REAL-TIME SHADOW DETECTION USING MULTIPLE VIEWPOINTS

A Thesis

Presented to

the Graduate School of

Clemson University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Electrical Engineering

---

by

Mathew Joseph

May 2004

Advisor: Dr. Adam Hoover

April 30, 2004

To the Graduate School:

This thesis entitled “Real-time Shadow Detection using Multiple Viewpoints” and written by Mathew Joseph is presented to the Graduate School of Clemson University. I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

Adam Hoover, Advisor

We have reviewed this thesis
and recommend its acceptance:

Ian D.Walker

Robert J. Schalkoff

Accepted for the Graduate School:

ABSTRACT

This thesis describes methods for shadow detection based on image data obtained from multiple viewpoints.Our methods are based upon image differencing, the subtraction of a live camera image from an image of the static background. We are motivated by the observation that the differences between the live and background intensity for all cameras or views seeing shadow tend to be similar. We use a two-step approach for our shadow detection method. The first step clusters differences from different views using one of four clustering methods. The second step classifies the world space point under observation as background, shadow or object using one of three rule-based decision making methods on the clusters obtained from the first step. We test different combinations of clustering and decision making methods in real-time using different scenarios.

ACKNOWLEDGMENTS

I am indebted to my advisor Dr. Adam Hoover for introducing me to the field of machine vision and his invaluable guidance, understanding and help during the last two and a half years. I also thank him for supporting me as a research assistant during this period.

I am grateful to Dr. Ian Walker for spending a lot of time and effort in discussing various technical doubts and questions.

I wish to thank Dr. Ian Walker and Dr. Robert Schalkoff for agreeing to serve on my Master's examination committee.

I would also like to thank my parents and Alex and Ellen for their love and support. Finally, I would like to thank my friends here, especially the soccer gang, for making the Clemson experience memorable.

TABLE OF CONTENTS

Page

TITLE PAGE i

ABSTRACT ii

LIST OF TABLES v

LIST OF FIGURES vi

1 Introduction 1
- 1.1 Related Work 3
- 1.2 Overview of Our Approach 8
- 1.3 Outline of the Thesis 13

2 Implementation of a Single-View Method and its Drawbacks 14
- 2.1 Method 14
- 2.1.1 Computational Color Model 14
- 2.1.2 Color Image Characteristics 16
- 2.1.3 Background Subtraction 16
- 2.1.4 Background Modelling 17
- 2.1.5 Pixel Classification 18
- 2.2 Experimental Results 20

3 Methods 28
- 3.1 Occupancy Map 28
- 3.1.1 Occupancy Map Initialization 29
- 3.2 Input Generation 30
- 3.3 Shadow Detection Algorithm 33
- 3.3.1 Clustering methods 34
- 3.3.2 Decision-making methods 45

4 Results and Observations 51
- 4.1 Occupancy Map results for different approaches 51

5 Conclusion 62

BIBLIOGRAPHY 65

LIST OF TABLES

Table Page

3.1 Observed intensities for point under observation (X) 33
3.2 Observed intensities for point under observation (X) 36
3.3 Example cluster recognition table 49

4.1 Performance Evaluation of various approaches for the detection algorithm . 61

LIST OF FIGURES

Figure Page

1.1 Example of incorrect detection of shadow as object 2
1.2 Examples showing shadow properties 4
1.3 Action performed for real-time shadow analysis 9
1.4 Real-time intensity variations due to shadow 10

2.1 Color model for approach in [8] 15
2.2 Detection results for varying camera gain settings 21
2.3 Detection results with variation in shininess of background 23
2.4 Detection results with variation in brightness of background 24
2.5 Detection results with varying depths of shadow 26
2.6 Effect of using an upper brightness threshold on detection results 27

3.1 An example occupancy map 29
3.2 View of a scene from six cameras to explain feature vector generation . . . 31
3.3 Example view from six cameras for clustering explanation 37

4.1 Test-bed scenario for Example 1 53
4.2 Occupancy Map results for Example 1 when clustering using static thresholds 54
4.3 Occupancy Map results for Example 1 when clustering using estimated cluster width 55
4.4 Occupancy Map results for Example 1 when clustering into fixed number of clusters 56
4.5 Occupancy Map results for Example 1 when clustering based on spatial adjacency 56
4.6 Test-bed scenario for Example 2 57
4.7 Occupancy Map results for Example 2 when clustering using static thresholds 58
4.8 Occupancy Map results for Example 2 when clustering using estimated cluster width 59
4.9 Occupancy Map results for Example 2 when clustering into fixed number of clusters 60
4.10 Occupancy Map results for Example 2 when clustering based on spatial adjacency 60

Chapter 1

Introduction

Using a camera to detect objects in a scene is a common problem in computer vision. For example, an automated surveillance system needs to detect people and vehicles moving through an area. An automated alarm system needs to detect the introduction of unauthorized objects, or the removal of guarded objects. An automated manufacturing inspection system needs to detect parts moving down an assembly line, and to detect part defects. Similar applications can be found in medical imaging and entertainment.

A common method applied to this problem is background differencing. Background differencing involves subtracting a static image of the background from each raw or live image captured by the camera. A threshold is applied to the subtraction, such that differences smaller than the threshold are considered unchanged, while differences larger than the threshold are considered changed. Areas of the scene that are unchanged are generally ignored, while areas of the scene that are changed are considered detected objects.

The largest cause for failure (incorrect detection) in background differencing is shadows. A deep shadow can cause part of a scene to appear significantly darker than its original intensity. The simple thresholding scheme would erroneously classify this area as changed, and therefore a detected object. An example of this type of failure is shown in Figure 1.1. In this figure, (a) is the background image, (b) is the live image and (c) is the thresholded

(a) Background Image (b) Live Image (c) Difference Image

Figure 1.1: Example of incorrect detection of shadow as object.

difference image. As observed, a large part of the shadow cast by the hand is falsely detected as an object.

Similarly, an object can closely match part of a scene in color, so that the difference in intensity is minimal. The simple thresholding scheme would erroneously classify this area as unchanged, therefore missing the object. Shadows in a scene can also be the cause of object merging and object distortion due to confusion between the outer boundary of an object and its shadow [5, 16].

In this thesis, we explore the potential for using several simultaneous observations of a scene to detect shadows during background differencing. Our testbed uses a network of six cameras, with overlapping fields of view that all observe the same space. The cameras are calibrated so that it is known which pixels in each camera see the same world point. This testbed provides a set of multiview background, live, and difference measurements for each world point. Our goal is to produce a classification of each world point as being background (unchanged), object (changed), or in shadow (shaded background). To our knowledge, this is the first time a multiview approach has ever been applied to the problem of automated shadow detection.

1.1 Related Work

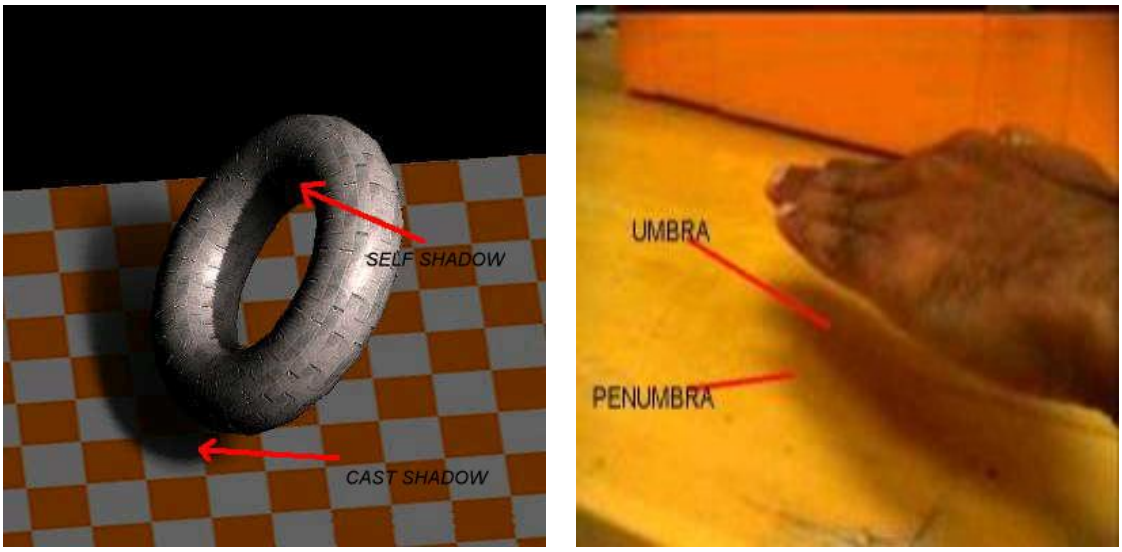
Research concerning shadows can be found in areas other than computer vision. In art, the crafting of shadows for realistic scene portrayal has been studied for hundreds of years. In painting, the term *Chiaroscuro,* derived from the Italian chiaro (light) and oscuro (dark), refers to a technique that contrasts bright illumination with areas of dense shadow. The skillful use of this technique is a particular feature in the works of such 16th-century Renaissance artists as Leonardo da Vinci and Raphael and such 17th-century baroque masters as Caravaggio and Rembrandt. The related field of photography is concerned with minimizing shadows through the creative placement of the camera and objects in the scene relative to the light sources. In photogrammetry, shadows have been used by human experts to measure things, for example the time of day or the height of a building. More recently, the field of computer graphics and animation has quantified many ideas in shadow portrayal, again for realistic scene rendering.

Through all these fields, as well as computer vision (perhaps the newest field to be concerned with shadows), several ideas have become generally understood.

- A shadow is created when a light source is occluded by an object. The shape of the shadow is a projection of a silhouette of the occluding object.

- The intensity of a shadow tends to be predictably darker than the surface on which it is cast.

- Shadows can be classified as either *cast shadows* or *self shadows* [11]. A self shadow is a projection of a portion of an object onto itself. For example, a nose can cast a shadow on a face. A cast shadow is a projection of the silhouette of an object onto another object or background. For example, a building can cast a shadow onto the ground. Figure 1.2(a) shows an example of each type.



(a) Cast and Self Shadows (b) Umbra and Penumbra regions

Figure 1.2: Examples showing some shadow properties.

- Points in a shadow fall into two categories: the *umbra*, which is the area in the shadow that is totally blocked from the light source by the object, and the *penumbra*, which is the area in the shadow that is only partially occluded from the light source. Figure 1.2(b) shows an example of each type.

- For objects in motion, the shadows tend to follow similar motion patterns, so that motion is not a good discriminator.

Over the past decade, low-cost computing systems have achieved enough power to be able to process standard video data at real-time frame rates (e.g. 10-30 Hz). With this new power, research into automated shadow detection has increased. Research with regards to shadows in images can be subdivided into two categories:

1. Detecting shadow regions in an image, and

2. Using shadow information in an image to understand or interprete the scene.

Various methods have been tried for shadow detection. Most approaches use background subtraction as their basic input [2, 8, 11, 13, 18, 21, 22]. The approaches differ in

how they use these differences to perform shadow detection. A key component in the background differencing method is the selection of the threshold. Rosin and Ellis [18] propose several methods to select an appropriate threshold. They present a hysteresis thresholding technique based on the Canny edge detector which gives reasonably good results. Another approach to thresholding was suggested by Gorman [15] and is based on image connectivity.

Color information in images has been used by researchers in the detection of shadows. In general, a shadow causes only a change in brightness with little or no change in chromaticity (color). A number of researchers have based their approaches on this premise [2, 3, 8, 13]. Horprasert et al [8] proposed a computational color model which separates the brightness from the chromaticity component. Pixels in the live image with similar chromaticity but lower or higher brightness levels are identified as belonging to a shadow or highlighted background region respectively. Pixels with a significant difference in chromaticity from the background are identified as those of an object. Mckenna et al [13] used color information together with gradient information to minimize incorrect detection of shadows as foreground objects while tracking groups of people. Here, each background pixel is modeled using gradient means and variances by applying Sobel masks in the *x* and *y* directions. For a live pixel, its spatial gradients for R, G and B are estimated using the Sobel operator. Thresholds are then set on these individual gradients to track foreground objects. Results obtained in [13] were better than in [8] because Mckenna used image gradient information together with chromaticity information.

Approaches such as presented in [2, 3, 22] model each pixel intensity as a mixture of Gaussians. A pixel is assumed to have one mean intensity and standard deviation when not shaded (background) and a second mean intensity and deviation when it is an object. Pixels are classified according to which of these distributions they best fit. In [2], shadows are then identified and seperated from objects by separating lightness information from chromaticity information. Friedman and Russell [3] learn a mixture-of-Gaussians classification

model for each pixel using an EM framework for the purpose of detection and tracking of road vehicles. Pixel values are classified into three separate distributions corresponding to vehicle color, road color and shadow color. A probabilistic classification of the current pixel value is used to update the models so that vehicle pixels do not become mixed in with the background model for slow moving traffic. Stauffer and Grimson's approach [22] is very similar but does not specifically account for shadows. In their approach, the value of each pixel is modelled as a mixture of gaussians. Pixels are identified as foreground pixels only if their value does not fit the Gaussian distribution for that pixel. The background pixel intensity is dynamically updated and modeled in this approach. Hence, a shadow present for a significant amount of time is likely to become background as it models the Gaussian representing those pixel values.

Another approach has been to utilize shadow geometry to detect shadows [4, 11]. Jiang and Ward [11] used a three-level analysis of shadow intensity and shadow geometry in an environment with simple objects and a single light source to determine shadow regions. The first level, termed the low level, extracts dark regions from images which include both shadows and regions with low reflectance. The next level, called the middle level, performs a feature analysis on the dark region to identify the penumbra and cast and self-shadow regions in this dark region. Object regions adjacent to the dark regions are also identified at this level. The final level or the high level, uses the information from the previous levels to make a final decision on shadow regions in the image. Funka-Lea and Bajcsy [4] used shadow geometry together with a color segmentation method to determine shadow regions and recover the penumbra and umbra regions of the shadow. They based their classification on the fact that the histogram in a color space of an image of a surface directly lit and in shadow is defined by the parametric form of a line. Thus, the color segmentation method segments an image into line-like or uniform color clusters where the line-like clusters are most likely shadow regions.

