

國立交通大學

電子工程學系 電子研究所碩士班

碩 士 論 文

靜態攝影機所拍攝影片內容之相對深度估計

**Relative Depth Estimation for Videos from
Static Cameras**

研 究 生：何開暘

指導教授：王聖智 教授

中 華 民 國 一 〇 〇 年 十 月

靜態攝影機所拍攝影片之相對深度估計

Relative Depth Estimation for Videos from Static Cameras

研究生：何開暘

Student：Kai-Yang Ho

指導教授：王聖智教授

Advisor：Prof. Sheng-Jyh Wang

國立交通大學

電子工程學系 電子研究所碩士班



Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of Master of Science

in

Electronics Engineering

October 2011

Hsinchu, Taiwan, Republic of China

中華民國一〇〇年十月

靜態攝影機所拍攝影片之相對深度估計


研究生：何開暘

指導教授：王聖智 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘要



在本篇論文裡，我們提出一套應用於固定式攝影機拍出影片的深度估計系統。藉由對現有的單張影像深度估計方法的擴展，我們希望能從影片中取出有用的時間資訊，將其加入單張影像深度估計演算法來產生更準確的背景深度的估計，接著給予移動物體適當的深度來產生整段影片深度的輸出。在固定式攝影機拍攝的影片之中，我們可以觀測到移動物體和場景中障礙物的遮蔽現象，然而在短時間之內資訊並不足夠，我們需要利用長時間累計的結果來做出正確的遮蔽邊界判斷。因此，我們利用統計的方式將移動物體的底部高度記錄於其所經過的地方，利用一段時間統計出來的結果來判斷靜止前景物體的位置及其深度範圍，我們將這些資訊加入單張影像遮蔽邊界演算法來產生邊界判斷及對應的相對深度輸出。對於移動的物體，我們用背景相減法將它們找出來並做基本的形態學處理，根據追蹤及遮蔽的判斷我們使用不同的方式給予它們深度並得到影片的深度輸出。

Relative Depth Estimation for Videos from Static Cameras

Student: Kai-Yang Ho

Advisor: Prof. Sheng-Jyh Wang

Department of Electronics Engineering, Institute of Electronics
National Chiao Tung University

Abstract

In this thesis, we propose a method to estimate the depth of a video taken from a static camera. By extending the single image depth estimation algorithm, we explore the temporal knowledge in the video. Combining both single image knowledge and temporal knowledge, we get better depth estimation of the background image. After that, we estimate the proper depth of the moving objects and get the video depth output. In a video, some boundaries can be detected when moving objects are occluded by the static occluding objects. However, short term temporal knowledge is not enough and we need long term temporal knowledge to find them. In our approach, we get the statistics by recording the object bottom height in the pixels the the object has passed. Using this statistical information, we find static occluding objects and their maximum depth. We add these cues into the boundary detection algorithm to get the boundary result and the relative depth. For those moving objects, we use the background subtraction method to identify them and perform morphological operations for post processing. Depending on the states of moving objects, we use different ways to estimate their depth. Finally, we get the depth of the video contents.

誌謝

在這裡我要感謝我的指導教授 王聖智老師這兩年來的指導，讓我了解從事研究的方法和態度，並對影像處理和分析有初步的認識，老師也指導我在報告時需要改進的地方，並給予我多次的練習，讓我有所進步。另外我也要感謝實驗室的成員這段日子的陪伴，學長們在我研究有問題時能給予我協助，同屆的同學們能夠和我討論，幫我查看我的想法是否正確，在研究之外的事也幫我解決了很多問題，學弟妹們則是為實驗室帶來歡笑。最後我要感謝我的家人及朋友，在我研究不順利時能給予我鼓勵，讓我能一步步的往前邁進。



Content

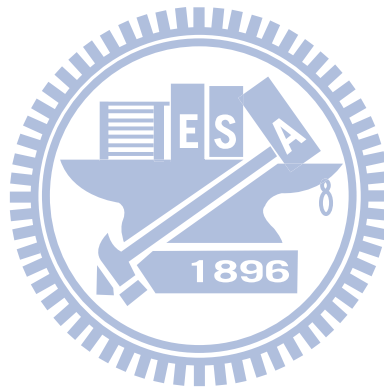
Chapter 1.	Introduction.....	1
Chapter 2.	Background.....	3
2.1.	Single Image Depth Estimation algorithms.....	3
2.2.	Depth Estimation from Occlusion Reasoning.....	8
2.2.1.	Surface Layout.....	8
2.2.2.	Occlusion Boundary.....	12
Chapter 3.	Proposed Method.....	16
3.1.	Temporal Knowledge.....	17
3.2.	Background Depth Estimation.....	27
3.3.	Video Depth Output.....	34
Chapter 4.	Experimental Results.....	37
Chapter 5.	Conclusions.....	41
References		42



List of Figures

Figure 2-1	The results of Battiato’s proposed system [1]	4
Figure 2-2	The results of Torralba’s proposed system [4]	4
Figure 2-3	The filters used by Saxena [5].....	5
Figure 2-4	The multiple scale structure [5].....	6
Figure 2-5	The result of Saxena’s proposed system [5].....	6
Figure 2-6	Result of Liu’s proposed system [8]	7
Figure 2-7	The labels of Hoiem’s system [9]	9
Figure 2-8	The cues used in Hoiem’s surface system [9].....	10
Figure 2-9	One multiple segmentation result of Hoiem’s system[9].....	11
Figure 2-10	The result of Hoiem’s surface layout estimation [9].....	11
Figure 2-11	One result of Hoiem’s occlusion boundary algorithm[10].....	12
Figure 2-12	The cues used in Hoiem’s boundary system [10].....	13
Figure 2-13	The flow chart of Hoiem’s occlusion boundary system[10].....	14
Figure 2-14	Illustration of five valid junctions [10]	14
Figure 2-15	The effect of Hoiem’s CRF model [10].....	15
Figure 3-1	The flow chart of our system	16
Figure 3-2	Left: Background image with boundary which is hard to detect. Right: Moving object passes through the boundary.....	17
Figure 3-3	When a car passes through the back of the tree, we can see the boundary in the background subtraction result last for a short time.	18
Figure 3-4	The man in the middle of the image walks in front of the tree, but we see the false boundary in the background subtraction result due to the similarity of color.	18
Figure 3-5	Left: The background image. Right: The summation of four minutes background subtraction results.....	19
Figure 3-6	Temporal knowledge collection steps	21
Figure 3-7	The statistical data of bottom height 100 and 165	21
Figure 3-8	An example of appearance number at different height of two points	22
Figure 3-9	The filtering process to find the occluding object region	24
Figure 3-10	The histograms of two points with the prediction and comparison steps	25
Figure 3-11	The depth range. Left: Minimal depth. Right: Maximal depth.....	26
Figure 3-12	The results of occluding objects detection using different statistics.....	27
Figure 3-13	The flowchart of Hoiem’s algorithm with some blocks adding temporal knowledge	28
Figure 3-14	The gradient result combining the original gradient result and the gradient	

of temporal statistical images.....	29
Figure 3-15 The comparison of the initial segmentation with and without using the temporal knowledge.	29
Figure 3-16 (a) Initial segmentation. (b) Boundary likelihoods from single-image features. (c) Boundary likelihoods from temporal cues. (d) Boundary likelihoods using both single image cues and temporal cues.....	31
Figure 3-17 The ground likelihood comparison	32
Figure 3-18 Comparison of background depth estimation	34
Figure 3-19 Object's bottom point is not always equal to the contact point on the ground	35
Figure 4-1 The background images of our demo videos.....	37
Figure 4-2 Some background subtraction results	38
Figure 4-3 Background depth estimation result.....	39
Figure 4-4 Video depth estimation result.....	40



Chapter 1.

INTRODUCTION

Depth estimation is an important issue in computer vision. Human can easily perceive and interpret the 3-D scene. We wish computer can also have the same ability to understand the 3-D scene itself. If a computer can understand 3-D knowledge, it would be of a great help in many works, like surveillance, navigation, and image editing. Hence, it is valuable to find some ways to get the depth information automatically. Up to now, many techniques have been explored. In a vision-based system, the most common and stable approach is to use binocular knowledge. With two or more images taken from the same scene with a little displacement of camera position, we can estimate the 3-D depth using geometric rules. However, in most images or videos, we do not have this kind of stereoscopic knowledge.

In the past few years, some people try to find the geometric properties in a single image and use them to estimate the depth [1,2,3]. However, they face the common problem: there are too few images. Usually, their approaches can only work at specific scenes. In recent years, with the progress of computer vision techniques, some people estimate the 3-D depth by using machine learning methods and by using images [5,8,10]. Their works seems to work better in getting some 3-D scene knowledge from the image. In our work, we aim to extend these learning-based methods from images to videos and plan to include the temporal information into the learning-based approach for the depth estimation of the 3-D scene.

In this thesis, we propose an algorithm to estimate the relative depth of the 3-D scene in the videos captured by a static camera. We first estimate the depth of the background and then calculate the depth of moving objects in the video frames. We explore the temporal knowledge which is helpful for background depth estimation. The most valuable knowledge is the occlusion effect which can help us find the boundaries and explore the figure/ground relationship. We use a background subtraction method to find the moving objects, and then use statistical method to record some useful data. By analyzing these data, we find the candidates of occluding objects and get their boundaries. After getting the depth of the background image, we assign depth to moving objects. Here, we use a simple tracking technique and use the maximum/minimum depth information to obtain more reliable estimation.

The greatest challenge is that the background subtraction method cannot perform well on complicated scenes. It often misses some pixels where the color of the foreground object is similar to that of the background. This might cause some false negatives in the detection of occluding objects. To obtain more stable and reliable depth estimation, we have to design a method that is not so sensitive to the occurrence of false negatives.

In this thesis, we first discuss some related works in Chapter 2. In Chapter 3, we will describe how to estimate the video depth. Some experimental results are shown in Chapter 4 and we will make brief conclusions on Chapter 5.

Chapter 2.

BACKGROUND

Vision-based depth estimation has been discussed for many years. Many algorithms have been proposed. In this thesis, our system works on video sequences from a single static camera only and do not have binocular knowledge about the scene. In this chapter we will introduce some works about single image depth estimation. In the first section, some relevant works about single image depth estimation will be introduced. In the second section, we will introduce the algorithms proposed by Derek Hoiem in detail which is most relevant to our work.

2.1. SINGLE IMAGE DEPTH ESTIMATION

ALGORITHMS

Depth estimation from a single image is a very difficult topic. Due to the lack of depth knowledge, we must restrict the images to satisfying some conditions. In the past years, depth reconstruction can be performed based on very specific settings. Some people estimate depth using vanishing lines. In [1], Battiato classifies images as indoor image, outdoor image with geometric elements, or outdoor image without geometric elements. By detecting vanishing lines and using the information collected in the classification step, a depth map is estimated. The result is shown in Figure 2-1.

