12-6-2016 12:00 AM

Color Separation for Background Subtraction

Jiaqi Zhou
The University of Western Ontario

Supervisor
Olga Veksler
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Abstract

Background subtraction is a vital step in many computer vision systems. In background subtraction, one is given two (or more) frames of a video sequence taken with a still camera. Due to the stationarity of the camera, any color change in the scene is mainly due to the presence of moving objects. The goal of background subtraction is to separate the moving objects (also called the foreground) from the stationary background. Many background subtraction approaches have been proposed over the years. They are usually composed of two distinct stages, background modeling and foreground detection.

Most of the standard background subtraction techniques focus on the background modeling. In the thesis, we focus on the improvement of foreground detection performance. We formulate the background subtraction as a pixel labeling problem, where the goal is to assign each image pixel either a foreground or background labels. We solve the pixel labeling problem using a principled energy minimization framework. We design an energy function composed of three terms: the data, smoothness, and color separation terms. The data term is based on motion information between image frames. The smoothness term encourages the foreground and background regions to have spatially coherent boundaries. These two terms have been used for background subtraction before. The main contribution of this thesis is the introduction of a new color separation term into the energy function for background subtraction. This term models the fact that the foreground and background regions tend to have different colors. Thus, introducing a color separation term encourages foreground and background regions not to share the same colors. Color separation term can help to correct the mistakes made due to the data term when the motion information is not entirely reliable. We model color separation term with $L1$ distance, using the technique developed by Tang et.al. Color clustering is used to efficiently model the color space. Our energy function can be globally and efficiently optimized with graph cuts, which is a very effective method for solving binary energy minimization problems arising in computer vision.

To prove the effectiveness of including the color separation term into the energy function for background subtraction, we conduct experiments on standard datasets. Our model depends on color clustering and background modeling. There are many possible ways to perform color clustering and background modeling. We evaluate several different combinations of popular color clustering and background modeling approaches. We find that incorporating spatial and motion information as part of the color clustering process can further improve the results. The best performance of our approach is 97% compared to the approach without color separation that achieves 90%.

Keywords: background subtraction, graph cuts, color separation, energy optimization

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Olga Veksler, who gave me great help by providing me with necessary materials, advice of great value and inspiration of new ideas. She walked me through all the stages of the writing of this thesis. It is her suggestions that draw my attention to a number of deficiencies and make many things clearer. Without her strong support, this thesis could not been the present form.

I would like to express my sincere thanks to Dr. Yuri Boykov, whose lectures have a great influence on my study in computer vision and image analysis. I feel grateful that I had the change to attend his lectures. The clear and detailed ways he explained a problem impressed me a lot.

Special thanks are given to the members of my examining committee, Dr. John Barron, Dr. Robert Mercer and Dr. Jagath Samarabandu.

I would like to thank Dr. Lena Gorelick for her valuable and helpful suggestions to my thesis. She taught me how to explain a problem in a more comprehensive way.

I also owe my heartfelt gratitude to my friends and other members in our vision group, who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis. I want to give special thanks to Meng Tang who did the prior work of this thesis. He answered my questions patiently when I came to him for help.

I would extend my thanks to the help and support of the staff members in the main office and system group.

I especially appreciate the support of my parents, who give me continuous encouragement and love through all these years.

Contents

List of Figures

List of Tables

Chapter 1

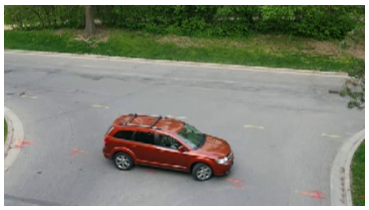
Introduction

1.1 Background Subtraction

Background subtraction is a vital step in many computer vision applications and it is mainly used for detecting moving objects in videos from static cameras. The rationale of this approach is to segment out foreground objects from their background in a video sequence by using the difference between the current image and a background image (see Figure 1.1). Here the background image represents the scene without moving objects. Background subtraction is composed of two distict steps, background modeling and foreground detection. These two steps are performed successively. In the background modeling, a background model is estimated in the beginning of the algorithm and usually it is updated regularly during the deployment of the algorithm. The need to update the background is due to multiple factors. For example, the illumination in the scene may change due to the change in the weather conditions, or due to the change in time of the day. Also, if an object was moving but then stops moving for a sufficient long time (for example a car that has been parked), then it should be considered as part of the background. Thus a reliable background model should adapt with time to the changing circumstances in the scene. In foreground detection, a decision is made about whether a pixel fits the background model or not. The result is ususlly fed back to background modeling, so that no foreground pixels are used when constructing the background model.

Many background subtraction approaches have been proposed previously. The simplest way to build a background model it to manually select a static image that represents the background and has no moving object. By thresholding the difference between the current image and the chosen image, the foreground can be detected, as shown in Figure 1.1.

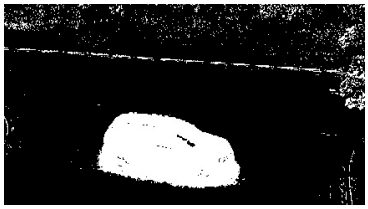
A static image is not the best choice. If the light changes suddenly, then the foreground segmentation may fail due to the large intensity difference in the background induced by the changing light. Some other previous works propose to use the frame previous to the current one instead of the static image [3]. This approach fails if the moving object has large areas of uniform color. In [21], they suggest to compute the arithmetic mean or weighted mean of the pixels in the previous frames as the background image. The main advantage of this method is that the background model is updated and might be able to handle large changes in the scene over time. Assuming that the background is more likely to appear in a scene, in [24] the authors proposed to use the median value of the previous several frames as the background



(a) Input video

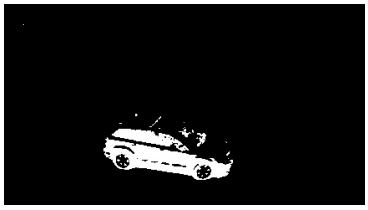
(b) Background model

(c) Difference between current frame and background model



(d) Foreground detection with a threshold 15

(e) Foreground detection with a threshold 25



(f) Foreground detection with a threshold 50

Figure 1.1: Example of input to the background subtraction algorithm. The background model is estimated from the scene without moving objects, as shown in (b). Usually, more than one frame is used for estimating the background model, to be robust to illumination changes. Given a new, previously unobserved frame (a), the task is to find pixels that belong to the new object (the red car in this case). It is done by thresholding the difference between the current frame and the background model, as shown in (c). The foreground detection results with different thresholds 15, 25, 40 are shown in (d), (e), (f) separately.

model. However, the approach has a high memory requirement with a buffer for the previous pixels.

After obtaining the background model, foreground detection is the next step. It can be done by thresholding the absolute difference computed from the current image and the background image. However, the detection results are sensitive to threshold selection, as shown in Figure 1.1. The works in [2][14][15][36] suggest other ways to improve the performance of foreground detection by using color, texture and edges features. Firstly, a simple approach to integrate the texture and color features for background subtraction is proposed in [36]. The method can handle the background that contain jittering background objects such as waving tree branches and detect the moving objects in the scene efficiently. In [15], the author presents a new background subtraction technique based on a combination of texture feature, color feature and intensity information. The approach works robustly both in rich texture areas and uniform areas and can also deal with noise and shadows effectively. A new algorithm for background modeling is presented by combining texture, RGB color information and Sobel edge detector [2]. The proposed method is robust and effective against illumination variation and scene motion.

In [34], Wren et al. propose to build the background model independently at each pixel location. It is based on an assumption that the intensity values of a pixel in the previous several frames fit a Gaussian distribution. So only two parameters need to be maintained for each pixel in background modeling, the average of previous pixels and standard deviation. But a single valued background is not an adequate model if the background changes are complex. In [30], the authors propose to model intensity of a pixel by a mixture of Gaussians distribution. This model has proved to be more appropriate as it provides a description of multiple background objects. After computing the difference between the current frame and previous average, the foreground detection can be done by comparing this value with standard deviation.

In this thesis, our main goal is to show that the color separation term is useful for the background subtraction energy. Thus rather than focusing on sophisticated background models, we chose three simple background modeling methods in our experiment: mean filter, Gaussian average and Gaussian mixture model, see Chapter 2 for details. If the color term is useful with simple models of the background, it should also be useful when more complex models are introduced.

In order to achieve a good performance in a practical application system, such as people tracking, traffic monitoring and video surveillance, foreground should be accurately segmented. Most recent studies on background subtraction are mainly about background modeling, the main difference between most methods is how the background model is represented. Few of them focus on foreground detection. Besides the practical needs, our work is motivated by the foregound segmentation through graph cuts, which achieves accurate and efficient foreground detection accuracy [13]. Segmentation with graph cuts to detect foreground is attractive because it allows formulation of an objective function that encodes the desired property of segmentation, such as color coherence of the object/background, smoothness of the boundary, etc, as well as providing a method for globally optimizing this objective function.

Binary energy optimization with graph cuts is an efficient approach for segmenting an image into foreground and background regions [4][29]. Many interactive segmentation approaches such as grabcut [29], lazy sanpping [22], [22], [23], are built upon it. Most segmentation functions include an appearance term to model foreground/background, and a smoothness

term to encourage smooth boundaries. If we want to apply the graph cut framework to background subtraction, what we need to consider is how to build the appearance model which is unknown in advance. In image segmentation, to estimate appearance models, a user input is needed first for initialization, usually through a box placed by the user roughly around the object. We can estimate the foreground model inside the box and the background model outside. After the initial appearance model is known, the image is segmented by a graph cut. Model estimation and segmentation are repeated until covergence to a local minimum [1]. However, we do not have user input for background subtraction within a video stream. So the appearance model can only be estimated by using the information from the background modeling. We follow the basic framework introduced in [13] to design an energy function for the background subtraction problem. The key difference is that we introduce the color separation term in the energy.

Color is a useful feature for segmentation. Background separation is also a type of segmentation problem, and so incorporating color should help to achieve better results. The main goal of this thesis is to incorporate color information into background subtraction. For this purpose, we incorporate the color separation term from [31] into the energy function for background subtraction.

In this thesis, we need to model color space efficiently. There are various approaches for modeling color, such as color histogram, kmeans, Gaussian Mixture Models (GMM), etc. Color space partitioning is usually performed independently from segmentation process, as a preprocessing step. There is a limited prior work, such as [25], where the authors propose to make color clustering an integral part of segmentation, by including a new color clustering term into the energy function. However, the approach in [25] is computationally intensive. Therefore, instead of integrating the search for a good color space as part of the energy function, we empirically search over a various ways to construct color space to determine what works the best for our application. In addition, there are multiple background modeling methods, and we search over them as well.

1.2 Our Approach

In this thesis, we formulate the background subtraction problem as a binary pixel labeling problem, where the goal is to assign each image pixel a ”background” or ”foreground” labels. We solve the pixel labeling problem using a principled energy minimization framework. We design an energy function composed of three terms: the data, smoothness, and color separation. The data term uses the background model. It assigns each pixel a cost of being assigned to the foreground and background labels. If a pixel fits the background model well, then its cost to be background is low. Otherwise the background cost is high. The smoothness term encourages the foreground and background regions to have spatially coherent boundaries. We use the standard pairwise term [5] [29] in this work. Commonly used pairwise potential is edge contract sensitive smoothness penalty. The higher the intensity contrast between two adjacent pixels, the smaller the smoothness penalty. This model is effective because the object boundaries are likely to align with strong image edges.

The data and smoothness terms have been used for the background subtraction problem before [13]. The main contribution of this thesis is the introduction of a new color separation

term into the energy function for background subtraction. This term models the fact that the foreground and background regions tend to have different colors. Thus, introducing a color separation term encourages foreground and background regions not to share the same colors. Color separation term can help to correct the mistakes made due to the data term when the motion information is not entirely reliable. We model color separation as L1 distance term using the model proposed in [31]. Color clustering is used to model the color space efficiently. Our energy function can be globally and efficiently optimized with a single graph cut [31], which is a big advantage of the framework.

Figure 1.2 shows the flow chart of our method, which can be roughly divided into three parts. Given an input video stream, we first build a background model including background initialization and background maintenance, based on a fixed number of frames. This model can be designed in various ways. The initialization should be done first, but it varies according to the chosen background modeling method. The background model needs to be upated during each step after the foreground detection process. These result images are analyzed in order to update the background model learned at the initialization step, with respect to a learning rate. After we get the background model, we compare the current frame with the background model and construct the data term in our energy function.

Another early step of our approach is to cluster the color space. This step is necessary for constructing the color separation term. We also found it helpful to augment the color feature with the spatial coordinates and motion features to obtain more accurate results.

The pairwise term in the eneryg function measures the smoothness of the boundary. This can be built using the intensity gradient information from the current frame. The last step is to use one graph cut to optimize the energy function and finish background subtraction for the current frame. This iteration comes to an end at the last frame of the video.

We evaluate our novel algorithm on an object detection dataset used in [7], wallflower dataset [32] and a video stream recorded by ourselves. Our approach with color separation term performs better compared with the results without color separation term and also better than results from previous background subtraction methods with the same modeling process. For instance, in Figure 1.3, we show a sample image from a video. We get more precise background subtraction for the moving object than the work in [30], which uses the Gaussian mixture model to build the background model. To make the comparison fair, in our proposed algorithm, we use the same background modeling method as the previous approach in the background subtraction process. The framework is shown in Figure 1.2. We also compare the result with and without color separation. Our approach reaches higer accuracy compared to the method [30]. More comparisons among different background modeling methods and color quantization methods are shown in Chapter 6.

This work contributes in many aspects, which are summarized as follows:

We propose to focus on the foreground detection in background subtraction by using binary energy minimization framework with the color separation term [31]. Unlike NP-hard multi-label problems discussed in [18], the high-order color separation constraint can be efficiently and globally minimized.

We propose to enhance the color separation term with spatial and motion information. Motion information is especially useful for the background subtraction problem. Thus the combination of the color and motion features performs more robustly.