The Entry-Exit method is another method for detecting the shadows of objects by segmenting and labelling the shadow boundary using information about the projection onto the image plane of the light source direction [1, 6]. This method is based on the fact that for a light ray originating at a single distant light source, the ray either enters or exits a shadow at its boundary. Shadow boundaries are classified into four types of segments: shadow lines, shadow-making lines, occluding lines and hidden shadow lines. Pairs of entry and exit segments with end points aligned along the light ray form a shadow-making and its corresponding shadow line. Occluding lines are outlines of the shadow casting object as seen by the viewer. Shadow lines created by a shadow making line not visible to the viewer are called hidden shadow lines.

Shadows in an image have been used extensively to gather other information about the scene, such as the shape of the objects causing the shadow and the location of light sources in the environment. For the detection of object shape and orientation in scenes from shadows, it is essential to solve the correspondence problem - that is, to determine the object causing the shadow. This subject has been dealt with in [9, 14].

In image interpretation, Shafer [19] has done extensive research on how shadows can be analyzed to determine 3D surface orientations. He showed how constraints can be applied to a special kind of curved surface, generalized cylinders, derivable from shadows in an image using orthographic projection, to give an accurate description of an object's shape. Kriegman [12] suggested that for an object viewed from a fixed viewpoint and for a finite set of light sources, there is an equivalence class of object shapes having the same set of shadows. This can be used to infer object shapes in images.

Shadows in aerial images have been used as cues to detect clouds and buildings and determine depth [9, 10, 20, 23]. These works base their classification on the assumption that dark regions in images with perceived objects adjacent to them are most probably shadow regions. Irvin and McKeown [10] identified object regions in images making assumptions on object shapes. Dark regions adjacent to the object regions are identified and their bound-

aries are approximated by straight lines. Based on if the lines form a 90 degree corner and the corner is concave towards the sun, the dark region is classified as a shadow region. Huertas and Nevatia [9] followed a similar approach. They explored the correspondence problem in aerial images and used this for detecting buildings and estimating depth in these images. Assuming all objects to be composed of rectangular components, they identified corners in images and labelled them as object or shadow corners. Now, if an object corner matched a shadow corner in the direction of sunlight, lines forming the shadow corner were labelled shadow and lines forming the object corner were labelled as object.

Features utilized for shadow detection have been extracted from three domains: spectral, spatial and temporal [16]. This implies that all shadow algorithms use either greyscale or color information, work at the region or pixel level and may use temporal redundancy to integrate and improve results. A comprehensive evaluation and comparison of various shadow detection methods is presented in [16] and [17].

All the methods described above have used only one viewpoint for their detection approaches. A domain for feature extraction which has not been mentioned in [16, 17] but has been used extensively for rendering purposes in computer graphics and animation is the multiple view domain. The approach described in this thesis attempts to relate the appearance of a point in one view to that from a different view and use this to facilitate detection. All the methods described in this thesis also use only greyscale information, are static in nature and work at the pixel level. However, our methods could be extended into other feature domains.

1.2 Overview of Our Approach

We motivate our approach to multiview shadow detection using the following experiment. Figure 1.3 shows the setup of the experiment. Six greyscale cameras are observing an empty area (the floor is the only thing of interest in the scene). A background image

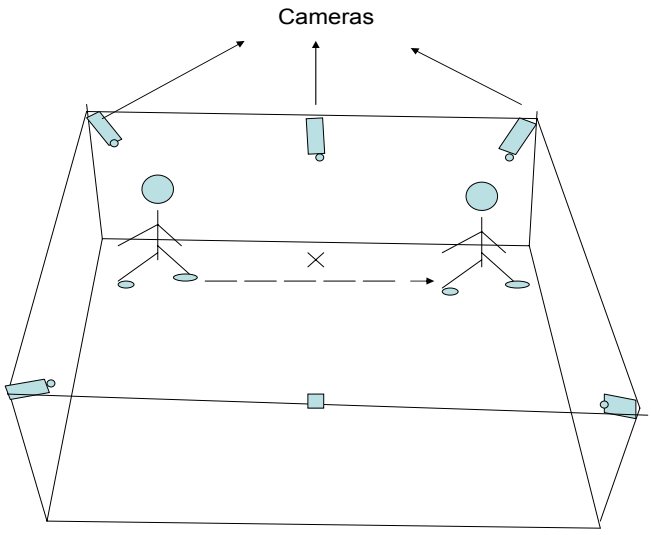


Figure 1.3: Action performed for real-time shadow analysis.

is acquired for each camera while the room is empty. A specific point on the floor is selected (marked X in Figure 1.3). A pixel in each camera's field-of-view is determined that corresponds to an observation of the floor point (e.g., these pixels "see" the selected point on the floor). The background intensity for these six pixels (one per camera) is recorded.

A person walks across the room, nears the point without ever actually stepping on it, and continues proceeding away from the point. This motion causes the selected floor point to come under shadow, gradually increasing until the person is nearest to the point and gradually decreasing as the person walks away. The raw (live camera) intensities are recorded for each of the six pixels previously selected.

Figure 1.4 shows a plot of the time (frame number) against the difference in intensity between the live and background image intensity, recorded for each frame. The y-axis shows the intensity differences for image pixels corresponding to the floor point under observation for all 6 cameras over time (x-axis).

At the outset of the experiment, and at the end, the person is far enough away from the scene point so that it is completely out of shadow. During some portions of the experi-

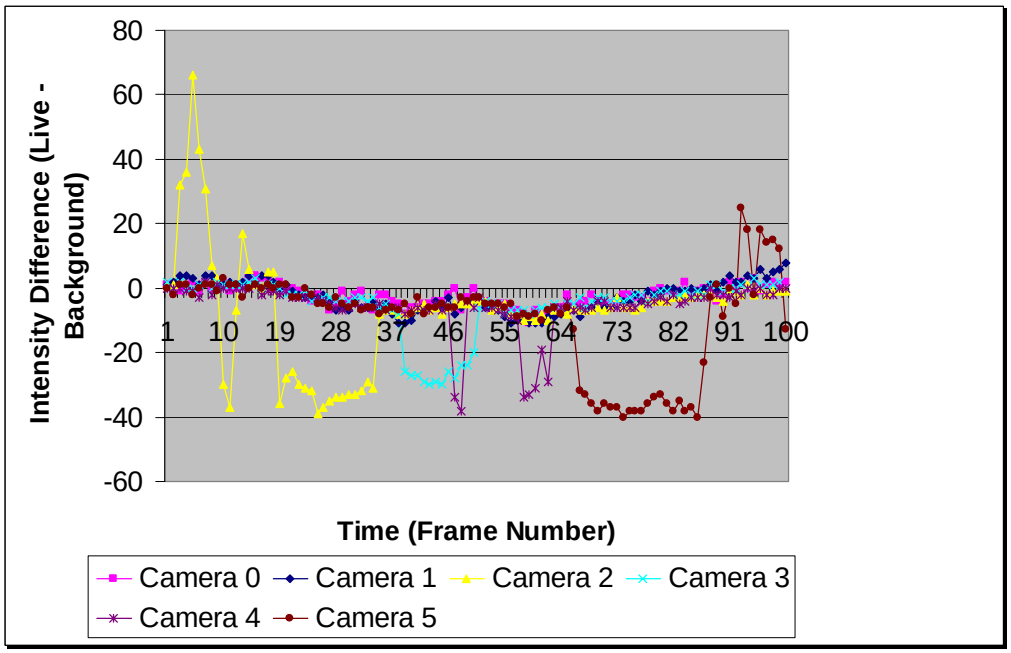


Figure 1.4: Intensity differences observed at point *X* in the 6 cameras corresponding to action performed as shown in Figure 1.1.

ment, various cameras are blocked from seeing the floor point by the person moving across the room. For example, during frames 1-17, camera 2's view is blocked, and so camera 2's differences are noticeably stronger. However, the main point of concern for us is the trending visible throughout this experiment. The shadowing of the point can be observed in Figure 1.4 as the downward curving of most of the data between frames 20-50. As the person approaches the point, the shadow deepens, so the differences increase. Similarly, there is an upward curving of most of the data between frames 50-75. As the person moves away from the point, the shadow lightens, so the differences decrease. *Notice that during these trends, the differences tend to cluster.*

Regardless of whether the shadow is light (as in frame 30), or deep (as in frame 60), the differences seen by all cameras (all those that see the shadow and not the occluding object) tend to be similar. This finding motivates us to use clustering methods, on multiview sets of differences, in order to detect shadows.

The shadow detection algorithm presented in this thesis attempts to use the above motivation towards achieving the following aims:

- To correctly detect and classify all floor points in the environment as either empty, occupied or shadow.
- To do it at a frame rate suitable for real-time applications.

To make real-time detection possible, we stick to simple approaches and avoid more advanced temporal and spatial techniques such as using data from multiple frames or region-growing. In short, we are exploring the simplest possible use of multiview data.

Our methods utilize the differences in pixel image intensity between the current or live frame and the background frame for each camera. Our approach consists of two steps:

1. Clustering: Grouping of individual differences into clusters
2. Rule-based decision-making: On the basis of the clusters, making a decision using a set of pre-defined rules as to whether the world point under observation is:

(a) Empty or Background (E), or

(b) Shadow or Shaded Background (S), or

(c) Occupied or Object (O).

Several variations on both the clustering and the decision making have been explored for this thesis. For clustering, four different methods have been explored. The first method classifies on the basis of simple thresholding. Here, static thresholds are applied to intensity differences for all pixels corresponding to the point under observation in all cameras. The second method attempts to group differences from all cameras seeing the same ground truth into individual clusters. It involves making an assumption on the maximum cluster width and grouping the differences into clusters whose range is less than this width. The third method is a slight variation to the previous method, in that it clamps the maximum possible number of clusters to a fixed value. The final clustering method clusters differences based on the circular or spatial adjacency of the cameras from which the differences are obtained.

For decision making, three different methods have been explored. The first classifies based on fixed priority - that is, a point is classified as belonging to an object only if all views see the object else it is classified as shadow or background. The second method classifies based on the cardinality of each cluster and the average image intensity of that cluster. The third method first labels each cluster based on its average intensity. Then, depending on the cardinality of each cluster, the corresponding cell is placed in a category. The final decision on the corresponding occupancy map cell is based by looking up the most likely scenario corresponding to that category in a cluster recognition table.

The methods described in this thesis are aimed at exploring the simplest possible use of multiview data. This is not to preclude the use of more advanced approaches. Future approaches could utilize data from adjacent regions or multiple frames or other advanced methods such as region growing or texture analysis. These ideas are discussed further in the conclusion.

1.3 Outline of the Thesis

In chapter 2, we reimplement a popular single-view shadow detection technique to analyze the problems associated with shadow detection, study the properties of shadows and finally to determine the drawbacks associated with this particular method. Various object or surface properties which might affect the nature of the shadows cast on or by them are studied in this chapter.

In chapter 3, we describe our methods for shadow detection. The concept of the occupancy map, and how it is initialized and processed in real-time are described in this chapter. The inputs for our shadow detection algorithm are identified along with a description of how they are obtained. Then, our method for shadow detection is explained with examples.

In chapter 4, we test our algorithms in a real-time environment. We take a look at the results for different variations of the shadow detection technique developed and compare them.

Finally, in chapter 5, cases in which our detection algorithm failed are analyzed. Other possible methods and data which could be used to overcome these failures and improve the detection rate using our testbed are suggested.

Chapter 2

Implementation of a Single-View Method and its Drawbacks

In order to understand the shadow detection literature, we reimplemented a popular technique. We did this with the following aims in mind:

- Determine problems associated with shadow detection.
- Determine the various parameters which affect the properties of a shadow.
- Determine cases in which this algorithm fails to detect the shadow correctly.

In this chapter, we describe this method and show some results. The following is a redescription of [8].

2.1 Method

2.1.1 Computational Color Model

Based on the fact that humans tend to assign a constant color to an object even under changing illumination over time and space (color constancy), the color model used here separates the brightness from the chromaticity component.

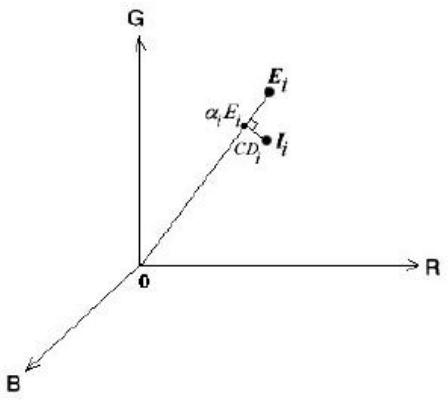


Figure 2.1: Color model for statistical non-parametric approach as described in [8].

Figure 2.1 illustrates the proposed color model in three-dimensional RGB color space. Consider a pixel, i, in the image. Let $E_i = [E_R(i), E_G(i), E_B(i)]$ represent the pixel's expected RGB color in the background image and $I_i = [I_R(i), I_G(i), I_B(i)]$ denote the pixel's RGB color value in a current image that we want to subtract from the background. The line OE_i is called the expected chromaticity line whereas OI_i is the observed color value. Basically, we want to measure the distortion of I_i from E_i. The proposed color model separates the brightness from the chromaticity component. This is done by decomposing the distortion measurement into two components, brightness distortion and chromaticity distortion, defined below.