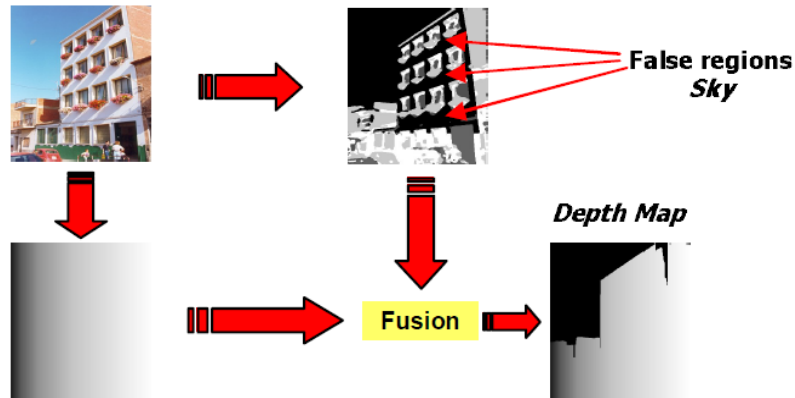


Figure 2-1 The results of Battiato's proposed system [1]

Shape from shading and shape from texture are traditional works to do 3-D reconstruction. They can be used to reconstruct relative depth. However, these methods can only be performed on uniform color and/or texture images. Some other works use known objects size to get the depth. However, there must exist some specific objects in the image and these objects should be detected before the estimation of depth.

To estimate depth with more images, people look for more general properties to connect depth and image. In [4], the authors estimate the depth scale of the scene. They analyze the Fourier transform and wavelet transform coefficients of the image and then infer the depth scale of the scene. Their result is shown in Figure 2-2.

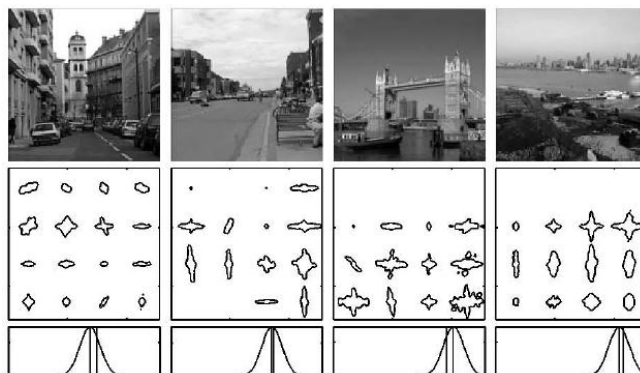


Figure 2-2 The results of Torralba's proposed system [4]

In recent years, with the progress of computer vision, people want to estimate image depth using machine learning techniques. Two types of method appear. One is to estimate depth from appearance features, while the other is to estimate depth by using image segmentation and figure/ground assignment. These methods can estimate depth in more images. In the rest of this section, we will introduce the first type of method and will leave the second type of method to the next section.

Saxena starts to estimate depth from image features. In [5], they first divide the image into small rectangular patches. In each patch, they get some cues from texture variation, texture gradient, and color. The filters they used are shown in Figure 2-3. There are two kinds of features in their algorithm. One is absolute depth feature, and the other is relative feature. The absolute depth features are calculated from the sum and the sum square of filter output in each patch. In their work, they assume the depth and absolute depth features have linear relationship. That is, the depth can be roughly estimated from the linear combination of the absolute depth features. Besides, these parameters can be trained beforehand.



Figure 2-3 The filters used by Saxena [5]

The relative features are calculated from the difference between the histograms of nearby patches. Because adjacent patches often have close depth, they penalizes the difference of depth between adjacent patches if the histograms are similar. They use MRF model to take account these two types of features. To get more global structure of the scene, they use multiple special scales of patch size and filter size. The multiple scale structure is shown in Figure 2-4 and some results are shown in Figure 2-5.

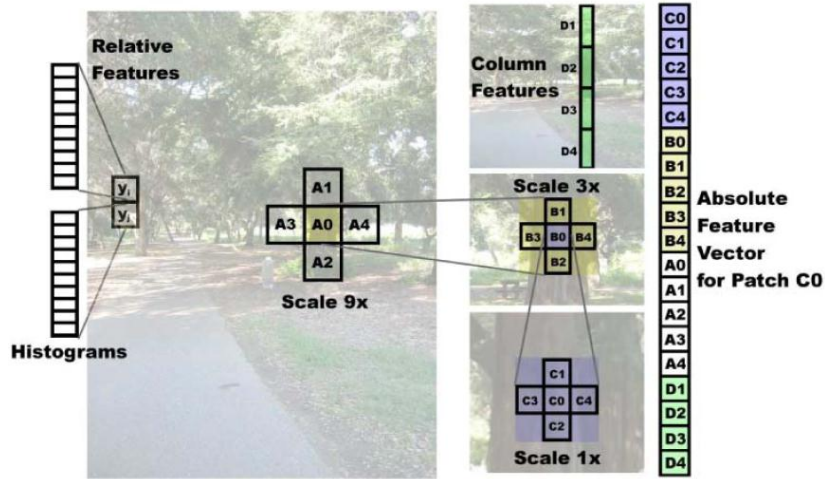


Figure 2-4 The multiple scale structure [5]

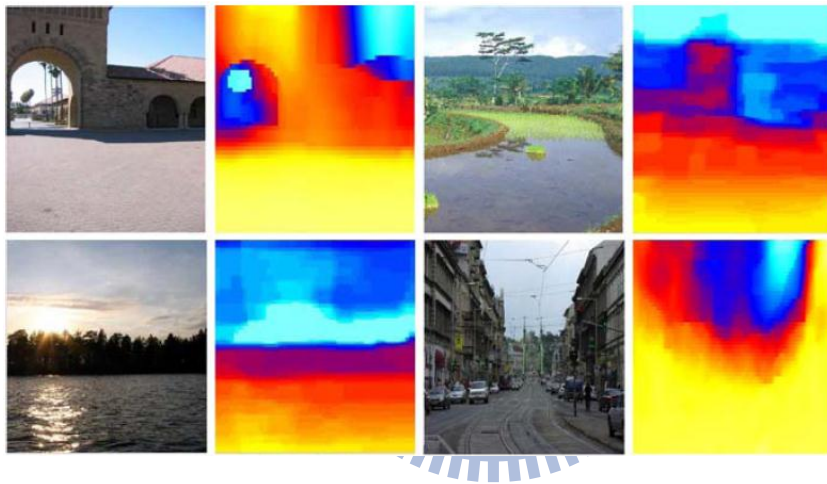


Figure 2-5 The result of Saxena's proposed system [5]

In [6], Saxena improves the depth estimation algorithm. Instead of dividing the image into rectangular patches, they use the superpixel segmentation algorithm [7] to divide the image into small uniform regions. Moreover, they assume each region is a planar surface. They want to estimate the 3-D position and orientation of each superpixel which is called the plane parameter. Having the plane parameter means we can get the depth of each pixel in the region. Like their previous work, they get some image features in each superpixel and use MRF model to consider the relationship between adjacent regions. Moreover, they consider the geometric structure between adjacent regions, which are connected structure, co-planar structure, and co-linearity.

Connected structure means two regions connect each other without a boundary between them. Co-planar structure means two regions are in the same plane so there is no edge between them. Co-linearity means two regions lie on a straight line in the 2-d image. When two nearby regions are more similar, their plane parameter should satisfy the geometric structure more.

In [8], Liu performs the semantic labeling before estimating depth. They think that directly estimating depth from image features is difficult because the relationship between appearance properties and depth in each object is different. For example, sky and water region have similar appearance features, but their relationships to depth are different. Hence, they roughly classify images into some semantic classes and train the parameters in each class. Once they get an image, they first predict the semantic label in each pixel. After that, they use the parameters in that class to estimate the depth. Some results of their algorithm are shown in Figure 2-6. Images from left to right are the original image, their semantic labels prediction, ground truth depth, and their depth prediction.

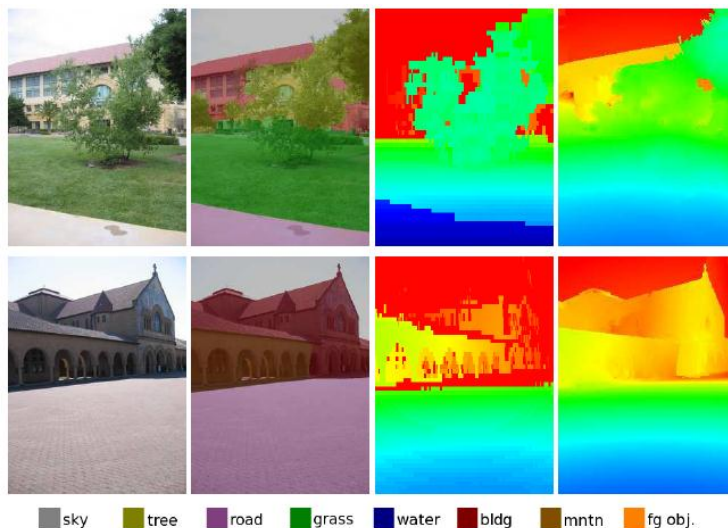


Figure 2-6 Result of Liu's proposed system [8]

Using appearance features, depth can be roughly estimated. However, the relationship between depth and appearance features is not so simple. Even after the classification of the image into semantic labels, the diversity in each class is still large. For example, there are many types of trees in the world. Different species of tree have different size, color, and texture. If we classified the image into many labels, classification accuracy in each label would be low. Hence, in this type of method, the images in the dataset could not have big diversity. That is, the images in the training data and the testing images have to be similar.

2.2. DEPTH ESTIMATION FROM OCCLUSION REASONING

Depth estimation from appearance features is difficult. Even human cannot predict depth well simply based on small-region features. In fact, we can perceive the depth usually because we know the whole structure in the scene. Besides, we can detect the objects in the image and get their position relationship. Hoiem proposed an algorithm in [9] to label the image into geometric classes. They use the classification result to find the occlusion boundaries in the image [10]. By knowing the surfaces and boundaries, they get the depth order of the image contents. In the following subsection, we will introduce the algorithms proposed by Hoiem.

2.2.1. SURFACE LAYOUT

To get the 3-D structure of the scene, Hoiem proposed an algorithm in [9] to

recover the rough surface layout of an outdoor image. They classify the image into geometric labels. Each pixel belongs to either ground plane, vertical surface, or the sky. The vertical surfaces are subdivided into planar surface facing left or right, or center and non-planar surface which is solid or porous. One classification result is shown in Figure 2-7. Different colors mean different main classes (ground, vertical, sky). The marks indicate the subclass labels of vertical regions.



Figure 2-7 The labels of Hotem's system [9]

In their work, they assume the camera axis is roughly aligned with the ground plane. They remove some pictures whose horizon is too high or too low. They find most pixels can be represented by three main classes. Because their classification is based on object's geometric orientation, it's possible that the same object belongs to different classes. The subcategories provide few geometric properties but are valuable in scene understanding. Many objects belong to the "solid" class and provide some cues for object recognition.

Like other semantic labeling task, they need some cues to estimate the geometric labels. In their work, they use location cues, color cues, texture cues and perspective cues. The detail cues are listed in Figure 2-8.