We show general usefulness of the color separation term by using different background

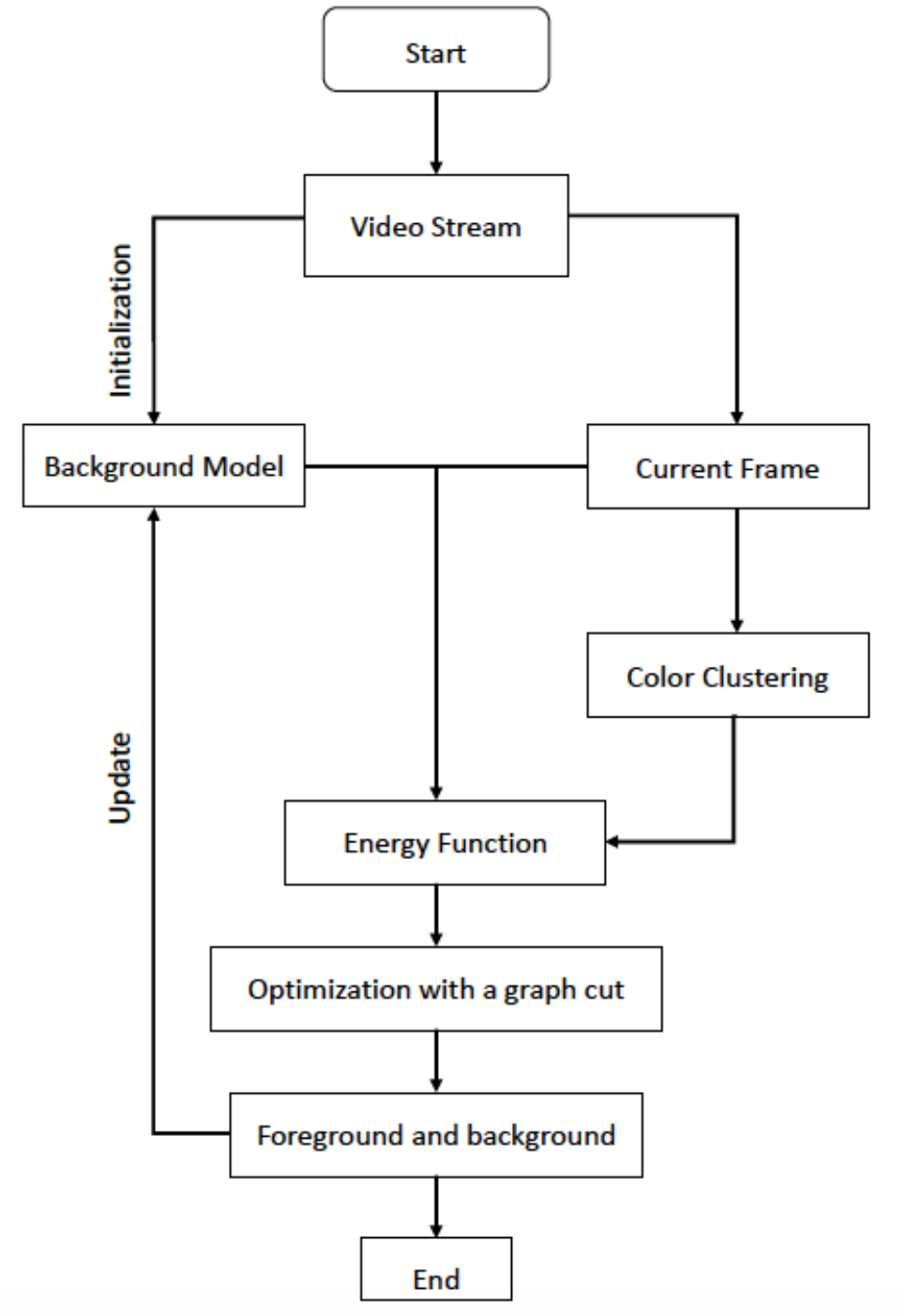


Figure 1.2: The flow chart of the algorithm

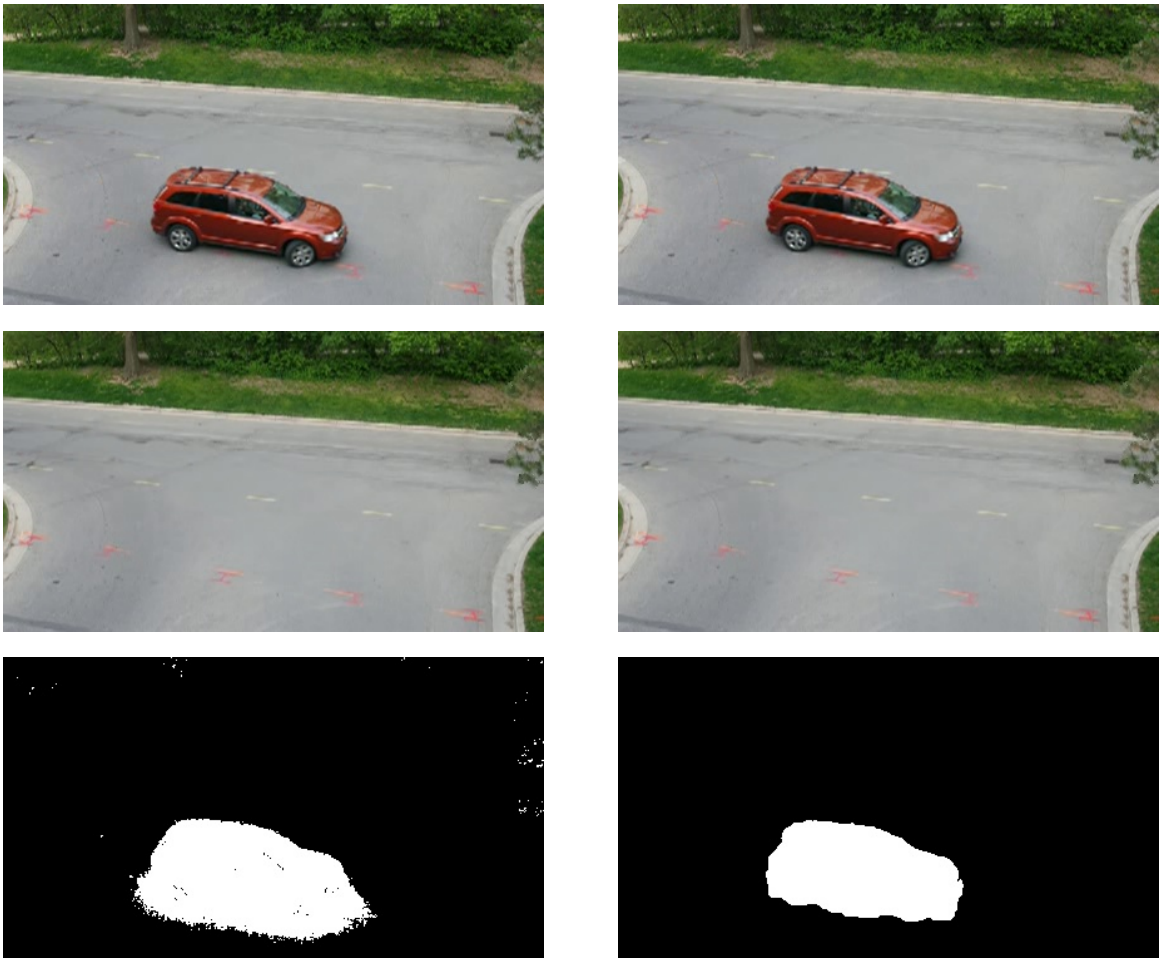


Figure 1.3: Example of background subtraction results using GMM (left) and our algorithm (right): (top) original image; (middle) background model; (bottom) foreground mask.

modeling methods and color clustering algorithms in experiments, such as mean filter, Gaussian mixture model, kmeans, meanshift, etc.

1.3 Outline of the Thesis

The thesis is organized as follows: **Chapter 2** is a review of the background subtraction techniques used in this work, including basic background subtraction model, mean filter, and statistical models such as Gaussian average (one Gaussian) and Gaussian mixture model. **Chapter 3** introduces some image segmentation algorithms we use in this work for color clustering. In **Chapter 4**, related work of binary optimization using graph cuts is analysed and the color separation term is indroduced. In **Chapter 5**, details of background subtraction using color sepatation term are explained. Furthermore, we show the graph construction for minimizing the energy function and present specific meanings for each term in background subtraction. Some experimental results are provided in **Chapter 6**. We apply the color separation term to background subtraction with different background modeling methods and different image segmentation methods. **Chapter 7** concludes the thesis work and point out future work.

Chapter 2

Background Subtraction Techniques

Given an input image, in most cases it is the objects in the scene that are of interest, not the scene itself. Examples include car tracking, people counting, etc. Background subtraction is widely used in such cases. Background subtraction is especially useful when no prior knowledge about object appearance is available. Background subtraction is among the most robust and efficient methods in computer vision.

Most background subtraction methods follow a similar procedure. It consists of two main stages: background initialization and maintenance and foreground detection. Background initialization builds an initial background model based on a fixed number of frames. In foreground detection, for each frame, the comparison is made between the current frame and the background model leading to the computation of the foreground of the scene. Usually the results of the foreground detection are fed back into the background modeling module to update the background model. This is called background maintenance.

In our principled energy minimization approach to background subtraction, we need to build a background model to be used as the data term in the energy function. We choose three background subtraction techniques for our experiments and compare the performance to find the method that shows the largest background subtraction accuracy. In this chapter, we will start with the basic background subtraction model including mean filter and further describe a simple statistical model, Gaussian average (one Gaussian). Finally, we introduce a robust statistical model, Gaussian mixture model (GMM).

2.1 Basic Models

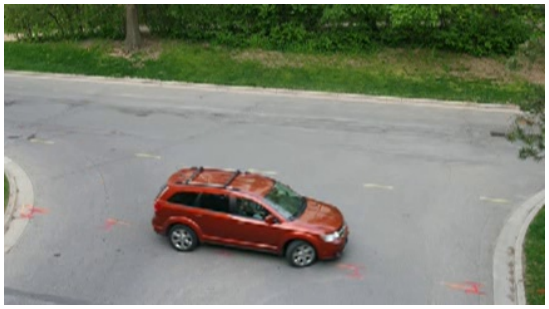
2.1.1 Static Frame Difference

Without any prior knowledge about the moving object, we first aim to build a background model. The simplest way is to manually select a static image that represents the background. This method is called the static frame difference. So we initialize the background with one static image. For each video frame, we then compute the absolute difference between the current frame and the selected static image. Based on the assumption that a moving object is made of colors that differ from those in the background, every pixel at time t whose color is significantly different from the ones in the background is more likely to be in motion. We can

apply a threshold, Th, to the absolute difference to get the foreground mask. In particular, the foreground mask image is defined as $M(x, y) = 1$, if the condition in Equation 2.1 is satisfied, and $M(x, y) = 0$ otherwise.

$$|I(x, y, t) - I(x, y, t_0)| > Th \tag{2.1}$$

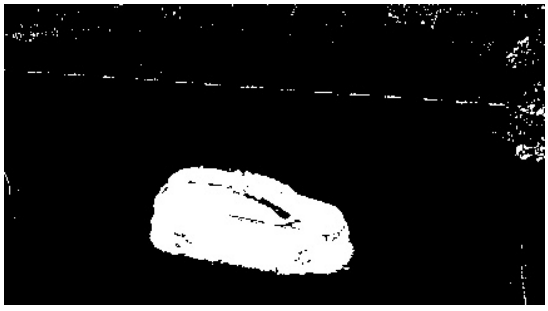
Here $I(x, y, t)$ denotes the intensity of a pixel at position (x, y), time t. $I(x, y, t_0)$ denotes the intensity of the pixel at the same position in background image. Instead of using just the intensity value at each pixel, we can use the full color information avaliable at that pixel to get a more accurate result. In this simplest approach to background modeling, there is not mechanism for updating the background model, which is a significant drawback. Figure 2.1 shows a sample frame in a video and background subtraction result using this method.



(a) Input video

(b) Background model

(c) Difference between current frame and background model



(d) Foreground mask M

Figure 2.1: Background subtraction using static frame difference

The obtained foreground is very noisy. This is because the background is very likely to undergo through changes due to illumination, etc. between the static image chosen some time in the past and the current frame.

2.1.2 Further Frame Difference

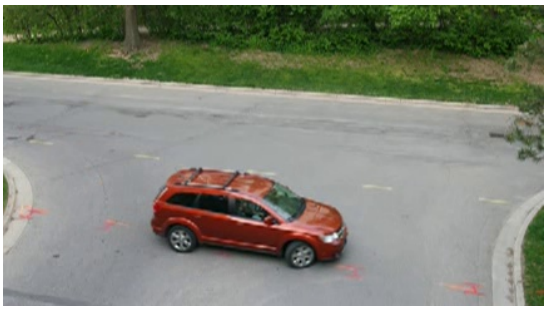
As discussed previously, a static image is not the best choice, if the background has changes, for example due to the changed lighting conditions, then the foreground segmentation may fail dramatically. Alternatively, one can use the previous frame rather than a static image to build the background model. This approach is called Further Frame Difference. The background is

initialized with the first input image. Here we maintain the background to be the previous frame of current frame in the algorithm. Based on the same assumption as Static Frame Difference, we also apply a threshold, Th. The background subtraction equation then becomes as follows:

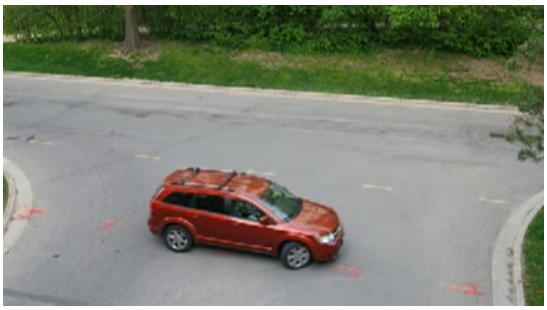
$$|I(x, y, t) - I(x, y, t - 1)| > Th \tag{2.2}$$

where $I(x, y, t)$ is the intensity of a pixel at position (x, y), time t and $I(x, y, t-1)$ is the intensity of a pixel at position (x, y), time $t - 1$. Similarly, one can also use multiple component color spaces, such as (R, G, B), (Y, U, V), (L, A, B), instead of intensity. As in the previous section, the foreground mask $M(x, y)$ is defined to be 1 for any pixel (x, y) that satisfies condition in Equation 2.2, and 0 otherwise.

Since any change in the background is much smaller between two constitutive frames, this methods is more robust to the changes in the background. However, if the object is moving slowly and has regions of uniform color, then motion of such pixels can go undetected. That is there may be no significant difference in color corresponding to the moving areas of a uniformly colored object. This can be observed in Figure 2.2. Notice that most of the background noise observed in Figure 2.1(d) is gone now. But now many interior parts of the car are not detected as the foreground, since the color difference between the two frames is small in the areas of uniform color. This figure shows the result on the same frame from the test video by using this method.

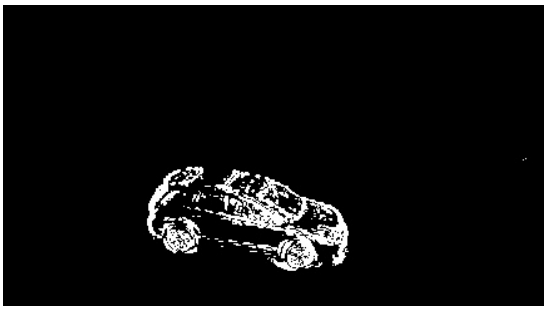


(a) Input video



(b) Background model

(c) Difference between current frame and background model



(d) Foreground mask

Figure 2.2: Background subtraction using further frame. We chose the threshold that leads to the best result.

2.1.3 Mean Filter

Another commonly used method is the mean filter. In this approach, the initialization and maintenance of the background model is performed by computing the arithmetic mean (or weighted mean) of the pixels between successive images [21]. The background model B is defined by:

$$B(x, y, t) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} I(x, y, t - i) \tag{2.3}$$

If implemented naively, the mean background model has a relatively high memory requirements, as all the frames to compute the average from have to be stored in memory. However, there is a memory efficient way to implement mean filter. First one uses equation 2.3 to initialize the background model. After the initialization, to perform the background maintenance, Equation 2.3 is computed recursively via:

$$B(x, y, t) = (1 - \alpha)B(x, y, t - 1) + \alpha I(x, y, t) \tag{2.4}$$

where α is a learning rate. The larger is the α the faster the background gets updated. The main advantage of the mean filter method is the adaptive maintenance of the background model so that it can handle changes of the background to some degree, see Figure 2.3.