- *Brightness Distortion (α):* The brightness distortion (α) is a scalar value that brings the observed color close to the expected chromaticity line and is obtained by minimizing

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

- *Color Distortion* (CD_i): Color distortion is defined as the orthogonal distance between the observed color and the expected chromaticity line. The color distortion of a pixel *i* is given by

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

2.1.2 Color Image Characteristics

The CCD sensors linearly transform an infinite-dimensional spectral color space to a three-dimensional RGB color space via red, green, and blue color filters. There are some characteristics of the output image, influenced by typical CCD cameras, which we should account for in designing the algorithm, as follows:

- *Color variation :* The RGB color value for a given pixel varies over a period of time due to camera noise and illumination fluctuation by light sources.

- *Band unbalancing :* Cameras typically have different sensitivities to different colors. Thus, in order to make the balance weights on the three color bands (R, G, B), the pixel values need to be rescaled or normalized by weight values. Here, the pixel color is normalized by its standard deviation (s_i) which is given by,

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

  where $\sigma_R(i)$, $\sigma_G(i)$, and $\sigma_B(i)$ are the standard deviation of the i^{th} pixel's red, green, blue values computed over N background frames.

- *Clipping :* Since the sensors have limited dynamic range of responsiveness, this restricts the varieties of color into a RGB color cube, which is formed by red, green, and blue primary colors as orthogonal axes. On 24-bit images, the gamut of color distribution resides within the cube range from [0, 0, 0] to [255, 255, 255]. Color outside the cube (negative or greater than 255 color) cannot be represented. As a result, the pixel value is clipped in order to lie entirely inside the cube.

2.1.3 Background Subtraction

Background subtraction basically involves subtraction of live image from a reference image. Typically, the algorithm consists of three basic steps:

1. *Background modeling*, where a reference image representing the background is constructed.

2. *Threshold selection* determines appropriate threshold values used in the subtraction operation to obtain a desired detection rate.

3. *Subtraction operation or pixel classification* determines whether the pixel is a part of the ordinary background, in highlighted background, shadow or if it is a foreground object.

2.1.4 Background Modelling

The background is modeled statistically on a pixel by pixel basis. A pixel is modeled by a 4-tuple $< E_i, s_i, a_i, b_i >$ where E_i is the expected color value, s_i is the standard deviation of color value, a_i is the variation of the brightness distortion, and b_i is the variation in the chromaticity distortion of the i^{th} pixel. The background image and some other associated parameters are calculated over a number of static background frames. The expected color value of pixel *i* is given by

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

where $\mu_R(i)$, $\mu_G(i)$, and $\mu_B(i)$ are the arithmetic means of the i^{th} pixel's red, green, blue values computed over N background frames.

Since the color bands have to be balanced by rescaling color values by pixel variation factors, the formulae for calculating brightness distortion and chromaticity distortion can now be written as:

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

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

Since different pixels yield different distributions of brightness and chromaticity distortions, these variations are embedded in the background model as a_i and b_i respectively which are used as normalization factors.

The variation in the brightness distortion and chromaticity distortion of the i^{th} pixel are given by a_i and b_i, which is defined as:

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

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

In order to use a single threshold for all of the pixels, we need to rescale α_i and CD_i. Therefore, let

$$\widehat{\alpha_i} = (\alpha_i - 1)/a_i$$

$$\widehat{CD_i} = (CD_i - 1)/b_i$$

2.1.5 Pixel Classification

Here, chromaticity and brightness distortion components are calculated based on the difference between the background image and the live image. Suitable threshold values are applied to these components to determine the classification of each pixel as follows:

- *Original background (B)*, if it has both brightness and chromaticity similar to those of the same pixel in the background image.

- *Shaded background or shadow (S)*, if it has similar chromaticity but lower brightness than those of the same pixel in the background image. This is based on the notion of the shadow as a semi-transparent region in the image, which retains a representation of the underlying surface pattern, texture or color value.

- *Highlighted background (H)*, if it has similar chromaticity but higher brightness than the background image.

- *Moving foreground object (F)*, if the pixel has chromaticity different from the expected values in the background image.

Based on these definitions, a pixel is classified into one of the four categories B; S; H; F by the following decision procedure.

$$M(i) = \begin{cases} F : \widehat{CD_i} > \tau_{CD}, \textit{ else} \\ B : \widehat{\alpha_i} < \tau_{\alpha 1} \textit{ and } \widehat{\alpha_i} < \tau_{\alpha 2}, \textit{ else} \\ S : \widehat{\alpha_i} < 0, \textit{ else} \\ H \end{cases}$$

where, τ_{CD} is a threshold value used to determine the similarity in chromaticity between the background image and the current observed image and, $\tau_{\alpha 1}$ and $\tau_{\alpha 2}$are selected threshold values used to determine the similarities in brightness.

A problem encountered here is that if a moving object in a current image contains very low RGB values, this dark pixel will always be misclassified as a shadow. This is because the color point of the dark pixel is close to the origin in RGB space and the fact that all chromaticity lines in RGB space meet at the origin, which causes the color point to be considered close or similar to any chromaticity line. To avoid this problem, we introduce a lower bound for the normalized brightness distortion $\tau_{\alpha lo}$. Then, the decision procedure becomes

$$M(i) = \begin{cases} F : \widehat{CD_i} > \tau_{CD} \textit{ or } \widehat{\alpha_i} < \tau_{\alpha lo}, \textit{ else} \\ B : \widehat{\alpha_i} < \tau_{\alpha 1} \textit{ and } \widehat{\alpha_i} < \tau_{\alpha 2}, \textit{ else} \\ S : \widehat{\alpha_i} < 0, \textit{ else} \\ H \end{cases}$$

2.2 Experimental Results

We demonstrate the performance of this algorithm over several indoor images. Results shown here are 640x480 images. In our set-up, the background image and the means and standard deviations associated with each of the color bands are determined utilizing twenty static background images. Values for the various thresholds to be utilized for pixel classification were set on-the-fly so as to obtain best results.

Results obtained as a result of this algorithm can be observed using the following examples. In the images, colors utilized for classification are as follows:

- *Original background* : Black
- *Highlighted background* : Green
- *Shadow* : Blue
- *Foreground object* : White

In the figures that follow,

(a) is the background or reference image that has been initially calculated based on 20 static background frames,

(b) is the live image, and,

(c) is the computed difference image showing classification results.

Also, to observe the effectiveness of the algorithm, images have been collected using different criteria:

1. *Brightness of camera*

   The brightness of the camera was varied to observe the performance of the detection algorithm for different gains in brightness of camera.

   (a) High gain

(a) Background Image (b) Live Image (c) Detection Result

(i) High gain

(a) Background Image (b) Live Image (c) Detection Result

(ii) Medium (normal) gain

(a) Background Image (b) Live Image (c) Detection Result

(iii) Low gain

Figure 2.2: Detection results for varying camera gain settings.

(b) Medium gain

(c) Low gain

Detection results can be seen in Fig 2.2. It is observed that as the camera gain is increased, the shadows tend to be increasingly identified as foreground and conversely, as the gain is decreased, the foreground is misclassified as shadow. This is most probably because as the gain of the camera is decreased, difference in brightness and chromaticity distortion caused by shadows and objects is reduced. Therefore, the algorithm fails if there is too much or too little variation in the brightness of camera.

2. *Shininess of Background*

   The detection algorithm was tried out with shadows cast on background of varying shininess.

   (a) Shiny background

   (b) Average background

   (c) Dull background

   Detection results can be seen in Fig 2.3. This method is robust to changes in the shininess of the background.

3. *Brightness of Background*

   The brightness of the background was varied to observe the performance of the detection algorithm for different levels of brightness of the background.

   (a) Almost white (Very bright background)

   (b) Average (Mid-level brightness of background)

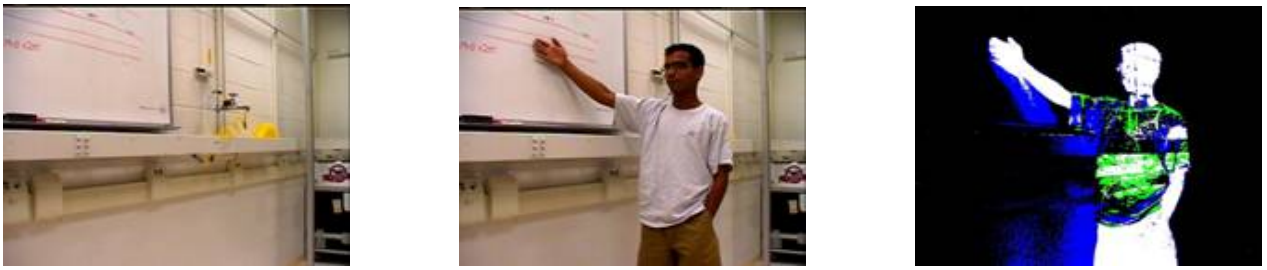
   (c) Almost black (Dark background)

   Detection results can be seen in Fig 2.4. Another case in which the algorithm appears not to work very well is as the background brightness gets lower. As observed in the image with the camera cover as the background, very little shadow region has been identified. Most of the shadow is detected as background.

4. *Depth of Shadow*

   The depth of the shadow was varied to observe the performance of the detection algorithm for different heights from which the shadow is cast resulting in different depths of the cast shadow.

   (a) Deep shadow



(a) Background Image (b) Live Image (c) Detection Result

(i) Shiny (high reflectance) background

(a) Background Image (b) Live Image (c) Detection Result

(ii) Average background

(a) Background Image (b) Live Image (c) Detection Result

(iii) Dull background

Figure 2.3: Detection results for varying reflective backgrounds.

(a) Background Image (b) Live Image (c) Detection Result

(i) Bright background

(a) Background Image (b) Live Image (c) Detection Result

(ii) Average background

(a) Background Image (b) Live Image (c) Detection Result

(iii) Dark background

Figure 2.4: Detection results for varying brightness in background.

(b) Medium shadow

(c) Light shadow

Detection results can be seen in Fig 2.5. It is observed that as the shadow gets deeper, they tend to be increasingly identified as foreground pixels. There are two possible reasons for this:

(a) The brightness distortion is very high in which case the upper brightness threshold is exceeded.

(b) There is a change in the chromaticity of the misclassified pixels. In other words, pixels in deep shadow do change color (somewhat) as well as brightness.

5. *Using Upper Brightness Distortion Threshold*

We introduced another threshold ($\tau_{\alpha hi}$) to reduce misclassifications of some foreground objects as highlighted background. The decision procedure now becomes

$$M(i) = \begin{cases} F : \widehat{CD_i} > \tau_{CD} \ \textit{or} \ \widehat{\alpha_i} < \tau_{\alpha lo} \ \textit{or} \ \widehat{\alpha_i} > \tau_{\alpha hi} \ , \ \textit{else} \\ B : \widehat{\alpha_i} < \tau_{\alpha 1} \ \textit{and} \ \widehat{\alpha_i} < \tau_{\alpha 2}, \ \textit{else} \\ S : \widehat{\alpha_i} < 0, \ \textit{else} \\ H \end{cases}$$

Results with and without this threshold have been observed:

(a) Without upper brightness distortion threshold:

(b) With upper brightness distortion threshold (set at $\tau_{\alpha hi} = 30$):

Detection results can be seen in Fig 2.6. The algorithm checks for misclassification of foreground pixels with very low RGB values as shadow by utilizing a lower bound for brightness distortion. However, certain foreground pixels tend to be misclassified as highlighted background since there is no upper bound for brightness distortion

(a) Background Image (b) Live Image (c) Detection Result
(i) Deep Shadow

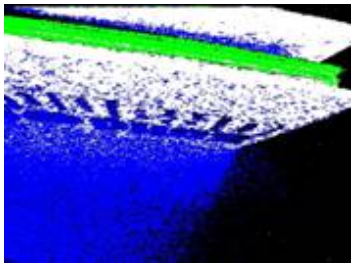
(a) Background Image (b) Live Image (c) Detection Result
(ii) Medium Shadow

(a) Background Image (b) Live Image (c) Detection Result
(iii) Light Shadow

Figure 2.5: Detection results with varying depths of shadow.

(a) Background Image

(b) Live Image

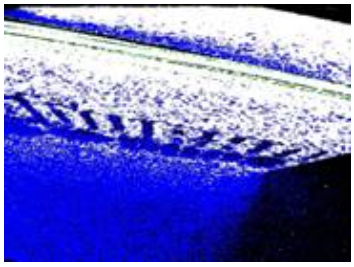


(c) Detection Result

(i) Original

(a) Background Image

(b) Live Image



(c) Detection Result

(ii) Using upper brightness threshold

Figure 2.6: Detection results showing performance improvement with our modification.

as seen in Fig 2.6 (i). Applying an upper bound removes this misclassification as observed in Fig 2.6 (ii).

Chapter 3

Methods

In this chapter, we describe our multiview method to detect shadows. The detection algorithm described here utilizes data from multiple video streams to create a two-dimensional spatial-temporal occupancy map in real-time. Each unit of this occupancy map is called a cell. We start with a description of the occupancy map - how it is initialized, computed and generated in real-time as described in [7]. Afterward, we describe the shadow detection algorithm.

3.1 Occupancy Map

Formally, let us assume the set-up consists of N cameras. Let (x, y, z) represent a right handed three dimensional coordinate system in the world. The occupancy map for the area under observation, in the plane of the floor, is defined by,

$$O[\,x, y\,],\ \ 0 \leq x < X,\ \ 0 \leq y < Y$$

It is assumed that x, y are integers. Now, the term, *cell*, is used to define a rectangle with area $(1/X \times 1/Y)$ in this plane. In this model, $O[\,x, y\,] = 0$ indicates a cell of freespace, $O[\,x, y\,] = 1$ indicates a cell in shadow and $O[\,x, y\,] = 2$ indicates a cell occupied by an object.