SURFACE CUES	
Location and Shape	
L1.	Location: normalized x and y, mean
L2.	Location: normalized x and y, 10 th and 90 th pctl
L3.	Location: normalized y wrt estimated horizon, 10 th , 90 th pctl
L4.	Location: whether segment is above, below, or straddles estimated horizon
L5.	Shape: number of superpixels in segment
L6.	Shape: normalized area in image
Color	
C1.	RGB values: mean
C2.	HSV values: C1 in HSV space
C3.	Hue: histogram (5 bins)
C4.	Saturation: histogram (3 bins)
Texture	
T1.	LM filters: mean absolute response (15 filters)
T2.	LM filters: histogram of maximum responses (15 bins)
Perspective	
P1.	Long Lines: (number of line pixels)/sqrt(area)
P2.	Long Lines: percent of nearly parallel pairs of lines
P3.	Line Intersections: histogram over 8 orientations, entropy
P4.	Line Intersections: percent right of image center
P5.	Line Intersections: percent above image center
P6.	Line Intersections: percent far from image center at 8 orientations
P7.	Line Intersections: percent very far from image center at 8 orientations
P8.	Vanishing Points: (num line pixels with vertical VP membership)/sqrt(area)
P9.	Vanishing Points: (num line pixels with horizontal VP membership)/sqrt(area)
P10.	Vanishing Points: percent of total line pixels with vertical VP membership
P11.	Vanishing Points: x-pos of horizontal VP - segment center (0 if none)
P12.	Vanishing Points: y-pos of highest/lowest vertical VP wrt segment center
P13.	Vanishing Points: segment bounds wrt horizontal VP
P14.	Gradient: x, y center of mass of gradient magnitude wrt segment center

Figure 2-8 The cues used in Hoiem's surface system [9]

In their perspective cues, they use vanishing line to infer the geometric structure. They first find the long lines in the image, and then find their intersections, which are possible vanishing points in the image. They classify the vanishing points into vertical vanishing points and horizon vanishing points. When more pixels in a region contribute to vertical (or horizon) vanishing points, the region is more likely to be a vertical (or support) surface.

Some cues listed before need a large spatial support to get information. Hence, how to get the appropriate region size is an issue of their work. In their algorithm, they first segment the image into superpixels. In each superpixel, they assume the pixel labels within it are the same. After that, they get some features in the superpixels and some features between nearby superpixels. Their system can roughly recognize the label of each superpixel independently. However, like other recognition works,

they need to consider the relationship between nearby regions. Instead of using MRF model, they merge the regions which are most likely to have the same label. This is because some cues, such as perspective information, can be effective only when a big region is considered. Since it may happen that two different label regions are merged, they use multiple segmentations to avoid commitment to any particular segmentation process. Because they don't know the proper segment number in each image, they use many different segmentation numbers. One of their segmentation results is shown in Figure 2-9.

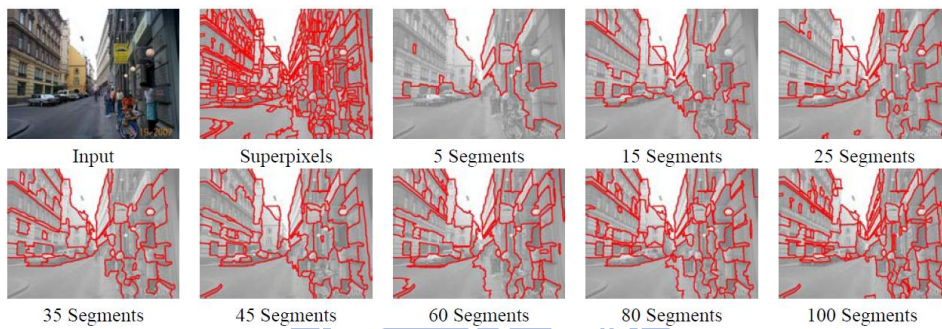


Figure 2-9 One multiple segmentation result of Hoiem's system[9]

After multiple segmentations, they use the pre-trained parameters to predict the label likelihood of each segment. Moreover, for each segment, they calculate the likelihood of being homogeneous as a weighting parameter. After that, they calculate the superpixel label likelihood by combining different segmentation results. Their surface label and likelihood estimating result is shown in Figure 2-10.

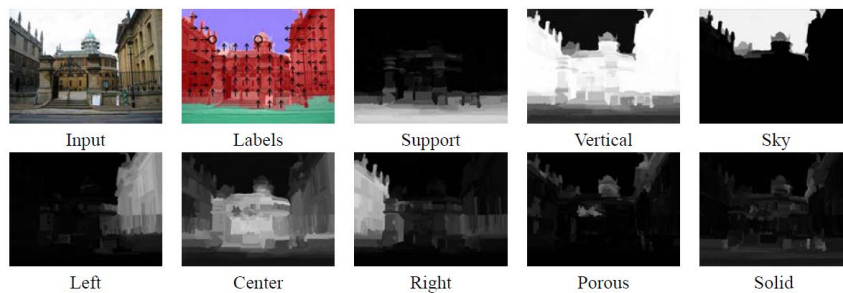


Figure 2-10 The result of Hoiem's surface layout estimation [9]

2.2.2. OCCLUSION BOUNDARY

With the help of surface layout estimation, Hoiem proposed an algorithm to recover the occlusion boundaries and depth ordering of an image. In their work, their goal is to segment the image into main objects and to know the figure/ground relationship between nearby objects. Using the vertical/ground structure, they estimate objects depth by detecting the contact points between objects and ground. Some regions are occluded by other regions. After that, they use the figure/ground relationship to know the max/min depth of those regions. One of their outputs is shown in Figure 2-11. In the left picture, the region to the left of an arrow is in front of the region to the right of the arrow.

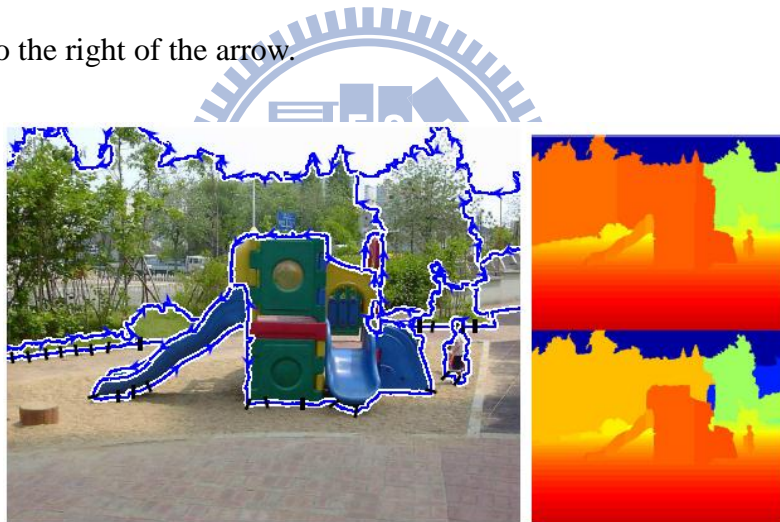


Figure 2-11 One result of Hoiem's occlusion boundary algorithm[10]

Because objects usually are not homogenous in appearance, recovering occlusion boundaries is a difficult work. Many objects are composed of many different color/texture regions and thus the cues used in many segmentation algorithms are usually not enough. In their work, they use many cues to recognize the boundaries. The cues can be classified as boundary cues, region cues, surface layout cues, and depth-based cues. The detail cues are listed in Figure 2-12.

Occlusion Cue Descriptions	Num
Boundary	7
B1. Strength: average Pb value	1
B2. Length: length / (perimeter of smaller side)	1
B3. Smoothness: length / (L1 endpoint distance)	1
B4. Orientation: directed orientation	1
B5. Continuity: minimum diff angle at each junction	2
B6. Long-Range: number of chained boundaries	1
Region	18
R1. Color: distance in $L^*a^*b^*$ space	1
R2. Color: difference of $L^*a^*b^*$ histogram entropy	1
R3. Area: area of region on each side	2
R4. Position: differences of bounding box coordinates	10
R5. Alignment: extent overlap (x,y,overall,at boundary)	4
3D Cues	34
S1. GeomContext: average confidence, each side	10
S2. GeomContext: signed difference of S1 between sides	5
S3. GeomContext: sum absolute S2	1
S4. GeomContext: most likely main class, both sides	1
S5. GeomTJuncts: two kinds for each junction	4
S6. GeomTJuncts: if both junctions are GeomTJuncts	2
S7. Depth: three estimates (log scale), each side	6
S8. Depth: diffs of S7, abs diff of first estimate	4
S9. Depth: diff of min overestimate, max underestimage	1

Figure 2-12 The cues used in Hoiem's boundary system [10]

In their approach, they use the result of their surface layout algorithm to get information about boundaries. These are very useful cues because most edges between different surface labels are boundaries. These labels can also reveal figure/ground knowledge. For example, solid regions are more likely to be in front of planar surfaces. Their experiment result demonstrates the importance of these cues.

In their algorithm, they start with an over-segmentation result of the image and then slowly remove the most unlikely edges to get the final boundaries. Their initial segmentation is produced from the watershed segmentation process with soft boundary map. Usually there are thousands of regions at the beginning and most boundaries are preserved in the edges between these regions. They use the cues listed before to predict for each edge the likelihood of being a boundary. Because many cues need a larger special support, they use a hierarchical segmentation process. At each time, they calculate the boundary likelihood of each edge and remove the most

unlikely edges until the boundary likelihood of all the remaining edges are up to a given threshold. Because regions have grown up and edges have been longer, they refine the boundary prediction and then remove the unlikely boundaries again. This process keeps going until no new region forms. The flow chart of their system is shown in Figure 2-13.

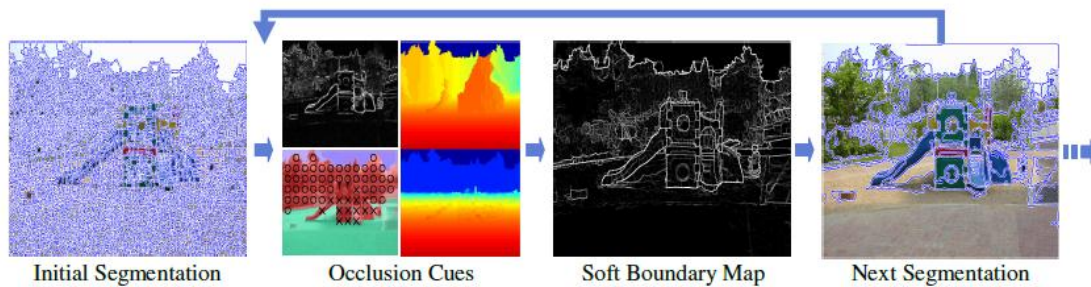


Figure 2-13 The flow chart of Hoiem's occlusion boundary system[10]

In their algorithm, they need to estimate the likelihood of boundary labels. They use a CRF (Conditional Random Field) model to enforce boundary continuity and consistency. That is, the boundary likelihoods of connected edges are related. Hence, they need to consider all possible labels of the image instead of estimating each boundary confidence alone. For example, in a junction where three edges are connected, 27 different labels can be assigned but only 5 types of labels are possible. The valid types are shown in Figure 2-14.