After building the background model, the next step is the foreground detection. Just as before, we compute the absolute difference between the current frame and the background model. At image positions where the difference is greater than some threshold the position is classified as a foreground pixel. The foreground detection equation is:

$$|I(x, y, t) - B(x, y, t)| > Th \tag{2.5}$$

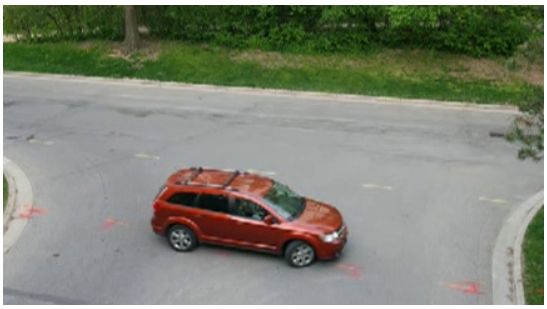
where $I(x, y, t)$ is the intensity or color of pixel (x, y) at time t. The background subtraction result from the same frame as in Figures 2.1 and 2.2 is shown in Figure 2.3.

Although the result is less noisy than in Figure 2.1 and less object parts are missing than in Figure 2.2, there are still many mislabeled background pixels. There is a shadow behind the object, since the color difference between the two frames is big in the some areas of background, it is not only present at colored object areas.

The three methods I have introduced above are easy to implement and use. All three are very efficient. And for the mean filtering method, the corresponding background model changes over time. But there is one global threshold for all pixels in the image in all these methods. And even a bigger problem, this threshold is not a function of time t. If the background appearance is bimodal or multi-modal, or if the light conditions in the scene change with time, these approaches will not give satisfactory results.

2.2 Gaussian Average

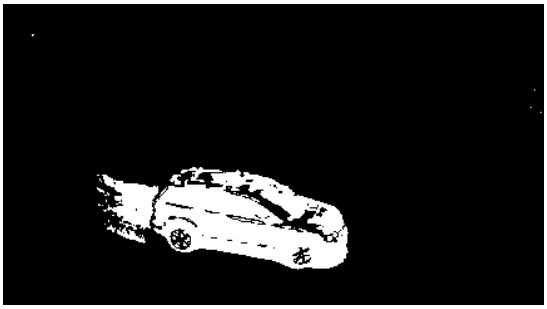
Building a statistical background model is one of the most popular methods. Each pixel at the same position in the successive images can be modeled independently by a Gaussian distribution [34]. Recall that the Gaussian distribution with mean μ and variance σ^2 is defined



(a) Input video

(b) Background model

(c) Difference between current frame and background model



(d) Foreground mask

Figure 2.3: Background subtraction using mean filter

as

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.6}$$

For each Gaussian function, only two parameters (μ, σ) need to be stored. Parameter μ is estimated as the average value of previous pixels. Parameter σ is the standard deviation and is estimated through previous pixel samples as well. There seems to be the same problem as with the mean filter, in order to estimate the parameters of the Gaussian function, we need to store all the previous images in the memory at each new frame time. This memory problem is solved as before, by computing and storing only the running average:

$$\mu(x,y,t) = \alpha I(x,y,t) + (1-\alpha)\mu(x,y,t-1) \tag{2.7}$$

where $I(x,y,t)$ is the current intensity of pixel at (x,y). Parameter α is the empirical learning rate which is a tradeoff between background stability and the speed of update. The lower the learning rate, the less quickly a background model can respond to the background changes. Meanwhile, the standard deviation can be computed in a memory efficient manner as:

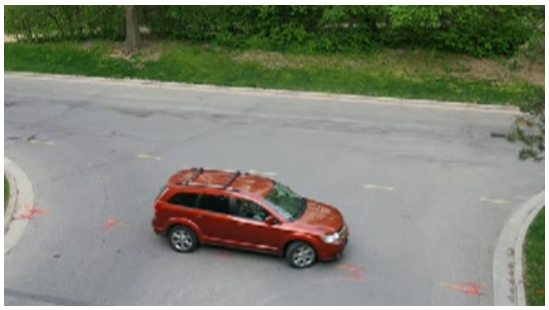
$$\sigma^2(x,y,t) = \alpha(I(x,y,t) - \mu(x,y,t))^2 + (1-\alpha)\sigma^2(x,y,t-1) \tag{2.8}$$

Instead of the buffer with previous images, this method consists of only two parameters for each pixel, which contributes to low memory requirement and fast speed.

At each frame time t, the current image pixel is labeled as a foreground pixel if it satisfies this inequality:

$$|I(x, y, t) - \mu(x, y, t)| > k\sigma(x, y, t) \tag{2.9}$$

otherwise, it will be classified as background pixel. The parameter k is usually set to a number between 2 and 3. This model can be extended to other color spaces, such as (R, G, B), (Y, U, V), (L, A, B) and etc. The background subtraction result from the same test frame is shown in Figure 2.4. There is a small amount of noise in the foreground mask. But almost the entire car



(a) Input video

(b) Background model

(c) Foreground mask

Figure 2.4: Background subtraction using Gaussian average

object is segmented, compared to the basic background subtraction methods.

In [19], the author suggests that the background model in Equation 2.7 is updated unnecessarily at pixels which are regarded as foreground. So a modified background model is proposed as:

$$\mu(x, y, t) = M\mu(x, y, t - 1) + (1 - M)(\alpha I(x, y, t) + (1 - \alpha)\mu(x, y, t - 1)) \tag{2.10}$$

where M is a binary image mask set to 1 if a pixel is classified as the foreground, and set to 0 otherwise. This method is known as selective background update.

2.3 Gaussian Mixture Model

A more robust method is to model the background with Gaussian mixture model which is proposed by Stauffer and Grimson [30]. In this algorithm, the distribution of the color of each

pixel is modeled as a sum of weighted Gaussian distributions defined in a given color space. Namely, the intensity is modeled as:

$$P(I(x,y,t)) = \sum_{i=1}^{K} w(i,x,y,t) \cdot G(I(x,y,t), \mu(i,x,y,t), \sigma(i,x,y,t)) \tag{2.11}$$

At any time t, what is known about a particular pixel is its history. The history is modeled by a mixture of K Gaussians G. Thus, the probability of occurrence of an intensity at a given pixel (x, y) is represented as Equation 2.11. G is the ith Gaussian model and $w(i, x, y, t)$ is its weight.

The mixture of Gaussians model is capable of modeling appearance as a mixture of several objects. If the background has animated textures such as waves on the water or tree branches shaken by the wind, or even just changing illumination conditions throughout the day, then a background pixel is best modeled as a mixture of several (usually a small) number of objects.

The parameters in the Gaussian functions, that is the means, standard deviations, and the mixing weights need to be initialized. We can initialize them randomly. But the standard deviation should be large enough, the weight of a new coming Gaussian should be small enough. Because the Gaussian model is not accurate when just initialized, we need to update it in the following process. If the standard deviation is larger, more samples can be included in one model, so that we get a more accurate model. Another way to initialize is with kmeans results obtained on training samples. We can set the value of μ as the average value for one cluster obtained by kmeans, σ can be computed from the same cluster and the corresponding weight is the ratio of samples from this cluster. Then, pixels which are at more than k (2 or 3) standard deviations away from any of the estimated Gaussian distributions are labeled "in motion".

$$|I(x,y,t) - \mu(i,x,y,t-1)| > k\sigma(i,x,y,t-1) \tag{2.12}$$

If a new pixel value, $I(x, y, t)$, can be matched to one of the existing Gaussians (within $k\sigma$), then the matched Gaussian's $\mu(i, x, y, t)$ and $\sigma(i, x, y, t)$ are updated as follows:

$$\mu(i,x,y,t) = (1-\rho)\mu(i,x,y,t-1) + \rho I(x,y,t) \tag{2.13}$$

$$\sigma^2(i,x,y,t) = (1-\rho)\sigma^2(i,x,y,t-1) + \rho(I(x,y,t) - \mu(i,x,y,t))^2 \tag{2.14}$$

where $\rho = \alpha G(I(x, y, t), \mu(i, x, y, t), \sigma(i, x, y, t))$ and α is a learning rate. Non-matched Gaussians are left unchanged. Prior weights of all Gaussians are adjusted as follows:

$$w(i,x,y,t) = (1-\alpha)w(i,x,y,t-1) + \alpha(M(i,t)) \tag{2.15}$$

where $M(i, t) = 1$ for the matching Gaussian and $M(i, t) = 0$ for all the others. In this model, objects are allowed to become part of the background.

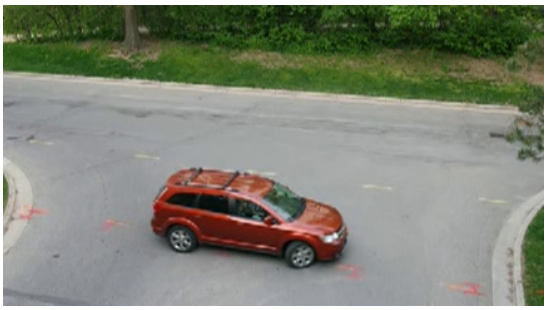
If $I(x, y, t)$ does not match to any of the K existing Gaussians, the least probably distribution is replaced with a new one. "Least probably" means in the w/σ sense. New distribution has $\mu(i, x, y, t) = I(x, y, t)$, a high σ and a low weight. Once every Gaussian has been updated, the K weights $w(i, x, y, t)$ are normalized so they sum up to 1.

The Gaussians with the most supporting weight and least variance should correspond to the background. Large weight means a large support area and smaller variance means larger

probability. The assumption is that the distribution with the largest area of support and low variance (high probability) belongs to the background. Then the smallest number b of distributions whose weights add up to a sufficiently large potion of the space T are chosen as the background model:

$$B = argmin_b(\sum_{i=1}^{b} \frac{w_i}{\sigma_i} > T) \tag{2.16}$$

where T is a threshold. Here is the background subtraction result from the same testing frame in Figure 2.5.



(a) Input video

(b) Background model

(c) Foreground mask

Figure 2.5: Background subtraction using Gaussian mixture model

There is still noise in the foreground mask. But among all these methods, Gaussian mixture model technique performs the best in terms of getting the whole foreground object. However, it performs much worse than the mean Gaussian average method in getting accurate results at object boundary.

Although a different threshold is selected for each pixel by Gaussian average and Gaussian mixture model, and these pixel-wise thresholds are adapted in time in this approach, it does not take the spatial information into consideration. That is each pixel makes a decision independently of other pixels, even though decisions are correlated. That is why the foreground mask boundary is not accurate even in GMM. In this work, our proposed algorithm models spatial interactions between pixels, encouraging a spatially coherent boundary.

Chapter 3

Image Segmentation Techniques

The goal of image segmentation is to assign labels to image pixels. As a result, the image is partitioned into distinct regions such that pixels within each region have high similarity and are very different between the regions. Image segmentation is a useful tool in many areas, such as object recognition, image processing, medical image analysis, 3D reconstruction, etc. In this work we focus on a specific segmentation task - background subtraction, where the goal is to segment the foreground object from the background in an image or a sequence of images. In this case the segmentation is binary, meaning there are only two labels. Our main contribution is incorporating color/feature separation energy term [31] into the segmentation energy. We show that color separation significantly improves background subtraction.

In this chapter, we will review three different segmentation techniques, the K-means clustering-based segmentation algorithm [26], the meanshift-based segmentation algorithm [9] and an efficient graph-based segmentation algorithm [11].

3.1 Kmeans

There is an extensive literature in computer vision that views segmentation problem as an instance of clustering [12]. Clustering is an unsupervised method in which given data points are partitioned into distinct groups/clusters based on their features. Many different clustering methods exist [17]. One of the most popular methods is K-means clustering. It was developed by J. MacQueen in 1967 and then by J. A. Hartigan and M. A. Wong around 1975. Given a set of data points, K-means partitions the points into k disjoint clusters, where the number of clusters is given by positive integer k. K-means algorithm consists of alternating between two major steps. In the first step, given current assignment of the data points to clusters, a centroid (mean point) is computed for each cluster. In the second step, each data point is assigned to the cluster with the closest centroid. These basic two steps are alternated until convergences, that is until there is no change in the cluster assignments. There are different approaches to compute the distances between data points and cluster centroids. The most commonly used is the Euclidean distance. As a part of this work, we have implemented one version of the K-means algorithm.

Given an input image, let $p \in \Omega \subset \mathcal{R}^2$ be a pixel and let f_p be a vector of features for pixel p. This vector can include geometric location of the pixel in the image as well as color and

other descriptors. Let L be the set of labels, namely $L = \{1 \dots k\}$. We denote by $l_p \in L$ the label of pixel p, namely the cluster to which p belongs. Let $1 \leq i \leq k$. We denote by Ω_i all the pixels that have label i. That is $\Omega_i = \{p | l_p = i\}$.
The algorithm is given by the following pseudo-code:
1. Initialize cluster assignments l_p.
2. Iterate till convergence
   2.1 Compute new cluster centroids

$$c_i = \sum_{p \in \Omega_i} f_p \tag{3.1}$$

   2.2 Compute new cluster assignment for each pixel

$$l_p = min_{i \in L} \|f_p - c_i\| \tag{3.2}$$

Although K-means is a very simple algorithm, it has some drawbacks. First, it is very sensitive to initialization. That is, the quality of the final clustering results strongly depends on the initial assignment of data points to clusters. Second, K-means might be computationally expensive as there is no guarantee of the convergence time.

There are many ways to initialize the cluster assignments to improve the results. Most initializations rely on Euclidean distances between pairs of the data points. In [27] they iteratively remove data points that are close to other data points from the original set, pruning the initial clustering. In this algorithm, they reported reduced running time without sacrificing the accuracy of clusters.

Madhu Yedla and etc [35] also focus on the centroid initialization to enhance the K-means clustering algorithm. They proved their proposed algorithm has more accuracy with less computational time compared to original k-means clustering algorithm. For each data point, they calculate the distance from origin. Then, the original data points are sorted according to the distances. After sorting, they partition the sorted data points into k equal sets. In each set take the middle points as the initial centroids. This algorithm does not require any additional parameters.

In this work we make use of both color and position of the pixels into account, which results in 5-dimensional feature space.

3.2 Mean Shift

As mentioned above, K-means segmentation requires the user to predefine the number of clusters. In many cases, the number of clusters is not known a priori and depends on the input data. Clustering algorithms that can automatically find the number of clusters during the processing are theretofore desirable. Mean shift is generally a non-parametric effective clustering algorithm that does not require the number of clusters as an input. It has become widely-used in the computer vision area since it was first proposed in [9] by D. Comaniciu and P. Meer. It was introduced as a robust approach toward feature space analysis. We give a brief review of the mean shift clustering algorithm for gray-scale images first and then explain how to use it on color images.

