edges. Then using the geometric properties the complete shadow is obtained. This method proved to be not very accurate in obtaining good invariant features but when the invariant images where good the method performed really well. So the next step was to obtain better invariant images. For that a more complex color model was used.

In [5] as well as in [6], it has been shown that the color space which the images are transformed to get invariant illumination varies for different cameras and illumination situations. It is based on Wien's approximation of Plank's law, as in [5]. The main idea is that it transforms the 3D color space into a 2D space where reflective surfaces under different illuminations form a line with the slope common to all other surfaces. If we project all the surface colors to an orthogonal line to this common slope, we obtain a grayscale image where every surface has a different gray value independent of illumination.

One problem with this method presents is that we need to have different illumination situations to calibrate the cameras and we might be in the situation where this is impossible, which is our case. We tried several modifications and some were more successful than others.

Finally, a very simple method was tried and seemed to be the best for aerial images. This method, inspired by [7], consists of using a simpler version of the Lambertian model, where the intensity of the image is a product of the reflectance of the object and illumination. By applying the logarithm to that intensity we transform the product into a sum. Now we only have to use a low pass filter because as light changes slower than the surface reflectance. We obtained were satisfactory results with aerial images.

3.2 Non-Calibrated Color Invariant Shadow Segmentation

A hypothesis about the presence of a shadow is first generated on the basis of initial and simple evidences, like shadows being usually darker than the surface where they are cast. The validity of this hypothesis is further verified in each detected region by making use of hypotheses on color invariance and geometric properties of shadows. Finally, an information fusion stage confirms or rejects the initial hypothesis for every detected region. To better understand how the hypotheses validation process works let us first define the spectral and geometric properties of a shadow.

3.2.1 *Spectral properties of a shadow*

To define the spectral properties, one needs to look at the physics of color generation and how that affects a shadow. The appearance of a surface is the result of the interactions among illumination, surface reflectance properties, and the responses of chromatic mechanisms, which in this case is the camera color filter response.

We model these physical interactions as follows. The radiance of the light $L_r(\lambda, \vec{p})$, [4], reflected at a given $\vec{p}$ on a surface in the 3D space, given some illumination and viewing geometry, is formulated as

$$L_r(\lambda, \vec{p}) = L_a(\lambda) + L_b(\lambda, \vec{p}) + L_s(\lambda, \vec{p}) \tag{3.1}$$

where $L_a(\lambda)$, $L_b(\lambda, \vec{p})$, $L_s(\lambda, \vec{p})$ are the ambient, body and surface reflection terms respectively and λ is the light wavelength. The ambient illumination is considered to be the same among all surfaces and does not vary depending on the geometry. Instead, if the point $\vec{p}$ is occluded it does not reflect body or surface reflection. Then, for a shadow the reflected light radiance is,

$$L_{r_{shadow}}(\lambda, \vec{p}) = L_a(\lambda) \tag{3.2}$$

Let us now model the chromatic mechanism, as in [4], as the three spectral color component sensitivities of the camera, $S_R(\lambda)$, $S_G(\lambda)$, $S_B(\lambda)$. Then the color components of the reflected intensity reaching the sensors at point (x,y) in the 2D image plane are

$$C_i(x, y) = \int_\Lambda E(\lambda, x, y) S_i(\lambda) d\lambda \tag{3.3}$$

where $i \in R, G, B$ and $E(\lambda, x, y)$ is the image irradiance at (x, y). Since image irradiance is proportional to scene radiance at a pixel position (x, y) representing $\vec{p}$ in direct light, the sensor measurements are

$$C_i(x, y)_{lit} = \int_\Lambda \alpha(L_a(\lambda) + L_b(\lambda, \vec{p}) + L_s(\lambda, \vec{p})) S_i(\lambda) d\lambda \tag{3.4}$$

and for a point in shadow the measurements are

$$C_i(x,y)_{shadow} = \int_\Lambda \alpha L_a(\lambda) S_i(\lambda) d\lambda \tag{3.5}$$

It is easy to see that for each of the three color components when passing from a lit region to a shadowed one, as everything is non-negative.

$$C_{i_{shadow}} < C_{i_{lit}} \tag{3.6}$$

In regions where we have a change between lit and shadow, all three components R,G,B suffer from a decrease in their spectral properties values.

A second spectral property of shadows can be derived by considering photometric color invariants. Photometric color invariants are functions that describe the color configuration of each image point discounting shading, shadows and highlights. These functions are demonstrated to be invariant changes in imaging conditions, like viewing direction, surfaces and illumination. One commonly used photometric color invariant is the normalized RGB (rgb), other can be hue (H), saturation (S), etc. The one we use is the $c_1c_2c_3$, defined as

$$c_1(x,y) = \arctan \frac{R(x,y)}{\max(G(x,y), B(x,y))} \tag{3.7}$$

$$c_2(x,y) = \arctan \frac{G(x,y)}{\max(R(x,y), B(x,y))} \tag{3.8}$$

$$c_3(x,y) = \arctan \frac{B(x,y)}{\max(R(x,y), G(x,y))} \tag{3.9}$$

This photometric color invariant space proved to be more stable than the rgb which is known to be unstable near the black vertex of the RGB space; also was much more stable than hue in its singularities along the achromatic axis.

3.2.2 *Geometrical properties of shadows*

It is obvious that the geometrical properties depend on object shape and the surface and to which it is projected, but there are some common characteristics we are able to identify.

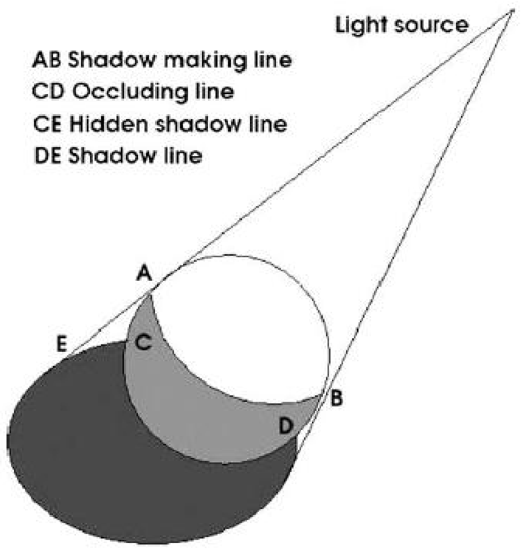

Figure 3.1: Shadow lines definition in [4]

In figure 3.1, each line has a name. The segment AB is called *Shadow-making lines* which separates the illuminated surface of the object and the shaded one. Segment DE is called *Shadow lines* for obvious reasons. Segment CD is called *Occluding lines* and separates the object from its cast shadow. Finally CE, it is defined as *hidden shadow line* because it is the shadow line of a non visible part of the object.

3.2.3 *Algorithm description*

The spectral and geometric properties explained earlier are going to be exploited to progressively strengthen or weaken an initial guess of shadow points. We now state the process of hypotheses generation.

1. We check if there has been a possible lit to shadow change in the neighborhood of a pixel, which may signal the presence of a shadow edge as explained in the first spectral property.

   $$S_c = \{(x, y) : R(x_r, y_r) > R(x, y), G(x_r, y_r) > G(x, y), B(x_r, y_r) > B(x, y)\} \tag{3.10}$$

   where x_r and y_r can be pixels within a neighborhood of (x, y). This property is tested as follows: First an edge detector is applied to all the three channels. The final edge map is made from a logical OR of these three maps. For every edge pixel, the gradient is calculated. Possible shadow pixels, that is $(x, y) \in S_c$, are then found when the sign of the gradient is the same for the three RGB channels.

2. Photometric color invariant property is exploited now, by performing edge detection for the three channels $c_1c_2c_3$. The AND operation is applied to obtain the invariant edge map. The shadow hypothesis is now strengthened wherever there has been non detection

   $$Inv(x, y) = (c_1(x, y), c_2(x, y), c_3(x, y)) \tag{3.11}$$

   $$S_e = \{(x, y) : Edge(Inv(x, y)) = 0\} \tag{3.12}$$

So the hypothesis modifies the earlier one as $S = S_c \bigcap S_e$.

3. Geometrical evidence is verified by checking the existence of the line DE and line CE in S. To obtain these segments, isolated pixels are erased and segments below 30% of the maximum length are deleted. Once we get the complete segment, the occluding line CD is finally exploited. First, the contact points between shadow contour and object contour $Edge(Inv(x, y))$ are detected. Finally, the occluding line is extracted from the $Edge(Inv(x, y))$ contour that connects both contact points.

There are some practical issues to take care of. The edges we find may not be good enough to exploit geometrical properties as described above. Connection points might be difficult to find as well as shadow segments contours. For that, operations like dilations and erasures of isolated points have to be done.

3.2.4 *Example*

For this Mandarin example (Figure 3.3 E), the one used in [4], we show in figure 3.2 all the three $c_1c_2c_3$ invariant images. It can be noticed that in c_1 the shadow barely appears and in the rest, it almost mixes with the background. In figure 3.3 the main steps of the algorithm are illustrated.

First we make the detection of edges that meet the first spectral property, resulting in figure 3.3 A. Also we put together all the edges we have previously detected in the three invariant images $c_1c_2c_3$. In [4], the authors use the AND operation to bring together the invariant edges, here the OR operation is used and therefore the threshold for the edge detectors is accordingly reduced, resulting in 3.3 B.

Once we have image 3.3 B, we perform morphological operations to dilate the edges and when applied as an inverse mask to 3.3 A, eliminate all of the Mandarin surface to keep only the shadow contour. After we eliminate the contours that are not greater than 30% of the longest shadow contour, we obtain the output shown in image 3.3 C.

Then comes the most complicated part. Once we have the shadow contour it has to be connected to the Mandarin contour. This is not a trivial operation. Those contours may not be completed and several contours can exist simultaneously. Picking the right contours is the toughest part of the process. It has been implemented using the geometric property. If we consider both contours as open ellipses we can rearrange all points that follow this model on both contours and finally connect these semi ellipses. After having connected the contours, we get image 3.3 D.

Images 3.3 E and F are the original image and final result for this Mandarin example.