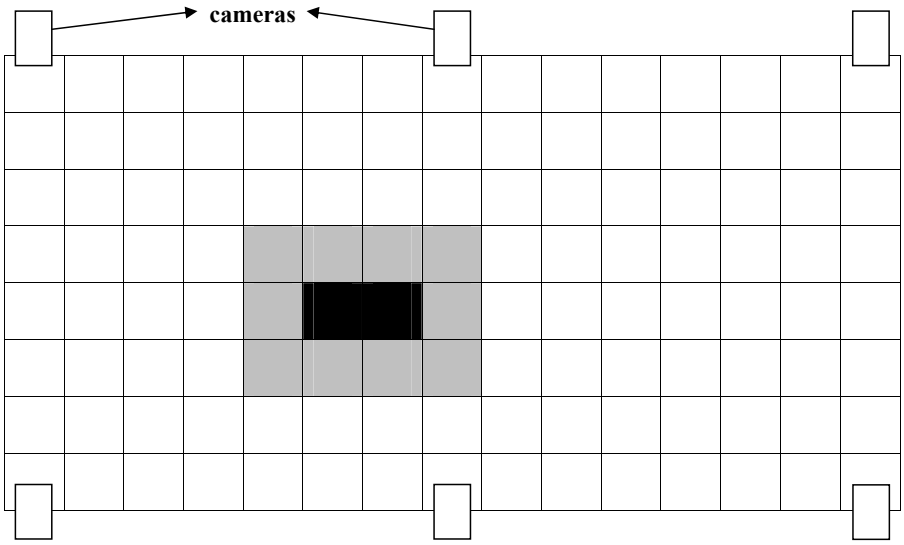


Figure 3.1: An example occupancy map.

The generation of the occupancy map is based upon the *volume intersection* paradigm. That is, we start with a model of occupied space and fill it with observed freespace and shadow. Based on the inputs from the multiple cameras and the results from the detection algorithm, pixels that are unchanged from the background are changed from occupied to empty, and the rest of the pixels are either changed from occupied to shadow or left unchanged.

Figure 3.1 shows a possible occupancy map. Here, white cells indicate empty space, grey cells indicate shadow and black cells indicate occupied space. This occupancy map is recomputed and updated on each new video frame sync signal.

3.1.1 Occupancy Map Initialization

Let $I[n, c, r]$, $0 < n \leq N$, $0 < c \leq C$, $0 < r \leq R$, n, c and r integers, define a set of C-columns x R-rows pixel images captured by N cameras. These cameras are calibrated to provide a transform, τ_n, from each camera's image space $I[n, c, r]$ to the (x, y, z) world space, given by

$$\tau_n : [\, n, c, r \,] \rightarrow (\, x, y, z \,) + (\, i, j, k \,)d = (\, x_g, y_g, 0 \,) \quad d > 0$$

where $(\, x, y, z \,)$ and $(\, i, j, k \,)$ are known for each pixel $(\, n, c, r \,)$.

A background image $B[n, c, r]$ is acquired for each camera while the floorspace to be monitored is empty. Further, a binary mask $M[n, c, r]$ is created for each background image, where $M[n, c, r] = 0$ signifies floorspace and $M[n, c, r] \neq 0$ signifies not floorspace. Thus, for $d > 0,\ 0 \leq x_g < X,\ 0 \leq y_g < Y,\ z = 0$ and $M[\, n, c, r \,] = 0$, a mapping, $\mathcal{F}$, given by

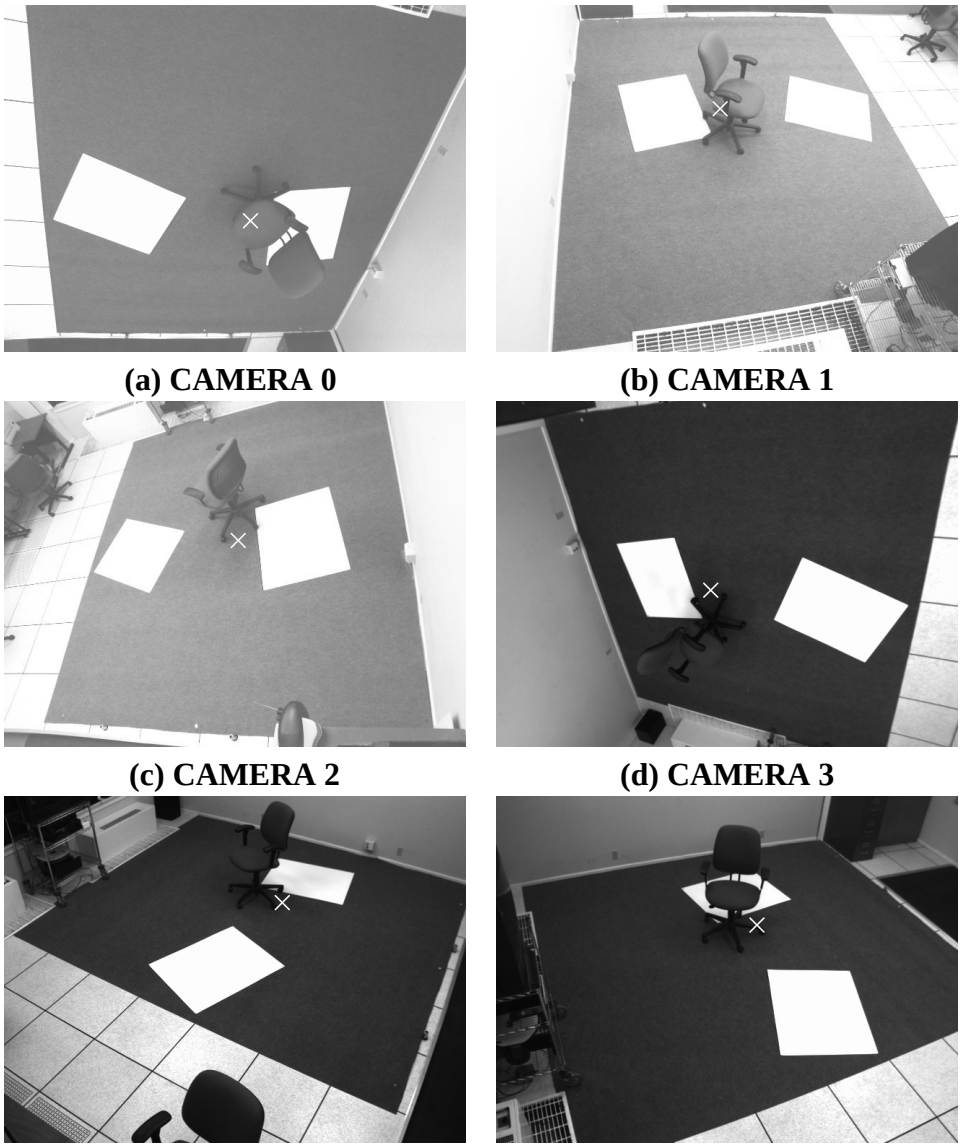
$$\mathcal{F} : I[n, c, r] \leftrightarrow O[x_g, y_g]$$

is established.

Since the solution for $\mathcal{F}$ is independent of image content, $\mathcal{F}$ can be computed off-line and stored as a look-up table. As $\mathcal{F}$ is a two-way mapping, we have two look-up tables, $L_1[n, c, r]$ and $L_2[x, y]$, the use of which lead to two different algorithms: image-based and map-based. $L_1[n, c, r]$ relates each image pixel for each camera to a unique occupancy map cell. $L_2[x, y]$ relates each occupancy map cell to a set of image pixels, which may include any number of pixels from each camera.

3.2 Input Generation

Since the occupancy map is generated by fusing the image data obtained from the N cameras, $L_1[n, c, r]$, which maps each image pixel for each camera to a unique occupancy map cell is called the forward look-up table. Consequently, $L_2[x, y]$, which provides us the image pixels (c,r) in each camera (n) corresponding to every occupancy map cell, $O[x, y]$, is termed as the reverse look-up table. We utilize the reverse look-up table in our map-based algorithm to generate the occupancy map. This is because we need the background and live image intensities for at least one pixel in each camera corresponding to each occupancy map cell to make a decision on the cell. These form the inputs and resultant feature vector for this detection system.



(a) CAMERA 0 **(b) CAMERA 1**

(c) CAMERA 2 **(d) CAMERA 3**

(e) CAMERA 4 **(f) CAMERA 5**

Figure 3.2: View of a scene point from six different cameras

To better explain how the map-based algorithm works, consider an example case shown in Figure 3.2 (a)-(f). These are live images where a chair and a poster board have been introduced into the environment. Background images with empty floorspace are acquired for each camera during system initialization. Particularly, let us consider how inputs for making a decision on a particular cell, $O[x,y]$, in the occupancy map are generated. This is done in the following steps:

1. Image pixels in each camera corresponding to $O[x,y]$ are identified by looking up the reverse look-up table, $L_2[x,y]$. These pixels have been indicated by an X in each live image.

2. Background and live image intensities corresponding to these pixels are now determined. The difference between these intensities for each camera forms the inputs to our detection algorithm.

Thus, the inputs for our system are the N differences between the background image intensity and live image intensity in each camera for pixels corresponding to a worldspace point under observation. For the rest of this thesis, we shall use the following notations for the above mentioned inputs:

Let $B_i = B[\ n,c,r\ ]$ be the background image intensity for cell under observation in camera i.

Let $L_i = I[\ n,c,r\ ]$ be the live image intensity for cell under observation in camera i.

Let $D_i = L_i - B_i = I[\ n,c,r\ ] - B[\ n,c,r\ ]$, be the difference between background and live image intensities in camera i for that cell.

Table 3.1 summarizes the inputs corresponding to the live images captured in Figure 3.2 (a)-(f).

Monochrome CCD (Charge Coupled Device) video cameras are used to output digital images as a 2-D array of typically 640×480 pixels. Each element in this array is filled with a value in the range 0 (black) to 255 (white). Values outside this range on both ends of the

CAMERA ID	B_i	L_i	$L_i - B_i$
0	135	92	-43
1	72	26	-46
2	51	35	-16
3	52	36	-16
4	51	36	-15
5	51	39	-12

Table 3.1: Observed intensities and differences at point X for all six cameras.

brightness spectrum are clipped to lie within this range. Noise might be generated in the camera because of any extraneous signal not generated by the object being imaged. This results in fluctuation in observed pixel intensity values. Thus, the same scene point being imaged might yield different pixel image intensity values at different times. Therefore, the sensor noise is taken into account while assuming a range of values defined by thresholds to identify differences in shadow, object or background range and while grouping into clusters.

3.3 Shadow Detection Algorithm

Our multiview method of shadow detection is motivated by the results obtained in Section 1.2. There it was observed that the differences in live and background intensities for cameras seeing shadow tend to cluster. The aim of this algorithm is to correctly identify this cluster by fusing the data available from the N cameras. The detection algorithm consists of two steps:

1. Clustering the inputs

2. Classification/ Decision-making based on the clustering

The following sections describe several variations on clustering and decision making that we have explored.

3.3.1 Clustering methods

For the various clustering methods and pseudocodes described below, the following variables and notations have been used:

- Let *cluster_shadow* be a cluster containing all differences identified as seeing shadow and *count_shadow* be the cardinality of this cluster.

- Let *cluster_empty* be cluster containing all differences identified as seeing background and *count_empty* be the cardinality of this cluster.

- Let *cluster_occupied* be cluster containing all differences identified as seeing an object and *count_occupied* be the cardinality of this cluster.

- Let C_i represent a cluster of intensity differences. Let the j^{th} element of this cluster be represented by C_{i_j}. The cardinality for this cluster is denoted by $||\ C_i\ ||$ and the average intensity of this cluster is denoted by (μ_{C_i}). Finally, let $max(C_i)$ and $min(C_i)$ hold the values of the highest and lowest elements of each cluster.

- Let T_b be a threshold set to take image noise into account. In other words, any deviation in live pixel image intensity from background pixel intensity within this range (positive or negative) is assumed to be due to noise.

- Let T_s ($< -T_b$) be a threshold set to take into account maximum intensity variation for a point in shadow from the background.

- Let *MAX_CLUSTER_WIDTH* denote the ”Maximum Cluster Width”, that is, the maximum possible intensity variation for any two elements within a cluster.

- We define two types of ”ground truth” for use in this thesis. The first is the ”world-space ground truth” and the second is the ”camera ground truth”. By ”world-space ground truth”, we mean what the point actually is in the world space or occupancy

map. By ”camera-space ground truth”, we mean what each camera is actually seeing. The ”world-space ground truth” might be blocked in individual camera views.

It is to be noted that the reverse look-up table returns sets of image pixel identities for each occupancy map cell. Set size can be any integer greater than or equal to zero. In our algorithm, we select one image pixel in this set for each camera corresponding to the entry in $L_2[x,y]$ for that occupancy map cell. Pseudocode for all the clustering algorithms has been written assuming this entry has been identified.

Four possible variations on clustering were attempted:

1. *Clustering based on static threshold ranges*

   For this method, we assume apriori fixed ranges of differences will always correspond to the same ”camera-space ground truth” of a floor point. We assume that a difference of zero and a small range around zero (for image noise) from the background image intensity will always indicate the camera is seeing a background point. We assume negative differences within a particular range will always indicate the point is in shadow. Everything outside these ranges is assumed to indicate the point belongs to an object. In essence, this method is frame differencing with a threshold applied to it.