Let $\Omega = p_1, ..., p_N$ be a set of N image pixels, each described by a feature vector f_p. Let $H = (b_1, b_2, ..., b_M)$ be the image intensity histogram with M bins. Here each bin i is represented by its center. Let $g(b_i)$ be the count of data points in bin b_i. A window of the histogram is a consecutive subset of $2w + 1$ bins centered at some bin b_i. The mean shift algorithm starts at a random bin as the center of the window. Then, the center of the window is shifted in the direction of the centroid of all the points that fall within the window, namely towards the higher density region. This is repeated until the window converges at a mode of the histogram. This mode represents a cluster and all bins that converge to the same mode are assigned to the same cluster. The process is repeated until all bins are associated with one of the computed modes/clusters. The details of the above iterative mode search are shown below.
1. Place a window center at a randomly selected bin b_i.

$$c_w = b_i \tag{3.3}$$

2. Repeat till convergence:
   2.1. compute the weighted centrorid which will be the new center of the window

$$c_w = \frac{\sum_{b_j \le c_w + w \text{ and } b_j \ge c_w - w} b_i g(b_i)}{\sum_{b_j \le c_w + w \text{ and } b_j \ge c_w - w} g(b_j)} \tag{3.4}$$

The shift procedure is illustrated in Figure 3.1 for a one-dimensional feature vector, e.g., intensity. In the figure, a nine bins window is shown, centered on bin five. The first five bins have count zero, while the other four bins have nonzero values giving nineteen in total. The new mean is computed as

$$\frac{1 \times 6 + 1 \times 7 + 7 \times 8 + 10 \times 9}{19} = 8 \tag{3.5}$$

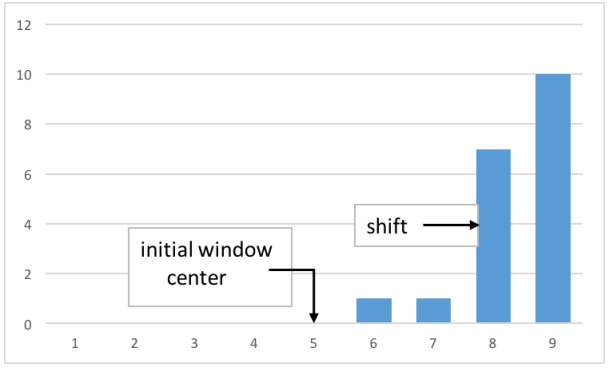


Figure 3.1: Example of mean shift for 1D histogram

Thus the mean will be shifted from bin five to bin eight, and the procedure continues until it converges. When using the mean shift technique for grayscale image segmentation, the shift procedure is run for each grayscale value and the mode at convergence gives the cluster assignment for all pixels with that gray-scale value.

Naturally the mean shift approach can be applied to multidimensional vectors. In our work, feature vectors for each pixel are comprised from color and location.

Finding the mode associated with each data point helps to smooth the image while preserving discontinuities. Two points that are far from each other in the feature space would not fall in the same window and therefore will likely converge to two different clusters. Hence, pixels on either side of a strong discontinuity will not attract each other. By controlling the size of the window in color and space with bandwidth (h_s, h_c), we can determine the resolution of the mode detection. If the window size is not appropriate the results can be noisy. In this case it is possible to run a filtering post-processing which will remove/merge small neighboring clusters.

In [9], a simple linkage clustering is used for grouping modes which are less than one window size apart and their corresponding data points are merged. This simple procedure effectively converts the noisy results into smooth. The authors in [8] suggest to build a region adjacency graph to hierarchically cluster the modes. But color information may not be sufficient, since there might be color overlap between the object and the background. They also proposed to combine edge information with color information to get better clustering results.

Figure 3.2 shows examples of mean shift segmentations for different values of the window bandwidth parameter h_c in the color space. Small changes in this parameter can make large differences in the segmentation results. Figure 3.2(a) shows the original image. When the color bandwidth is small, we get an over-segmented image in Figure 3.2(b). As we increase the value of the color bandwidth, we get reasonable segmentation results with clear objects boundary in Figure 3.2(f). Figure 3.2(g) shows a segmentation result for a large color bandwidth.

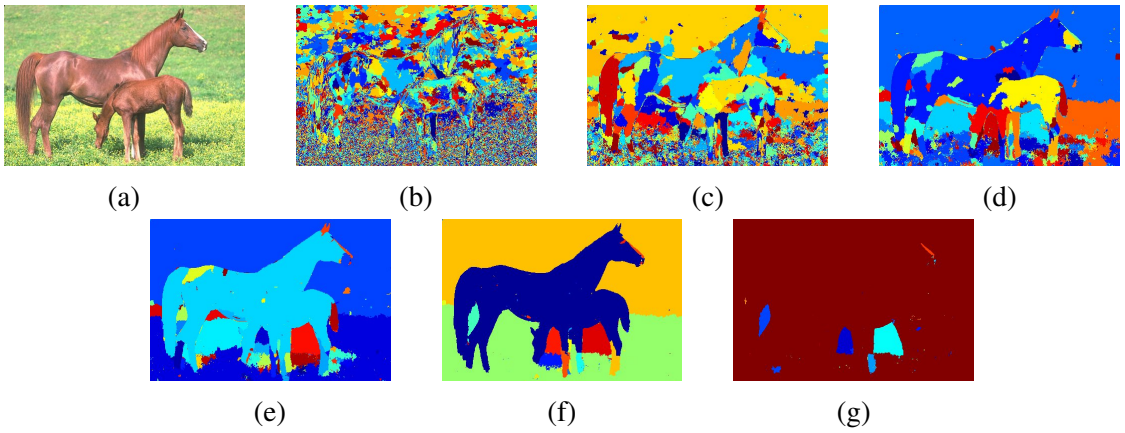


Figure 3.2: Example of mean shift segmentation with different color features: (a) Origianl image, (b)-(g) mean shift segmentation using scale bandwidth 7 and color bandwidths 3, 7, 11, 15, 19 and 23 respectively. [28]

Mean shift works well for color quantization. As we discuss in the experiments section, this algorithm is a reliable building block for our background subtraction with color separation. For some input images mean shift algorithm might produce noisy segmentation. Moreover, it might be sensitive to variations in lighting, e.g. in consecutive video frames.

3.3 Efficient Graph-based Image Segmentation

Another method to cluster image pixels in the feature space is efficient graph-based image segmentation as described in [11]. Different from mean shift, this method does not need to perform a filtering step first. It directly works on the data points in feature space and uses a adaptive threshold instead of a constant threshold on single linkage clustering. In this section, we first introduce the graph-based representation of the image, then describe the metric for comparing the dissimilarity inside one region and between regions, and give a brief overview of the complete algorithm at last.

3.3.1 Graph-based segmentation

Let $G = (V, E)$ represent an undirected graph, in which V is the set of vertices and E is the set of edges. It is the set of vertices V that needs to be segmented in image segmentation. To measure the dissimilarity between neighboring vertices v_i and v_j, we use an edge between them with a corresponding weight $w(v_i, v_j)$. This dissimilarity between the vertices can be simply measured by the difference in intensity. Or even more complex, it can be based on color, depth, motion, location or any other local attributes. In [11], an edge weight is based on the color information in the RGB space. Then, the goal of this method is to partition the vertices into two sets such the variance within each set is low compared to the boundary between the sets. The principle of such a partition is that the data points which belong to the same parts are similar and yet data points which belong to different parts are dissimilar. The edge weights give a measurement of dissimilarity between two vertices as we described above, so that edges between vertices belonging to the same region will have low weights whereas edges between vertices belonging to different regions will have high weights.

3.3.2 Pairwise region comparison metric

Using a constant threshold as a criterion for merging clustering does not consider the variability between different regions. This may cause some problems. For example, neighboring regions with low contrast between them may be merged together, whereas neighboring regions with high internal variability may be separated into several components. To avoid this, a metric which has an adaptive threshold taking into account the variability of the region is used. The metric has two dissimilarity components. One is to measure the dissimilarity between two neighboring regions by comparing the data points along their boundary. The other is to measure the dissimilarity of the data points within each region. Internal difference is defined as follows:

$$\mathrm{Int}(C) = \max_{e \in \mathrm{MST}(C,E)} w(e) \tag{3.6}$$

where, $G = (C, E)$ is a subgraph and $\mathrm{MST}(C, E)$ is the Minimum Spanning Tree built on this subgraph. A minimum spanning tree is generated by connecting all data points in the subgraph with edges so that the resulting tree has the minimal sum of weights. Thus, the internal difference can be explained as the largest edge weight in the minimum spanning tree of the subgraph.

The difference between two different regions is measured as the minimum edge weight connecting these two parts. External difference is defined as follows:

$$\text{Diff}(C_i, C_j) = \min_{v_i \in C_i, v_j \in C_j, (v_i, v_j) \in E} w(v_i, v_j) \tag{3.7}$$

If there is no edge between C_i and C_j, it means that these two regions are not neighbors, so that $\text{Diff}(C_i, C_j) = \infty$. And then if the external difference between the two regions is less than the internal difference inside either of the region plus a variable threshold term, the two regions are allowed to be merged. We define the merged difference as follows:

$$\text{MInt}(C_i, C_j) = \min(\text{Int}(C_i) + \tau(C_i), \text{Int}(C_j) + \tau(C_j)) \tag{3.8}$$

More specifically, we compare Equation 3.7 with the formula 3.8. If Equation 3.7 is less than the Equation 3.8, the regions are merged. Therefore, the regions are not combined if there is a strong evidence of a boundary between them. A strong evidence of a boundary is required for small regions and vice versa, so the threshold function is written as follows:

$$\tau(C) = k/|C| \tag{3.9}$$

The above threshold function implies that a larger k produces larger regions and yet smaller k produces smaller regions. Instead of constant threshold in traditional linkage clustering, the key to success of this algorithm is this data-dependent thresholding. It is also possible to define $\tau(C)$ based on prior information to favor some desired shape. Using adaptive dissimilarity metric above is robust to outliers, but makes the partition problem NP-hard [11].

3.3.3 Algorithm

Below we describe the algorithm in detail. First, the graph $G = (V, E)$ is formed with m edges and n vertices. Each vertex is a pixel. The final segmentation will be $S = (S_1, .., S_r)$ where S_i is a cluster of data points. The pseudo-code for the efficient graph-based image segmentation is illustrated in Algorithm 1.

Algorithm 1 Efficient Graph-based Image segmentation

Input: $G = (V, E)$ and $w(v_i, v_j)$ $\forall v_i, v_j \in V$ and $v_i \neq v_j$
1: Sort E into $E' = (e_1, ..., e_m)$ in non-decreasing order
2: start with a segmentation S^0 where each vertex v_i is in a component by itself
3: Let $e_q = (v_i, v_j)$. Repeat step 3 for $q = 1, ..., m$ to find S^q given S^{q-1}
4: **if** v_i and v_j are in disjoint components of S^{q-1} **then**
5: **if** $w(e_q)$ is less than $\text{MInt}(C_i, C_j)$ where $v_i \in C_i$ and $v_j \in C_j$ **then**
6: Merge C_i and C_j
7: **end if**
8: **end if**
Output: S^m

The merging metric in 3.8 allows efficient graph-based clustering to be more sensitive to edges in regions of low variability and less sensitive to edges in the regions of high variability.

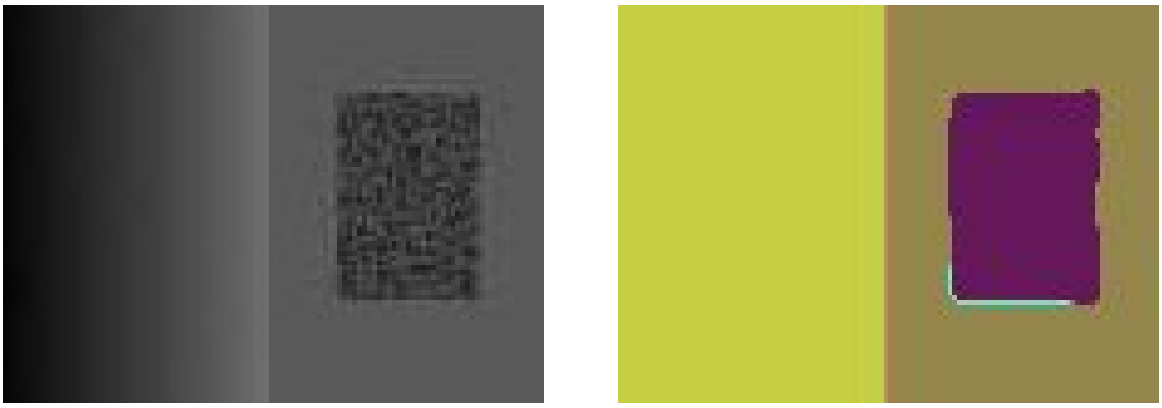


Figure 3.3: A synthetic image (320 × 240 grey image), and the segmentation results (σ = 0.8, k = 300) [11].

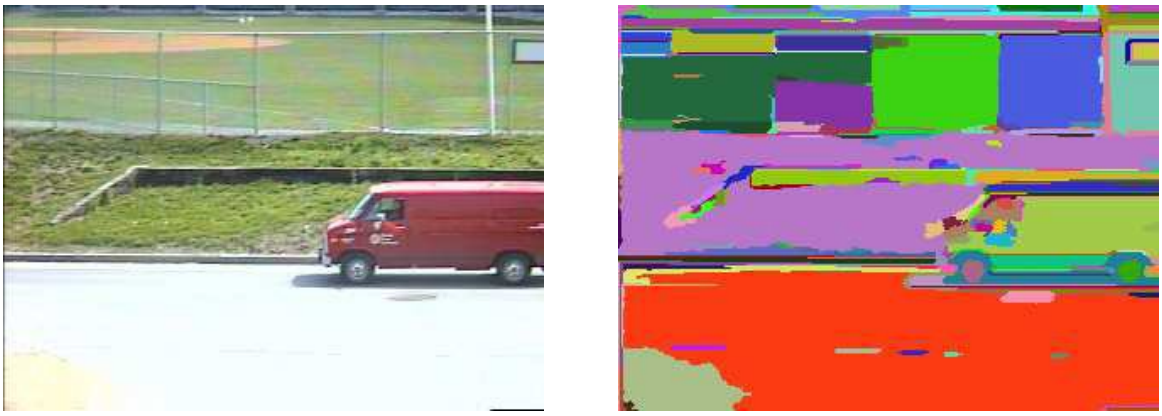


Figure 3.4: A street scene (320 × 240 color image), and the segmentation results (σ = 0.8, k = 300) [11].

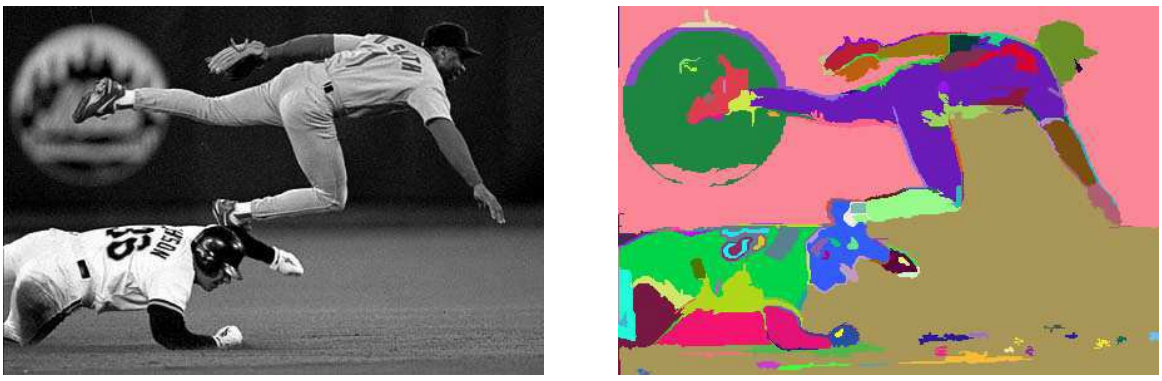


Figure 3.5: A baseball scene (432×294 grey image), and the segmentation results (σ = 0.8, k = 300) [11].

Gaussian filter is often used first to remove artifacts and smooth the image before segmentation. Some segmentation results implemented in [11] are shown here.