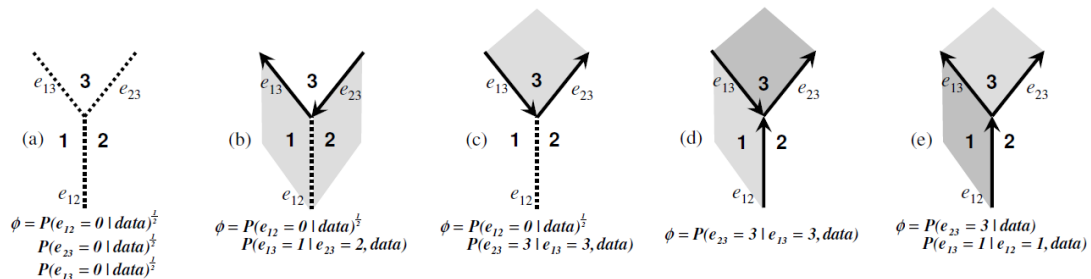


Figure 2-14 Illustration of five valid junctions [10]

In this figure, the arrow line means there is a boundary between two regions and the region to the left of the arrow is occluding the region to the right of the arrow. The

dot line means there is no boundary. The surface figure/ground relationship can also be revealed by the colors, where a darker surface is closer to the camera. They calculate the junction likelihood using the likelihoods of outgoing arrows and dot lines. Usually the likelihood of an outgoing arrow is conditioned on the data of the incoming arrow. They estimate these kinds of likelihood using some cues such as the relative angle of two adjacent boundaries. They also consider the consistency of surface labels, putting penalty to some cases such as there is no boundary between different surface labels. Since the max-product inference of their model is intractable, they use Yuille's algorithm to provide a soft-max likelihood. The effect of using CRF model is shown in Figure 2-15. Small errors are corrected by the CRF model.

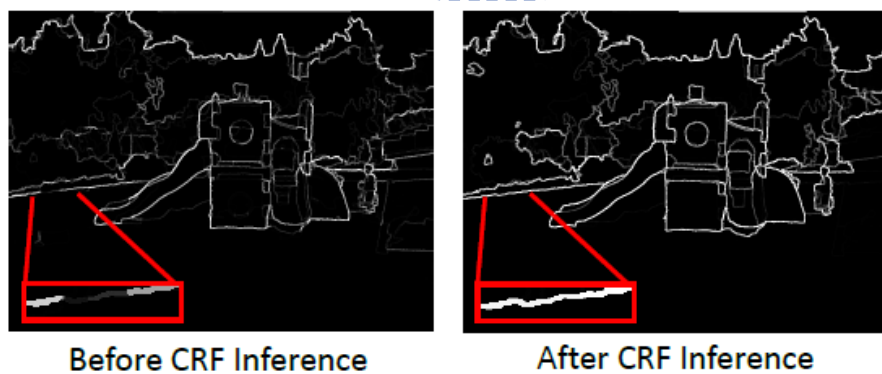


Figure 2-15 The effect of Hoiem's CRF model [10]

Chapter 3.

PROPOSED METHOD

The goal of our work is to estimate the depth map of videos taken from static cameras. Because the backgrounds in the video frames are stationary, we first estimate the depth map of the background. Using Hoiem's single-image depth estimation algorithm, we add the temporal information in it and get the depth map of the background. After that, we get the depth of the moving objects and have the video depth output. The flow chart of our system is shown in Figure 3-1. In this chapter, we will introduce the detail of our system.

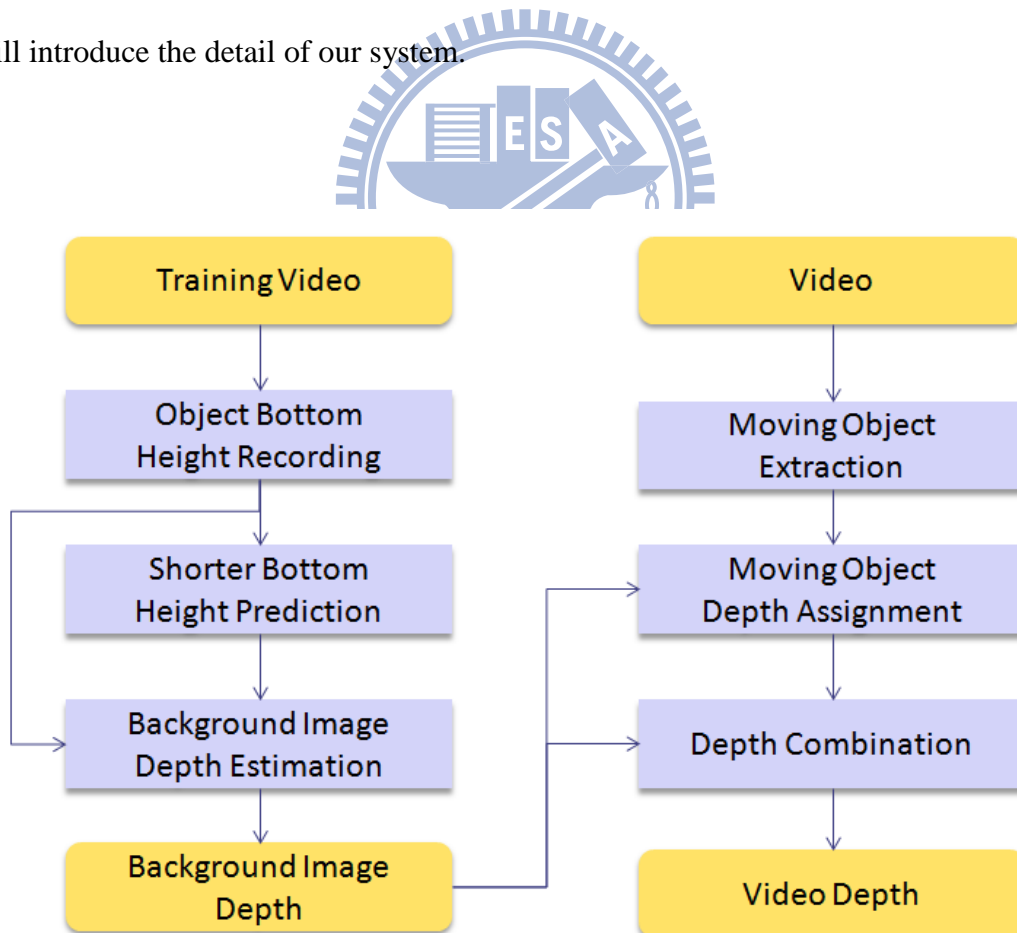


Figure 3-1 The flow chart of our system

3.1. TEMPORAL KNOWLEDGE

Video is composed of many images, so a way to have the video depth output is to estimate the depth of each image using single-image depth estimation. However, general single image depth estimation algorithms take a few minutes to process one single frame. It would be impractical to estimate the depth frame by frame. In our system, since the camera is static and the background is stationary, we only need to estimate the background depth once. However, here comes a question: what temporal knowledge can we get from the moving objects in the video to help us estimate the background depth? In many cases, objects give little knowledge when they just pass in front of the camera. On the contrary, when objects pass behind some obstacles in the image, we can detect the boundaries of obstacles more easily. Figure 3-2 shows an example. When only the background image is given, it is hard to detect the boundary within the red circle based on boundary detection algorithm. However, when a car passes through it, it becomes easier to find this boundary.



*Figure 3-2 Left: Background image with boundary which is hard to detect.
Right: Moving object passes through the boundary.*

We want to utilize moving objects to obtain temporal knowledge, but we don't want to spend too much time in it. An efficient way to detect moving objects is based on background subtraction. In our approach, we perform the background subtraction

process proposed in [11] and use the result to detect some boundaries in the background image. However, current background subtraction algorithms still cannot perform well in complex scenes. In most videos that contain complicated scenes, the background subtraction results are unstable. In the situation that a moving object has a color different from the background, we can see the boundary in the background subtraction result when the moving object passes through the back of an occluding object. However, when the color of the moving object and the background are similar, we might detect some false boundaries even if the moving object is not occluded by any object. Some examples are shown in Figure 3-3 and Figure 3-4.



Figure 3-3 When a car passes through the back of the tree, we can see the boundary in the background subtraction result last for a short time.



Figure 3-4 The man in the middle of the image walks in front of the tree, but we see the false boundary in the background subtraction result due to the similarity of color.

That is, short-term temporal information like motion history image (MHI) is not enough. We need a sequence of video frames and a way to detect the possible boundaries. An easy way to collect temporal information is to count the number of objects passing over each pixel. We can do this by summing the value of background

subtraction result. The equation is expressed in Eq. 3-1, where B_k means the k -th background subtraction image.

$$S = \sum_k B_k \quad \text{Eq. 3-1}$$

Using this process, if there is an occluding object in the image and some moving objects pass behind it, we may find that the accumulated count at a pixel on the occluding object is less than that on nearby background pixels. However, the color similarity problem still exists. We find that pixels with a dark color are more likely to get an accumulated count that is less than the true value. This might be due to the low contrast of the camera which makes the objects within the dark regions look similar to the background.

Figure 3-5 shows an example of the summation of four minutes background subtraction results. In the scene, people walk on the sidewalk and cars run on the road. The contours of the trees can be seen in this picture which can be used to infer the boundaries. However, there is also strong contrast near the tire and the chassis of the cars which is caused by the drawback of the background subtraction algorithm.



Figure 3-5 Left: The background image. Right: The summation of four minutes background subtraction results

Hence, how to make the system less sensitive to the drawback of the background subtraction process is a crucial issue of our work. Before, we counted the frequency of

objects passing through each pixel and used the difference of value as boundary cues. In fact, moving objects may pass through the occluding object from the back or from the front side. If there is no moving object passing through the front of an occluding object, the value counted on the pixels of the occluding object would be very small. Hence, if we know the appearance times of moving objects passing through the back side, we can detect the boundaries more precisely.

Although we don't have the depth of moving objects, we have another cue which is relevant to the depth. That is the bottom height of the moving object. Usually the bottom height gives the roughly object depth if the object is not severely occluded. The steps we collect the bottom height statistics are shown in Figure 3-6. First, we use the background subtraction method proposed by Barnich [11]. We use the morphological opening process to clear some noisy regions in the subtraction result. After that, we merge 10 successive frames and do the morphological closing process. The reason for using 10 frames is that there could be a big loss of moving object pixels in the background subtraction result. By combining the results of 10 frames, we can fill in some of these pixels. The selection of 10 frames is due to the fact that the period of time a moving object passes through a pixel last roughly 10 frames. Besides, the object depth usually does not have a big change within 10 frames. After the above processes, we eliminate small regions that are less than 100 pixels and merge some nearby regions. At this time we get some regions and can easily get their bottom heights. For each region we record the bottom height at all the pixels within the region.