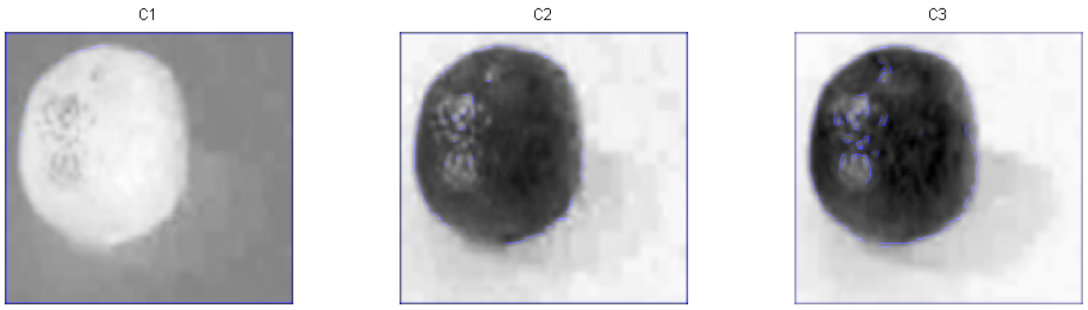

Figure 3.2: Invariant images

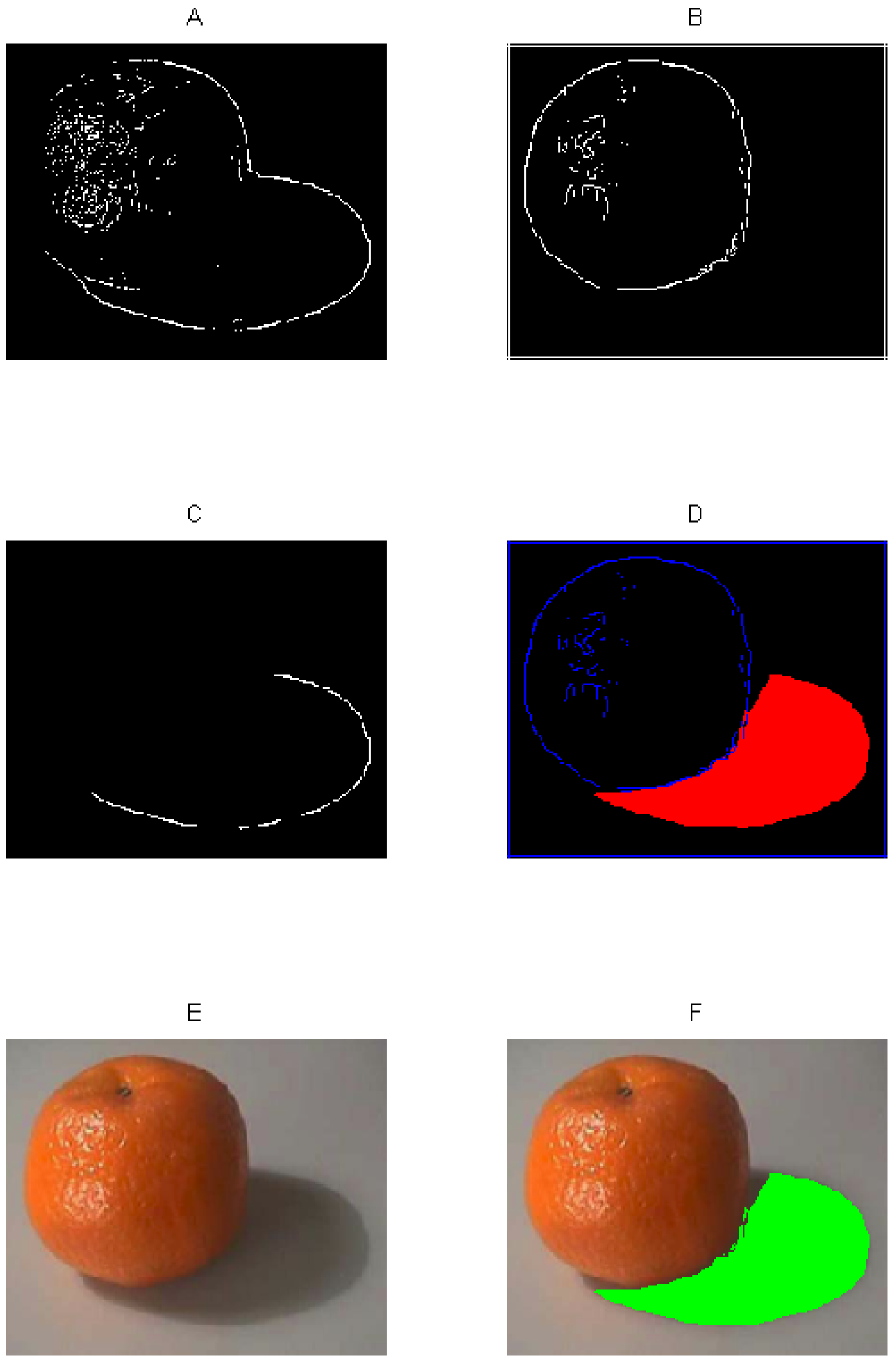

Figure 3.3: Mandarin shadow detection

3.2.5 *Results for Aerial Images*

In this section the algorithm described in the previous section is tested with some modifications to meet the low resolution conditions often found in aerial images. The first problem, that is shown in the figure 3.4, is due to illumination conditions on the image. This makes the invariant method described before not to be that effective for obtaining invariant edges. We might be capturing shadow edges along with object edges.

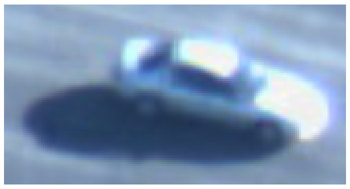

Figure 3.4: White car

In figure 3.4 we see that the shadow is really dark compared to rest of the image. Also the car is too bright. The contrast between the shadow and the ground is very high. Intuitively, it is hard to make the shadow mix with the background-ground. This can be seen in the invariant images.

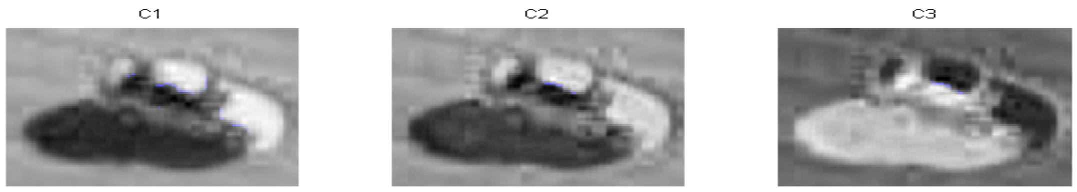

Figure 3.5: Invariant images white car

In figure 3.5 we can verify that the shadow does not disappear at all. Thus, the invariant edge detection may fail. One possible solution is to bring down as much as possible the threshold of the edge detector for $c_1c_2c_3$ images to make sure that all the detected points belong to car edges. If we do so, we will obtain a small number of edges, but at least they will be part of the car and not belong to the shadow. Then we will have to do much more dilation operations to eliminate the maximum number of edges on the car as we obtain the shadow contour.

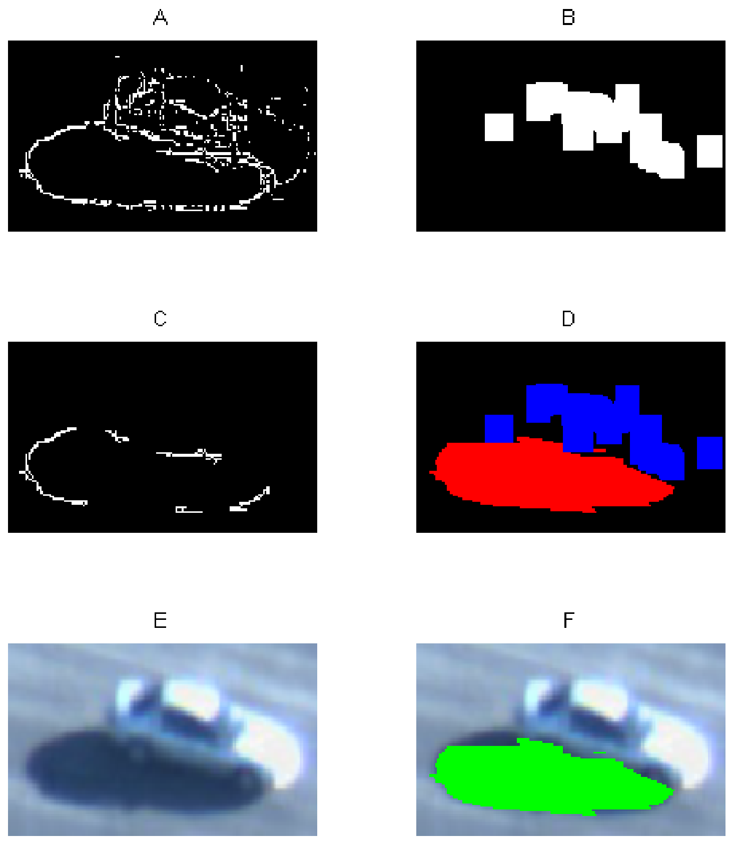

Figure 3.6: Algorithm for white car

We can see the extent dilation needed in figure 3.6 B. Real contours are lost by dilation but at least we are eliminating a lot of the car edges in the shadow detection. As the resolution is very low, loosing real contours is not that critical. The result shown is not the best but the shadow has been almost completely detected.

Some other examples are shown next.

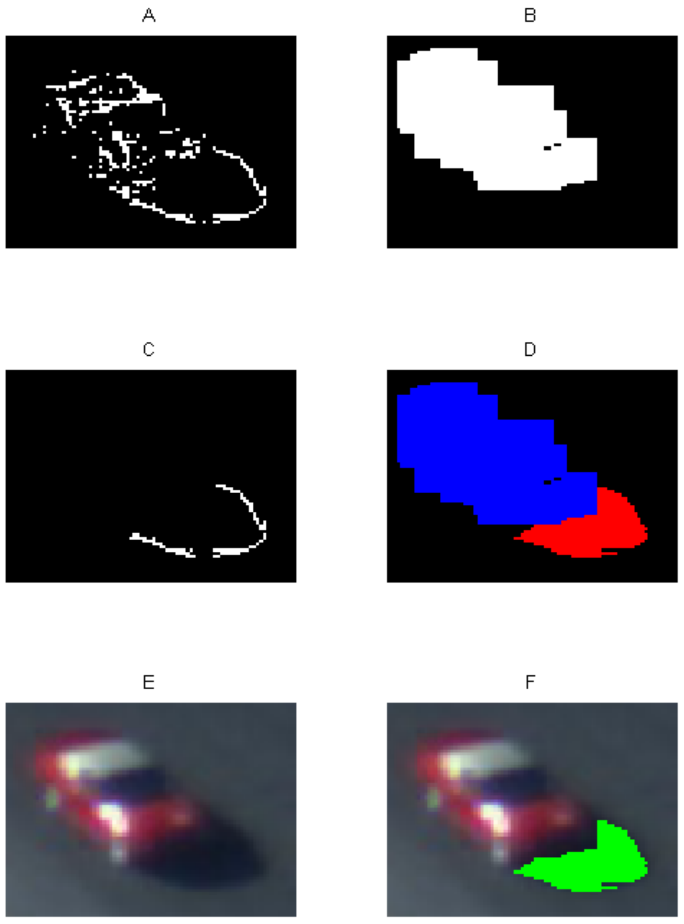

Figure 3.7: Red car example

In the red car example, in figure 3.7, the algorithm works really well. This may be due the clear differences between the car, ground and the shadow. All of these components seem to be very different color wise. This is not true for the case of the blue car in figure 3.8.

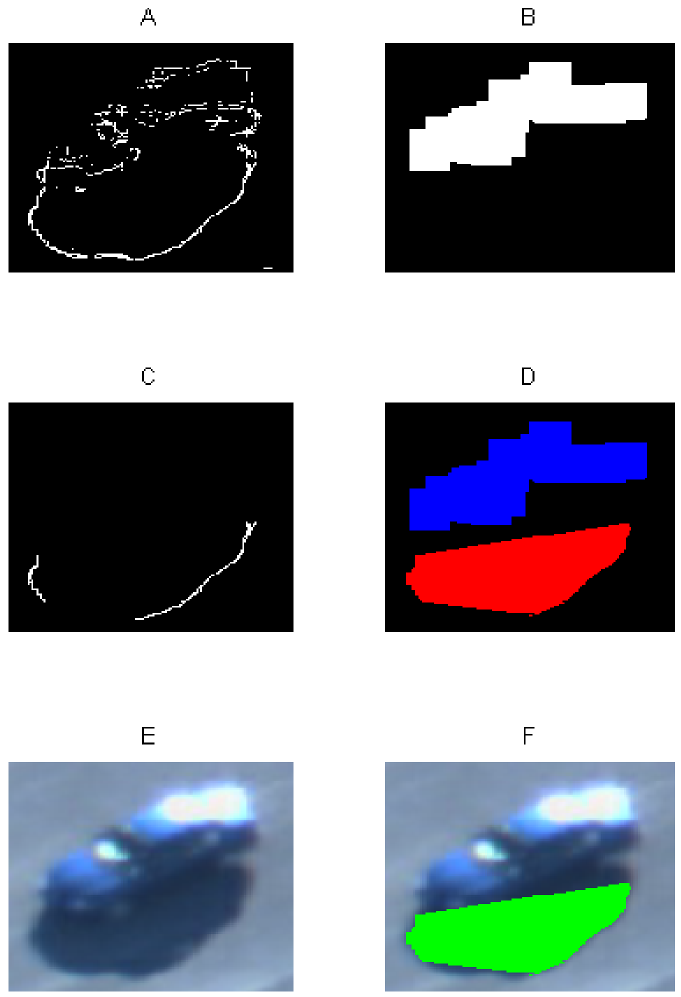

Figure 3.8: Blue car

In the blue bar example, it is really hard, even for a human eye, to find edges between the car and the shadow. Thus, it is impossible for the algorithm to refine the shadow contour at places where it meets the car contour. The shadow we obtain is not the real shadow but still most of the shadow is captured.

Stability is also another issue to worry about. As the image is captured with a lot of noise, edges are never at the same place, which makes the shadows not to maintain the same shape all the time. It also looses some edges between frames. All of this makes the shadow detection process very unstable as shown in figure 3.9, one second clip at 30 fps of a white car.

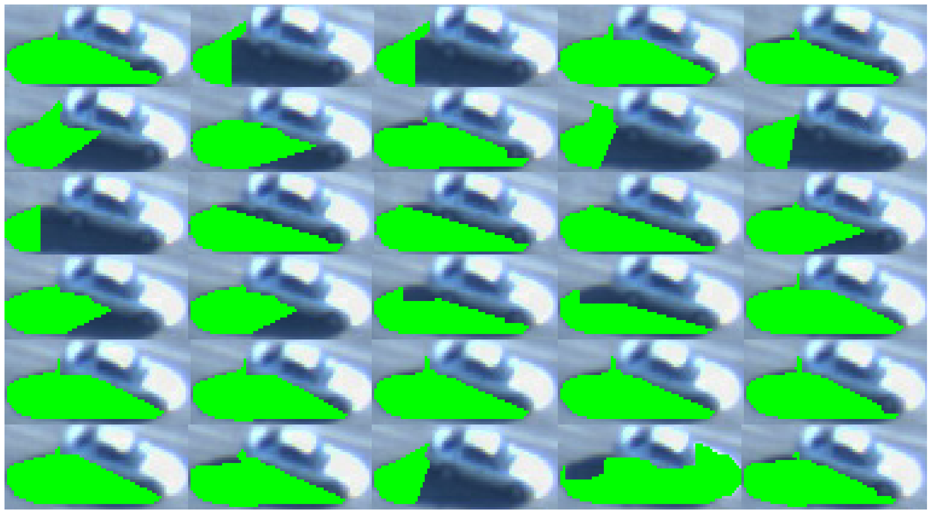

Figure 3.9: White car 30 frames sequence

This stability issue combined with the fact that the invariant images are not that effective for capturing the car edges, the result can be very bad. In the next sequence, shown in figure 3.10, we begin to see that this algorithm relies on too many random factors that can make it fail. But still the outcome is acceptable.

It has been stated that the most difficult part of the process is obtaining good invariant set of images. In the next example, it is shown how severely this problem can affect the rest of the algorithm.