Figure 3.3 shows a synthetic image with a region which has high variability in the right part. This algorithm can segment out such region due to lower sensitivity to edges in this region. Figure 3.4 and Figure 3.5 show two real world scenes and the segmentation result produced by the method.

Chapter 4

Energy Minimization using Graph Cuts

For each frame in a video, we formulate background subtraction as a binary labeling problem where the goal is to assign each image pixel either a "foreground" (or object) or a "background" label. We solve this binary pixel labeling problem in a principled energy minimization framework. The advantage of using energy minimization framework is that we can encode useful problem constraints, such as the requirement that the foreground and background regions are spatially coherent. After the energy function is formulated, we can use optimization algorithm to do energy minimization. The advantage of our energy is that it can be efficiently and globally optimized with a single graph cut [5].

In this chapter, we introduce the labeling problems and a common form of energy function first. Then we give an overview of the energy minimization method based on graph cuts and finally describle a new color separation energy term used in the energy function, which can be globally minimized with a single graph cut [31].

4.1 Labeling Problem and Energy Minimization Framework

Many problems in computer vision can be formulated as labeling problems. In a labeling problem, one has a set of sites and a set of labels. The set of sites is usually all the image pixels. The set of labels are application dependent. The goal is to to assign each site a label from the label set.

To describe the labeling problem more formally, let $\mathcal{P}$ be the set of sites and $\mathcal{L}$ be the set of labels. The set of sites could be image pixels, medical volume voxels, and any other image entity to which we wish to assign a label. In this thesis, we will assume that sites are image pixels. A labeling problem now can be described as assigning a label f_p to each pixel $p \in \mathcal{P}$. The collection of all pixel label assignments will be denoted by f. The labels can have a semantic meaning, such as "house", "person", "vehicle", or geometric meaning such as "left oriented surface", "right oriented surface", etc. Many other meanings of labels are possible. In this work, we need only two labels, the "foreground" and the "background". We identify the foreground with label 1 and the background with label 0. Thus our label set is binary and is equal to $\{0, 1\}$.

If the label set for all pixels is the same, i.e. $\mathcal{L}$, then the labeling f belongs to the set

$$\mathfrak{L} = \mathcal{L} \times \mathcal{L} \times ... \times \mathcal{L} \tag{4.1}$$

Thus the total number of different possible labelings f is exponentially large, even in the case when the label set $\mathcal{L}$ is binary. Any naive method for optimization, such as exhaustive search, is not feasible.

To solve the labeling problem in the energy minimization framework, we design an energy function $E(f)$. The energy function $E(f)$ measures whether labeling f is a good solution to the problem or not. That is $E(f)$ should output a small value if labeling f is considered good labeling, and $E(f)$ should output a large number if f is not satisfactory. A number of criteria can be used by the energy function to judge whether f is a good labeling or not. Most commonly $E(f)$ consists of two terms, the data and the smoothness, both evaluating different aspects of how good a labeling f is. The typical form of energy $E(f)$ is as follows:

$$E(f) = E_{data}(f) + \lambda \cdot E_{smooth}(f) \tag{4.2}$$

In Equation 4.2, the E_{data} is called data term, because this term penalizes labels inconsistency with the observed data according to some data model known a priori, or estimated from the user marked regions, or learned from a dataset with ground truth. E_{smooth} term is called smoothness term, it measures how smooth or regular the boundaries between regions with different labels are. A typical model used in the smoothness term penalizes any two adjacent pixels that have different labels. If the label set has many labels, then typically a larger label difference is penalized more. However in our case, there are only two labels. Thus either two nearby pixels have exactly the same label, in which case there is no penalty, or they have different labels, in which case there is a penalty. The weight $\lambda > 0$ measures the relative importance of smoothness versus the data terms. If λ is small, then the smoothness term is not so important and we are looking for a labeling which fits the data terms closely. If λ is large, then the data term is less important and a labeling with less discontinuities between the labels of nearby pixels is preferred. Thus the choice of λ is very important. It is chosen either through a parameter learning technique, or estimated experimentally on validation data through grid search.

The typical form for data term is as follows:

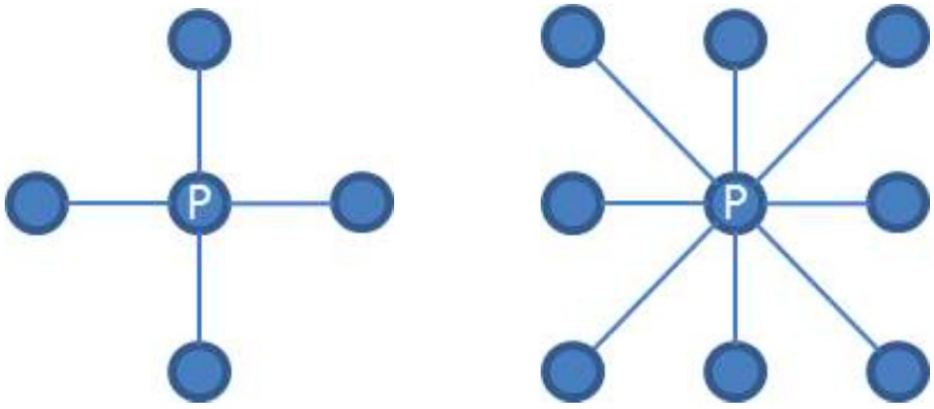
$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{4.3}$$

where D_p measures the penalty for assigning pixel p to the label f_p, according to the data model. The data model can be based on image intensity, color and etc. For example, let us consider a simple case. Suppose we know a priori that the background should have intensity 20 or lower and the object should have intensity 220 or higher. Then a good choice for the data term is $D_p(0) = max(I_p - 20, 0)$ and $D_p(1) = max(220 - I_p, 0)$.

The smoothness term is typically written as,

$$E_{smoothness}(f) = \sum_{pq \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{4.4}$$

where $\mathcal{N}$ is the set of pairs of neighboring pixels. The structure of $\mathcal{N}$ is most often given by 4-connected or 8-connected grid. Each pixel is connected to its four nearest neighbors if using the 4-connected neighborhood, and to its eight nearest neighbors if using the 8-connected neighborhood. Figure 4.1 shows the construction of the neighborhood systems.



(a) 4-neighborhood system (b) 8-neighborhood system

Figure 4.1: Illustration of 4 and 8 neighborhood systems in images

The smoothness term penalizes the discontinuties between neighboring pixels. The more similarity the two neighboring pixels have, the larger is the penalty if they are assigned different labels. Parameter λ is used to adjust the relative weight of the smoothness term versus the data term.

4.2 Optimization with Graph Cuts

A graph cut is a popular energy optimization algorithm in computer vision. It has been successfully used in many applications [3]. In this work, the optimization algorithm is also based on computing a graph cut of minimum cost [5]. In this section, we will briefly describe the graph cut optimization algorithm which is based on the max-flow/min-cut theorem. Then we discuss how a graph cut is applied to the binary labeling problem.

Let $G = (V, E)$ be a connected weighted graph with vertices V and edges E. The set of vertices contains two special vertices, called the source s and the sink t. Figure 4.2 shows an example of a graph with source and sink vertices. An $s - t$ cut $C = (\mathcal{S}, \mathcal{T})$ segments vertices V into two sets $\mathcal{S}$ and $\mathcal{T}$, such that $s \in \mathcal{S}$, $t \in \mathcal{T}$ and $V = \mathcal{S} \cup \mathcal{T}$. More specifically, a cut C is a subset of edges E, such that when edges in C are removed from the graph G, vertices V is partitioned into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$. The cost of the cut C is computed as follows,

$$|C| = \sum_{e \in C} w_e \tag{4.5}$$

where w_e is the weight of edge $e \in E$. Figure 4.3 shows an example of $s - t$ graph cut with terminals s and t. It is shown for a 4-connected neighborhood. The thickness of the edge is directly proportional to the edge weight. The aim of the minimum cut problem is to find a cut with the minimum cost among all possible cuts in the graph G. The green dashed curve shown in Figure 4.3 illustrates a minimum cut, that is the cut with the lowest cost.

According to the max-flow/min-cut theoriem proved by Ford and Fulkerson [16], the minimum cut of a graph can be easily computed by finding a maximum flow from the source s

to the sink t. In [16], they describe each edge of the graph as a pipe with a capacity that is equal to its weight. Thus the flow that one can push through an edge cannot be larger than the capacity of that edge. Then, maximum flow is explained as the maximum "amount of water" that can be sent from the source to the sink [6]. A maximum flow from source to tink saturates a set of edges in the graph. An edge is saturated if the flow through that edge is equal to the weight (or capacity) of that edge. These saturated edges divides the nodes into two disjoint parts corresponding to a minimum cut [16]. Thus, the minimum cut problem is equivalent to the maximum flow problem.

More specifically, the flow from one node to another is denoted as a function $f : V \times V \rightarrow \mathbf{R}$. The value of the flow for the graph $G = (V, E)$ is defined as follows:

$$|f| = \sum_{v \in V} f(s, v) \tag{4.6}$$

where $f(s, v)$ means the flow sent out of the source s to v. If the edge (u, v) is non saturated, we call the additional amount of flow as residual capacity $c_f(u, v)$.

$$c_f(u, v) = c(u, v) - f(u, v) \tag{4.7}$$

where $c(u, v)$ denotes the capacity of the edge, $f(u, v)$ denotes the flow sent from u to v.

Consider a path $p = (s, v_1, v_2, ..., t)$ with no repeated vertices from s to t, we define the residual capacity of the path as below:

$$c_f(p) = min\{c_f(u, v)|(u, v) \in p\} \tag{4.8}$$

If we iteratively send flows from the source to the sink till no more flow can pass through the edges, each path reach the lowest residual capacity with a saturated edge, then we have sent the maximum flow from the source to sink.

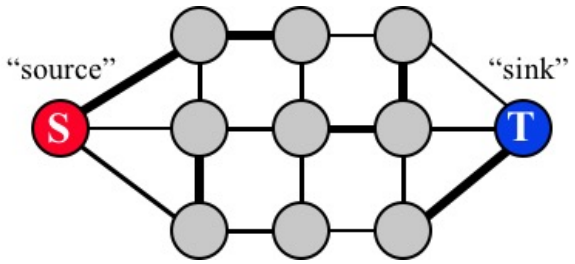


Figure 4.2: An example of a graph with source and sink terminals. [Image credit: Yuri Boykov]

Then, the process of max-flow/min-cut algorithm is described as following steps. Figure 4.4 also gives an intuitive description.

1. First, find a flow path f from s to t along non saturated edges, which means that the capacity of the flow f is smaller than all edges weight along the path.

2. Increase flow f along this path until some edge saturates. The edge become saturated when the capacity of the flow f is equal its edge weight.

3. Iterate step 1 and step 2 until all paths from s and t have at least one saturated edge.

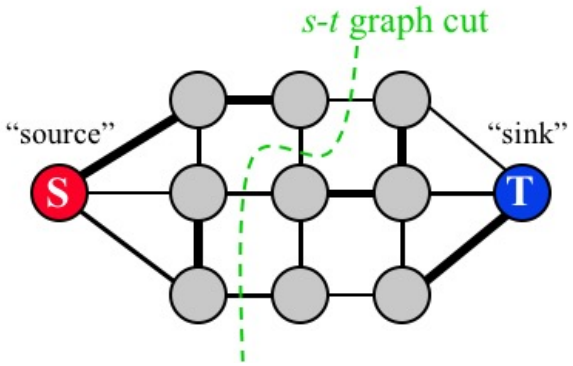


Figure 4.3: An example of a $s - t$ cut on a graph. [Image credit: Yuri Boykov]

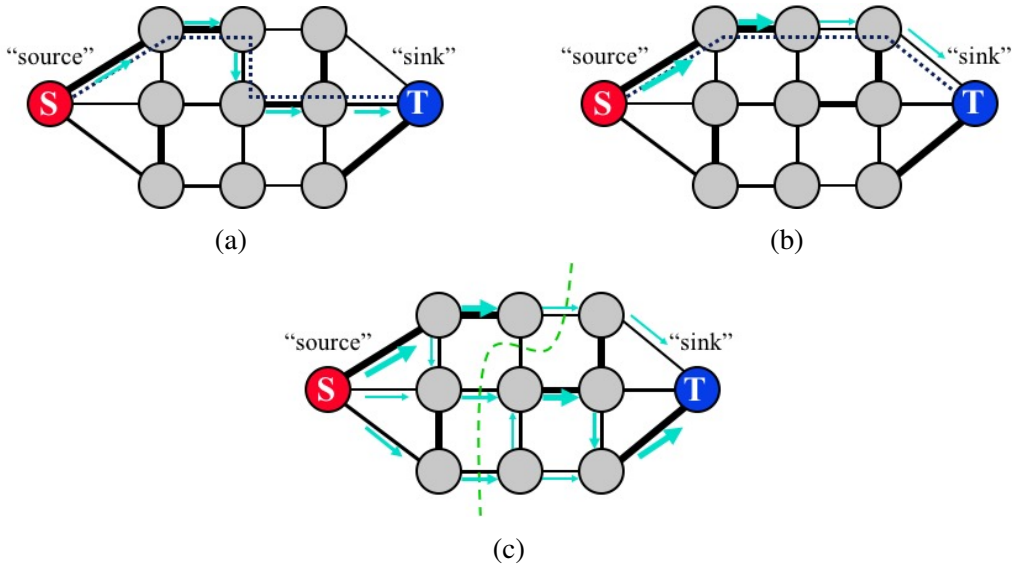


Figure 4.4: Illustrates the max-flow/min-cut algorithm. [Image credit: Yuri Boykov]

4. All the saturated edges found in the former steps form the minimum cut.

The algorithm introduced before is one of the popular algorithms to compute the maximum flow of a given graph. It is the basic maximum flow algorithm proposed by Ford-Fulkerson. Edmonds-Karp presented another method to compute the maximum flow, which is an improvement of Ford-Fulkerson's algorithm in terms of computational efficiency. More details are in [10].

Generally, the optimization of the energy Equation 4.2 is an NP-hard problem, even when dealing with the binary energy. But in some special cases, a global optimum of the energy function can be found by finding a minimum cut on a certain graph. In particular, if a binary energy function is submodular, then it can be minimized exactly with finding a minimum cut on a certain graph. In this thesis, we choose to use the max-flow/min-cut algorithm developed by Boykov and Kolmogorov [6] as it is particularly efficient in practice for the graphs that arise in computer vision problems.

For binary energies, Kolmogorov [20] showed that the energy function is submodular if

$$V_{pq}(0,0) + V_{pq}(1,1) \leq V_{pq}(0,1) + V_{pq}(1,0) \tag{4.9}$$

where V_{pq} is the pairwise smoothness term in the energy function,

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \sum_{pq \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{4.10}$$

Consider a simple 3 by 3 image as an example for binary energy optimization. As shown in Figure 4.5, we first need to construct a graph $G = (V, E)$ based on the original image. The pixels in the image correspond to graph vertices V and the neighboring pixels are connected by graph edges E. There are two special terminal nodes source s and sink t, corresponding to the two different labels in binary optimization problem. Each pixel node in the constructed graph is connected to the terminal nodes source s and sink t by an edge usually called $t-link$. The data term $D_p(f_p)$ in the energy function is used for the weight of $t-link$, and it corresponds to the penalty of assigning label f_p to pixel $p \in \mathcal{P}$. In this work, $D_p(f_p)$ is computed based on the difference between background model and the current image, where $f_p \in 0, 1$.