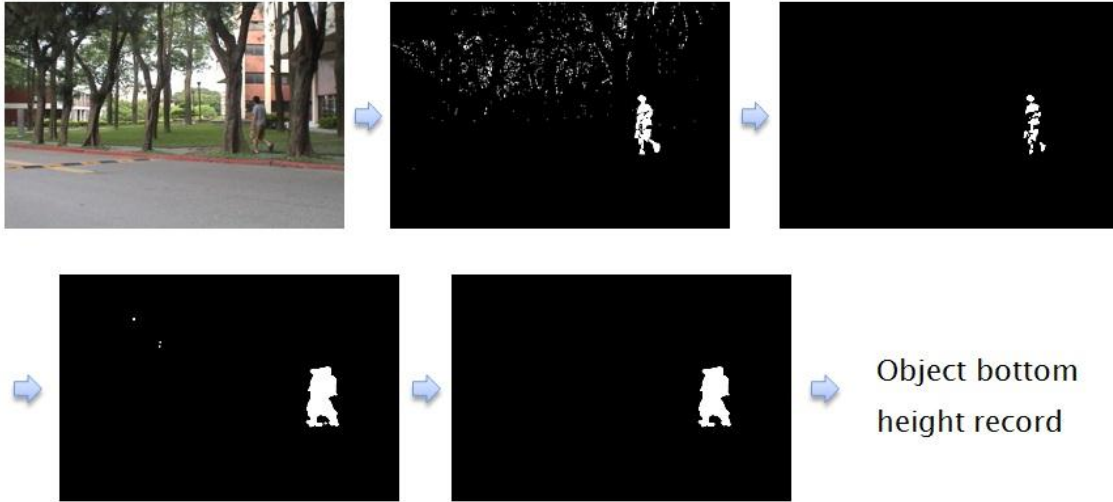


Figure 3-6 Temporal knowledge collection steps

The recording equation can be written as Eq. 3-2. S_i means the i -th statistical 2-d data recording the number of passing objects corresponding to the bottom height i in each pixel. Because our video size is 480 by 720, there are 480 possible bottom heights. In Figure 3-7, we show two examples after a period of time of recording. These two examples correspond to the bottom heights 100 and 165. The brighter region has a bigger counting number. The label n means the n -th object in the frame. Function $h(n)$ means the bottom height of the n -th object. $B_{k,n}$ means the binary image of k -th frame with the value 1 only on the pixels of the n -th object. (One example is the last image in Figure 3-6.)

$$S_i = \sum_k \sum_n \phi(i - h(n)) B_{k,n} \quad \text{Eq. 3-2}$$



Figure 3-7 The statistical data of bottom height 100 and 165

When two nearby regions have a large difference of numbers at some bottom

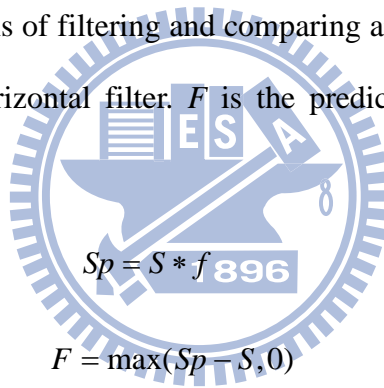
heights, there might exist an occluding object in the region that has a smaller number. Hence, we can detect the occluding object. Figure 3-8 shows an example. If we count the number of object passing through the red and blue dot at different bottom heights, we might have the statistical results like the right one. The red histogram means the appearance number statistics. On the red dot, moving objects can pass through it both on the sidewalk and the road and there are two groups in its histogram. On the other hand, on the blue dot, only those moving objects on the road can pass through it. When the moving objects walk on the sidewalk, they are occluded by the tree and we don't see their appearance on the blue dot. In this case, the curve has only one group. The big difference of the appearance on the height of the sidewalk gives the cue that there is an occluding object on the blue dot.



Figure 3-8 An example of appearance number at different height of two points

After the statistical analysis, we want to find some occluding objects in the image. A way to find the occluding objects is to predict and compare. In the prediction step, we use a horizontal filter to predict the statistics without occlusion. In the comparison step, we compare the predicted one with the observed one. Here we use the total number statistics (which is equal to summing the 480 statistical data) to explain these steps. If we count the total number of moving objects passing through

each pixel in the scene (a) of Figure 3-9, we may get the result (b) of Figure 3-9 (the lighter pixel has a larger number). In this image, we see the contour of the trees. We can detect the boundaries by taking gradients, but we may detect false boundaries on the top and the bottom of the path. Because we take the video shot on the eye level, moving objects statistics are quite uniformly distributed on the x axis in the image if there is no occlusion. Hence, we can do a horizontal filtering to ‘predict’ the appearance number at each pixel without occlusion. The predicted image is shown in (c) of Figure 3-9. By comparing the difference of the observed appearance number and the predicted one, we can find the regions where the appearance number are shorter and are likely to be the occluding region. The detection result is shown in (d) of Figure 3-9. The equations of filtering and comparing are written in Eq. 3-3 and Eq. 3-4. The label f is the horizontal filter. F is the predicted likelihood of occluding regions.



$$Sp = S * f \tag{Eq. 3-3}$$

$$F = \max(Sp - S, 0) \tag{Eq. 3-4}$$

However, using the total number statistics usually produces many false alarms due to the aforementioned drawback of the background subtraction process.

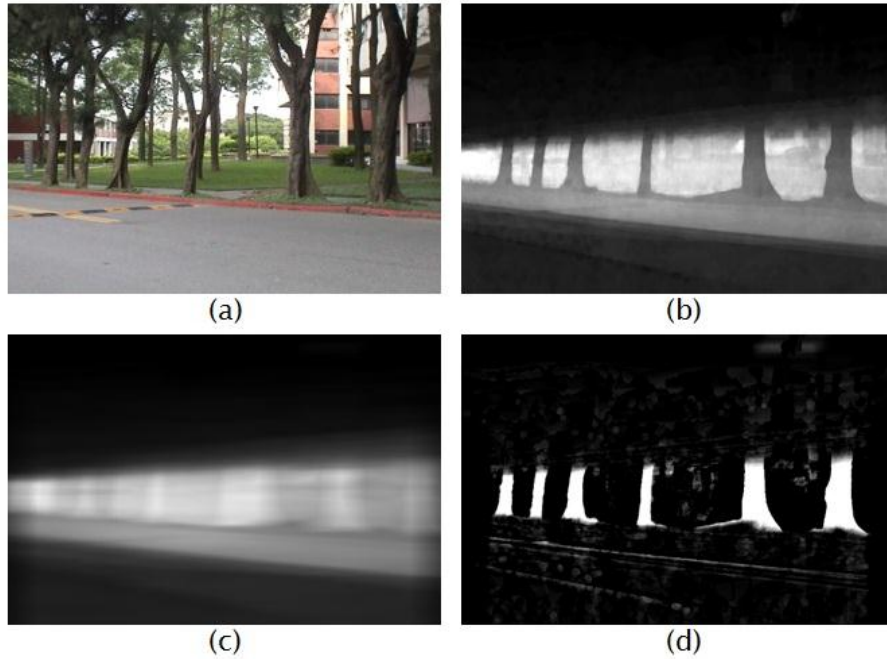


Figure 3-9 The filtering process to find the occluding object region

We do the prediction and comparison processes separately at each bottom height to erase the background subtraction similarity problem. We use the same predicting filter but adjust the way of comparison. Only when the predicted number is a few times larger than the observed number, we think that pixel is on an occluding object and count the number of difference. The equations are written in Eq. 3-5 and Eq. 3-6.

$$Sp_i = S_i * f \quad \text{Eq. 3-5}$$

$$F_i(x, y) = \begin{cases} Sp_i(x, y) - S_i(x, y), & Sp_i(x, y) > 5S(x, y) \\ 0, & \text{else} \end{cases} \quad \text{Eq. 3-6}$$

If we perform these steps over the statistics in Figure 3-8, we would get the result shown in Figure 3-10. The histograms from left to right are the object bottom height histogram, predicted object bottom height histogram, and the occluded object bottom height histogram. In this example, we can detect the tree because it occludes the pedestrian on the sidewalk.

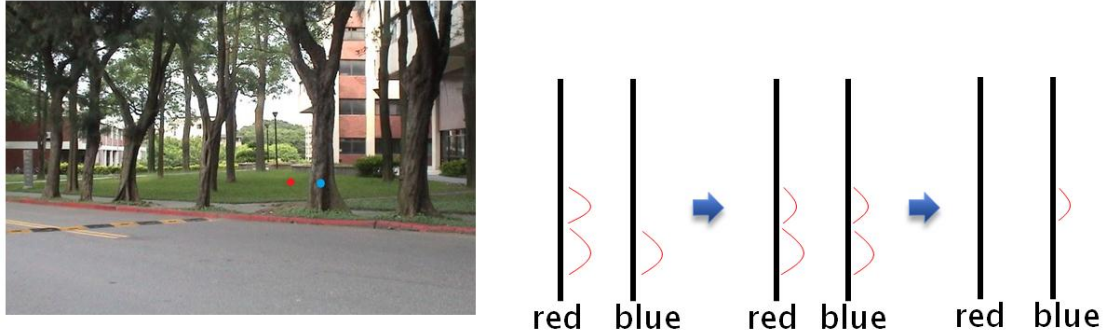


Figure 3-10 The histograms of two points with the prediction and comparison steps

Because moving objects do not always walk parallel to the camera, we average the statistical result at each bottom height with nearby bottom heights and then do the filtering separately. There exists one problem, that is, moving object is not fully occluded by the occluding object. Its bottom is occluded but we can still see the remaining part. In this case, the bottom height jumps from one value to another value and makes both side have a big change at some heights. To deal with this situation, we also consider the total number change at nearby regions. The occluding object possibility would be suppressed if the total number change is small.

Now we have two kinds of statistics. One is the object bottom height statistics and the other is the occluded object bottom statistics. Using these statistics, we can get the depth range at each pixel. From the object bottom height statistics, at each point we know the roughly depth of objects that might occlude it. Hence, we know that the depth of this point is larger than the depth of these objects. From the occluded object bottom height statistics, at each point we know the roughly depths of the occluded objects. Hence, we know that the depth of this point is smaller than the depth of these objects. Here we use the maximum 2% of object bottom height as the minimal depth and use the minimum 2% of occluded object bottom height as the maximal depth. Combing with the maximal depth range of ground, we get the maximal and minimal depths, as shown in Figure 3-11.

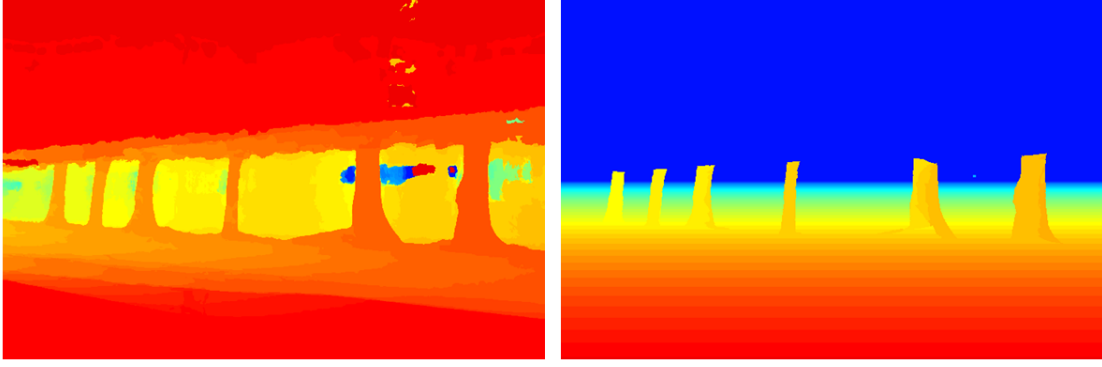


Figure 3-11 The depth range. Left: Minimal depth. Right: Maximal depth.

In the occluded object bottom height statistics, there are 480 statistical 2-d data recording the number of occluded objects at each bottom height. If we sum up the 480 2-d data, we can get the result image in (d) of Figure 3-12, which shows the predicted likelihood of occluding region. The detection results using other statistics are shown in Figure 3-12, including the image (a) that counts the total number at each pixel (refer to Eq. 3-7), the image (b) that averages the objects bottom height passing through each pixel (refer to Eq. 3-8), and the image (c) that combines the first two statistics (refer to Eq. 3-9).

$$S = \sum_k \sum_n B_{k,n} \quad \text{Eq. 3-7}$$

$$S = \frac{\sum_k \sum_n h(n) B_{k,n}}{\sum_k \sum_n B_{k,n}} \quad \text{Eq. 3-8}$$

$$S = S_a S_b \quad \text{Eq. 3-9}$$

Statistical method using Eq. 3-7 is likely to be affected by the aforementioned color similarity problem. Statistical method using Eq. 3-8 can deal with this color similarity problem. However, it is likely to be affected by the noise in some regions where objects rarely pass through. Statistical method using Eq. 3-9 is to find the interaction of the result using Eq. 3-7 and Eq. 3-8. It produces better result than the

other two methods. The statistical method that produces Figure 3-12 (d) spends the longest time but is the most stable one. Using this statistics, we can roughly get the maximal depth range.