In figure 3.11, a lot of different edges can be found. Still the human eye can

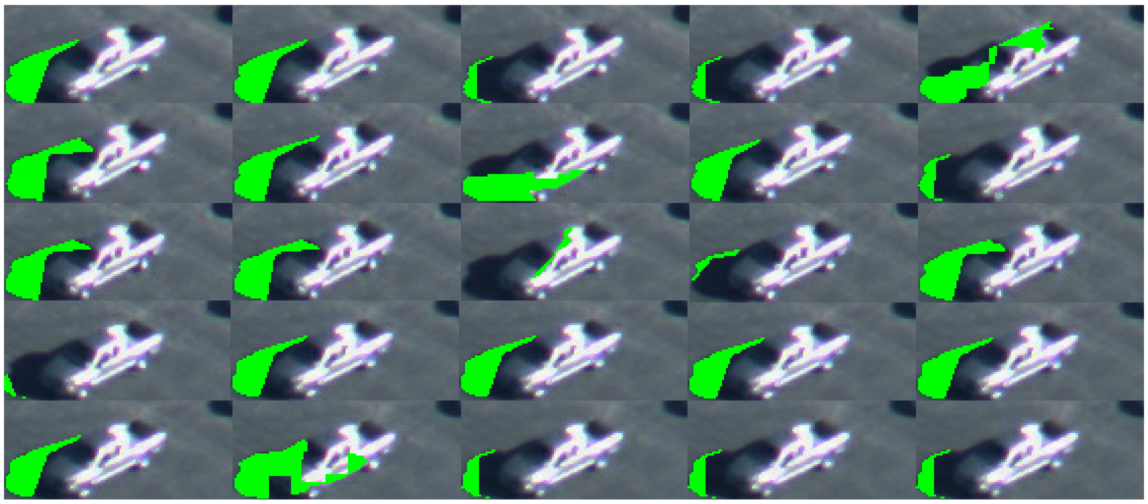

Figure 3.10: Silver truck 30 frames sequence

differentiate the many shadows that appear. There are more shadows than the one belonging to the car.

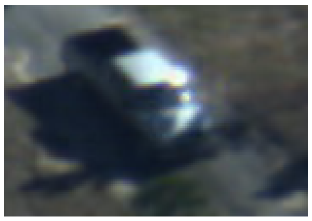

Figure 3.11: Truck 2

In the invariant images, shown in figure 3.12, it can be seen that the differences between shadow and ground have been enhanced. Still we are able to obtain edges belonging to the truck. The fact that we have to reduce the threshold of the invariant edge detector a lot to only find truck edges makes the number of these edges to be very small. Also the fact that the original image has a lot of different edges, some belonging to the truck, some belonging to the shadow of the truck and the rest

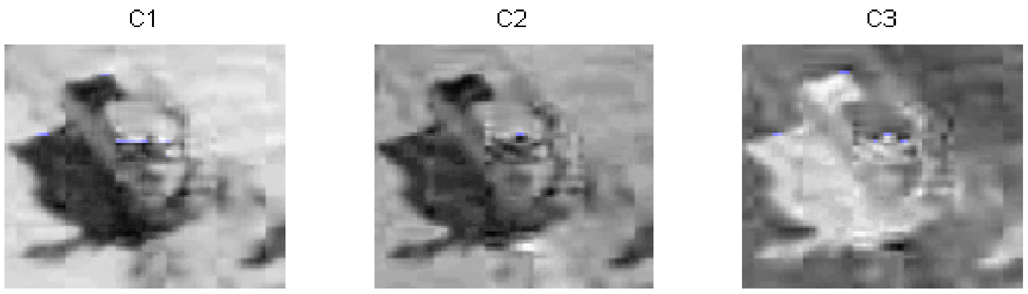

Figure 3.12: Invariant Images of Truck 2

belonging to tree shadows, road, etc, will make the algorithm find something that is erroneous.

In figure 3.13 A, we can see all the edges detected, including possible shadow edges. In 3.13 B we see a small number of invariant edges found in the truck. When we use these invariant edges to eliminate the truck edges in 3.13 A, we eliminate some but we do not obtain only the shadow contour, as can be seen in 3.13 C. Finally, when we build the complete shadow we have a wrong one. That's because we are including edges that do not belong to the truck or the shadow in the detection process.

One solution is to bring down the threshold for detecting possible shadow edges, A. But that has been tried and if we do so we tend to loose a lot of real shadow edges. That is also one of main drawbacks of this algorithms, i.e., the results are too sensitive to threshold.

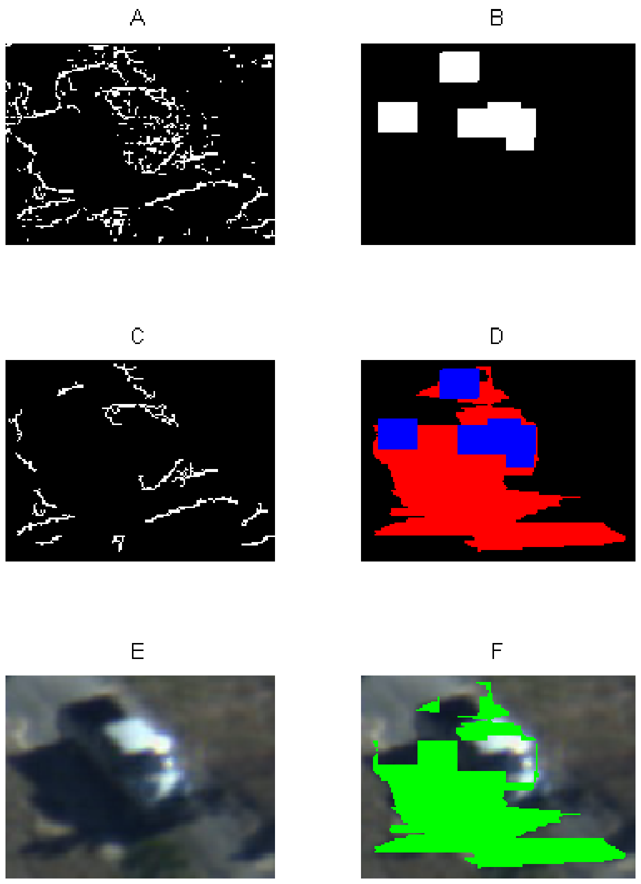

Figure 3.13: Algorithm of truck 2

3.2.6 *Evaluation*

In this algorithm, a first approach to shadow detection exploiting its illumination properties has been tested with decent but not enough successful results. The illumination invariance space proposed seems to be right for some cases and be inappropriate for others. Multiple scenarios have been tried because robust detection was one of the goals. For that reason new methods for illumination invariance generation of images are suggested.

Another problem is that even having a really good invariant space, where only object edges are detected, so that the right contour of the shadow and its connection to the object can be found is not an easily solvable problem. It has to do with shape detection and some processing has to be done. The way it has been done was very simple and prone to fail. If the shadow is not an open ellipse then the method used to close contours can give undesirable results.

Robustness is something desirable and is not being achieved either. As we extract the shadow from an object to get the shape of that object, we want the extracted part to be the same or almost the same for every frame. This is another problem that is addressed in later sections.

Finally, a practical drawback arises as has been already explained, due to the algorithm being too sensitive to thresholds. Under the same illumination condition the thresholds for different objects may remain quite stable, but if the illumination condition is changed then all thresholds must be readjusted to new illumination conditions. This is critical when an algorithm that does need fine tuning for every scenario is desired.

3.3 Illumination Invariance through Camera Color Calibration

In this section, we use a different approach to obtain illumination invariant images. It is based on the Planckian model of light and how it reflects on objects. If lightning is approximately Planckian, then as illumination color changes, a log-log of 2D $\{\log(\frac{R}{G}), \log(\frac{B}{G})\}$ values for a single surface will form a straight line. Therefore, the lightning change reduces to a linear transformation along almost a straight line. Using this line we transform our 2D log-log to a single 1D grayscale invariant image. One way to do it is to try a panel with different colors on it and then change the illumination intensity. Another way is to capture images from the same location at different times of the day. Once that line is found, obtaining the invariant image will be just a linear transformation of all those 2D points into the ”orthogonal line” to the one holding all the different illumination conditions.

3.3.1 *Invariant Image Formation*

In figure 3.14, it is shown how illumination reflects on a surface. The next step is to understand how this illumination is registered into a camera producing the RGB colors. The invariant formation for Lambertian surfaces is determined by:

$$\rho_k = (a \cdot n) \int_{\Lambda} E(\lambda) S(\lambda) Q_k(\lambda) d\lambda$$

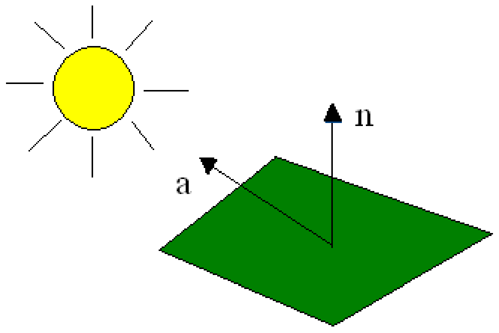

Figure 3.14: Illumination of a surface with normal $\vec{n}$ and lightning direction $\vec{a}$

Here $k = 1..3$ represent the RGB components, $E(\lambda)$ is the illumination and $S(\lambda)$ is the reflectance of the surface as a function of the wavelengths respectively.

$Q_k(\lambda)$ is the sensitivity of the sensor for each RGB channel as a function of the light wavelength.

Planckian Light

This is the first assumption we make, following [5]. Actually we use Wien's approximation to Plank's law. This is:

$$E(\lambda) = Ic_1\lambda^{-5}e^{-\frac{c_2}{\lambda T}}$$

Here c_1 and c_2 are constants, I is the intensity, T the temperature and λ the light wavelength.

Narrow Band Sensor

This is the second assumption we make. For simplicity we assume that the camera sensitivity for each RGB channel is a delta function. The Lambertian reflection model then gets simplified and that assumption is not very far from reality. Thus, our sensitivity is:

$$Q_k(\lambda) = q_k\delta(\lambda - \lambda_k)$$

If we apply both assumptions to our Lambertian model, this is going to be simplified a lot, as follows:

$$\rho_k = (a \cdot n)\int_{\Lambda} E(\lambda)S(\lambda)Q_k(\lambda)d\lambda = c_1(a \cdot n)IS(\lambda_k)\lambda_k^{-5}e^{\frac{c_2}{\lambda_k T}}q_k$$

Then here comes the linearization part of the equation using the log function:

$$\log \rho_k = \log((a \cdot n)I) + \log(c_1 S(\lambda_k)\lambda_k^{-5} q_k) - \frac{c_2}{\lambda_k T}$$

Here $\log((a \cdot n)I)$ depends only on illumination and shading, $\log(c_1 S(\lambda_k)\lambda_k^{-5} q_k)$ depends on the surface reflectance and camera sensors (q_k, λ_k) and $\frac{c_2}{\lambda_k T}$ is determined by illumination and camera sensors. We can eliminate the first illumination term very easily if we define a new variable called $r = \log(\frac{\rho_1}{\rho_2})$. Therefore,

$$r = \log \frac{S(\lambda_1)\lambda_1^{-5} q_1}{S(\lambda_2)\lambda_2^{-5} q_2} + (-\frac{c_2}{\lambda_1} + \frac{c_2}{\lambda_2})\frac{1}{T} = \alpha_r + \frac{\beta_r}{T}$$

Here $\alpha_r = \log \frac{S(\lambda_1)\lambda_1^{-5} q_1}{S(\lambda_2)\lambda_2^{-5} q_2}$ and $\beta_r = (-\frac{c_2}{\lambda_1} + \frac{c_2}{\lambda_2})$

Similarly,

$$b = \log(\frac{\rho_3}{\rho_2}) = \log \frac{S(\lambda_3)\lambda_3^{-5} q_3}{S(\lambda_2)\lambda_2^{-5} q_2} + (-\frac{c_2}{\lambda_3} + \frac{c_2}{\lambda_2})\frac{1}{T} = \alpha_b + \frac{\beta_b}{T}$$

Here $\alpha_b = \log \frac{S(\lambda_3)\lambda_3^{-5} q_3}{S(\lambda_2)\lambda_2^{-5} q_2}$ and $\beta_b = (-\frac{c_2}{\lambda_3} + \frac{c_2}{\lambda_2})$

If we parametrize these two equations into one by isolating the temperature T, we get the linear relationship;

$$r - \alpha_r = (b - \alpha_b)\frac{\beta_r}{\beta_b}$$

So, for every pair of points, (r, b), belonging to the same surface will be all lying on a line following the linear relationship in the previous equation. The slope will be determined by the ratio between $\frac{\beta_r}{\beta_b}$ which is:

$$\frac{\beta_r}{\beta_b} = \frac{\lambda_1 - \lambda_2\lambda_3}{\lambda_3 - \lambda_2\lambda_1}$$

and will be determined by camera sensitivity. This is where camera calibration has to be done. Once we have determined this ratio, the line slope, we can build the invariant image. This is an illustration, not a real experiment, to show how this calibration process can be carried out.