The links between neighboring pixel nodes are usually called $n-links$. The smoothness term is used for the weight of $n-link$, which penalizes the discontinuity between the neighboring pixel nodes. More specifically, the more similar the two neighboring pixel nodes p and q are, the higher the V_{pq} is to prevent assigning them to different labels. The measurement of the weight between neighboring pixel nodes is performed in different ways. For example, in grayscale images the similarity can be computed on local gradient of intensity.

Any cut in the constructed graph corresponds to a binary labeling of pixels in the image. If a pixel stayed connected to the source, it is assigned label 1. If it stayed connected to the sink, it is assigned label 0. In this way, the cost of any cut is equal to the energy of the corresponding label assignment. Then, the max-flow/min-cut algorithm in [6] is applied to find the cut with the minimum cost. Finally, all the nodes that correspond to pixels are segmented into two disjoint sets. The process to optimize the energy function by using graph cuts is shown in Figure 4.5.

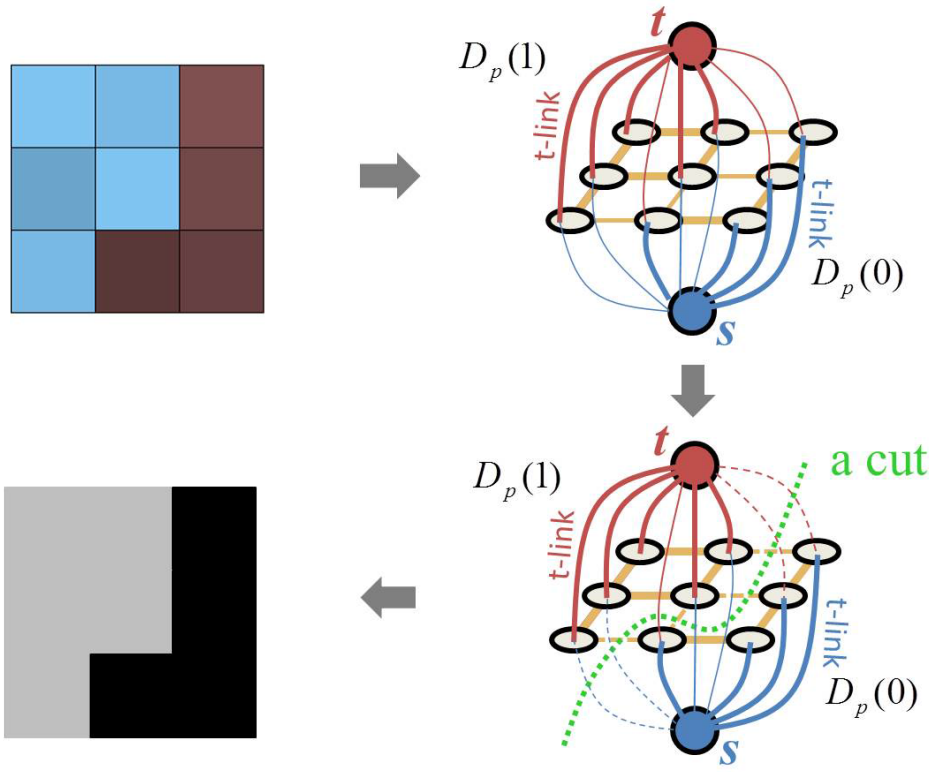


Figure 4.5: Binary segmentation for 3 by 3 image. (top-left): Original image; (top-right): Graph constructed based on original image with source and sink terminals; (bottom-right) A minimum cut for the graph separating all pixels into two disjoint sets; (bottom left): Segmentation result, one color stands for one label. [Image credit: Yuri Boykov]

4.3 Color Separation Term

In [31], the authors focused on the energy minimization for image segmentation problem and proposed a new energy term to measure the L_1 distance between foreground and background color histograms. This new color separation term is submodular, so that the whole energy function can be globally minimized in one graph cut. The new color separation term leads to an improved segementation performance in many applications. The main idea of this thesis is to include the color separation term in the energy function in order to improve the performance of the foreground detection process. In this section, we will introduce the color separation term in detail.

As previously, let $f_p \in \{0, 1\}$ be the binary labes for pixel p, 1 represents foreground and 0 represents background. Let I_p be the intensity of pixel p. Given a fixed foreground and background appearance model θ^1 and θ^0, for each pixel p, the data term in Equation 4.10 can be represented by an appearance-based log-likelihood term $lnPr(I_p|\theta_p^f)$. The smoothness term gives penalties for discontinuities between neighboring pixels. Denote $\mathcal{N}$ is the set of pairs of neighboring pixels. The smoothness term is commonly set as $|\partial F| = \sum_{pq\in\mathcal{N}} w_{pq}|f_p - f_q|$. The more difference between the two neighboring pixels, the less penality is given in the pairwise energy term when they are assigned different labels. w_{pq} in the fomular decides the contrast sensitive metric for the smoothness term. If it is equal to a constant value, the smoothness term is not edge-contrast sensitive and vice versa. Combine the log-likelihood data term and the edge-contrast sensitive smoothness term, the binary image segmentation energy function in Equation 4.10 can be written as

$$E(F|\theta^1, \theta^0) = -\sum_{p\in\mathcal{P}} lnPr(I_p|\theta_p^f) + |\partial F| \tag{4.11}$$

where $\mathcal{P}$ is the set of image pixels and F is the set of foreground pixels , that is the pixels with $f_p = 1$. The data term in Equation 4.11 is a unary term and the smoothness term is a pairwise term.

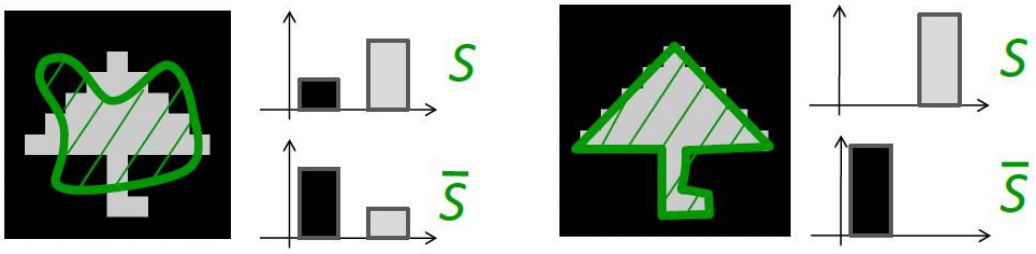
A simple way to build foreground and background appearance models is by color histograms, the energy function 4.11 can be rewritten as:

$$\begin{aligned} E(F|\theta^1, \theta^0) &= -\sum_{f_p=1} lnPr(I_p|\theta^1) - \sum_{f_p=0} lnPr(I_p|\theta^0) + |\partial F| \\ &= -\sum_k n_k^F ln\theta_k^1 - \sum_k n_k^{\bar{F}} ln\theta_k^0 + |\partial F| \\ &= -|F|\sum_k \theta_k^F ln\theta_k^1 - |\bar{F}|\sum_k \theta_k^{\bar{F}} ln\theta_k^0 + |\partial F| \\ &= |F| \cdot H(\theta^F|\theta^1) + |\bar{F}| \cdot H(\theta^{\bar{F}}|\theta^0) + |\partial F| \end{aligned} \tag{4.12}$$

where n_k^F and $n_k^{\bar{F}}$ represent the number of pixels in the k^{th} color bin in foreground and background respectively. Terms θ^F and $\theta^{\bar{F}}$ represent histograms in foreground F and background $\bar{F}$. Based on the theoriem of cross entropy inenquality $H(\theta^F|\theta^1) \geq H(\theta^F)$, minimization of enery function 4.11 equals to the minimization of energy

$$E(F) = |F| \cdot H(\theta^F) + |\bar{F}| \cdot H(\theta^{\bar{F}}) + |\partial F| \tag{4.13}$$

Equation 4.12 only depends only on the foreground F, so that the global minimum dose not depend on the initial foreground and background appearance models θ^1 and θ^0. The entropy terms of this energy prefer segments with more peaked color distributions that give lower entropy, which means smaller overlap between foreground and background histograms. Figure 4.6 shows a black-white image as an illustration. There are only two color bins in the histograms. When the white pixels are completely separated from the black pixels, we get the lowest entropy value.



(a) High entropy example (b) Low entropy example

Figure 4.6: Color separation gives segments with low entropy. [Image credit: Meng Tang]

Then the energy function 4.13 can be rewritten further as,

$$\begin{aligned} E(F) &= -|F| \sum_k \theta_k^F ln\theta_k^F - |\bar{F}| \sum_k \theta_k^{\bar{F}} ln\theta_k^{\bar{F}} + |\partial F| \\ &= -\sum_k n_k^F ln\theta_k^F - \sum_k n_k^{\bar{F}} ln\theta_k^{\bar{F}} + |\partial F| \\ &= -\sum_k n_k^F ln\frac{n_k^F}{|F|} - \sum_k n_k^{\bar{F}} ln\frac{n_k^{\bar{F}}}{|\bar{F}|} + |\partial F| \\ &= |F|lnF + |\bar{F}||ln|\bar{F}| - \sum_k (n_k^F ln\, n_k^F + n_k^{\bar{F}} ln\, n_k^{\bar{F}}) + |\partial F| \end{aligned} \tag{4.14}$$

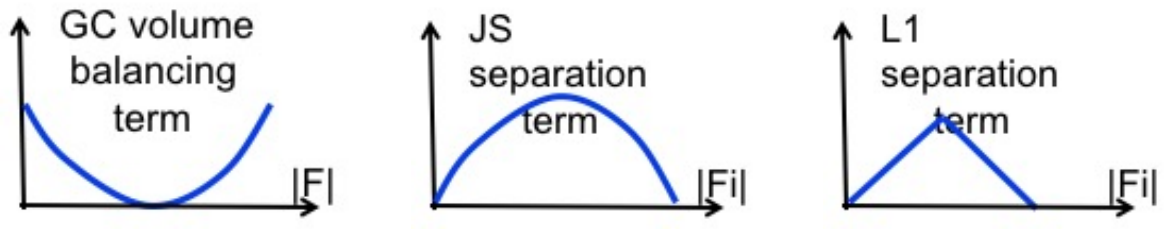
Therefore the two entropy term in function 4.13 can be equivalently written as,

$$h_{\mathcal{P}}(F) - \sum_i h_{\mathcal{P}_i}(F_i) \tag{4.15}$$

where $h_A(B) = |B| \cdot ln|B| + |A/B| \cdot ln|A/B|$ is standard Jensen-Shannon(JS) divergence functional for subset $B \subset A$ [31]. $_{\mathcal{P}_i}$ represents the set of all pixels in color bin i and F_i is the subset of pixels with color bin i inside the foreground region. $h_{\mathcal{P}}(F)$ is volume balancing term, which shows that the energy function 4.11 prefers segmentation of foreground and background with equal size. Term $-\sum_i h_{\mathcal{P}_i}(F_i)$ shows preference of color separation in the energy function. They propose to use a L_1 color separation term $-|F_i - \bar{F}_i|$ instead of $-h_{\mathcal{P}_i}(F_i)$. Figure 4.7 shows the rough plots for these terms.

F is the set of the pixels labeled as foreground by f, θ^F is the foreground color histogram and $\theta^{\bar{F}}$ is the background color histogram. Then the L_1 color separation term gives penalty for the overlap between the color bins in foreground and background appearance model. The L_1 color separation term is defined as,

$$E_{L_1}(\theta^F, \theta^{\bar{F}}) = -|\theta^F - \theta^{\bar{F}}|_{L_1} \tag{4.16}$$



(a) $h_{\mathcal{P}}(F)$ (b) $-h_{\mathcal{P}_i}(F_i)$ (c) $-|F_i - \bar{F}_i|$

Figure 4.7: Plots for different energy terms.

In [31], they have proved that this simple effective color appearance term is submodular, so the L_1 color separation term can be optimized by one graph cut. Below we will explain how to construct the graph by using L_1 color separation term. To gain more intuition, they rewrite the L_1 term as,

$$E_{L_1}(\theta^F, \theta^{\bar{F}}) = \sum_{k=1}^{K} min(n_k^F, n_k^{\bar{F}}) - \frac{|\mathcal{P}|}{2} \tag{4.17}$$

where K denotes the number of color bins. Terms n_k^F and $n_k^{\bar{F}}$ are the number of foreground and background pixels in color bin k respectively. It is easy to see that L_1 color separation term penalizes the case when pixels in the same color bin are assigned to different labels. That is to say, labeling continuity is encouraged among pixels in the same color bin. Figure 4.8 shows the graph construction for one color bin when optimizing the L_1 color separation term in one graph cut.

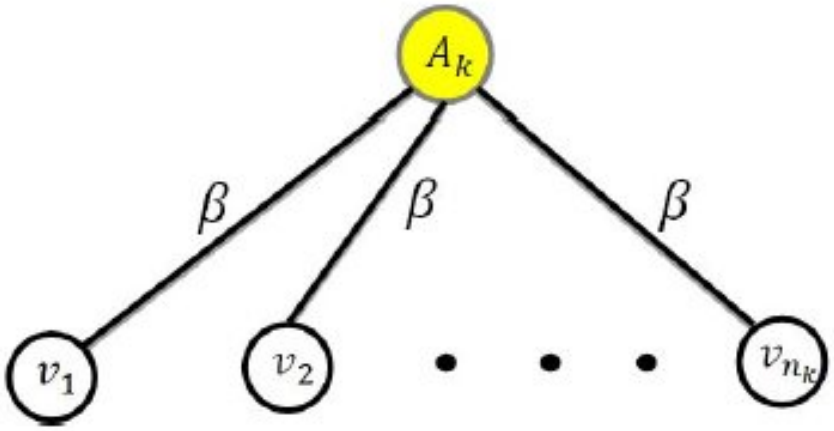


Figure 4.8: Graph construction for L_1 color separation term in one color bin. [Image credit: Meng Tang]

In Figure 4.8, they ignore the links for other energy terms, such as $n-links$ between neighboring pixels. A new auxiliary node A_k is added and it is connected to nodes $v_1, v_2, ..., v_{n_k}$ which correspond to the pixels in the color bin k using undirected links. The edges capacity β gives

penalty of color appearance model overlap. Figure 4.9 construct the overall graph structure of energy with L_1 color separation term. K auxiliary nodes $A_1, A_2, ..., A_K$ corresponding to the $1, 2, ..., K$ color bins are added to the structure. Each pixel in the graph lies in its corresponding color bin, so that edges are built to connect the auxiliary A_k with the pixels in k^{th} color bin. In the end, every pixel is assigned to one auxilliary node. If the pixels in color bin k are separated into foreground and background with n_k^F and $n_k^{\bar{F}}$ pixels inside two parts respectively, $min(n_k^F, n_k^{\bar{F}})$ is the number of links that need to be cut in graph cut. The L_1 color separation optimization is achieved in this by finding a cut with minimum cost.