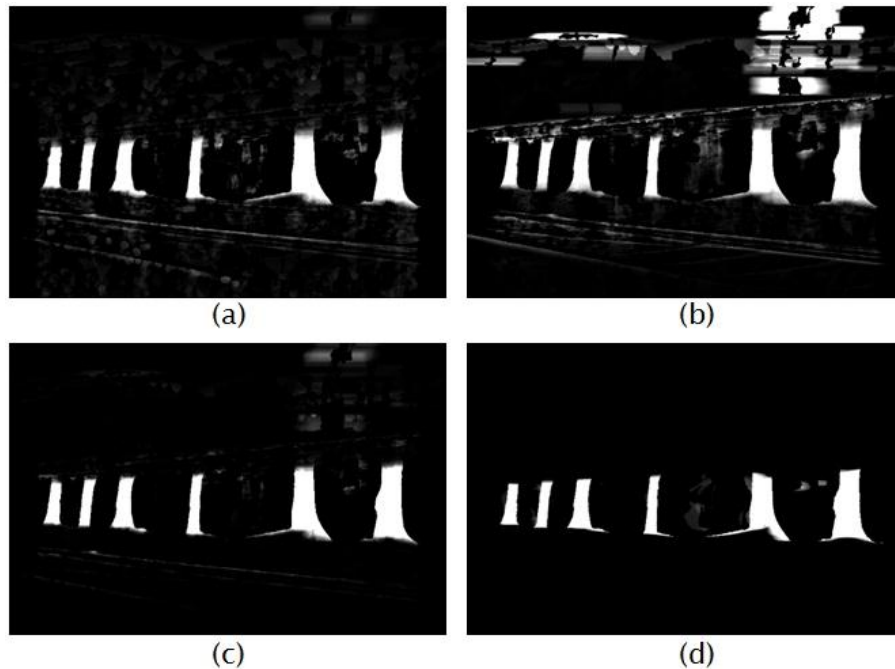


Figure 3-12 The results of occluding objects detection using different statistics

3.2. BACKGROUND DEPTH ESTIMATION

After we collected the moving object information, we want to estimate the depth of the background image by combining single-image cues with it. We do this work by extending Hoiem's algorithm [10]. Here, let's review Hoiem's single-image depth estimation algorithm first. To estimate the depth of an image, they need to know the surface orientations and the boundaries in the image. They use their surface estimation algorithm to estimate the surface orientations and use them as cues to recover the occlusion boundaries in the image. After that, they combine the two results to get the relative depth map. In our algorithm, we collect some useful

temporal knowledge and add it in some modules of Hoiem's algorithm. The flowchart of Hoiem's algorithm and the modules that we have added in the temporal knowledge are shown in Figure 3-13. In this section, we will introduce the adjustments we have made over the flowchart.

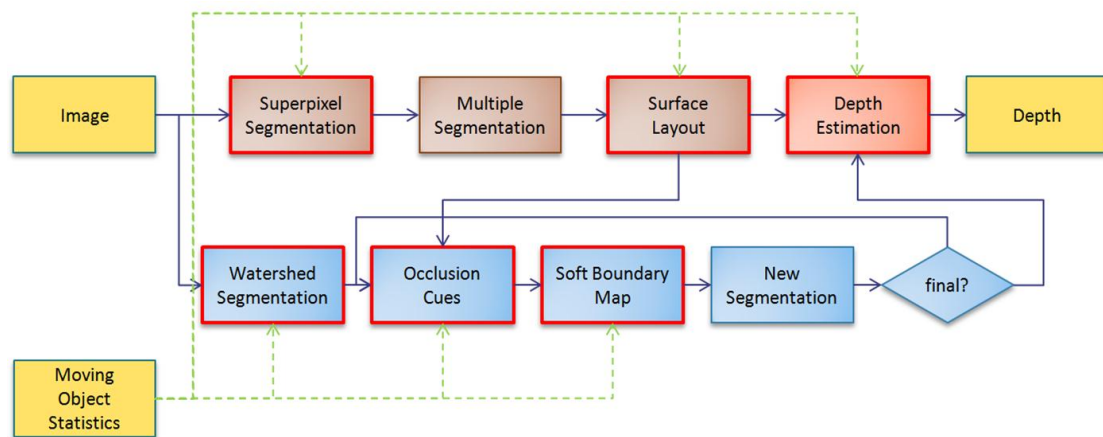


Figure 3-13 The flowchart of Hoiem's algorithm with some blocks adding temporal knowledge

In their work, they start with a watershed segmentation process over the image. Although they segment the image into thousands of regions, their boundaries might be a little displaced from the actual places. With the help of temporal knowledge, we want to preserve the precise boundaries in the initial segmentation. Since the watershed segmentation is based on image gradients, we make some modification over the image gradients to adjust the segmentation result. In our statistical images ((b) and (d) in Figure 3-9), we see the contours of the occluding objects. Taking gradient to each single one would yield some false boundaries. Hence, we take gradients to both of them and find their intersection. We get the image with a large value at the pixels on the boundaries. We add this result to the original gradient and then do the watershed segmentation. The gradient images and the segmentation results are shown in Figure 3-14 and Figure 3-15. From the segmentation images we can see that the watershed segmentation does better on detecting the boundaries of occluding objects.

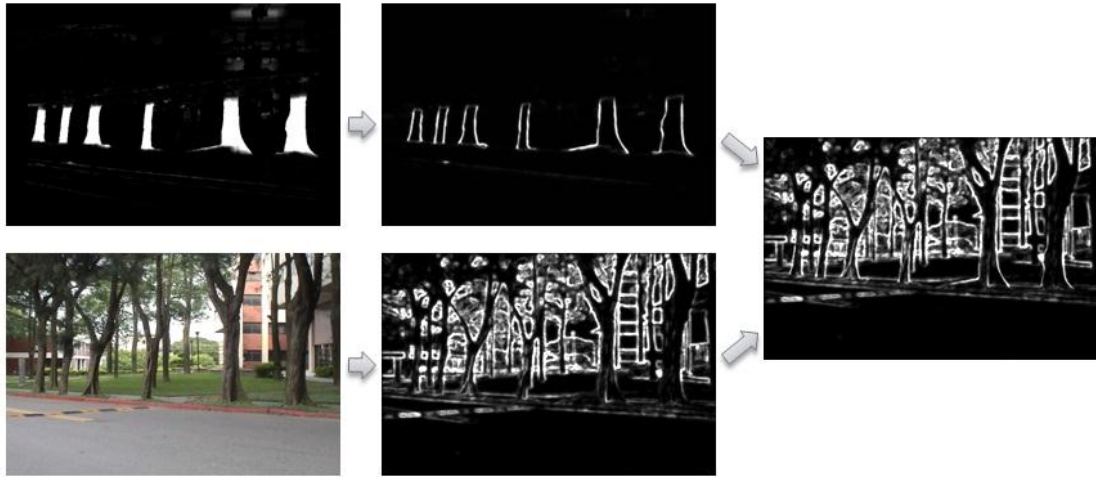


Figure 3-14 The gradient result combining the original gradient result and the gradient of temporal statistical images

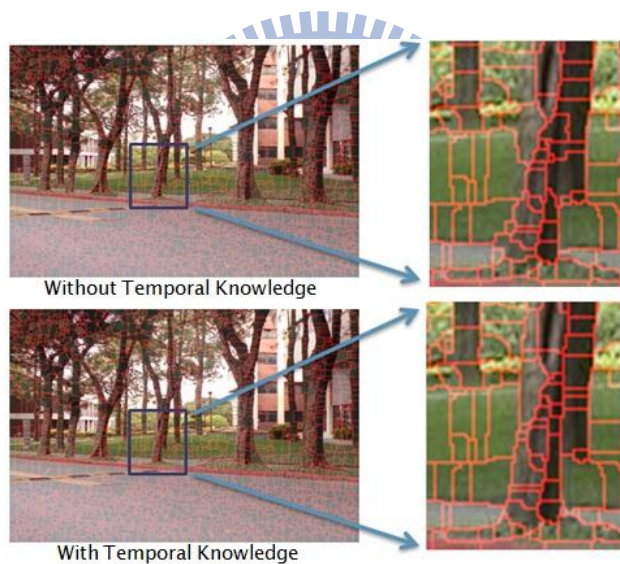


Figure 3-15 The comparison of the initial segmentation with and without using the temporal knowledge.

Having this initial segmentation, we want to get the boundary likelihoods. Beside the cues used in Hoiem's algorithm, we add in the temporal cues. First we get the statistics of each small region. The statistics include the existent part and the shorter part. After that, we get the boundary likelihoods between nearby regions by checking

whether the shorter part in one region is in the existent part of another region. The boundary likelihood is positively related to the counting numbers and is negatively related to the region sizes. The equation can be written as below. Function $p(x,y,k)$ returns 1 when the segment label of pixel (x,y) is k ; otherwise, it returns 0. $L_{a,b}$ is the estimated likelihood of two nearby regions a and b . Sk_i means the i -th object bottom height statistics of region k . Fk_i means the i -th occluded object bottom height statistics of region k . The function g adjusts the region size weight.

$$Sk_i = \sum_{x,y} p(x,y,k) S_i(x,y), \quad \text{Eq.3-10}$$

$$Fk_i = \sum_{x,y} p(x,y,k) F_i(x,y) \quad \text{Eq.3-11}$$

$$L_{a,b} = \sum_{i=1}^{480} \frac{Sa_i \circ Fb_i + Sb_i \circ Fa_i}{g(\sum_{x,y} p(x,y,a))g(\sum_{x,y} p(x,y,b))} \quad \text{Eq.3-12}$$

We calculate the likelihoods and add them to the original likelihoods to make the boundaries more likely to be preserved. When regions merge, instead of calculating the boundary likelihoods by using the composed edges, we calculate the statistics of the new region by combing the statistics of the composed small regions and then compute the boundary likelihood of nearby regions. This can prevent the situation that the occluding and occluded regions merge across weak edges. Figure 3-16 shows an example to show how the temporal knowledge works.