   Formally, the pseudocode for this method is defined as:

   loop . . . time

   loop x = 0 . . . *map_columns*

   loop y = 0 . . . *map_rows*

   loop n = 0 . . . *N*-1

   if $-T_b \leq D_i \leq T_b$

   increment *count_empty* by 1, and

   add D_i to *cluster_empty*.

   else if $T_s \leq D_i < -T_b$

CAMERA ID	B_i	L_i	$L_i - B_i$
0	97	77	-20
1	62	25	-37
2	93	248	155
3	67	179	112
4	86	220	134
5	62	30	-32

Table 3.2: Observed intensities at point X in Figure 3.3 for all six cameras

increment *count_shadow* by 1, and

add D_i to *cluster_shadow*.

else

increment *count_occupied* by 1, and

add D_i to *cluster_occupied*. end loop N-1

end loop *map rows*

end loop *map columns*

end loop time

Example :

To demonstrate this clustering method, consider the image intensities observed at the point on the floor, X, in Figure 3.3. Table 3.2 contains live (L_i) and background (B_i) image intensities at that point and their differences.

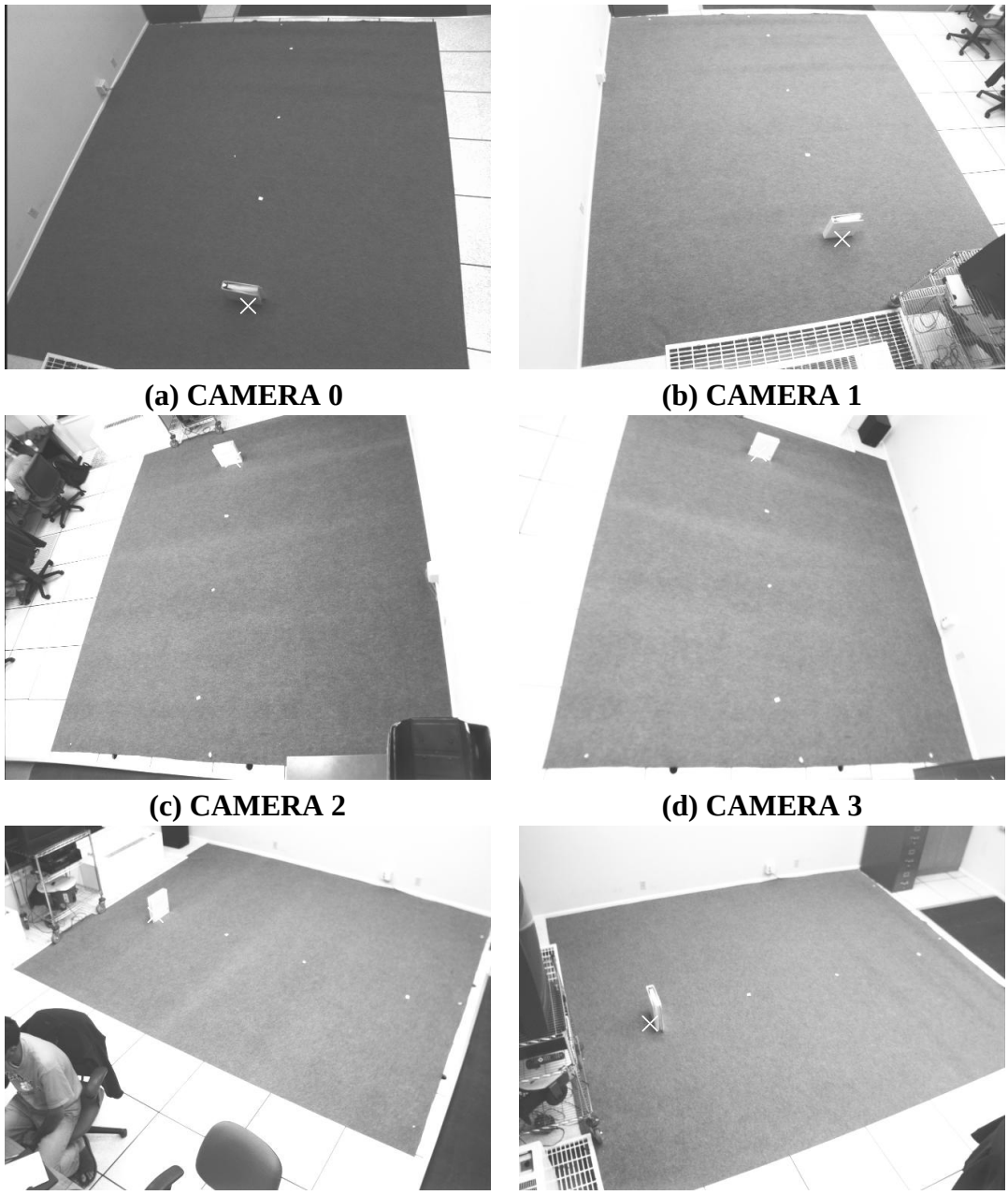
Assume the thresholds are set to: $T_b = 7$ and $T_s = -35$.

Running the algorithm on these values gives us the following clusters:

(a) No differences lie in the range -7 to 7. Therefore, *count_empty* $= 0$ and *cluster_empty* contains no elements.

(b) Two cameras (cameras 0, 5) have differences in the range -35 to -7. Therefore, *count_shadow* $= 2$ and *cluster_shadow* contains 2 elements.

cluster_shadow = {-20, -32}



(a) CAMERA 0 **(b) CAMERA 1**

(c) CAMERA 2 **(d) CAMERA 3**

(e) CAMERA 4 **(f) CAMERA 5**

Figure 3.3: Views of point X from six different cameras for clustering explanation

(c) Four cameras (cameras 1, 2, 3, 4) have differences lesser than -35 or greater than 7. Therefore, *count_occupied* $= 4$ and *cluster_occupied* contains 4 elements.

cluster_occupied = {-37, 112, 134, 155}

2. *Clustering based on expected cluster widths*

There are two major flaws in the first clustering algorithm:

- Values falling in different static threshold ranges will never be clustered together - however close their differences may be. For example, consider the clustering in Example 1. Intuitively, viewpoints with differences -20, -32 and -37 are most probably seeing the same world-space ground truth but they have not been grouped together.
- Objects with differences in the shadow range will automatically be classified as shadows. We shall henceforth refer to such objects as *chameleons*.

For cameras seeing the same camera-space ground truth, the differences will tend to cluster together. Setting static thresholds will group intensities into different clusters irrespective of how close they might be to each other. This clustering method aims to overcome this drawback by making an assumption on the ”maximum cluster width”. That is, it groups all differences into clusters whose range is less than this width. The ”maximum cluster width” is a variable that corresponds to the maximum intensity difference that might be observed in the image intensity of a point (background, object or in shadow) from viewpoints seeing the same camera-space ground truth. The value set for this variable should take into consideration such factors as changes in intensity observed for the same point from different views due to reflectance and other surface properties. Formally, the pseudocode for this method is defined as:

loop . . . time

loop x = 0 . . . *map columns*

loop y = 0 . . . *map rows*

loop n = 0 . . . *N*-1

Sort the N differences in ascending order.

Initialize number of clusters, $total\_clusters = N$.

Initialize each cluster, C_i by adding $D_i,\ 0 \leq i < N$ to that cluster. Thus, we start with N clusters.

Search for the smallest inter-difference, *inter_diff*, between any two consecutive clusters. Inter-difference between two consecutive clusters, C_i and C_k, $k > i$, is defined as $inter\_diff = max(C_k) - min(C_i)$.

Merge clusters if and only if : $inter\_diff \leq MAX\_CLUSTER\_WIDTH$.

Repeat last two steps until : $inter\_diff > MAX\_CLUSTER\_WIDTH$.

Calculate the average cluster intensity for each cluster, (μ_{C_i}), as :

$$(\mu_{C_i}) = \Sigma C_{i_j} / \mid\mid C_i \mid\mid,\ \ 0 \leq j < \mid\mid C_i \mid\mid$$

end loop *N*-1

end loop *map rows*

end loop *map columns*

end loop time

Example :

Again, consider the same set of values as given in Table 3.2. A reasonable assumption, $MAX\_CLUSTER\_WIDTH = 20$ is made. Now, running the above clustering algorithm on these values yields the following clusters:

(a) We have the original differences:

-20, -37, 155, 112, 134, -32

(b) These differences are sorted to get:

-37, -32, -20, 112, 134, 155

(c) Next each difference is initialized into its own cluster. Let a box represent a cluster. The initial clusters for this example are:

-37	-32	-20	112	134	155

(d) Now, the smallest inter-difference - difference between the maximum of a cluster and the minimum of the preceding cluster - is found to be 5 between clusters 0 and 1. Therefore, we merge clusters 0 and 1. Number of clusters is now equal to 5 given by :

-37, -32	-20	112	134	155

(e) Clusters 0 and 1 have the smallest inter-difference of 17. Therefore, we merge clusters 0 and 1. Number of clusters is now equal to 4 given by :

-37, -32, -20	112	134	155

(f) Now smallest inter-difference = 21 ($>$ *MAX_CLUSTER_WIDTH*) between clusters . Therefore, stop clustering at this point.

(g) Individual cluster averages are calculated to be: (μ_{C_0}) = -29.67, (μ_{C_1}) = 112, (μ_{C_2}) = 134 and (μ_{C_3}) = 155.

3. *Clustering into fixed number of clusters*

This clustering method is closely tied to a unique decision making process. The idea is that specific cluster configurations are likely to be caused by specific scene conditions. For example, if there is a cluster of four cameras seeing object-range differences and a cluster of two cameras seeing background-range differences, then the scene point is most likely a background point.

With three different types of clusters, of any possible size, this ”cluster recognition” is conceptually intractable. In order to limit the complexity, we describe a clustering method that limits the maximum number of clusters to three.

This method, in essence, uses the same principle as the previous method. The previous method clusters differences into any number between 1 and N total clusters

where N is the number of cameras in the set-up. The fact that the number of clusters can take any value in the range (1,N) increases the number of possible cases and hence, the complexity of the decision-making process. This method reduces this complexity by restricting the number of clusters to a maximum of three. This is based on the assumption that most practical situations produce at most 3 clusters.

For our shadow detection algorithm approach, we are not concerned with object segmentation. Therefore, all differences definitely in the object range, irrespective of whether they belong to different objects can be grouped together into one cluster. Differences due to shadow or background or due to objects with differences in shadow or background range are again grouped together. Assuming that the view of the floor point is not blocked by more than one objects, our decision to clip the number of clusters to 3 seems to be a reasonable assumption. Formally, the pseudocode for this method is defined as:

loop . . . time

loop x = 0 . . . *map columns*

loop y = 0 . . . *map rows*

loop n = 0 . . . *N*-1

Sort the N differences in ascending order.

Initialize number of clusters, $total\_clusters = N$.

Initialize each cluster, C_i by adding $D_i,\ 0 \leq i \leq N$ to that cluster. Thus, we start with N clusters.

Search for the smallest inter-difference, *inter_diff*, between any two consecutive clusters. Inter-difference between two consecutive clusters, C_i and C_k , $k > i$, is defined as $max(C_k) - min(C_i)$.

Merge clusters if and only if :

$total\_clusters > 3$, or,

$inter\_diff \leq MAX\_CLUSTER\_WIDTH$

Calculate the average cluster intensity for each cluster, (μ_{C_i}), as :

$$(\mu_{C_i}) = \Sigma C_i / \mid\mid C_i \mid\mid$$

end loop *N*-1

loop i = 0 . . . *total_clusters*-1

if $-T_b \leq (\mu_{C_i}) \leq T_b$

$count\_empty = count\_empty + \mid\mid C_i \mid\mid$ else

if $T_s \leq (\mu_{C_i}) \leq -T_b$

$count\_shadow = count\_shadow + \mid\mid C_i \mid\mid$ else

else $count\_occupied = count\_occupied + \mid\mid C_i \mid\mid$

end loop *total_clusters*-1

end loop *map rows*

end loop *map columns*

end loop time

Example :

To show how the clustering here differs from the previous method we use the same set of values again. As before, *MAX_CLUSTER_WIDTH* is set to 20. Running the above algorithm groups the above differences into clusters in the following manner:

(a) We have the original differences:

-20, -37, 155, 112, 134, -32

(b) These differences are sorted to get:

-37, -32, -20, 112, 134, 155

(c) Next each difference is initialized into its own cluster. Let a box represent a cluster. The initial clusters for this example are:

-37	-32	-20	112	134	155

(d) Now, the smallest inter-difference - difference between the maximum of a cluster and the minimum of the preceding cluster - is found to be 5 between clusters 0 and 1. Therefore, we merge clusters 0 and 1. Number of clusters is now equal to 5 given by:

-37, -32	-20	112	134	155

(e) Clusters 0 and 1 have the smallest inter-difference of 17. Now smallest inter-difference = 17. Therefore we merge clusters 0 and 1. Number of clusters is now equal to 4 given by:

-37, -32, -20	112	134	155

(f) Now, clusters 2 and 3 have the smallest *inter-difference* of 21 which is greater than *MAX_CLUSTER_WIDTH*. However, the number of clusters is still greater than 3 and so we merge clusters 2 and 3. Number of clusters is now equal to 3 given by:

-37, -32, -20	112	134, 155

(g) At this point, the total number of clusters is 3 and the smallest inter-difference between two clusters is 43 ($>$ *MAX_CLUSTER_WIDTH*). Therefore, we stop clustering at this point.

(h) Individual cluster averages are calculated to be:

(μ_{C_0}) = -29.67, (μ_{C_1}) = 112 and (μ_{C_2}) = 145.5.

4. *Clustering based on circular adjacency*

The previous methods ignore the spatial pattern of the cameras. This method assumes that pairs of adjacent cameras are more likely to see the same foreground object or shadow. Therefore, we apply clustering in the circular order of the cameras according to how they are spatially arranged around the room. Differences are only allowed to group with differences obtained from neighboring cameras or through a chaining of consecutive neighbors.

Formally, the pseudocode for this algorithm is as follows:

loop . . . time

loop x = 0 . . . *map columns*

loop y = 0 . . . *map rows*

loop n = 0 . . . *N*-1

Initialize $C_i = D_i, 1 \leq i \leq N$, and cardinality, $||\ C_i\ || = 1$. Thus, we have N clusters, each initialized with the respective D_i value and cardinality, 1.