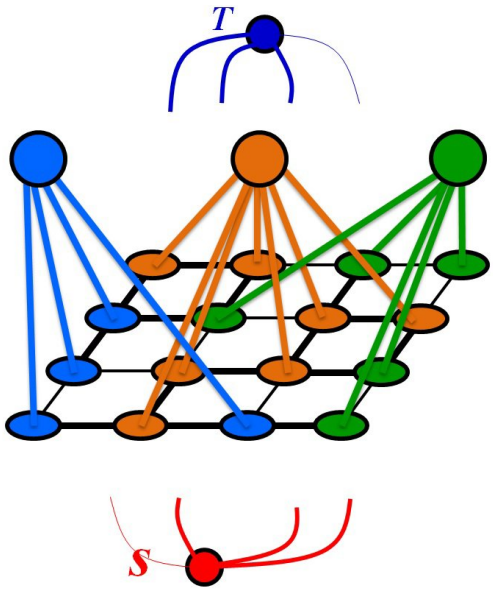


Figure 4.9: Overall graph construction for energy with L_1 color separation term. This example shows three different color bins, blue, orange and green. Three auxiliary nodes are added for these color bins. All pixels are connected to the corresponding auxiliary node. [Image credit: Meng Tang]

Notice that the volume balancing term $h_{\mathcal{P}}(F)$ in energy function 4.15 is not submodular. If it is replaced by a submodular term, then the energy in Equation 4.15 can be globally optimized. For example, in binary segmentation with bounding box, a ballooning term is used to replace the volume balancing term.

$$E_F = |\bar{F} \cap R| - \beta \|\theta^F - \theta^{\bar{F}}\|_{L_1} + \lambda |\partial F| \tag{4.18}$$

where R is the foreground bounding box, which is provided by a user. The ballooning term here encourages larger foreground inside R and helps to avoid the trival image segmentation solutions. The energy optimization can be achieved by using one graph cut. Figure 4.10 shows some image segmentation results by using one cut.

In this work, depending on the previous work of Meng [31], we use different image segmentation techniques to get more accurate color bins and apply the energy funtion with $L1$ color separation term in background subtraction hoping to achieve better performance than traditional background subtraction methods. In a word, the energy function proposed by [31]

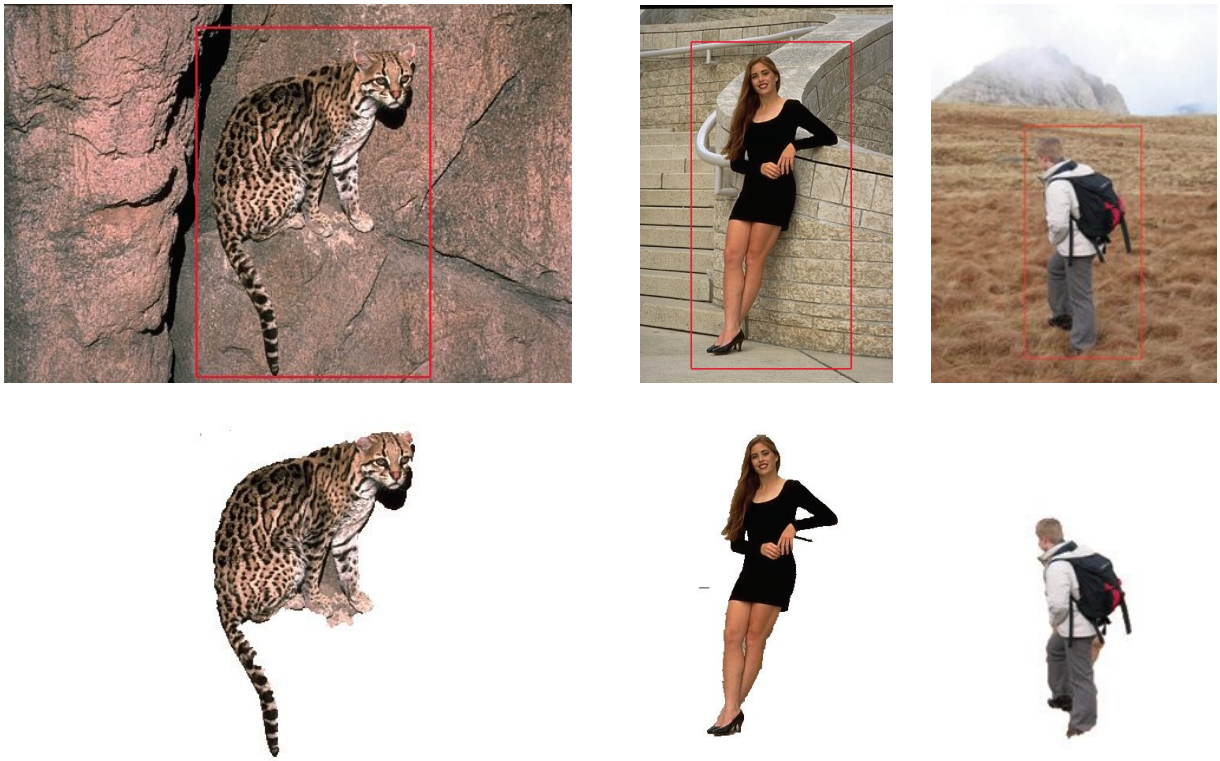


Figure 4.10: Example of segmentation results with one graph cut. [Image credit: Meng Tang]

is the heart of our thesis. We will introduce how we use it in background subtraction in the following chapter.

Chapter 5

Background Subtraction using Color Separation Term

Our approach to background subtraction is based on minimizing a binary energy function with L_1 color separation term proposed by [31]. Energy function with L_1 color separation term can be used in many other applications, such as interactive segmentation, shape matching, saliency segmentation and etc. The advantage of using L_1 color separation term is that many problems that were formulated before with NP-hard energies can now be reformulated with tractable energies. The results can be improved both in terms of time complexity and the resulting accuracy [31].

In this chapter, we first introduce the overall framework of our background subtraction method. It is mainly composed of three parts, background model construction and maintenance, color clustering and foreground detection based on one graph cut. We will explain each step in our algorithm in detail.

The vital part in this thesis is to formulate our energy function for background subtraction. There are three terms in our background subtraction energy function. First, we explain the data term, which is based on the background modeling. The pixels that change significantly between the observed image and background image prefer to be foreground, while the other pixels with only slight changes prefer to be the background. We use three different ways to build the background model and therefore three different methods to compute the data term. The second term in the energy function is the L_1 color separation term, which gives a penalty for the color overlap in foreground and background. Three image segmentation techniques are used in our work to perform clusterinig needed to obtain the color clusters. The resulting clusters respresent the color appearance for each pixel. To enhance the performance of color separation term in background subtraction problem, we add spatial coordinates and motion information to features during clustering. The experimental results with and without motion information are shown is Chapter 6. At last, we introduce the smoothness term. The smoothness term used here is contrast-sensitive. More penalty is given for discontinuity between two neighboring pixels with lower contrast. This standard smoothness term encourages segments along places of high image gradient, as those are more likely to correspond to object boundaries.

5.1 Overview

Binary energy optimization is a popular method to segment an image into foreground and background. Most segmentation functions include an appearance term to model foreground and background, and a smoothness term to encourage smooth boundaries. In [31], the authors focused on appearance models and proposed a simple tractble L_1 color separation term. The new energy term with color separation term can be optimizaited in one graph cut. We propose to use this new efficient energy function in background subtraction in order to improve performance. Our background subtraction methods consists of the following steps:

1. Build the initial background model in background initialization step.

2. Compute the difference between each pixel in the current observed image with pixel in the same position in the background model. The data term and motion information about the moving objects comes from this step.

3. Perform color clustering for the current observed image. Motion information obtained from step 2 can also be added to the color features in order to get clustering result more related to the background subtraction aim. The L_1 color separation term is built using the clustering result in this step.

4. Build the smoothness term in current observed image to penalize discontinuities between neighboring pixels.

5. Use one graph cut to do binary energy optimization for segmenting the current observed image into foreground and background.

6. Background model is updated for next observed image in the video.

7. Go to step 2 till the all the frames from the video are finished in background subtraction.

Figure 5.1 shows the flow chart of our algorithm.

We have described the background modeling methods in Chapter 2. The three background modeling methods used here are the mean filter, Gaussian average and Gaussian mixture model. These background modeling methods comes from previous background subtraction methods. As explained previously, background subtraction has two main parts, background modeling and foreground detection. These approaches have a lot of shortcomings in foreground detection process. Mean filter still uses a global threshold for all pixels in the image. And even a bigger problem, this threshold is not adapted in time. This might bring artifacts in the final background subtraction result. Though a different threshold is selected for each pixel by the Gaussian average and Gaussian mixture model and these pixel-wise thresholds are adapting in time, these methods do not take the spatial information into account, that is the neighboring pixels make decisions whether to be labeled background or foreground independently of each other. Background subtraction results produced by these traditional methods are shown in Chapter 2. Figure 5.2 shows the background subtraction result using Gaussian average model on a sequence of images. While we use relatively simplistic background modeling techniques, our algorithm can be easily adapted to use more advanced background models.

To achieve the tractable appearance based algorithm in [31], they replace the non-tractable volume-balancing term with application-dependent unary term. In this way, the energy is no longer NP-hard and therefore can be globally optimized with a single graph cut. When applying the framework from [31] to our background subtraction problem, the application-dependent unary term is set as a data term, which is related to the motion information computed from the current frame and background model.

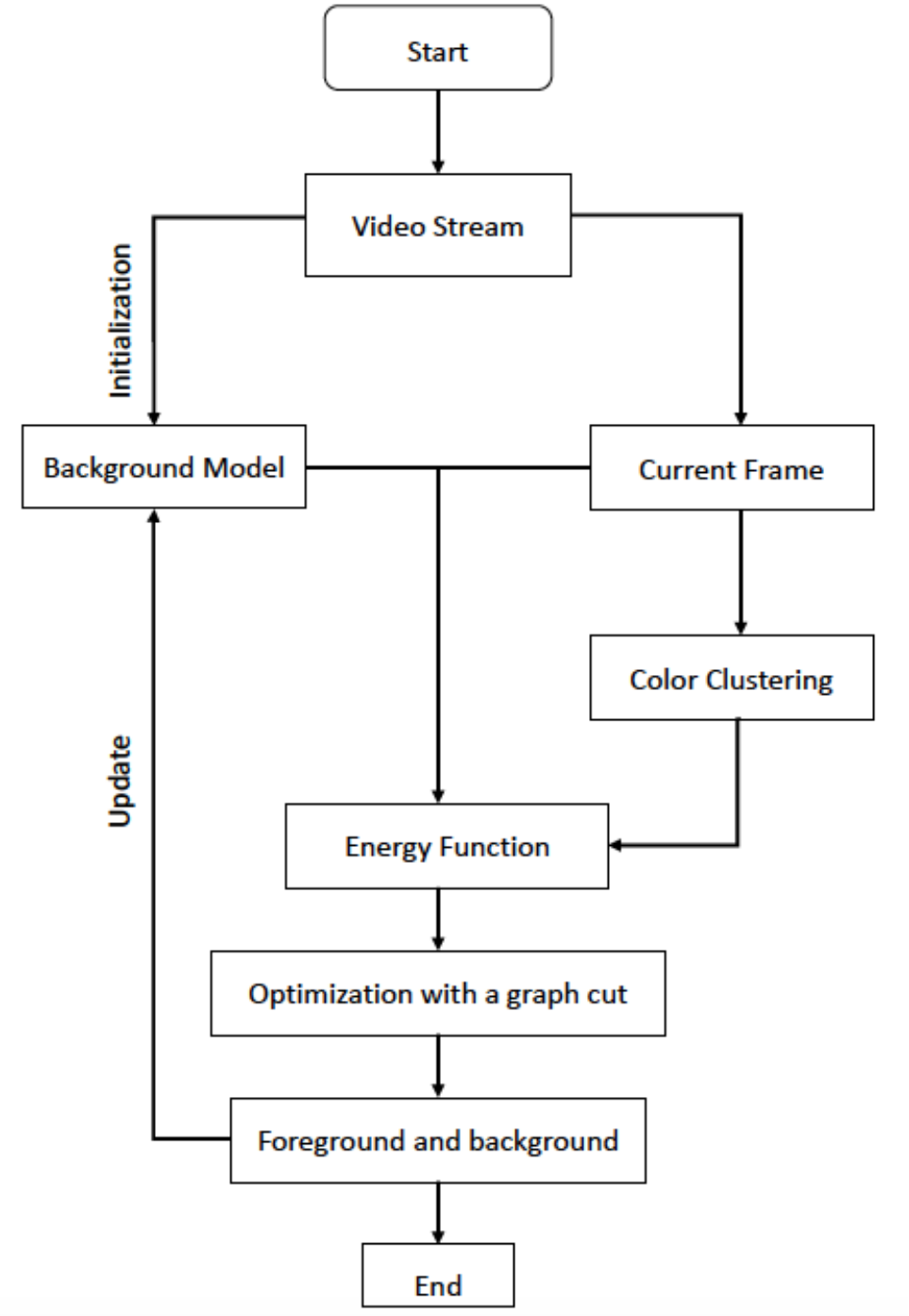


Figure 5.1: The flow chart of the algorithm

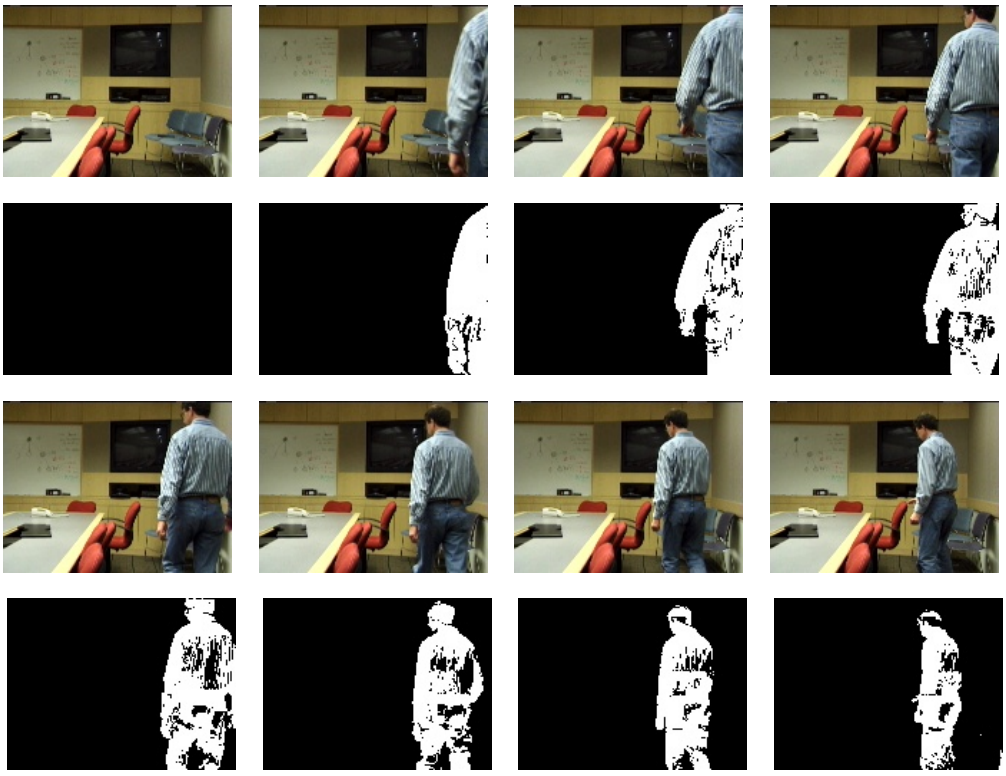


Figure 5.2: An example of background subtraction using Gaussian average model: a sequence of images from a video stream (first row and third row) and corresponding background subtraction results (second row and last row).