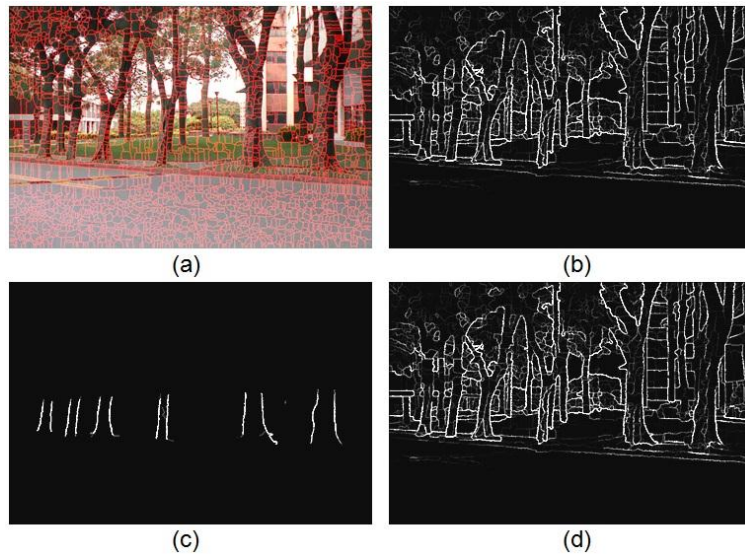


Figure 3-16 (a) *Initial segmentation.* (b) *Boundary likelihoods from single-image features.* (c) *Boundary likelihoods from temporal cues.* (d) *Boundary likelihoods using both single image cues and temporal cues.*

Using the temporal cues, boundaries where objects often pass can be preserved. Sometimes, however, occluded regions near the occluding objects have wrong surface labels and may yield wrong segments if their color difference with respect to the occluding objects is not big enough or their region size is too small. We check this problem in Hoiem's algorithm and find there are two main reasons. One is due to the wrong initial segmentation of the surface layout algorithm and the other is due to the wrong spatial support in multiple segmentations.

In their surface layout algorithm, they start with superpixel segmentation and yield hundred of regions. This segmentation is fast but not very accurate. Sometimes, it merges different label regions into one single region. This would cause wrong surface labels. Although the segmentation method used in their occlusion boundary algorithm is more precise, it doesn't work well in the surface layout algorithm. This might due to the randomness of multiple segmentation method used in the surface layout algorithm. Moreover, it's hard to make adjustment in the superpixel segmentation algorithm, which makes decision to merge regions by checking whether

the color difference between the mean of nearby regions is smaller than the variability in each single region. Hence, we only make adjustment in the original background image over which we add some value in the occluding object regions to force the separation of the occluding and occluded regions.

The wrong spatial support is caused by the merging of different label regions in their multiple segmentation process. Usually this happens at those small occluded regions that merge with occluding objects. We find that in these small regions, the single region surface layout estimation often does better prediction than the value of multiple segmentation result. Hence, we increase the weighting of the single region surface layout estimation in those regions near the occluding objects.

The ground likelihood prediction comparison is shown in Figure 3-17. Picture (a) shows the ground likelihood prediction of Hoiem's algorithm. Picture (b) shows the ground likelihood prediction after our adjustment. The likelihoods of regions between the trees after adjustment are more accurate. Hence, they would not be mistakenly classified as vertical regions in the final result.



Figure 3-17 The ground likelihood comparison

After a repetition of merging process, we get the final segmentation result. We need to assign the depth to each region. For those unconnected vertical regions which are occluded by the same region, we calculate their similarity. If they are similar

enough, we think they are the same object and we calculate their depth simultaneously. This process is due to the fact that some vertical occluded regions are separated by some occluding objects. If we calculate their depth separately, they might get different depth values due to the displacement of contact point with the ground. By combining them together, the result looks more accurate.

To know the depth range in each region, besides the figure/ground relationship used in Hoiem's algorithm, we have obtained the maximal and minimal depths information in some regions by using the temporal statistics. The maximal depth is gotten from the minimal appearance height of the shorter part in each region. The minimal depth is get from the maximal appearance height of the existent part in each region. We use these features to constrain the depth range of some objects and to get better object depth estimation.

One example of background depth estimation is shown in Figure 3-18. We get the segmentation result in (a) and estimate the depth in (b). The result of Hoiem's algorithm which uses only single-image features is shown in (c) and (d). In this picture, we get some wrong estimation results mainly on the leaves. This is due to the fact that the upper part of the training video usually has deeper depth. Hence, it's still difficult to get accurate depth estimation result in this kind of scene. However, since we have obtained some useful depth information over some regions, such as the trunk of the trees, based on the temporal knowledge, we can do better depth estimation over those regions.

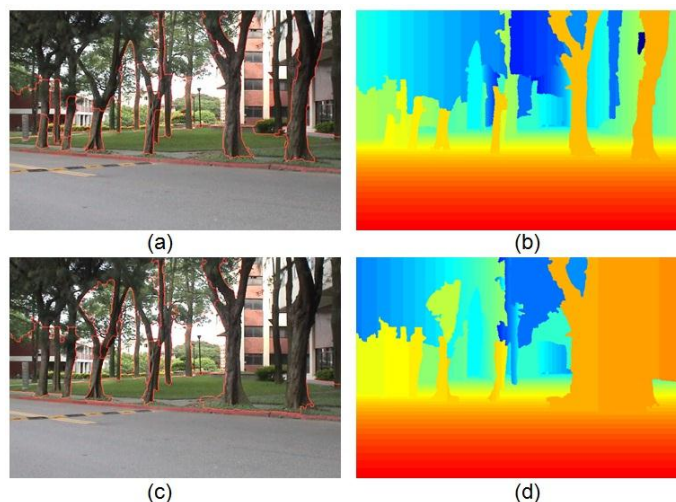


Figure 3-18 Comparison of background depth estimation

3.3. DEPTH OF VIDEO CONTENTS

After we have gotten the depth of the background image, we aim to estimate the depth of the video contents. The video used here can be the same or different from the video we used to collect temporal knowledge. We want to fast find the moving objects and assign them the correct depth. Hence, we use the background subtraction method to identify the moving objects and assign them the appropriate relative depth.

To assign the right depth to moving objects, we need to know the rough positions of them. After doing background subtraction, we apply morphology processes to clear noisy results and to merge nearby pixels. Here we get the foreground regions and find the bottom point of each object. However, we can't directly treat the bottom point as the contact point on the ground. This is because the moving object may be occluded by some other occluding objects, as shown in Figure 3-19. Even if the moving object is not occluded, the instability of the background subtraction method may also cause the bottom point to be different from the actual contact point. Hence, we need to do some tracking to better estimate the depth of the moving object.

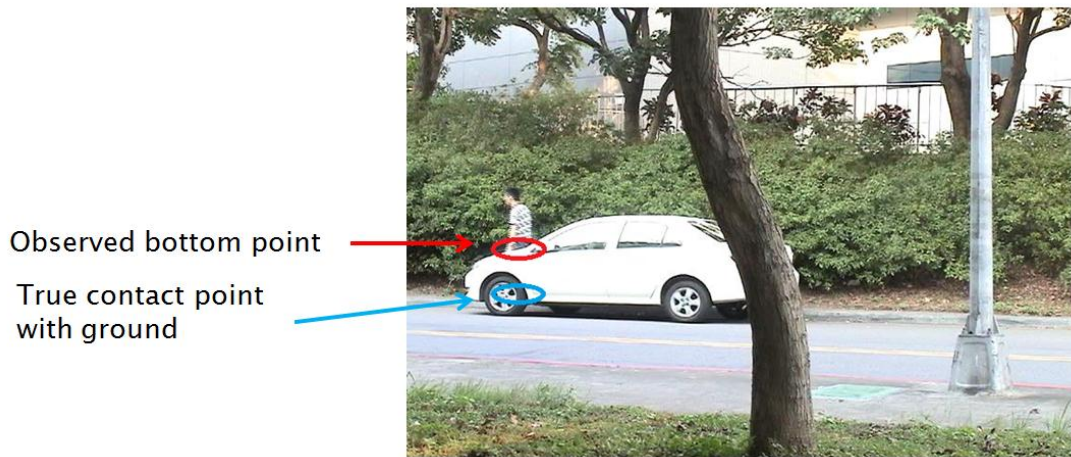


Figure 3-19 Object's bottom point is not always equal to the contact point on the ground

For each moving object, we know it is occluded or not by checking whether there are vertical regions near its bottom point. Besides, we can find whether there are objects appear around the position in the previous frame. If the object is not occluded and we have found its depth in the previous frame, we estimate its depth based on its bottom point and its previously estimated depth. If the object is occluded and we have found its depth in the previous frame, we estimate its depth based on the previously estimated depth. If the object is not occluded and we cannot find its depth in the previous frame, we estimate its depth based on the bottom point. If the object is occluded and we cannot find its depth in the previous frame but only know its maximal and minimal depths. In this case, we estimate its depth by averaging the depth of the moving objects that have passed through this region. This is because objects usually have similar paths. Hence, we can record the depth of each pixel in advance and propagate the recorded depth information to those objects with unknown depth.

In our method, we do tracking by recording the bottom height, top height, and the location on the x-axis in the image. We do not record the properties of the object

like shape or color. When an object is occluded by an occluding object, using the bottom height may make some mistakes. Hence, we relax the bottom height restriction. In our tracking data, we clear the data when the object doesn't appear for a period of time. This happens when some objects get fully occluded for a short time and then appear again.

Some situations which can't be solved in background subtraction could affect our result. One problem is the inter-occlusion among moving objects. In background subtraction method, the algorithm identifies which pixels are static and which pixels are non-static. The algorithm doesn't separate pixels of different objects. Hence, in our algorithm, it is possible that different moving objects are grouped into a single object. In this situation, the depth of the object on the back would be underestimated. A possible way to solve this problem is to perform more accurate tracking. On the opposite, we may get broken foreground detection result when the color of the object is similar to that of the background. In this case, our algorithm doesn't perform well.

When we estimate the depth of each moving object, we need to check whether it is deeper than its background which is an impossible case. Hence, we use the mean background depth in this region as the upper limit. Moreover, we find that there could be tiny holes between the occluding objects and moving objects when some moving objects pass through the back of occluding objects. To solve this problem, after we have gotten the depth of the moving objects, we perform some morphology processes over these vertical background regions with shallower depth to get a proper depth estimation.

Chapter 4.

EXPERIMENTAL RESULTS

In this chapter, we will show and discuss our experimental results. In our experiments, the first stage is to capture the video. To show the assistance of temporal knowledge in our algorithm, we chose the scenes that have some occluding objects that block moving objects behind them. To avoid some situations that might decrease the background subtraction performance, we chose the scenes that do not have huge luminance change and dramatically shaking trees. We took some videos in the campus. The background images of two videos are shown in Figure 4-1. In each scene, we took the video for about four minutes. There are about 25 objects pass through within the period. The occluding objects in the first scene are the trees near the road. Moving objects mainly move on the road or on the sidewalk. The occluding objects in the second scene are the tree, the streetlamp, and the car. Moving objects move on the road or on the two sidewalks of the road. The camera is the DCR-TRV60 digital video camera with the frame rate of 29 frames per seconds.



Figure 4-1 The background images of our demo videos

Over the video, we perform background subtraction. Here, we use the ViBe program proposed in [11]. We choose the default parameter setting to run the program. It takes about five and half minutes to run the background subtraction algorithm over the 4-min videos. In the result, there are some holes in the moving object regions and there are some false alarms over the leave regions. Examples of the background subtraction results are shown in Figure 4-2.