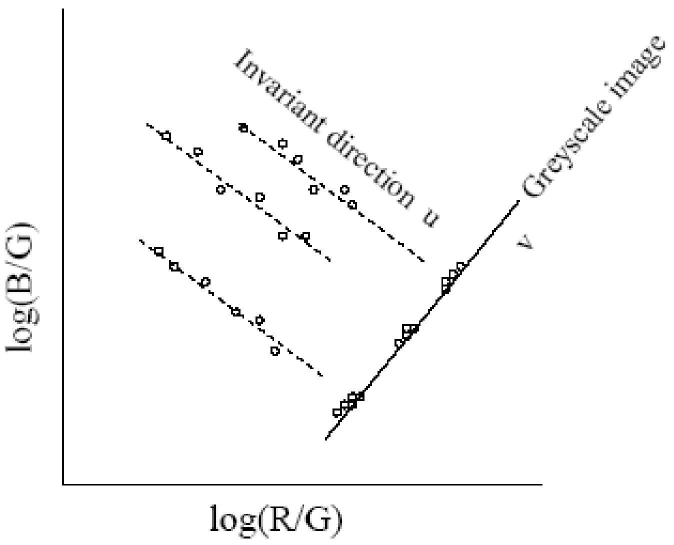

Figure 3.15: Calibration Illustration

In figure 3.15, the main steps of the calibration process are shown. First, we capture a couple of colors with different illuminations. In this case we have three different colored surfaces captured with 6 different illumination conditions. Then, as expected, every point of each surface lies on a line defined by the slope previously stated. All surfaces follow the same invariant direction u. If we find the orthogonal direction to u, denoted as v, for all the different illumination conditions we will have the same grayscale value. Thus, we have created an illumination invariant grayscale image.

There are a couple of ways to obtain the invariant direction u. We could take one single color and then fit the best line using the least squares method. Thus, for all the colors we could check if we are getting the same line. Another method is to find the direction that minimizes the entropy of the grayscale image. This is reasonable because we would like to obtain a Grayscale image with the least variance around every single color spot. We formulate this as,

$$gs = r \cdot \cos\theta + b \cdot \sin\theta$$

Here gs is the grayscale value when we project every single (r, b) point into the orthogonal direction described by θ. We then estimate the probability of grayscale values by calculating the histogram of gs. We call this $\hat{p}(gs|\theta)$. Thus, to obtain the best orthogonal direction we only need to minimize,

$$\hat{\theta} = \{\theta | \min - \sum_{gs} \hat{p}(gs|\theta) \cdot \log \hat{p}(gs|\theta)\}$$

3.3.2 *Invariance without calibration*

As has already been explained in our case we do not have the training image to learn where to detect the shadow under different illuminations nor we have the camera calibrated. Thus, under this scenario how does one use prior information to remove shadows? A very simple idea has been implemented. We know there is a direction where the shadow should mix with the background due to illumination invariance, therefore we could from a single image obtain this direction that makes the shadow vanish in the background. In the next figure we can see how the grayscale image varies as a function of the direction θ.

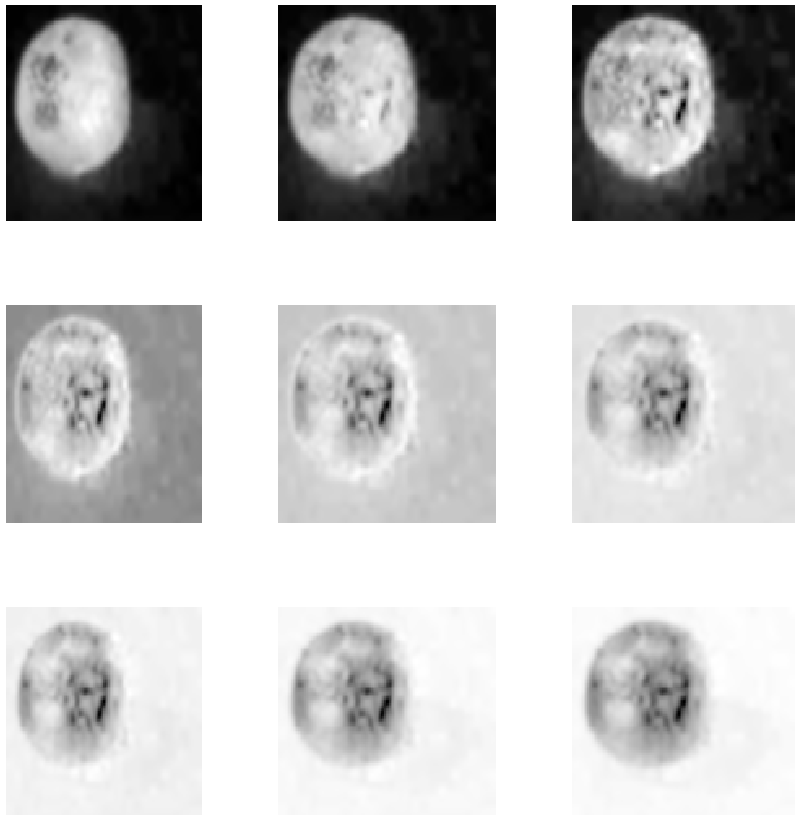

Figure 3.16: Mandarin under different θ projection from 0^0 to 90^0

A low-pass filter has been applied to smooth the images. In figure 3.16 we see that there is a particular direction where the shadow mixes most with the background. This direction will have the least entropy among the rest because, in theory, we should be obtaining a gray value for the object surface and another for the background. We see that our guess seems to be right. So we look for the invariant image as the one with the least entropy as if we were calibrating the camera following the process described earlier.

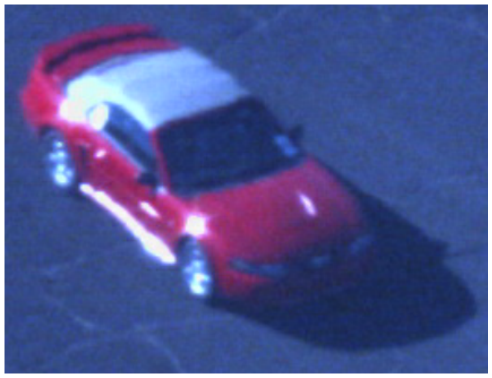

Figure 3.17: High resolution red car

In figure 3.17 we have a red car in a higher resolution than we normally see in an aerial image. For a human eye it is pretty easy to follow all the contours between the car and the shadow. If we see some projections of it over different θ, in figure 3.18, there are some projections where the shadow almost disappears in the ground. We plot the entropy for each projection and see if there is a minimum.

We obtain figure 3.19 by calculating the entropy for several θ in the range from $(0, \pi/2)$. It can be seen that there is a minimum but also that the minimum is very wide. The resulting image after the projection is shadow free. This can be seen in the minimum entropy projection in figure 3.20. We have finally eliminated all the shadow.

In figure 3.21, we have a set of four different examples with various illumination conditions. In all of them is seen the convex shape of the entropy with a global minimum in the $(0, \pi/2)$ range. The Mandarin example is the one we have already seen many times. Particularly good is the apple example. We almost see

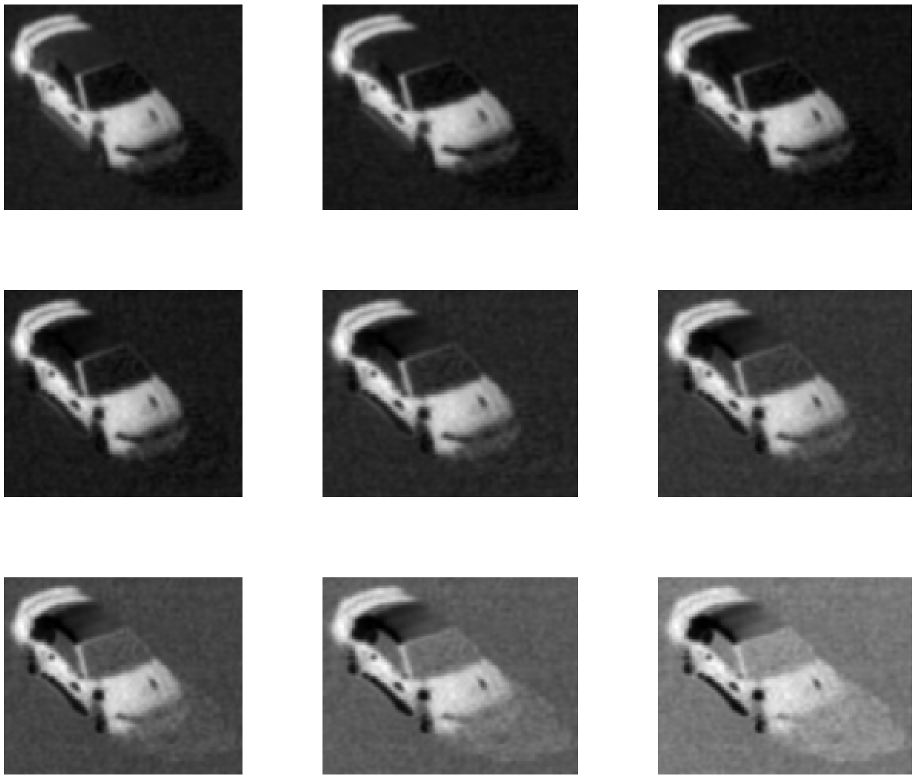

Figure 3.18: High resolution red car projections

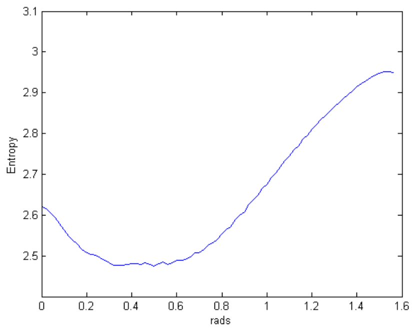

Figure 3.19: Entropy for different θ of the red car

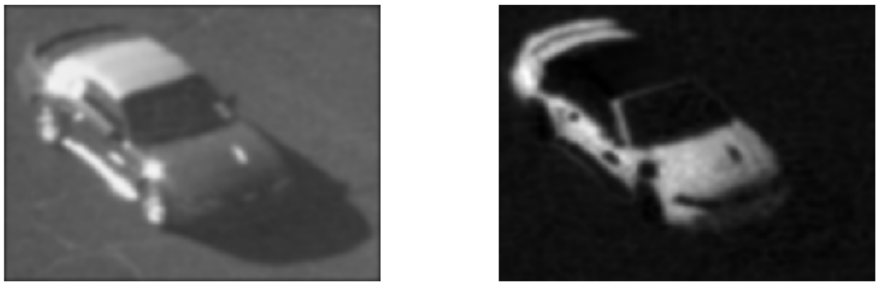

Figure 3.20: Original grayscale and Invariant grayscale

the complete shape and the shadow is all but gone. With the white car we are not that successful. By eliminating the shadow, we are also eliminating the side of the car which has a big component of shading. Thus, we will not be able to obtain a good shape of the car in this case. Last but not least, we have the best result for this set of examples. From a very narrow entropy function, with a very clear minimum, we obtain an invariant image where the shadow is completely eliminated from the ground. In this case we can say we clearly obtained the invariant to illumination direction.

From the examples shown, we conclude that this method is effective for obtaining illumination invariant images, but for aerial images this method fails.

Aerial Images

As has been explained already, when we are dealing with really low resolution images, detection methods might perform differently than for higher resolution ones.

The fact that the resolution is very low makes the noise dominant in image capture. We do not have nice surface colors, shadings and shadows. Instead, we are