Perform the following steps until the maximum cluster range is less than *MAX_CLUSTER_WIDTH*:

Calculate smallest cluster-difference, *cluster_diff*, between $C_{(i)mod(N)}$ and $C_{(i+1)mod(N)}$, and check if it is less than *MAX_CLUSTER_WIDTH*,that is, *lowest* $\mid C_{(i)mod(N)} - C_{(i+1)mod(N)} \mid \leq MAX\_CLUSTER\_WIDTH$.

If yes,

Merge $C_{(i)mod(N)}$ *and* $C_{(i+1)mod(N)}$.

Set *N*= *N*-1.

Calculate average cluster intensity for each cluster.

end loop *N*-1

end loop *map rows*

end loop *map columns*

end loop time

Example:

Again, consider the same set of values as in Table 3.2 obtained by recording background and live image intensities at point X in Figure 3.3. Let $MAX\_CLUSTER\_WIDTH = 20$. Running the above algorithm forms clusters as follows:

(a) We start with six clusters initialized with the six differences:

-20	-37	155	112	134	-32

(b) Now, clusters 0 and 4 have the smallest cluster inter-difference of 12. Therefore, we merge clusters 0 and 5. Number of clusters is now equal to 5 given by :

-32, -20	-37	155	112	134

(c) Now smallest cluster inter-difference = 17 between clusters 0 and 1. Therefore, we merge clusters 0 and 1. Number of clusters is now equal to 4 given by :

-32, -20, -37	155	112	134

(d) Now smallest cluster inter-difference = 21 ($>$ *MAX_CLUSTER_WIDTH*) between clusters . Therefore, STOP clustering at this point.

(e) Individual cluster averages are calculated to be: (μ_{C_0}) = -29.67, (μ_{C_1}) = 155, (μ_{C_2}) = 112 and (μ_{C_3}) = 134,

3.3.2 Decision-making methods

Once the differences from the N cameras has been grouped into clusters, a decision-making method needs to be applied to decide whether the actual point under observation is :

1. Empty or Background (E): $O[x, y] = 0$
2. Shadow or Shaded Background (S): $O[x, y] = 1$ or,
3. Occupied or Foreground object (O): $O[x, y] = 2$

Four possible decision-making methods were attempted:

1. *Fixed priority*

   Here, the point under observation is classified on a priority basis. This is based on the knowledge that if any one of the N cameras in the network is positively seeing a shadow, then the point under observation is in shadow. By positively viewing, we mean confirmation of the fact that it is not a foreground object with intensity difference in the shadow range. This is a safe assumption since a point in

shadow/background can be blocked by objects in other views but not vice versa. Therefore, if any cluster is determined to be a group of cameras seeing shadow, the point under observation is said to be in shadow. Next, if any cluster is determined as seeing background, the point under observation is identified as being a background point. Finally, a point under observation is determined to belong to a foreground object only if all clusters are identified as seeing foreground.

Using the same variables defined earlier, the pseudocode for this method is:

if *count_shadow* ≥ 1

$O[x,y] = 1$ else

if *count_background* ≥ 1

$O[x,y] = 0$ else

$O[x,y] = 2$

Example

Consider the following results obtained from the clustering step:

count_shadow = 2, *count_empty* = 1 and *count_occupied* = 3.

Occupancy Map cell under observation will be classified as shadow and $O[x,y]$ is set to 1 since atleast one difference in the shadow range is identified.

2. *Cluster Cardinality*

Once the differences have been grouped into clusters, this decision-making method classifies the point under observation based on the cardinality and average intensity of each cluster. This method reduces misclassifications of objects with differences in shadow or background range, or objects blocked by such objects in some views, as shadow or background. This is done by making an assumption that the cluster with maximum cardinality sees the ground truth. Unless an object blocking the view of the ground truth is really large or surrounds the point under observation, this assumption

holds good. In the case of two clusters having the same maximum cardinality, a decision is made by applying the fixed priority rule.

The pseudocode for this method is given below. For this pseudocode, the following additional variables have been used:

Let *Label*$[i]$ be a label assigned to each cluster based on its average intensity. This variable can only take on values 0, 1 and 2.

Let *Max_Cardinality* contain the identity of the cluster with maximum cardinality.

ALGORITHM:

Set *Max_Cardinality* = 0.

loop i = 0 . . . *total_clusters* - 1

if $-T_b <= (\mu_{C_i}) <= T_b$

Label$[i] = 0$ else

if $T_s <= (\mu_{C_i}) <= -T_b$

Label$[i] = 1$ else

Label$[i] = 2$.

if $||\ C_i\ || > ||\ C_{Max\_Cardinality}\ ||$

Max_Cardinality $=$ *i*.

end loop *total_clusters*-1

if *Label*$[$*Max_Cardinality*$] = 0$, $O[x, y] = 0$ else

if *Label*$[$*Max_Cardinality*$] = 1$, $O[x, y] = 1$ else

$O[x, y] = 2$

3. *Cluster Recognition*

This decision-making method is used for the case where the maximum number of clusters is restricted to being below or equal to three. Each occupancy cell is placed in a category based on the number of clusters, individual cluster elements and the

cluster cardinality. Finally, a decision is made based on what the world-space ground truth for that category is most likely to be.

The various categories and the most probable ground truth corresponding to each category is stored in a table called the *cluster recognition table*. Our cluster recognition table for the case where the maximum possible number of clusters is 3 is shown in Table 3.

The pseudocode for this decision-making method is given below. The following additional variables are used are used in this algorithm:

Let *Category* be the category in which each occupancy map cell is placed.

Let *NUMBER_CASES* be the total possible number of categories.

Let *Case*$[i][0]$ be the value of *count_empty* for the i^{th} category.

Let *Case*$[i][1]$ be the value of *count_shadow* for the i^{th} category.

Let *Case*$[i][2]$ be the value of *count_occupied* for the i^{th} category.

Let *Result*$[i]$ hold the most probable ground truth for the i^{th} category.

loop i = 1 . . . *NUMBER_CASES*

if *count_empty* $=$ *Case*$[i][0]$ and *count_shadow* $=$ *Case*$[i][1]$ and *count_occupied* $=$ *Case*$[i][2]$

Category $= i$

$O[x, y] =$ *Result*$[i]$

break loop *NUMBER_CASES*

end loop *NUMBER_CASES*

The cluster recognition table was constructed by considering a number of examples for each category and manually observing the ground truth for each case. Then, the most common ground truth for each case was calculated. This value was stored in *Result*$[i]$, signifying the most probable ground truth for that category.

Category(i)	*Case[i][0]*	*Case[i][1]*	*Case[i][2]*	*Result[i]*	*Likely Ground truth*
1 (6O)	0	0	6	2	Occupied
2 (6B)	6	0	0	0	Empty
3 (6S)	0	6	0	1	Shadow
4 (5O,1B)	1	0	5	2	Occupied
5 (5O,1S)	0	1	5	2	Occupied
6 (5B,1O)	5	0	1	0	Empty
7 (5B,1S)	5	1	0	0	Empty
8 (5S,1O)	0	5	1	1	Shadow
9 (5S,1B)	1	5	0	1	Shadow
10 (4O,2B)	2	0	4	0	Empty
11 (4O,2S)	0	2	4	1	Shadow
12 (4B,2O)	4	0	2	0	Empty
13 (4B,2S)	4	2	0	0	Empty
14 (4S,2O)	0	4	2	1	Shadow
15 (4S,2B)	2	4	0	1	Shadow
16 (3O,3B)	3	0	3	0	Empty
17 (3O,3S)	0	3	3	1	Shadow
18 (3B,3S)	3	3	0	0	Empty
19 (3O,2B,1S)	2	1	3	0	Empty
20 (3O,2S,1B)	1	2	3	0	Empty
21 (3B,2O,1S)	3	1	2	0	Empty
22 (3B,2S,1O)	3	2	1	2	Occupied
23 (3S,2O,1B)	1	3	2	1	Shadow
24 (3S,2B,1O)	2	3	1	2	Occupied
25 (2O,2B,2S)	2	2	2	2	Occupied
26 (1B,1S,4O)	1	1	4	2	Occupied
27 (1B,4S,1O)	1	4	1	1	Shadow
28 (4B,1S,1O)	4	1	1	0	Empty

Table 3.3: Cluster recognition table indicating likely ”world-space” ground truth for each category when total number of clusters is restricted to 3.

Example

To explain how the cluster recognition table is interpreted, suppose the clustering step gives us the following three cluster cardinalities:

- *count_empty* = 3: three differences in background range,
- *count_occupied* = 2: two differences in object range, and,
- *count_shadow* = 1: one difference in shadow range.

This corresponds to case 21(3B,2O,1S) in the cluster recognition table and the most likely ground truth for this case is *Empty*. Therefore, the corresponding occupancy map cell is classified as unoccupied and $O[x, y] = 1$.

Chapter 4

Results and Observations

In this section, all the algorithms described in the previous section were run and tested on a system consisting of six cameras and a single CPU general purpose computer. The six input video streams are processed at full resolution (480 x 640 standard NTSC) producing a 480 x 640 resolution occupancy map. Using a PC with an Athlon AMD 1200MHz processor, these algorithms were able to compute and display a new occupacy map frame almost every 30-60 ms, that is at a frame rate of around 10-20Hz.

In this section, we combine a clustering method and a rule-based decision method and test it in our environment. We try to show, firstly, how the detection method works as a whole, how some combinations of clustering and decision-making methods work better than others and finally cases in which the methods described in this thesis fail and analyze the reasons behind these failures.

4.1 Occupancy Map results for different approaches

This section contains the occupancy maps obtained when running the different combinations of a clustering and a decision-making approach. In each example, Figures (a) - (f) contain live images from the various viewpoints in the environment to get an idea of the scene. The occupancy maps computed for various combinations of clustering and decision-

making are displayed after that. The coloring scheme used in the occupancy maps is as follows:

- Black cells : Background or Empty region (E)
- Gray cells : Shaded Background or Shadow region (S)
- White cells : Foreground or Object region (O)

Example 1

Consider a case where a chair introduced into the scene as seen in the live image captures in Figures 4.1(a)-(f).

Occupancy map computed for different combinations of clustering and decision-making are seen in Figures 4.2, 4.3, 4.4 and 4.5. All possible combinations have been tested to be able to make a comparitive evaluation.

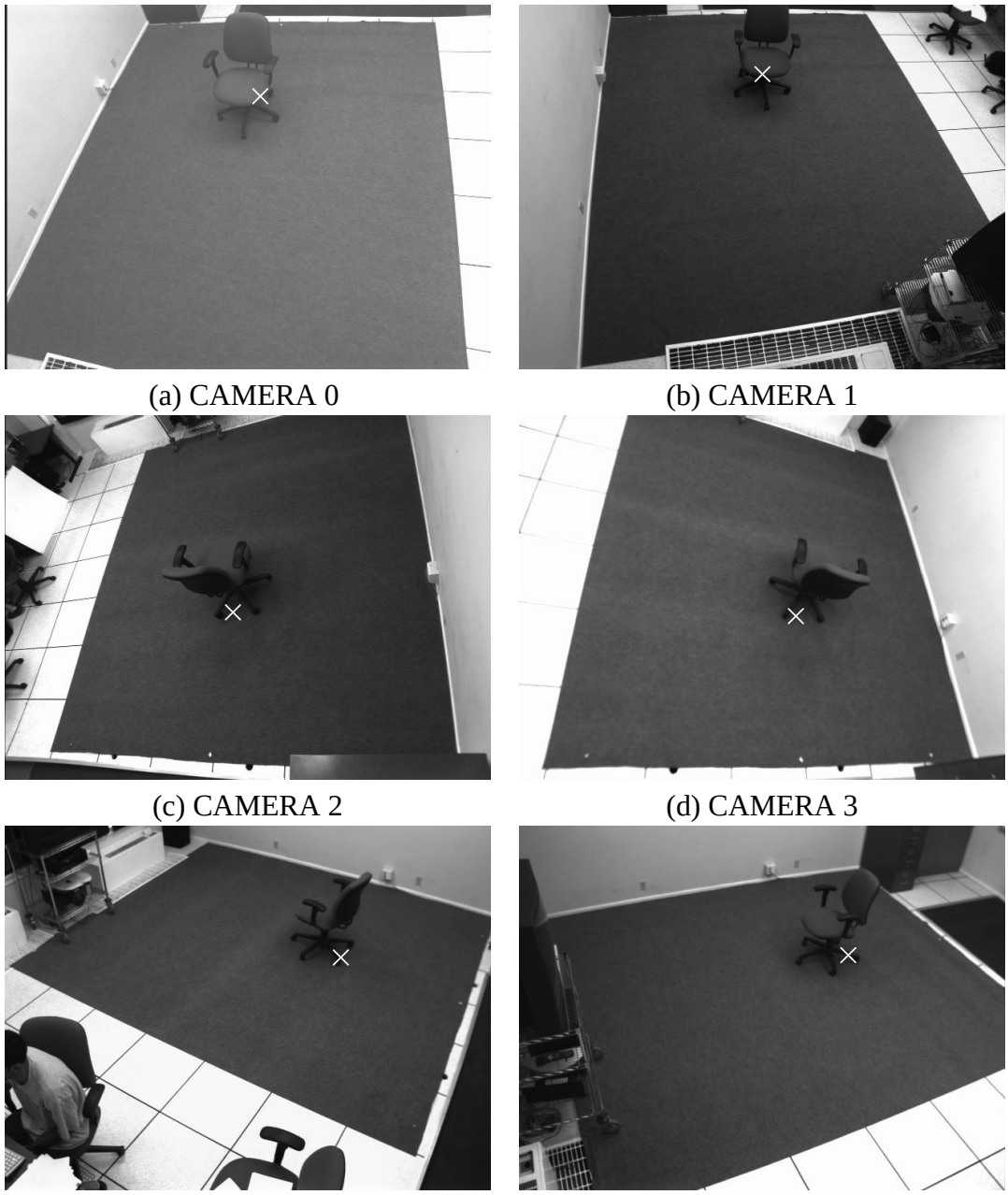
It is observed that the following combinations give good detection rates:

- Clustering using static thresholds and making a decision based on cluster cardinality.
- Clustering using estimated Cluster Widths and making a decision based on cluster cardinality.
- Clustering into fixed number of clusters and making a decision based on cluster recognition.
- Clustering based on spatial adjacency and making a decision based on fixed priority.