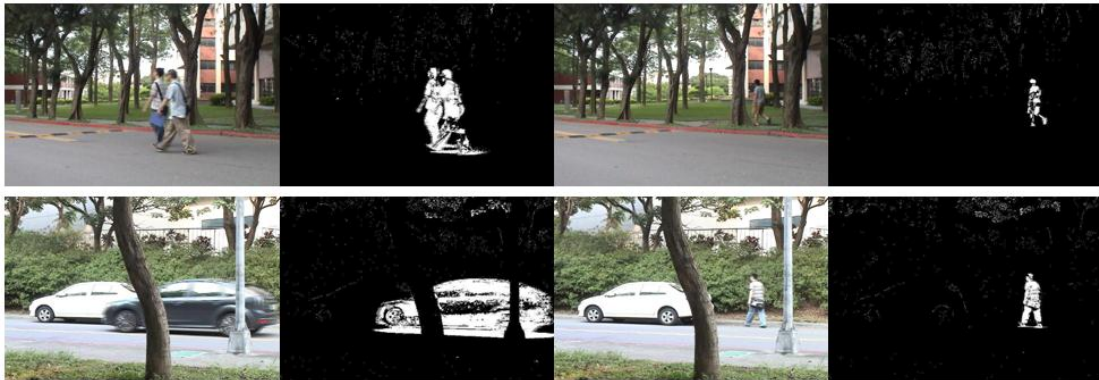


Figure 4-2 Some background subtraction results

Having the background subtraction result, we do the morphology processes and collect the temporal knowledge. This step spends a lot of time because we have thousands of frames. Even only performing the opening and closing operations would takes a few minutes. After that, we combine the temporal knowledge with Hoiem's algorithm to get the depth estimation of the background image. Here, we use a rule-based approach to adjust the boundary likelihood. The results are shown in Figure 4-3.

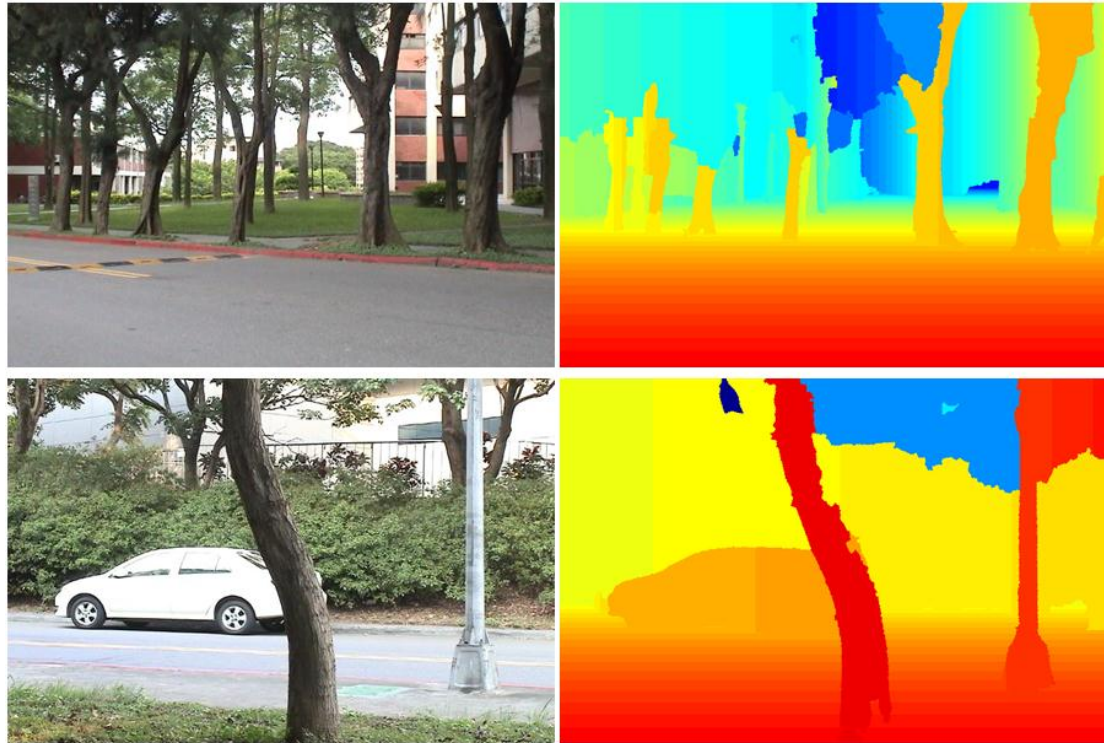


Figure 4-3 Background depth estimation result

With the background image depth estimation, we assign depth to the moving objects and then get the relative depth of the video contents. We compare our results with the original single-image estimation results. Here we use Hoiem's algorithm to estimate the depth of each image separately. The results are shown in Figure 4-4.

Compared with the results using Hoiem's method, our algorithm does better at a few places. First, the boundaries with objects passing are more likely to be preserved. For example, the trunks of trees in the first video have occluded some moving objects. Their depths are better estimated. Second, we can avoid the depth discontinuity on the background, which can be shown is the second video. The groves in the image are separated by some occluding objects. Using our algorithm, we can treat them as continuous objects that have continuous depth. Third, the depths of moving objects are better estimated. In fact, using Hoiem's method to estimate the depth of each frame is very inefficient. Their estimation results are usually inconsistent from frame

to frame.

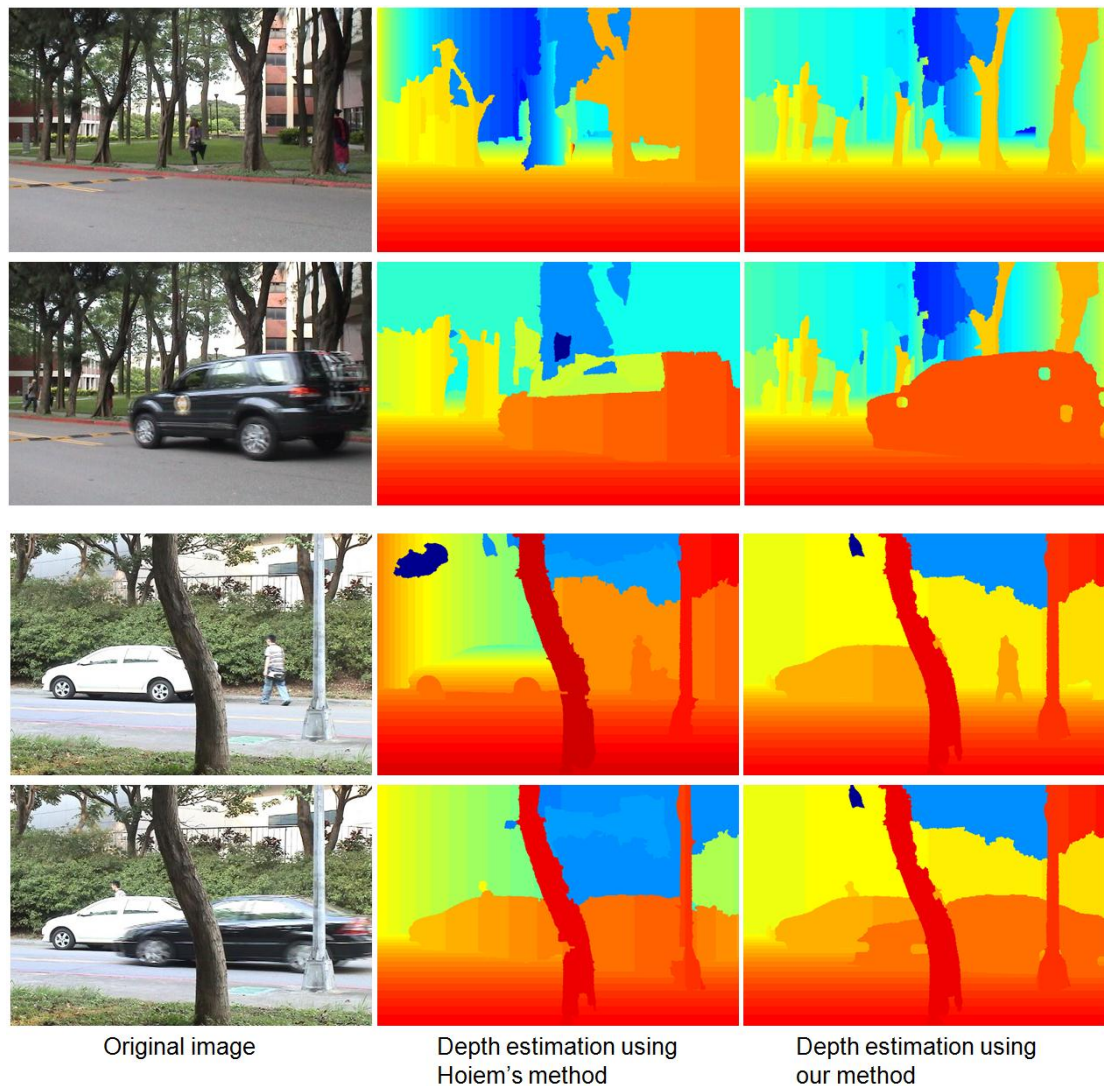


Figure 4-4 Video depth estimation result

There are some restrictions in our system. First, if there is no occluding object that occludes some moving objects in the video, we cannot obtain some useful temporal information for depth estimation. Second, our system is based on the result of background subtraction. Hence, it won't perform well if the background subtraction result is poor. Moreover, in our method, we do not use specific object knowledge and we do not use complicated tracking technique. It would be better if we can take into account more object knowledge and adopt more advanced tracking techniques in the future.

Chapter 5.

CONCLUSIONS

In this thesis, we proposed a method to estimate the depth of videos taken from static cameras. We accomplish this by first estimating the depth of background image and then assigning the depth to moving objects to get the final depth estimation of the video contents. Compared with single-image depth estimation, we concern the temporal information offered by the moving objects and the obstacles that might occlude the moving objects. Our algorithm can provide useful information to other processes, like video surveillance and video synthesis. For surveillance systems, we can know whether a moving object is occluded by the obstacle and use the information as prior knowledge. For video synthesis, we can add some synthesized moving objects into the video and know whether some part of the moving object should be occluded by some occluding objects in the scene.

REFERENCES

- [1] S. Battiato, S. Curti, M. L. Cascia, M. Tortora and E. Scordato, “Depth Map Generation by Image Classification”, in *Proc. SPIE, Three-Dimensional Image Capture and Applications VI, 2004*, vol. 5302, pp. 95-104.
- [2] B. J. Super and A.C. Bovik, “Shape from Texture Using Local Spectral Moments” , *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 333–343, April 1995.
- [3] Y. G. Leclerc and A. F. Bobick, “The Direct Computation of Height from Shading,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp.552-558, June 1991.
- [4] A. Torralba and A. Oliva, “Depth Estimation from Image Structure”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1226–1238, September 2002.
- [5] A. Saxena, S. H. Chung and A. Y. Ng, “3-D Depth Reconstruction from a Single Still Image,” *International Journal of Computer Vision*, vol. 76, no.1, pp. 53-69, 2008.
- [6] A. Saxena, M. Sun and A.Y. Ng, “Make3D: Learning 3-D Scene Structure from a Single Still Image,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, May 2009
- [7] P. Felzenszwalb and D. Huttenlocher, “Efficient Graph-Based Image Segmentation,” *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.
- [8] B. Liu, S. Gould and D. Koller, “Single Image Depth Estimation from Predicted Semantic Labels,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp.1253-1260, June 2010.
- [9] D. Hoiem, A. A. Efros and M. Hebert, “Recovering Surface Layout from an Image,” *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151-172, 2007.
- [10] D. Hoiem and A. A. Efros, “Recovering Occlusion Boundaries from an Image,” *International Journal of Computer Vision*, vol. 91, no. 3, pp. 328-346, 2011.
- [11] O. Barnich and M. V. Droogenbroeck, “ViBe: A Universal Background Subtraction Algorithm for Video Sequences,” *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709-1724, June 2011.