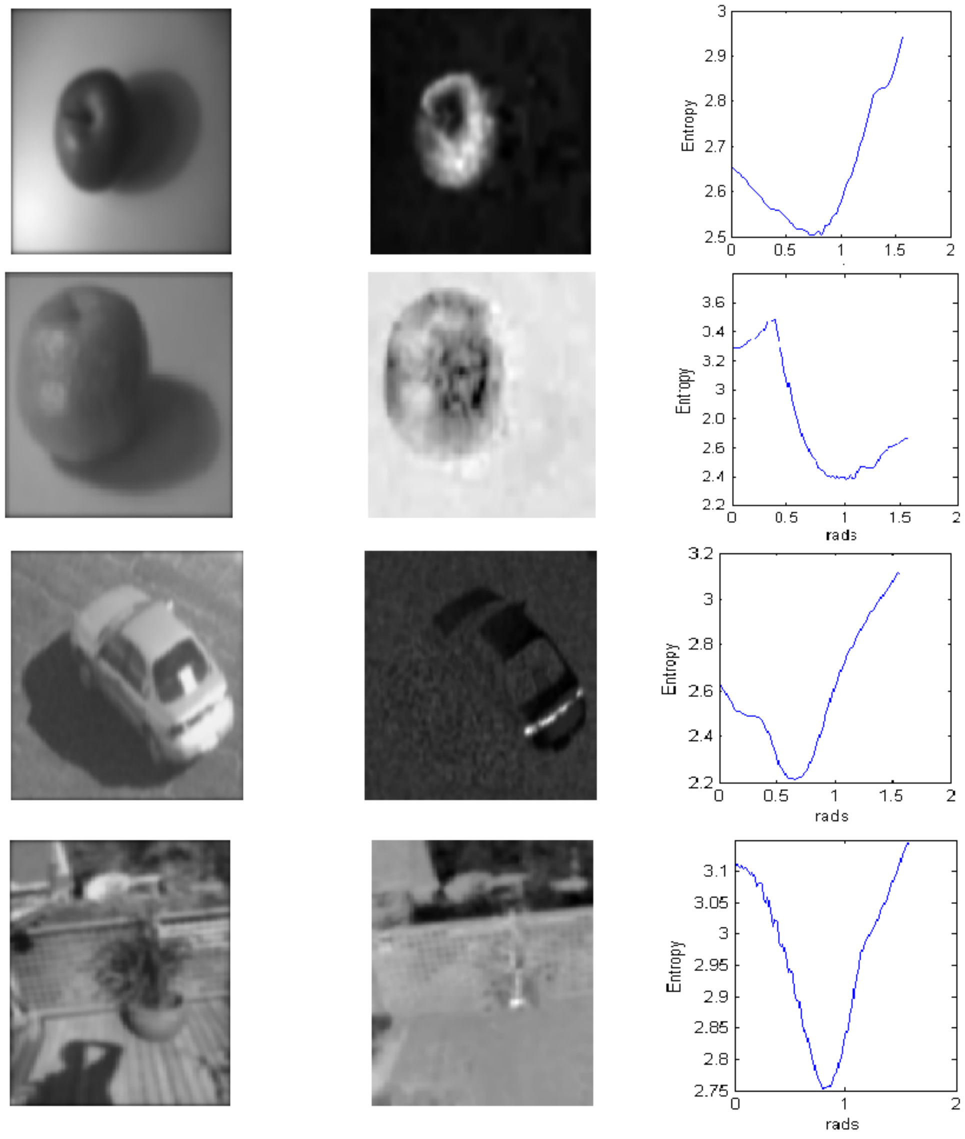

Figure 3.21: Invariant formation examples

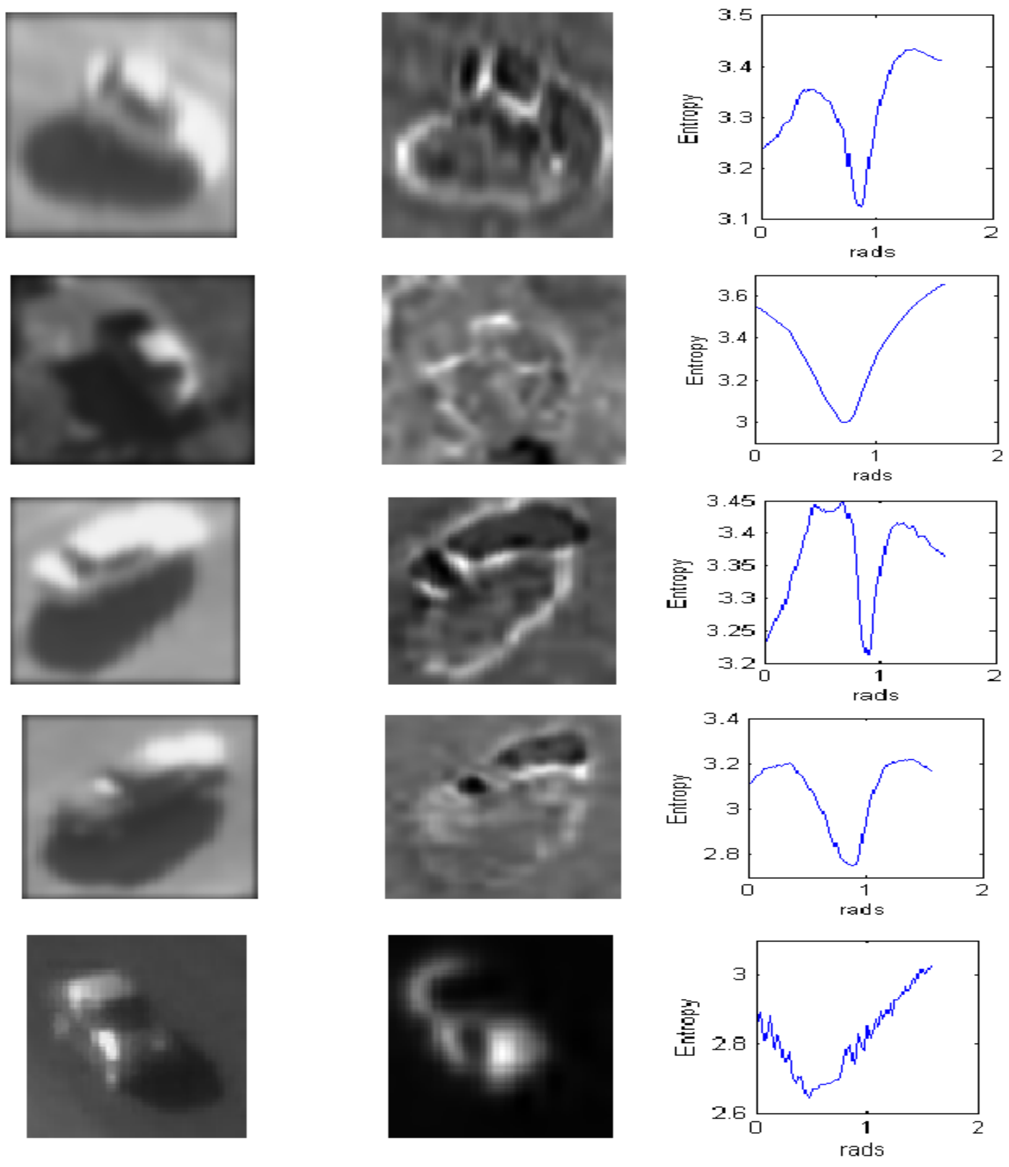

Figure 3.22: Aerial invariant examples

dealing with noise that changes surface colors in a significant way. Also transitions between shaded edges are stronger than in images at higher resolutions. The fact that the shadows merge with the ground but we still see its edges even more than before makes it difficult to have good detections. Only for the last two examples we obtain good shadow detection. Actually the last one behaves almost identical to its higher resolution version. As this method did not work that well for aerial images, in the next section a new method is proposed. It is based on a similar idea with simpler implementation.

3.4 Illumination Invariance through Homomorphic Filtering

The intensity of an image, as has been described earlier by the Lambertian model, is the reflection over a surface of an incoming illumination. Let us assume a simpler model than the ones used earlier [7]. Thus, we can state that the intensity we see in an image is,

$$y(k) = i(k) \cdot r(k)$$

Here k is the pixel index, i is the illumination and r is the reflectance component. The goal is to separate r from i and then explore the characteristics of both separately. In many realistic cases the illumination component is slowly varying through the image and the reflectance component is rapidly changing due to surface shape. We use this fact to low pass filter the illumination component.

First we transform the multiplication factor into a summation, by applying the log function. Then,

$$\log(y(k)) = \log(i(k)) + \log(r(k))$$

Although this transformation modifies the spectral content of illumination and reflectance components, it is not very far from reality to assume that log-illumination to be slowly varying too. It is also important to state that in general we have to introduce a gamma factor to the intensity formation due to the camera non-linearity. Then,

$$y(k) = y_{in}^{\gamma}(k)$$

but when we apply log to y, this gamma factor multiplies both r and i and hence does not affect the filtering process.

The last part of the process is to exponentiate again the filtered components of r and i. For detection purposes we could stay in the log domain but it is better to return to the original domain as otherwise we will be propagating the noise to a non-linear space.

In figure 3.23, the complete process is shown. The low pass filter used is a binomial filter,

$$f(k,n) = \begin{pmatrix} K \\ k \end{pmatrix} \begin{pmatrix} N \\ n \end{pmatrix}$$

Here K and N are the height and width of the filter and the pair (k,n) indicates the pixel position.

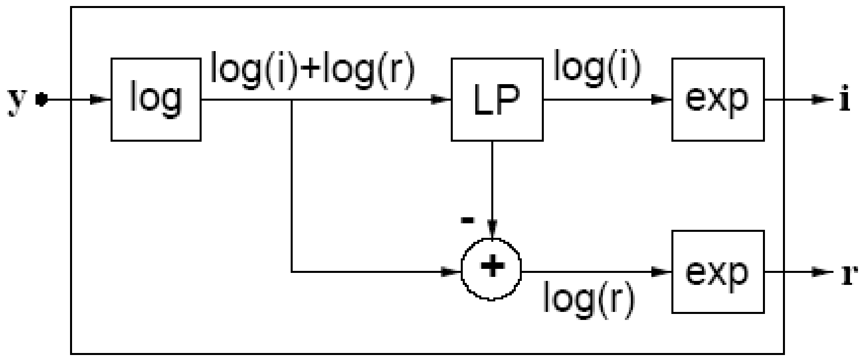

Figure 3.23: Illumination and reflectance component separation

3.4.1 *Algorithm description*

A summary of all the steps is explained as:

1. Conversion from RGB to HSV is made to get a stable intensity value V. We do not use color information for this method.

2. With the intensity value corresponding to y, a log operation is made to make the multiplication a sum and also to get rid of the non-linearity of the camera.

3. A 2D filtering scheme is implemented using the binomial kernel described before to get the $\log(r)$ and the $\log(i)$ components separately.

4. An exponentiation transformation is applied to both components independently to get the r and i values. But as we are more interested in shadows, an inverse illumination map is used instead. Thus, we exponentiate $-i$ instead of i.

5. Once we get the inverse of illumination map, we threshold it in order to get the shadowed regions of the image.

3.4.2 *Example*

The typical Mandarin example is used again, even though it is not the one that gives best results. As seen in figure 3.24 the reflectance map is not good enough to extract the shadow as we were able to do in previous methods, because we still see the contour of the shadow in the invariant reflectance map. We have to use a new technique. The idea is to threshold the inverse illumination map.

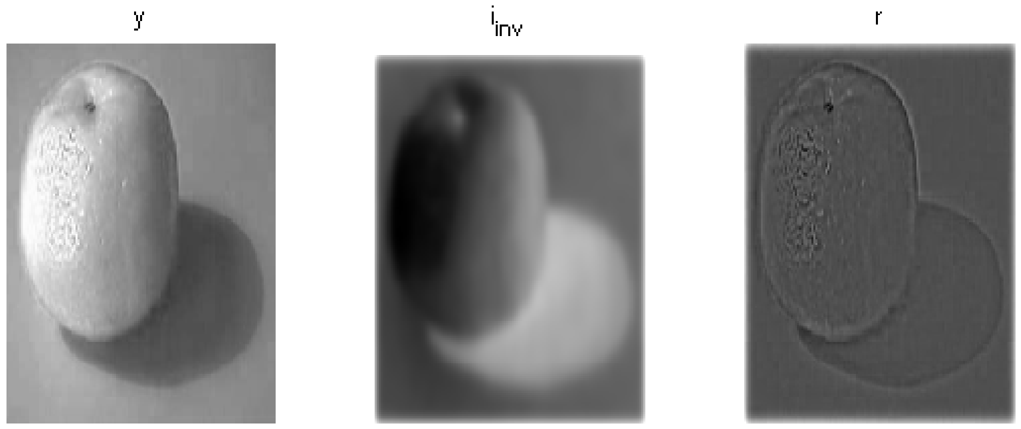

Figure 3.24: Image y, inverse illumination i_{inv} and reflectance r of the mandarin

The final result is presented in figure 3.25. We see almost all the shadow detected but still it is not a perfect result. If we bring the threshold down we will capture the whole shadow along with some shaded region of the fruit.

In figure 3.26 we see how the method performs perfectly for detecting the photographer's shadow. In the first example, we see that the shadow generated by the pot, the leaves of the plant, the tree shadows and some more are detected. They seem bigger than they really are due to low pass filtering. The result is very good.

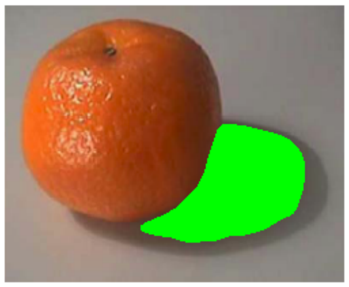

Figure 3.25: Mandarin shadow detection

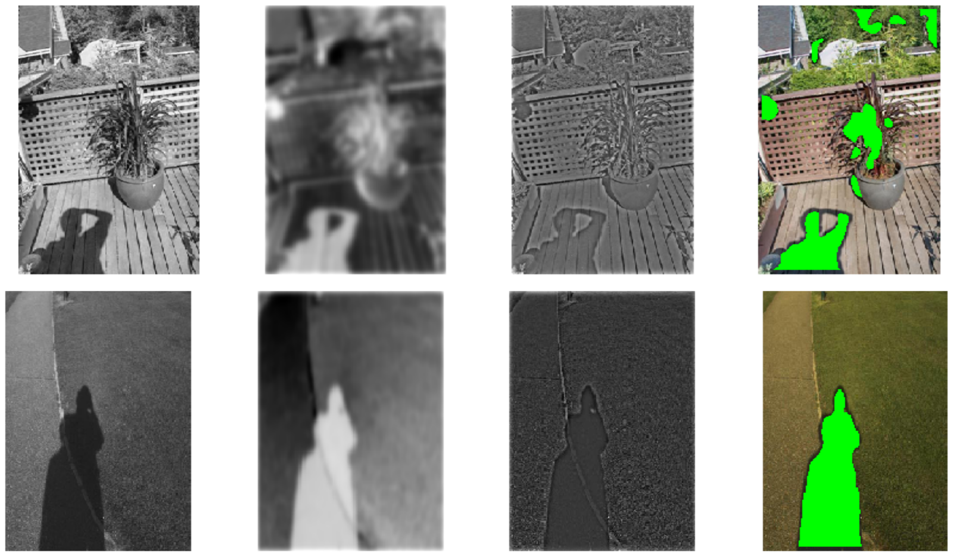

Figure 3.26: Human shadow under different scenarios

Now we see how the algorithm behaves when we are dealing with aerial images.