Example 2

Another scenario with two chairs and a chart board introduced into the scene is seen in the live image captures in Figures 4.6(a)-(f). Occupancy map computations for different combinations of clustering and decision-making are seen in Figures 4.7, 4.8, 4.9 and 4.10.

The different combinations of clustering and decision-making methods were tried on various scenes. The scenes were changed by:



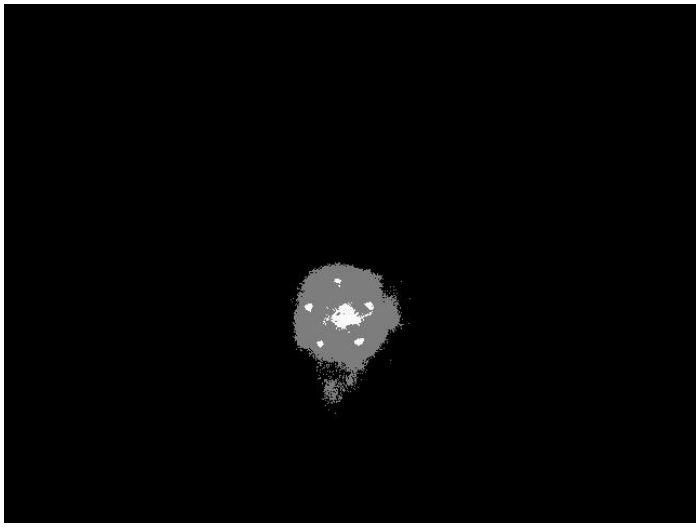
(a) CAMERA 0 (b) CAMERA 1

(c) CAMERA 2 (d) CAMERA 3

(e) CAMERA 4 (f) CAMERA 5

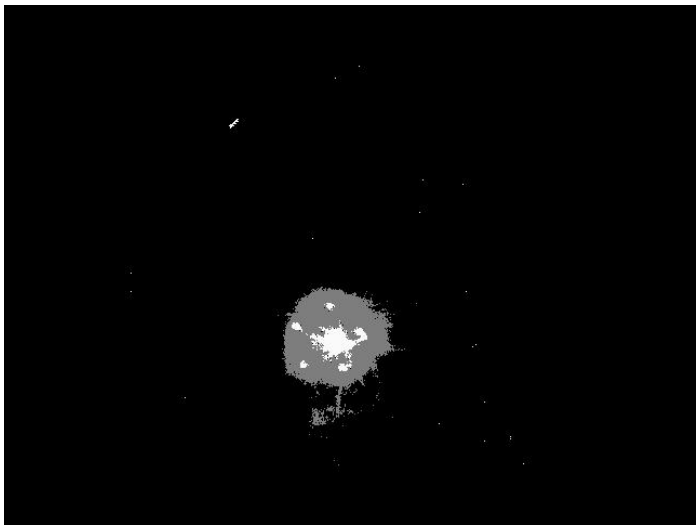
Figure 4.1: Scene as described in Example 1.

Using fixed priority decision-making and with thresholds $T_s = -45$ and $T_b = 10$

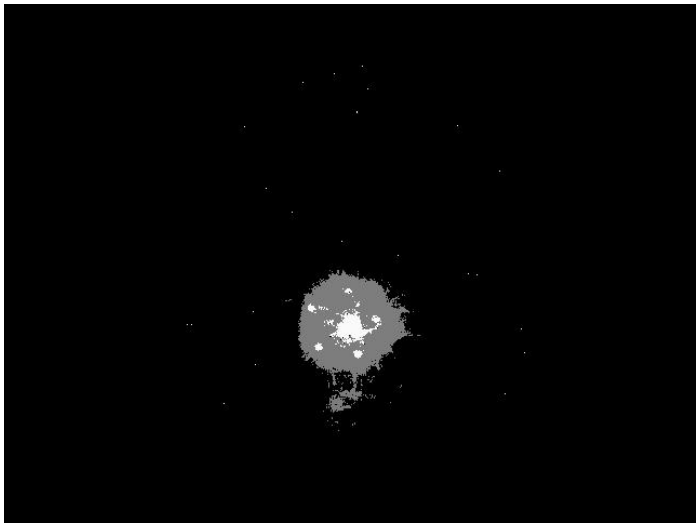


Making decision based on cluster cardinality and with thresholds set as $T_s = -45$ and $T_b = 10$

Figure 4.2: Occupancy Map results using static thresholding on scene described by the live image captures in Figures 4.1 (a) - (f).

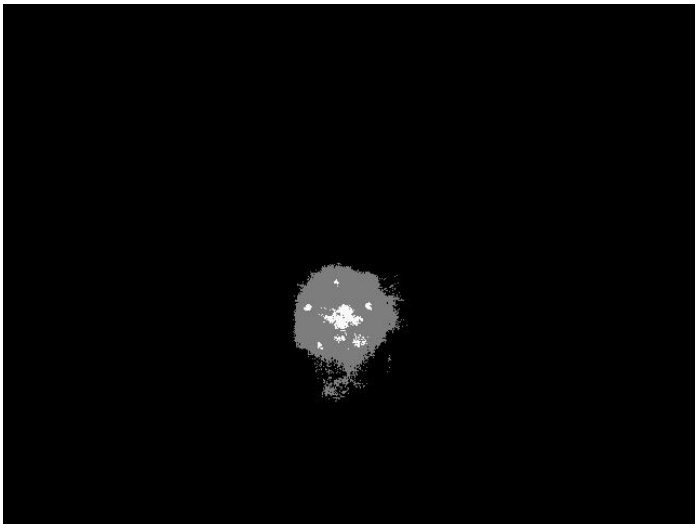


Using fixed priority decision-making and with thresholds $T_s = -45$ and $T_b = 10$



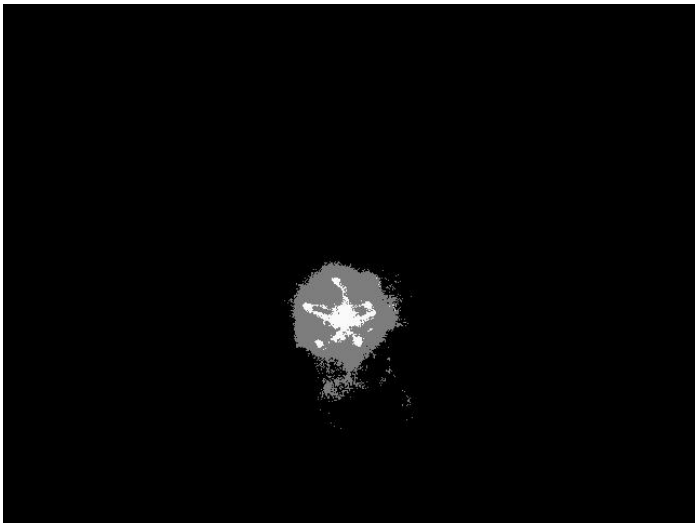
Making decision based on cluster cardinality and with thresholds set as $T_s = -45$ and $T_b = 10$

Figure 4.3: Occupancy Map results when clustering using estimated cluster width on scene described by the live image captures in Figures 4.1 (a) - (f).



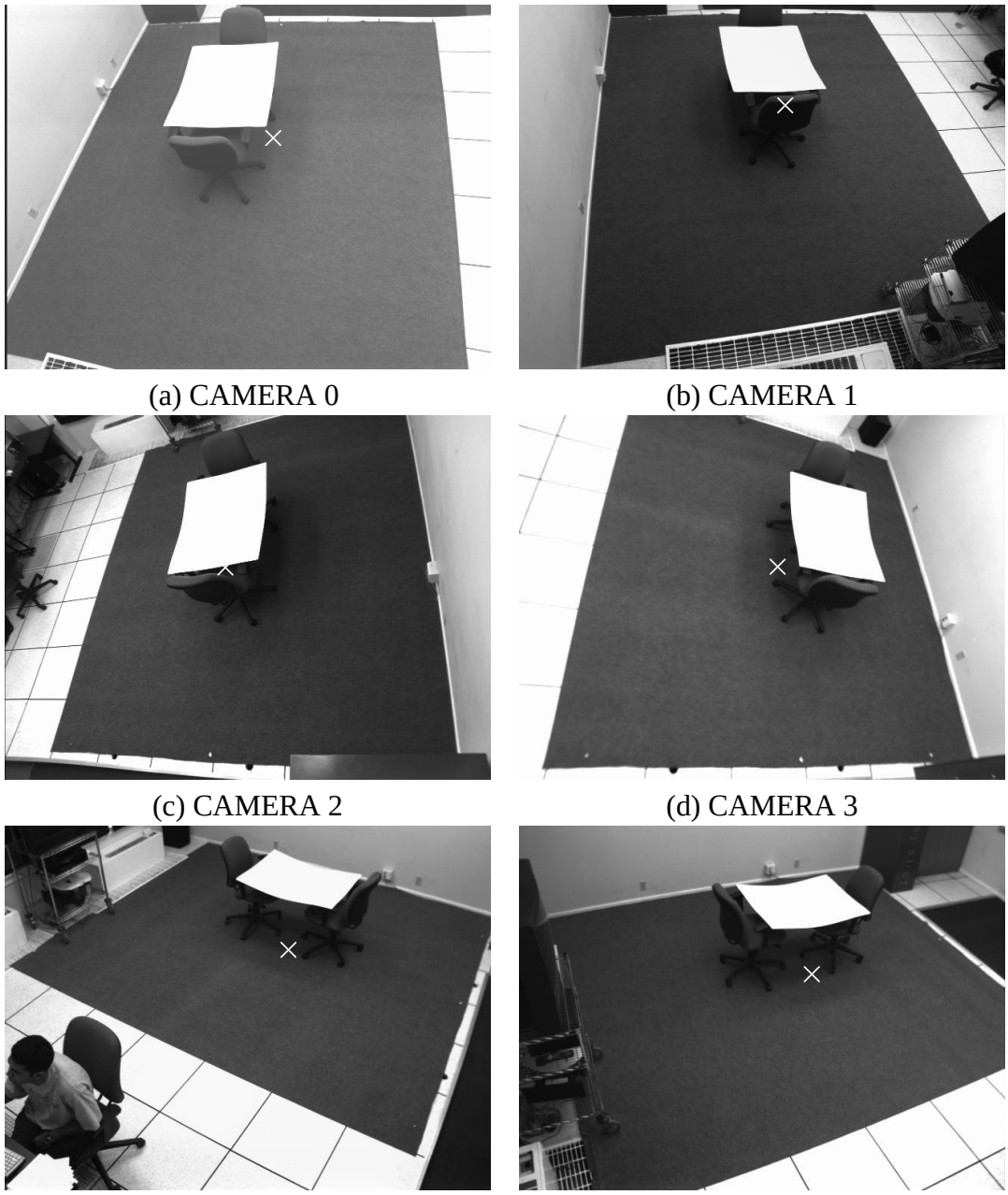
Using cluster recognition based decision with thresholds $T_s = -45$ and $T_b = 10$

Figure 4.4: Occupancy Map results when clustering into fixed number of clusters on scene described by the live image captures in Figures 4.1 (a) - (f).



Using fixed priority based decision with thresholds $T_s = -45$ and $T_b = 10$

Figure 4.5: Occupancy Map results when clustering based on spatial adjacency on scene described by the live image captures in Figures 4.1 (a) - (f).

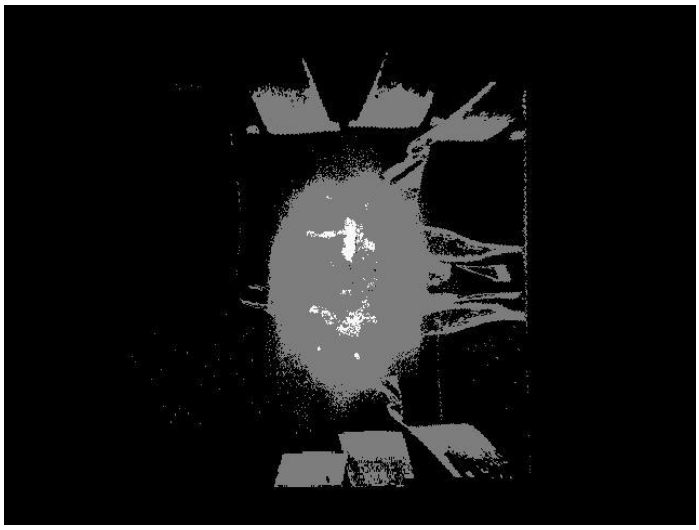


(a) CAMERA 0 (b) CAMERA 1

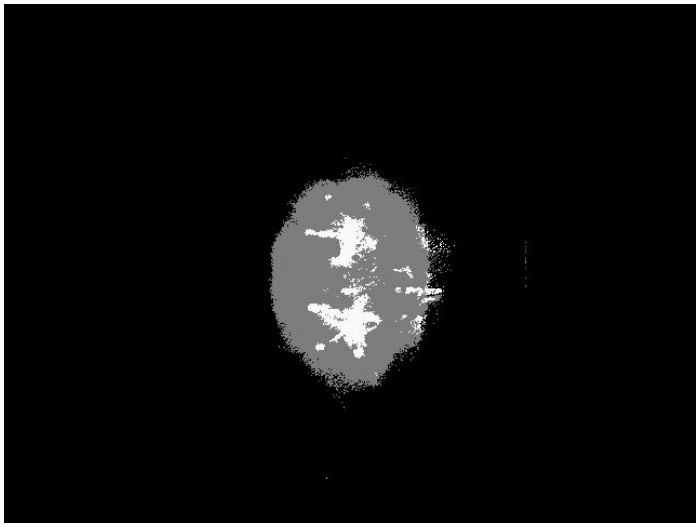
(c) CAMERA 2 (d) CAMERA 3

(e) CAMERA 4 (f) CAMERA 5

Figure 4.6: A test scene as described in Example 2.