3.4.3 *Results for Aerial Images*

This method is the one that works best for typical vehicle detections that have been tried. This can be seen in figure 3.27. Detection in all three cases is almost perfect.

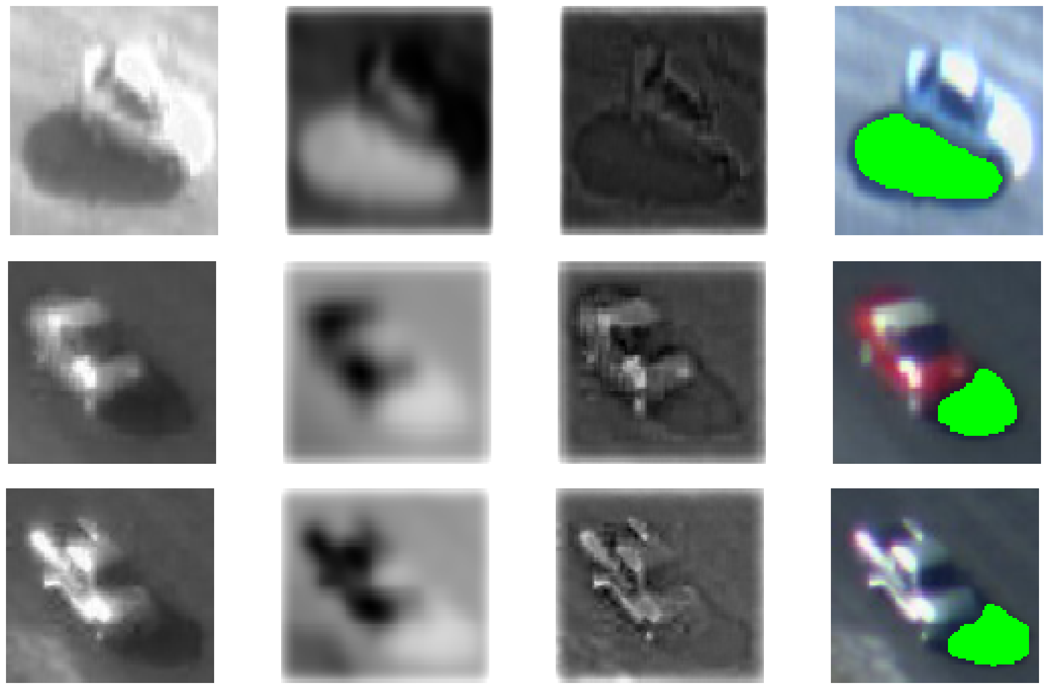

Figure 3.27: Some aerial images

Challenging Scenarios

In figure 3.28 we show that even when the previous methods failed to detect the shadows in aerial images, this method performs quite well. In figure 3.28, shadow is shown in blue and the lightest illumination on the image in red.

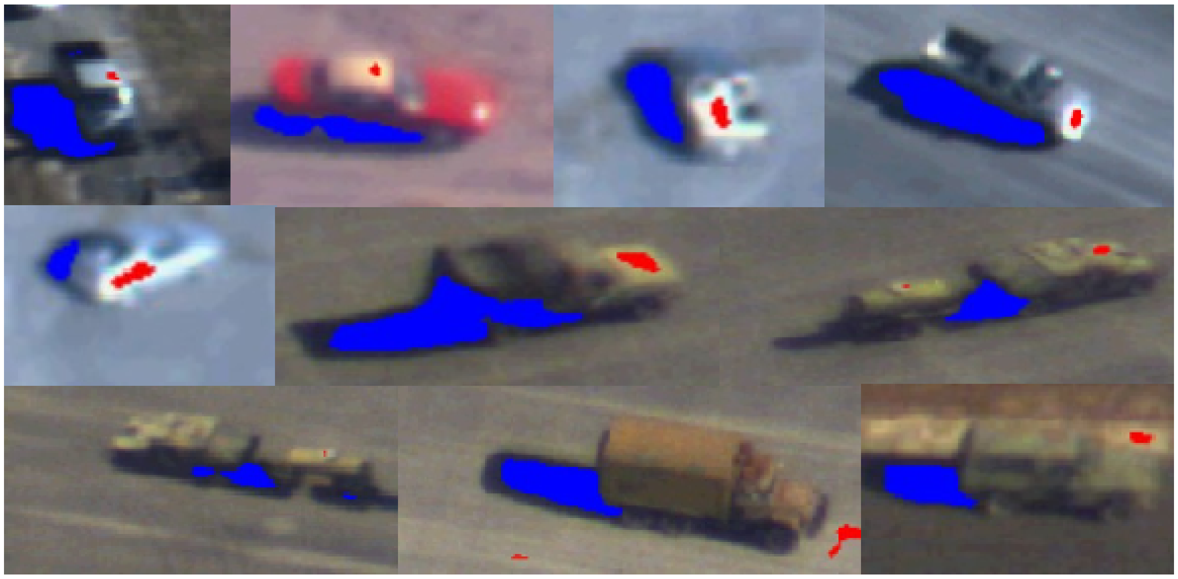

Figure 3.28: Challenging scenarios

Stability

Finally, in figure 3.29, it can be seen that the algorithm behaves in a stable manner compared to the earlier methods and stability is one of the major achievements that will enable enhaced tracking.

3.5 Conclusions

We have demonstrated several methods for obtaining invariance to lighting images from colored input images. It has been shown how important it is to obtain these kind of images to be able to detect shadows properly. Hence, through the

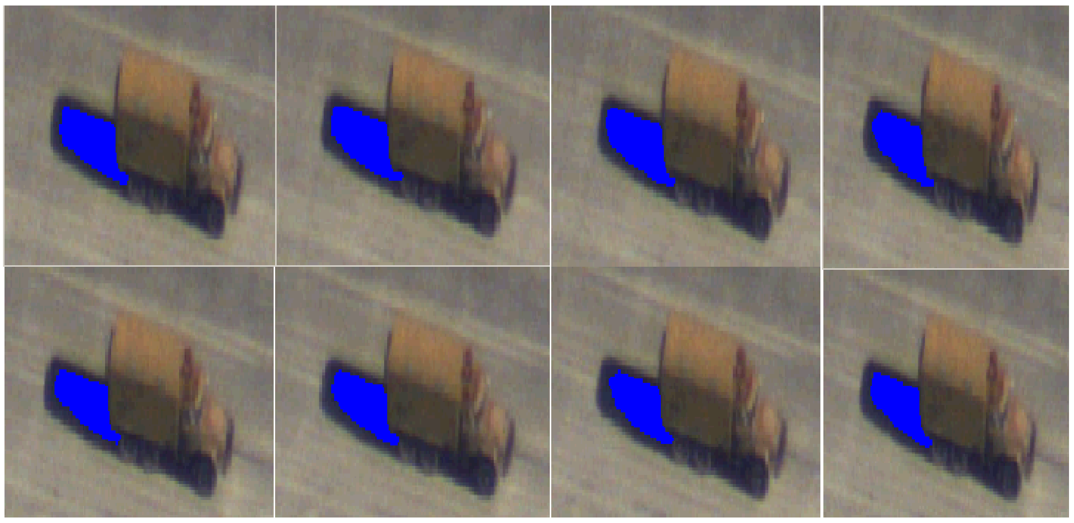

Figure 3.29: Stability in the detection

segmentation of the shadow, better detection of objects is possible and the performance of tracking algorithms also improves significantly. The segmentation problem has also been extensively studied. First, an intuitive method to obtain the shadow contour was presented. This method is effective when the invariant image is good, and when the shape of the shadow follows the assumed model. In order to get better invariant images, a more complete lighting model has been explained and tried. Due to the size of our input images and the noise added while capturing them it was difficult to apply the complex lighting model. Thus, we suggest using a simpler model which proved to be the one yielding best, stable results.

When dealing with shadow detection, to eliminate them from the object we want to detect, stability is as important as good detection. Even if we manage to good detections, if the shape of the shadow changes constantly, then the shape of the object also changes making harder for the tracker to characterize the object.

Therefore we have to focus on detection accuracy and on stability. It turns out that homomorphic filtering for detecting shadows is the best method ground and airborne videos. It's true that the first invariant method was effective sometimes in detecting the shadow but it was not stable.

Several steps can be taken to improve the performance of the proposed methods. One is to have good camera calibration. Another one is to explore the random properties of the shadows and try to come up with good statistical methods to detect them. One can also study the casting of shadows and explore model-based methods to find shadows; this is possible if we know a priory or we have at least an estimate of the shape of the object and the sun direction, a rough estimate of the shape of the shadow can be made. All these methods remain to be studied and compared with the methods presented in this chapter.

Chapter 4

Model Based Method

4.1 Introduction

Recognition systems can be improved incorporating a prior model of the object that we wish to detect-recognize. In this chapter a simple method is suggested to recognize shadows and use the information to classify an object as a human being or a vehicle.

When we are dealing with aerial images it is more complicated to model humans compared to when we are dealing with ground based surveillance videos. Also, the algorithms we use to detect shadows in surveillance videos are derived from the background subtraction. As we may not have any prior information on backgrounds, we need other methods. Illumiation invariance methods may work, but as has been already shown may not always be effective.

We need first to define the problem. We only expect the image to have moving humans or vehicles in our field of view. When the presence of a big shadow region may mislead standard recognition algorithms, as we model human shadow as a line oriented in a particular direction, established by the sun, and vehicle shadow does not fit this model. We find the *Hough lines* in the edges of the image and see how consistent they are with our assumption.

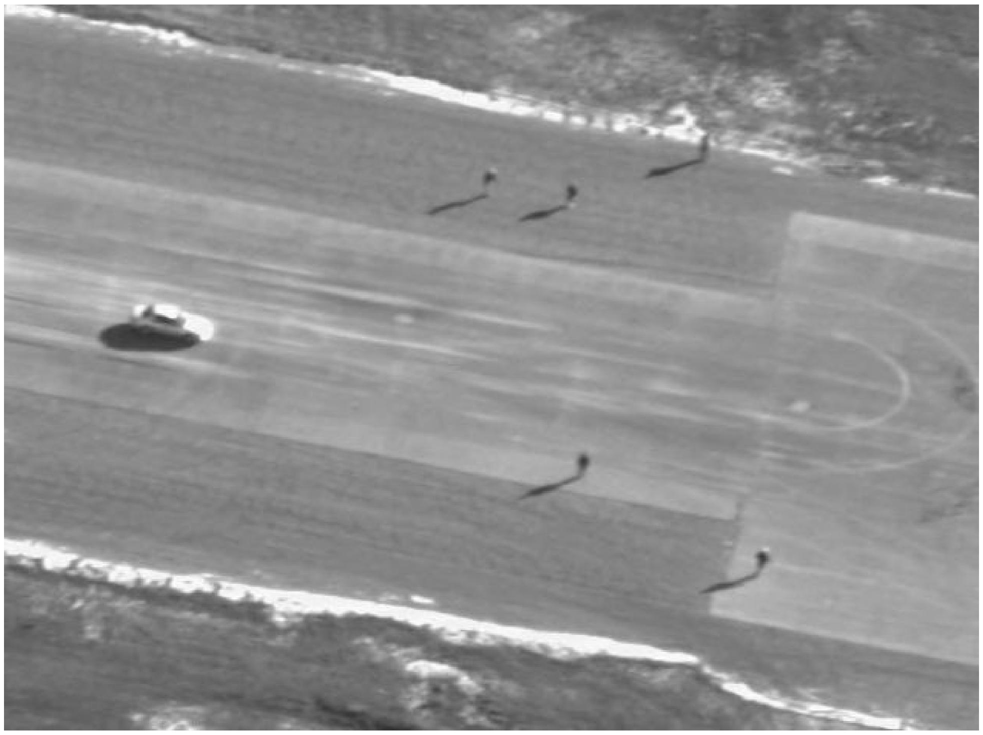

Figure 4.1: Input Image

After we have segmented the regions where movement was detected, we perform an edge detection in every frame. We then get the main *Hough line*, as it is going to be explained. The edge detection results are aggregated. The final result at the right shows how all the extracted *Hough lines* were added to that final image, where black means more presence of that line during the frame addition process.

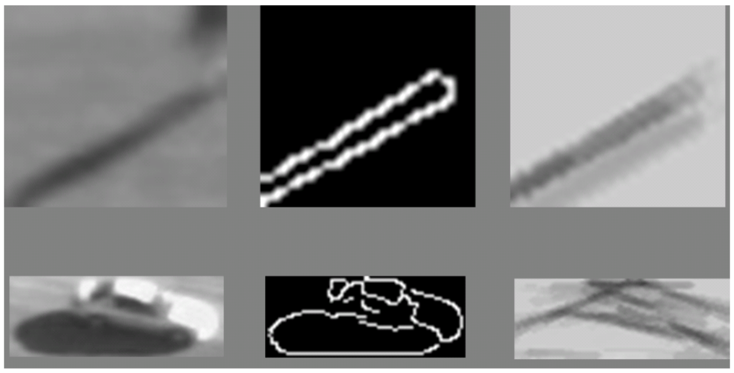

Figure 4.2: Hough lines after 30 frames

It is clear from figure 4.2 that the lines coming from a human are more stable in its direction than the ones coming from a vehicle. As the direction is established by the position of the sun only, it is common to all humans appearing on the scene. This is a key issue when dealing with classification.

It is obvious that the line we extract from of the *Hough lines* will be relevant when we are dealing with humans, otherwise we get random lines only from the contours of the vehicle. Thus, our algorithm only works for detecting human shadow.

Despite in the previous examples we have used thirty frames to show how the lines are obtained and their consistency, we need only a few frames, and that makes the algorithm to have quite a small delay. Very desirable when we need real time algorithms.

4.2 Algorithm Overview

Before stepping into the algorithm, I will briefly explain the method used to obtain the *Hough lines.* As is commonly known the Hough transform is a 2D transform of all (x, y) points of an image to an (ρ, θ) domain defined as follows,

$$\rho = x_i \cos(\theta) + y_i \sin(\theta) \ for \ \forall(x_i, y_i)$$

If the image is not black and white we scale ρ by its grayscale value. For every line that passes through a pixel we get curves in the (ρ, θ) domain, as it is shown in figure 4.3.

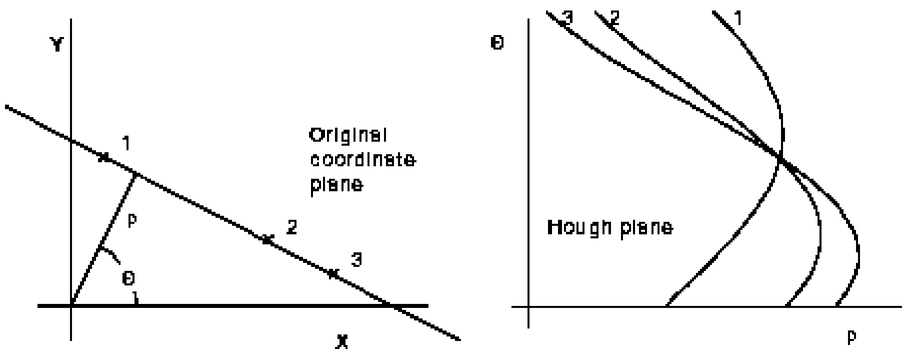

Figure 4.3: Hough Transform

We see that the intersection of the curves correspond to the (ρ, θ) that parametrize the colinearity of three points. Thus, if we compute a histogram of all points in (ρ, θ) domain, and identify the dominant *bins,* lines in the (x, y) domain can be identified.

A particular version of this algorithm is the Probabilistic Hough transform that it does not take all the image points to get the lines, instead it randomly picks some of them. That is proven not to perform much worse than the deterministic Hough transform when the image has only a few long lines, as the case for human shadows. In [8], a comparison of deterministic and probabilistic Hough transform has been presented.

The algorithm to obtain human shadows is described below

- First, we align all our frames in order to have shadow lines aligned.
- Second, an initial detection of potential shadow regions is done, which is aimed at getting rid of possible parts of the object that may have strong line edges, but not from shadows. That is done by clustering the colors on the image along with their position and then thresholding the darkest ones.
- Once we thresholded the original image edge detection is performed.
- We obtain the dominant lines formed by edges using a Probabilistic Hough transform.
- The mean slope of all the extracted lines is calculated using all frames.
- From all the lines, the one with the closest slope to the mean is chosen as the line shadow from our sequence. This is reasonable, as the person is walking, his/her shadow is also moving along the person's feet.
- Finally, with the "closest to the mean" shadow we segment the original image and obtain the shadow free image.

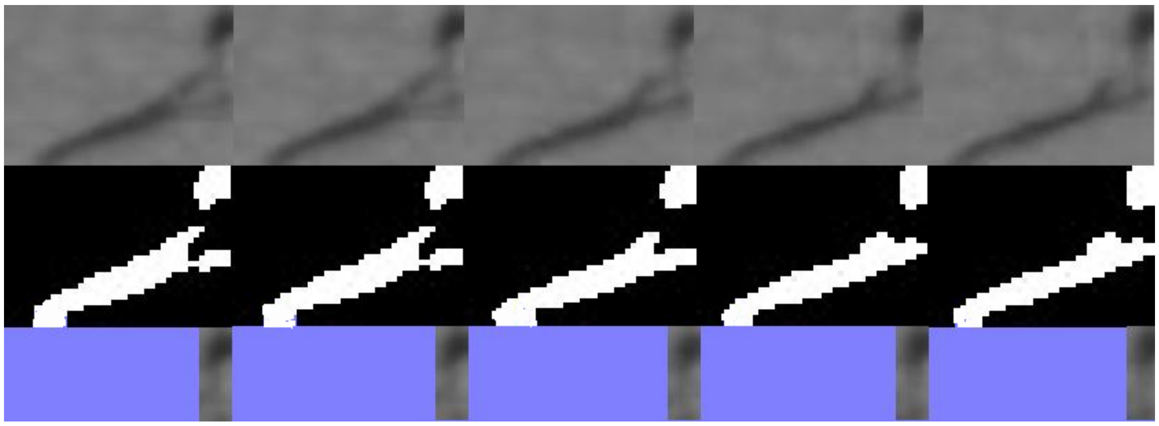

Figure 4.4: Algorithm steps

In figure 4.4, the results of the main steps of the algorithm are presented. In order to know to cut the image from its shadow, it is necessary to know the sun position. In this work, we just assumed the sun direction.

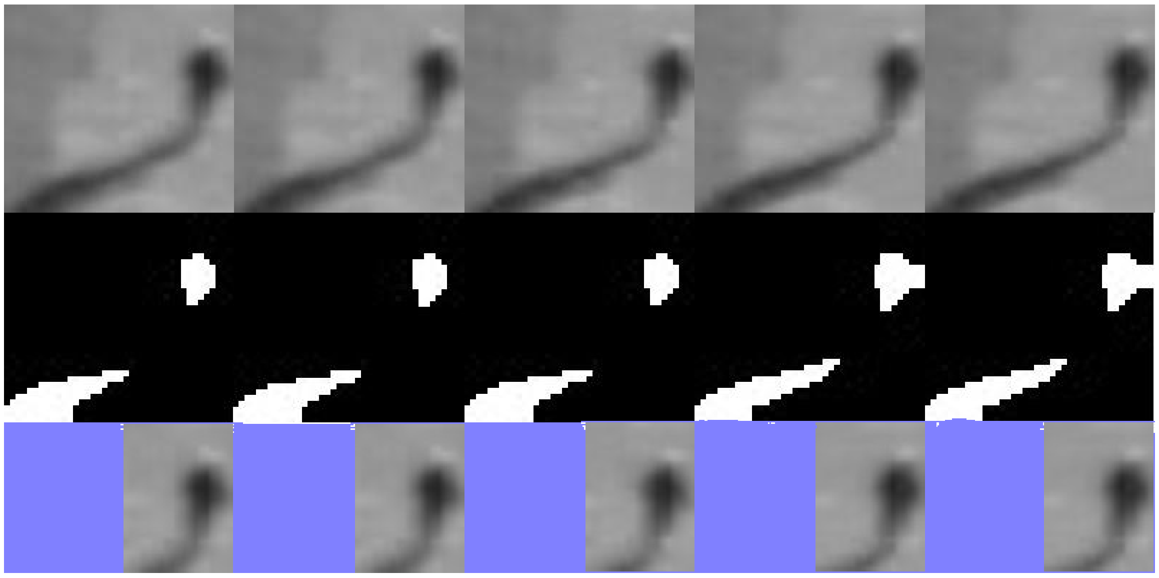

Figure 4.5: Shadow Removal 2

In figure 4.5, we see that we are not eliminating the whole shadow but at least a big part of it has been properly detected and removed. It is also clear from figure 4.6 the ineffectiveness of this method when dealing with vehicle shadows; the re-cropped segmentation image does not have the shadow region but most of the car is also removed.

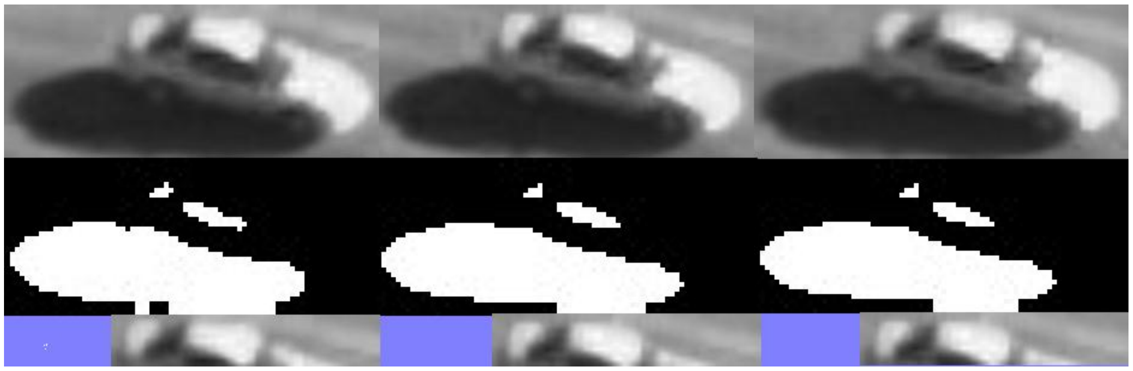

Figure 4.6: Erroneous Car Shadow Removal

4.2.1 *Evaluation*

The shadow detection/removal algorithm can be implemented in a frame by frame basis More robust shadow removal can be accomplished by aggregating a couple of frames. As we do not have any model of the background, the algorithm is much faster than regular shadow detection algorithms that rely on background subtraction. Also the probabilistic Hough transform computationally fast; most of time spent is on the initial alignment.

Obviously, when the presence of shadow is not that strong, the shape line becomes diffuse and has shape. We should be able to estimate the size of the shadow, if the sun position is known. Thus, position of the sun is a key to make the algorithm robust.

We conclude that the proposed method is very simple, fast and way to obtain shadows.

4.3 Algorithm for Classification

We also use Hough line classify objects as to whether they are humans or not. We do this by means of the presence of a big line forming its shadow. One idea tried first was to see how much variance the slopes of the Hough lines detections presented through a set of frames. Humans are expected to have low variance on that shadow lines, while cars and trucks have higher value. This heuristic works reasonably fine but still can be improved. The method do not take into account the fact that all the shadow lines are aligned in one same direction established by the sun. We also need to take care of the fact it is possible that if the vehicle may have a strong shadow edge aligned with the direction of the sun when the car or truck is towards the sun and leaves a reasonable shadow on its back. That is why we introduce another feature which was the ratio between the perimeter of the contour of the object and the area of it. This measure is not invariant to scale but help us to discriminate between narrow shapes of human body and the shadow they cast and wider and bigger shapes of cars and trucks.

The outline of classification algorithm is given below,

- We perform edge detection as before over a dark thresholded region of the detected object.

- Two feature variables are extracted from the edges. The first variable is the slope of the main line; the second is the ratio of the perimeter of the outer contour and the area of that contour.

- With these two features, we cluster all the objects detected with an initial heuristic about the presence or not of humans. This heuristic is based on the second feature, the ratio, and we need to have an estimate of the scale of the image in order to establish a proper threshold. If we cluster without having humans, the result will be unpredictable.

- We establish the shadow cluster as the one with the smaller ratio center.

4.3.1 *Example 1*

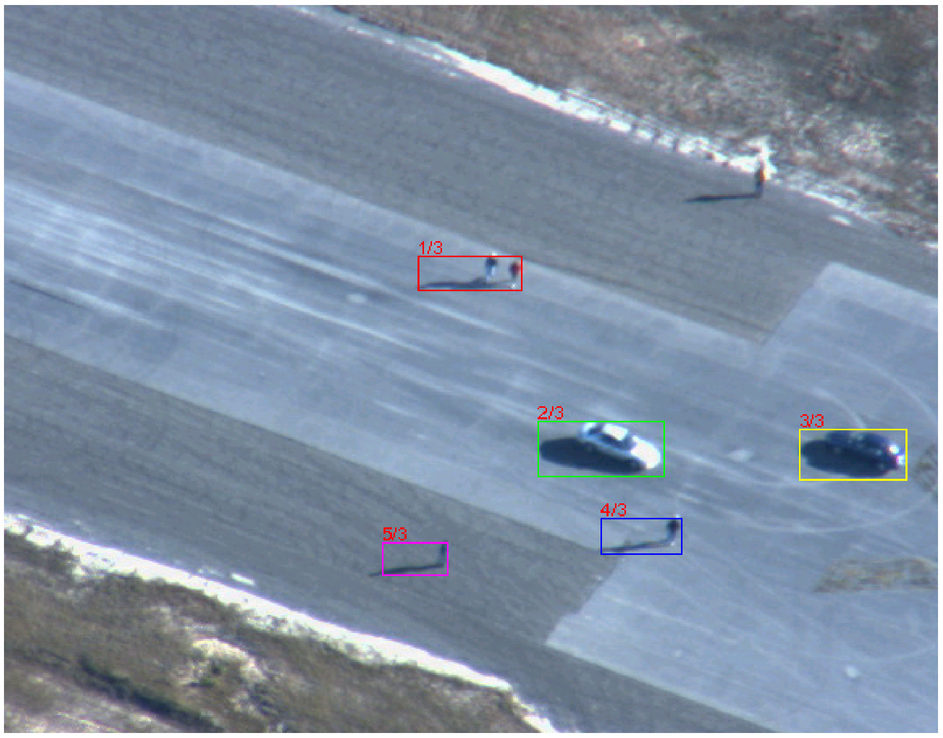

Figure 4.7: Input Image with Detections

In figure 4.7, we see a fairly good detection where the two numbers separated by a slash indicates the object number and the number of frames since its first detection. In object labeled as 1, we detect two humans in one object alone.