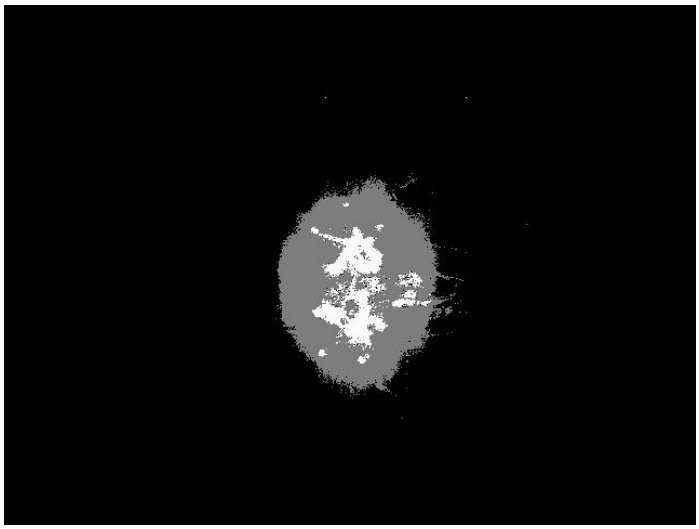


Using fixed priority decision-making and with thresholds $T_s = -45$ and $T_b = 10$

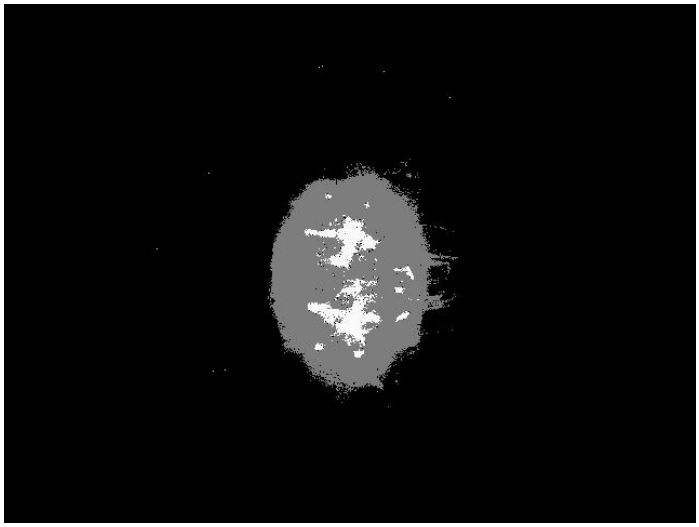


Making decision based on cluster cardinality and with thresholds set as $T_s = -45$ and $T_b = 10$

Figure 4.7: Occupancy Map results using static thresholding on scene described by the live image captures in Figures 4.6 (a) - (f).

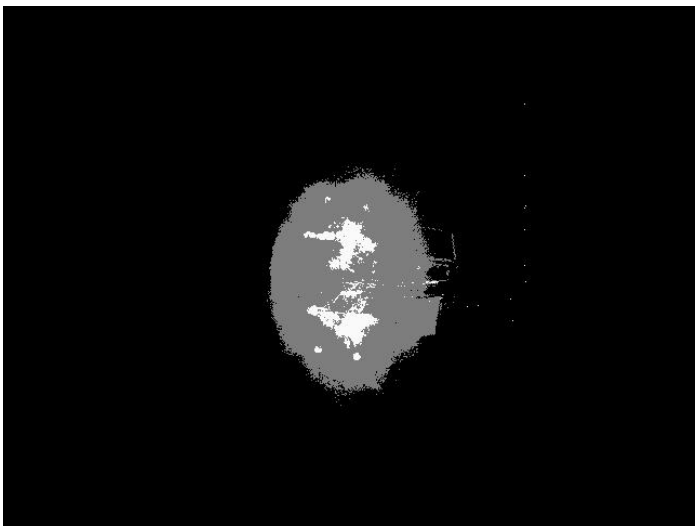


Using fixed priority decision-making and with thresholds $T_s = -45$ and $T_b = 10$



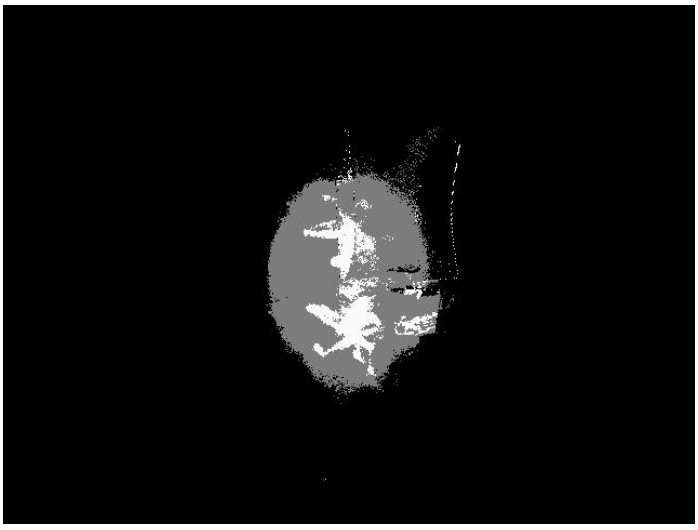
Making decision based on cluster cardinality and with thresholds set as $T_s = -45$ and $T_b = 10$

Figure 4.8: Occupancy Map results when clustering using estimated cluster width on scene described by the live image captures in Figures 4.6 (a) - (f).



Using cluster recognition based decision with thresholds $T_s = -45$ and $T_b = 10$

Figure 4.9: Occupancy Map results when clustering into fixed number of clusters on scene described by the live image captures in Figures 4.6 (a) - (f).



Using fixed priority based decision with thresholds $T_s = -45$ and $T_b = 10$

Figure 4.10: Occupancy Map results when clustering based on spatial adjacency on scene described by the live image captures in Figures 4.6 (a) - (f).

Clustering method	*Decision-Making method*	*Performance*
Static Thresholding	Fixed Priority	Poor
Static Thresholding	Cluster Cardinality	Very Good
Estimated Cluster Width	Fixed Priority	Average
Estimated Cluster Width	Cluster Cardinality	Good
Fixed no. of clusters	Cluster Recognition	Very Good
Spatial Adjacency	Fixed Priority	Good

Table 4.1: Performance Evaluation of various approaches for the detection algorithm

1. Changing the number of objects in the scene

2. Trying different objects with varying colors and levels of brightness in the scene

3. Changing the depth of shadows caused by objects in the scene

Based on occupancy map computations for various combinations of clustering and decision-making, we evaluated the performance of each approach on a qualitative scale. Table 4.1 contains this evaluation.

Chapter 5

Conclusion

Detection of objects is an essential part of many computer vision applications. Background differencing is a common method used for object detection. This method subtracts live or raw images, captured by cameras, from the background image, to determine objects that have been newly introduced into the scene. The presence of shadows in these images is the primary cause for incorrect detection of objects in a scene.

All methods for shadow detection until now have used only one viewpoint to determine whther a point is in shadow or not. This thesis suggested a new approach to shadow detection - fusing data from multiple viewpoints to determine whether a point is in shadow or not. The algorithm consists of two steps: Clustering and rule-based decision making. Various approaches to both these steps were studied, tested and analyzed.

The clustering algorithms used four different approaches:

- Direct application of static thresholds to intensity differences obtained from multiple cameras.
- Clustering differences from multiple cameras based on proximity of observed differences.
- Clustering differences into a fixed number of clusters based on proximity of observed differences.

- Clustering based on spatial adjacency of cameras observing the environment.

One of three possible decision-making methods was then used to decide whether the point under observation is an object, in shadow or background.

- Fixed priority - if any cluster/camera sees shadow, the point is identified as being in shadow, then background and finally object.
- Cluster Cardinality - cluster with maximum cardinality is assumed to be seeing true world-space ground truth.
- Cluster Recognition - decision making method tied to fixed clustering where decision is made based on pre-determined classification based on clustering results.

While the classification algorithms worked correctly for most cases, the primary cause for failure was the presence of foreground objects with intensity differences similar to either the background or shadow. These *chameleons* cause misclassification of shadow/empty points as occupied and vice-versa. Also, in a few cases it was observed that while the final decision on the point under observation is correct, a correct decision on what each camera is seeing is not made.

This thesis showed the potential for multiview shadow detection. However, we are interested in making the algorithms suitable for running in real-time. Hence, more complicated approaches have been avoided keeping in mind the increased processing time. Many directions are available for further research. There are a number of other inputs which could be used independently or incorporated into this algorithms to get better detection rates. These include:

- Points in shadow closer to the object will be deeper (umbra) as compared to points in shadow away from the object (penumbra). Further, it is very likely that a point in deep shadow (umbra region) will be blocked by the object causing the shadow in at

least one view. Distance of the point under observation from each camera could be used to confirm such cues towards the presence of shadows in the environment.

- Using color cameras and chromaticity information. The fact that shadows cause only a significant change in brightness from the background whereas objects differ significantly in both brightness and chromaticity from the background can be used to improve shadow detection.

- Using temporal information, that is information on world-space points from previously processed occupancy maps to make a decision on current frame.

Bibliography

[1] Raphael Charit and Murray S. Loew. Complex shadow-boundary segmentation using the entry-exit method. In *IEEE Computer Vision and Pattern Recognition*, pages 536–541, June 1988.

[2] Ahmed Elgammal, David Harwood, and Larry S. Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision*, volume 2, pages 751–767, June 2000.

[3] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, Aug 1997.

[4] Gareth Funka-Lea and Ruzena Bajcsy. Combining color and geometry for the active, visual recognition of shadows. In *Proceedings of IEEE International Conference on Computer Vision*, pages 203–209, June 1995.

[5] Gareth D. Funka-Lea. *The Visual Recognition of Shadows by an Active Observer*. PhD thesis, University of Pennsylvania, Dec 1994.

[6] Larry N. Hambrick, Murray H. Loew, and Robert L. Caroll Jr. The entry-exit method of shadow boundary segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume PAMI(9), No. 5, pages 597–607, September 1987.

[7] Adam Hoover and Bent David Olsen. A real-time occupancy map from multiple video streams. In *IEEE Conference on Robotics and Automation*, volume 3, pages 2261–2266, May 1999.

[8] Thanarat Horprasert, David Harwood, and Larry S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE International Conference on Computer Vision Frame Rate Workshop*, 1999.

[9] A. Huertas and R. Nevatia. Detecting buildings in aerial images. In *Computer Vision Graphics and Image Processing*, volume 41(2), pages 131–152, February 1988.

[10] R. B. Irvin and D. M. McKeown. Method for exploiting the relationship between buildings and their shadows in aerial images. In *IEEE Transactions on System, Man and Cybernetics*, volume 19(6), pages 1564–1575, November 1988.

[11] C. Jiang and M. O. Ward. Shadow identification. In *IEEE Computer Vision and Pattern Recognition*, pages 606–612, June 1992.

[12] David J. Kriegman and Peter N. Belhumeur. What shadows reveal about object structure. In *European Conference on Computer Vision*, volume 2, pages 399–414, June 1998.

[13] Stephen J. Mckenna, Sumer Jabri, Zoran Duric, Harry Wechsler, and Azriel Rosenfeld. Tracking groups of people. In *Computer Vision and Image Understanding*, volume 80, No 1, pages 42–56, October 2000.

[14] F. O'Gorman. Light lines and shadows. Technical report, 1975.

[15] L. O'Gorman. Binarization and multi-thresholding of document images using image connectivity. In *Computer Vision, graphics and Image Processing*, volume 56(6), pages 494–506, November 1994.

[16] Andrea Prati, Rita Cucchiara, Ivana Mikic, and Mohan M. Trivedi. Analysis and detection of shadows in video streams: A comparative evaluation. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 571–576, December 2001.

[17] Andrea Prati, Rita Cucchiara, Ivana Mikic, and Mohan M. Trivedi. Comparative evaluation of moving shadow detection algorithms. In *Workshop on Empirical Evaluation Methods in Computer Vision*, December 2001.

[18] Paul L. Rosin and Tim Ellis. Image difference threshold strategies and shadow detection. In *British Machine Vision Conference*, pages 347–356, September 1996.

[19] Steven A. Shafer. *Shadows and Silhouettes in Computer Vision*. Kluwer Academic Publishers, Hingham, Masachusettes, 1985.

[20] James J. Simpson, Zhongai Jin, and James R. Stitt. Cloud shadow detection under arbitrary viewing and illumination conditions. In *IEEE Transactions on Geoscience and Remote Sensing*, volume GeoRS(38), No. 2, pages 972–976, March 2000.

[21] Jurgen Stauder, Roland Mech, and Jorn Ostermann. Detection of moving cast shadows for object segmentation. In *IEEE Transactions on Multimedia*, volume MultMed(1), No. 1, pages 65–76, March 1999.

[22] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 246–252, June 1999.

[23] C. Wang, L. Huang, and A. Rosenfeld. Detecting clouds and cloud shadows on aerial photographs. In *Pattern Recognition Letters*, volume 12, pages 55–64, 1991.