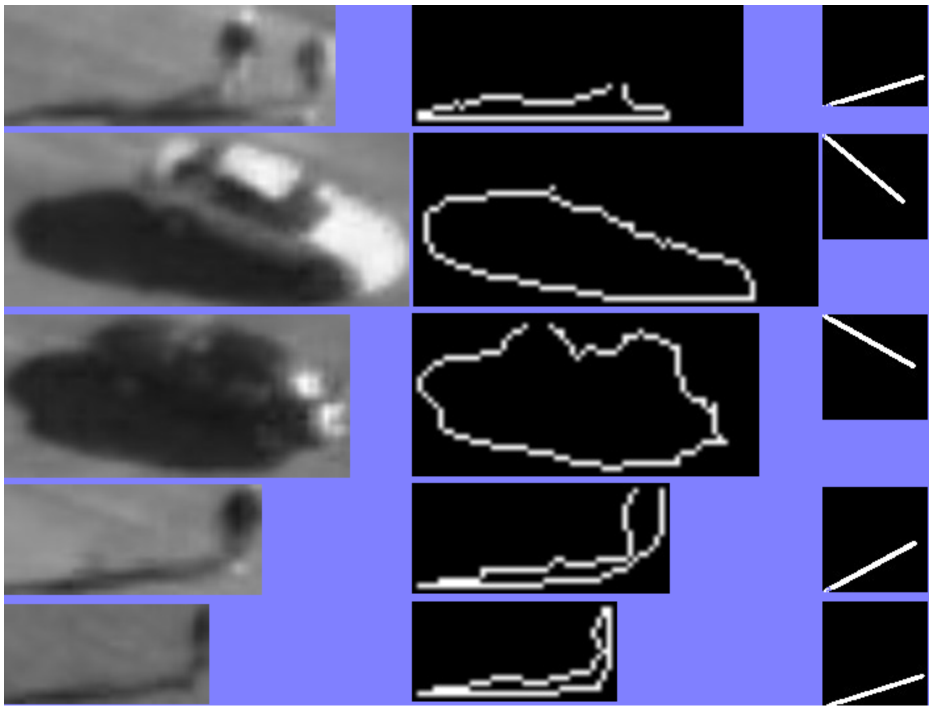

Figure 4.8: Algorithm steps

In the first column of figure 4.8, we see the object detected following the order shown in figure 4.7. In the second column, the outer contour where the ratio is measured, and the last column shows the slope of the shadow. That slope might seem not to coincide with the real one but it is because for representation purposes the ratio is changed to a square. For recognition we keep the original slope. In this case, the algorithm performs perfectly because all the criteria we established to classify are meet, i.e., narrow human contours vs wider vehicles and shadow slopes aligned with a particular direction. In the next figure we see how the clusterization worked.

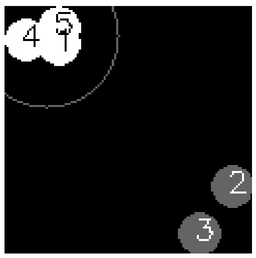

Figure 4.9: Clusters

In figure 4.9, the x-axis represents the slope variable and the y-axis is the ratio between the area and the perimeter of the outer contour. Both variables have been normalized to 1, so the distances between clusters are also normalized. The circle is the decision criteria as a function of the center of the supposed human cluster.

It is interesting to notice that in object one even though we have two people detected, still the shadow they produce is narrow enough and aligned that we can say it comes from a human.

4.3.2 *Example 2*

In figure 4.11, we see how the algorithm performs. We still get a perfect detection, again due to all the criteria being met.

Finally in figure 4.12, the final clusterization and detection is done. It is interesting to see that the performance for this algorithm was almost perfect except in a couple of frames, getting the best results for this video.

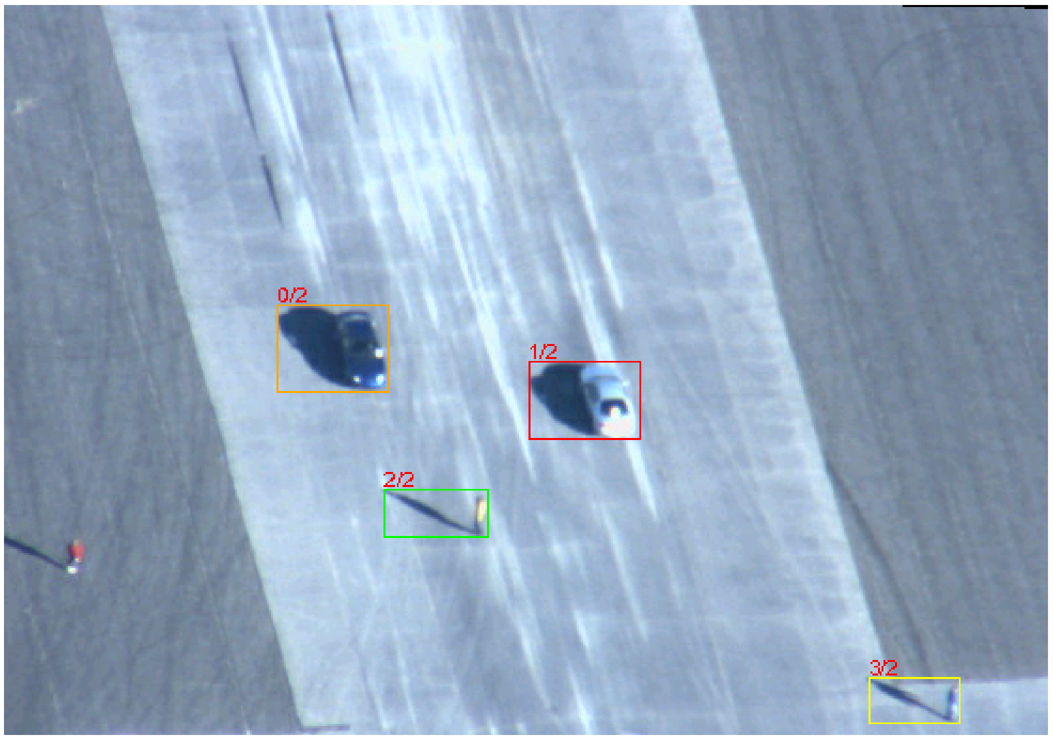

Figure 4.10: Input Detection for Example 2

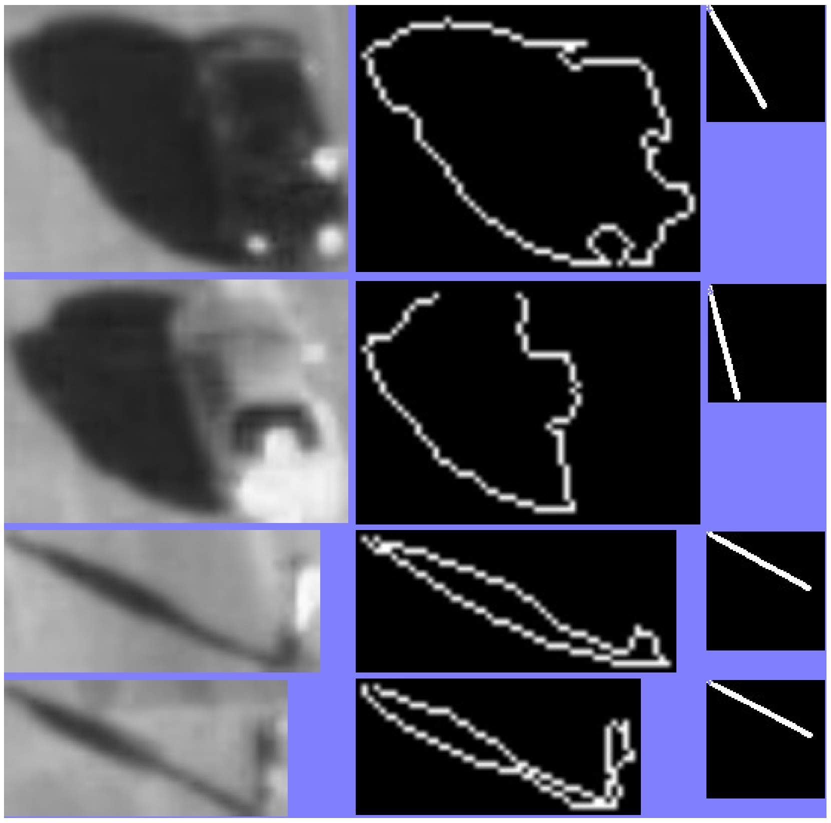
Figure 4.11: Algorithm Performance for Example 2

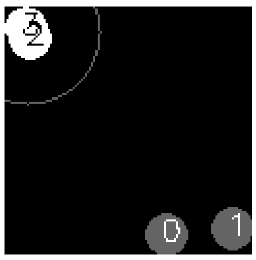

Figure 4.12: Example 2 Final Clusters

4.3.3 *Evaluation*

We have seen a couple of cases where the algorithm performed perfectly, and it does so on many other similar videos, but it is obvious that it relies too much on the characteristics of the video. It is required that a big clean shadow be always present.

Getting a robust algorithm to work in many situation was not what was looked for in this chapter. A different method was presented to show how it is possible to come with very specific solutions and yet very effective for some cases.

The performance we got was by far the best among all the methods we tried with this videos in order to come up with a good human vs vehicle classification classification.

4.4 Conclusions

In this chapter a novel and yet very simple strategy has been implemented to solve the detection problem.

The goal of removing the shadows was to get a good detection of humans in order to extract their periodic motion and use it for classification. When using this shadow removal technique, we already are getting enough information about whether if the object was a human or not. So the next step of extracting periodic information is unnecessary. Almost perfect classification was achieved.

In conclusion,we could make the performance more robust by adding a good estimation of the sun position. We have been working on that. With the metadata the videos contain we can get the geo position, camera parameters, time, etc... With all that we can estimate the position of the sun and then project it to the ground to obtain the orientation of the shadows.

Chapter 5

Conclusions

It is hard to make significant general conclusions as the methods have very little in common and the results are of diverse nature. A way we could evaluate them more rigorously would be having ground truth of the shadow, but as we do not have that we will rely with our visual intuition. All of them showed some strength and some weaknesses and depending on the scenario one method seemed to be better option than another. For indoor scenes the non-parametric method proved to be really good but it failed a little more in outdoor videos; still if the size of the detections was considerably big, the non-parametric method performed effectively. When we do not know what we are trying to classify, but at least the statistical model (usually Gaussian) we can apply the EM algorithm, using the parametric method, to obtain the parameters that describe the statistics of the different classes we want to obtain. The parametric method can use ideas from physics of light formation to improve parameter estimation. Thus the parametric method appears to work better. Either bringing more knowledge to the estimation process or using that knowledge to make the non-parametric method, the importance of understanding how light behaves and how that is reflected in our camera capturing process is one way to succeed in the shadow detection problem. In chapter 2 we presented a model for the physics of light, but it is deterministic. It does not matter how much knowledge we may have, there will be always randomness, and in future work, we will incorporate as much knowledge as possible in a statistical framework.

None of the statistical methods showed the accuracy as the homomorphic filtering for illumination invariance. Speed and robustness are the strong points of homomorphic filtering method. Homomorphic filtering was a simple way of incorporating of our understanding of the physics of light when trying to implement the detection algorithm. We still need to work a little more on that subject to get better illumination invariance images. The main problem is that at the end what we are doing is nothing else but thresholding dark regions on the image after some filtering. This is very inaccurate and can lead to unexpected errors; like dark objects. Actually, we barely experienced errors of such nature in our experiments which made us believe in the reliability of the method. Detections were pretty satisfactory as can be seen in the respective examples.

BIBLIOGRAPHY

[1] Nir Friedman and Stuart Russell, *Image Segmentation in Video Sequences: A Probabilistic Approach*, Uncertainty in Artificial Intelligence, 1997. Proceedings.

[2] Ivana Mikic, Pamela C. Cosman, Greg T. Kogut, Mohan M. Trivedi, *Moving Shadow and Object Detection in Traffic Scenes*, ICPR, 2000. Proceedings.

[3] Thanarat Horprasert, David Harwood and Larry S. Davis, *A Statistical Approach for Real-time Robust Background Subtraction and Shadow Detection* CVL, University of Maryland.

[4] Elena Salvador, Andrea Cavallaro and Touradj Ebrahimi, *Cast shadow segmentation using invariant color features* (Computer Vision and Image Understanding 95, 2004), 238-259.

[5] Mark S. Drew, Graham D. Finlayson and Steven D. Hordley, *Recovery of Chromacity Image Free from Shadows via Illumination Invariance* (ICCV, Nice, France, 2003).

[6] Hao Jing and Mark S. Drew, *Shadow-Resistant Tracking in Video*(ICME, 2003).

[7] Daniel Toth, Til Aach and Volker Metzler, *Illumination-Invariant Change Detection*

[8] J.R. Bergen and H. Shvaytser, *A Probabilistic Algorithm for Computing Hough Transforms* J. of Algorithms, vol. 12., no. 4, pp. 639-656, 1991.