Let us consider one pixel p in the current frame and p' is its corresponding pixel at the same position in background image. If the intensity or color information changes significantly, the pixel is more likely to be in the foreground. These pixels form a foreground mask. Otherwise, if the difference is little, p is more likely to be in the background. These pixels form a background mask. The difference $|I(p) - I(p')|$ gives information about presence of motion for pixel p, where $I(p)$ is the color or intensity of pixel p. In case of color, $|I(p) - I(p')|$ stands for taking the norm between the color vectors. Figure 5.3 is a visualization of the motion information from one image. In this example, we used the mean filter. With blue and red spectrum we show pixels with large $|I(p) - I(p')|$. With green and yellow spectrum we show pixels with smaller $|I(p) - I(p')|$. The data term prefers segments with larger foreground in the foreground mask and larger background in the background mask. An example of the foreground mask and background mask computed from Figure 5.3 by thresholding given in Figure 5.4. More details about specific data term is discussed in the following section.

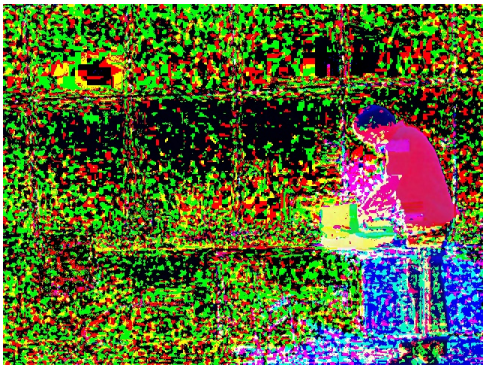


Figure 5.3: Illustrates motion information using the mean filter background modeling.

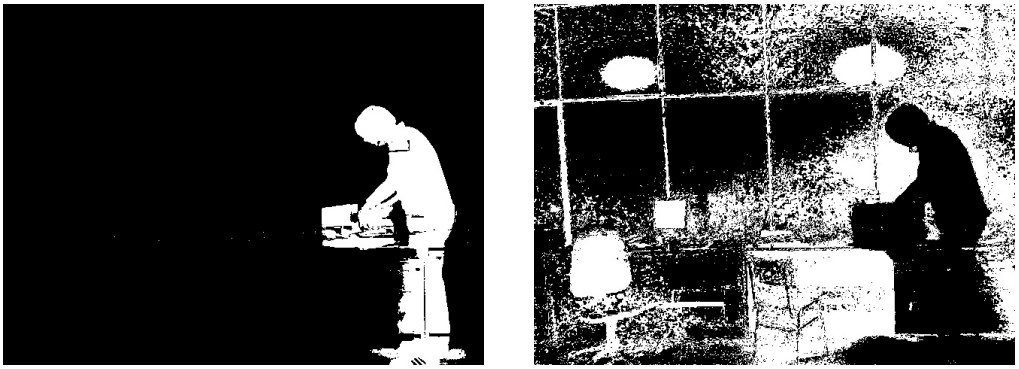


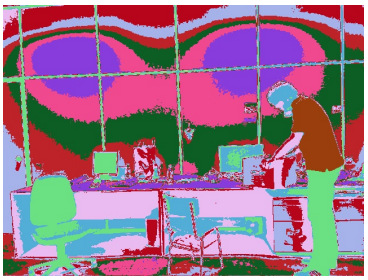
Figure 5.4: An example of the foreground mask (left) and background mask (right) using mean filter. White area in the two images represents the pixels that prefer to be foreground and background respectively.

Color information is very usefull to build appearance models. When the appearance model is known, an optiminal segmentation can be achieved by a graph cut. In this work, it is the most important part for forming the color separation term in the energy functon. The color

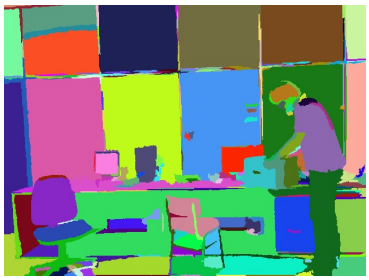
space should be handled efficiently due to its relatively high dimensionality. In [31] they adapt a simple way to handle color by using a binned histogram. The binning is done uniformly in each color channel. In this thesis, we evaluate several color clustering techniques. We also found it helpful to include the coordinates of a pixel into color clustering. Thus in addition to the standard color clustering techniques such as k-means, we evaluate some standard segmentation algorithms such as [11], which can be thought of as performing clustering with features consisting of the pixel color and pixel coordinates.

When the color space is partitioned into clusters (or bins), all pixels in an image are quantized. Including clustering information into the energy function can help enhance the foreground and background distinction. To observe the effect of different color space clustering algorithms, color space patitioning is performed by mean-shift, kmeans and an efficient graph-based segmentation algorithm [11] in this work. By including a new color separation term, the energy function prefers labeling pixels that are in the same color bin with the same label, as much as possible. Color quantization maps produced by different color clustering methods are shown in Figure 5.5.

However in some the background subtraction cases, the moving objects might have color similar to the background color. Thus to obtain more accurate background subtraction results, motion information should be helpful to include into the clustering process, as a feature, in addition to color and coordinates. We enhance our feature vectors with motion information obtained in part 2, and perform clustering.



(a) color space clustering using kmeans



(b) color space clustering using meanshift

(c) color space clustering using an efficient graph-based segmentation algorithm

Figure 5.5: Color clustering results using different algorithms.

Since the boundary information is also very important in obtaining an accurate segmenation result, a smoothness term is required in the energy function. This term gives a penalty for assigning nearby pixels different labels. Hence, the penalties all lie on the boundary of the segmentation. The boundary smoothness term prefers the segments with a shorter ”length” along the boundary. The penalty cost in this work is contrast-sensitive. It means that the penalty cost is inversely proportional to the color (or intensity) difference between pixel pairs. Figure 5.6 shows the visualization of edge contrast between neighboring pixels.

Section 5.2 provides a more detailed description on the energy function, the construction of the graph and corresponding weight of graph edges.

(a) original image (b) visualization of edges

Figure 5.6: An example of the visualization of edge contrast between neighboring pixels.

5.2 Graph Construction for Energy Function

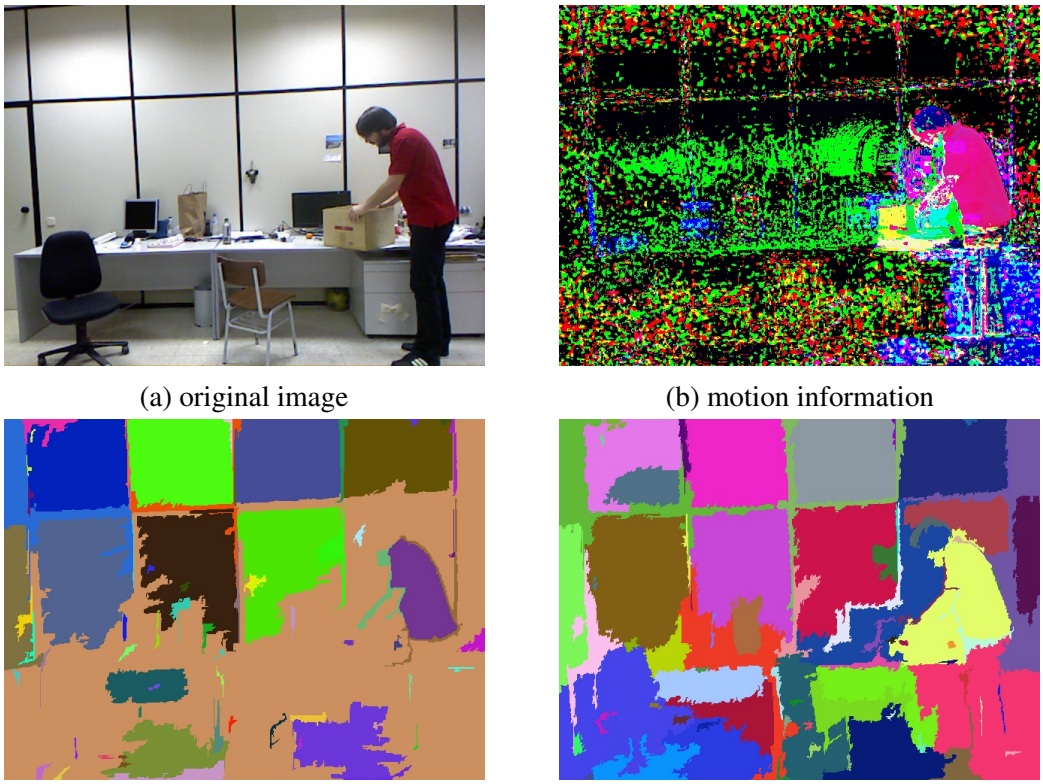
In this section, we apply the energy optimization framework to the problem of foreground segmentation in the background subtraction framework. It is a binary labeling problem. The goal is to segment the current image into two sets for each frame in a video stream, the foreground and background. As explained previously, a new energy term explicitly measuring the L_1 distance between foreground and background color clusters is added in the energy function to minimize the appearance overlap between the foreground and background regions. To change the NP-hard problem to a tractable one, the non-tractable volume-balancing term should be replaced with an application-dependent unary term, so that the energy function can be optimized in one graph cut. In this work, we use our background subtraction based unary data term in the function. The complete energy function $E(f)$ is given by:

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) - \beta |\theta^f - \theta^{\bar{f}}| + \lambda \sum_{pq \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{5.1}$$

where $\mathcal{N}$ is a set of neighboring pixels in the 4-neighborhood system, $\mathcal{P}$ is a set of image pixels, θ^f and $\theta^{\bar{f}}$ are unnormalized color histograms for foreground and background regions, respectively. $D_p(f_p)$ is the data term, $|\theta^f - \theta^{\bar{f}}|$ is the L_1 color separation term for all color bins and $V_{pq}(f_p, f_q)$ is the smoothness term for a pair of pixels. Parameters λ and β are the relative weights of smoothness term and color separation term in the energy function.

In this work, the motion information obtained from the background image is used for two purposes. First, it is used to form the data term in the energy function, which connects the background subtraction problem to energy minimization framework. That is to say, the pixels with large motion indicators are more likely to be in the foreground. Second, the motion information is also used for color quantization. When taken into account it can help to reduce the color appearance overlap between moving objects and background. This is especially helpful when a moving object has colors similar to that in the background. In this case, these object pixels can get clustered into the bin with other object pixels that have a similar motion, rather than to the bin with the background pixels of similar color. Figure 5.7 shows an example of color clustering results using the efficient graph-based segmentation algorithm with and without motion information. The parameters are the same in the two cases. We can see that many

parts of the person is clustered into one brown color bin as the background before adding the motion information. After the motion information is added, the color bins within the person are clustered more accurately with different color bins from most part of the background, such as the yellow color bin.



(a) original image

(b) motion information

(c) color clustering result without motion information

(d) color clustering result including motion information

Figure 5.7: An example of color clustering results using the efficient graph-based segmentation algorithm.

5.2.1 Data Term

This section explains the data term in the energy function. In this work, we use mean filer, Gaussian average and Gaussian mixture background models to build the data term. However, different background modeling methods have different measurements. First, they compute the motion information of pixels in different ways, because the background image varies between different background modeling approaches. The motion information here can be defined as the absolute difference between the observed current image and the background image. Denote $I(x, y, t)$ as the intensity of a pixel at position (x, y), time t in current image. Similarly, we use multiple component color spaces, (L, A, B) instead of intensity. $B(x, y, t)$ is the intensity of the pixel at the same position in the current background image. The motion information for this pixel is $|I(x, y, t) - B(x, y, t)|$. Pixels with high information of motion are more likely to

be in the foreground. Otherwise, pixels with low information of motion are more likely to be in the background. But how to measure the degree of motion differs in background modeling methods. Mean filer just compares the motion information with a global threshold which is the same for all pixels in the image. Gaussian average and Gaussian mixture model compares the motion information with the standard deviation for each pixel, which can be updated with time. We explain the data term for different background modeling methods separately.

As introduced in chapter 2, mean filter computes the background model by computing the arithmetic mean (or weighted mean) of the pixels between successive images. So the background model B is defined by:

$$B(x, y, t) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} I(x, y, t - i) \tag{5.2}$$

Once we get the background model, we apply a threshold $Th1$,

$$|I(x, y, t) - \frac{1}{n} \cdot \sum_{i=0}^{n-1} I(x, y, t - i)| > Th1 \tag{5.3}$$

where $I(x, y, t)$ is the intensity or color of pixel (x, y) at time t. The results of thresholding in Equation 5.3 can be used as an indicator of the presence of motion. Those pixels that pass the test in Equation 5.3 are likely to be the foreground.

By applying thresholding in different direction and with a possibly different value, we can find a set of pixels that are likely to be the background:

$$|I(x, y, t) - \frac{1}{n} \cdot \sum_{i=0}^{n-1} I(x, y, t - i)| < Th2 \tag{5.4}$$

Pixels that satisfy equation 5.3 form a foreground mask O and pixels that satisfy equation 5.4 form a background mask N. The foreground mask and background mask determined by the mean filter modeling method are used to model the data term. Recall that $f_p = 0$ means the background and $f_p = 1$ means the foreground. Then the data term for foreground and background are defined as:

$$D_p(0) = \begin{cases} Z & \text{if } p \in O \\ 0 & \text{if } p \in N \end{cases} \tag{5.5}$$

and

$$D_p(1) = \begin{cases} Z & \text{if } p \in N \\ 0 & \text{if } p \in O \end{cases} \tag{5.6}$$

where Z is a positive penalty value. The capacity of the $t - links$ between terminal sink t and the pixels is set as $D_p(1)$ and the capacity of the $t - links$ between terminal source s and the pixels is set as $D_p(0)$.

For Gaussian average modeling method, only two parameters (μ, σ) need to be stored. Parameter μ is the average value of previous pixels, which represents the background model.

Parameter σ is the standard deviation. The maintance of μ and σ has been introduced in chapter 2, we don't show much details here. At each t frame time, then we can get the estimate of foreground at the current image pixel by finding pixels that pass the test below,

$$|I(x, y, t) - \mu(x, y, t)| > k\sigma(x, y, t) \tag{5.7}$$

Pixels that do not pass the test in Equation 5.7 are more likely to be in the background. Parameter k in the inequality 5.7 is usually chosen in the range from between 2 and 3. In this work, we set $k = 3$.

For the Gaussian model, instead of using the same penalty for all pixels passing a threshold test, we now use a different way to model the data term. In particular, our data term is:

$$D_p(0) = |I(x, y, t) - \mu(x, y, t)| \tag{5.8}$$

$$D_p(1) = k\sigma(x, y, t) \tag{5.9}$$

where the position of pixel p is (x, y). The model in equations 5.8 and 5.9 imposes a small penalty for pixels with intensity not far from the mean to be the background. Pixels with standard deviation far from the norm are encouraged to be the foreground. The capacity of the $t - links$ between terminal sink t and the pixels is set as $D_p(1)$ and the capacity of the $t - links$ between terminal source s and the pixels is set as $D_p(0)$.

Our last model, the mixture of Gaussians is handled similarly to the single Gaussian model just discussed. Suppose we have K existing Gaussians, then $\mu(i, x, y, t)$ and $\sigma(i, x, y, t)$ are the two parameters of the i^{th} Gaussian function with a weight $w(i, x, y, t)$. All the K weights $w(i, x, y, t)$ are normalized so they sum up to 1. The updating of Gaussian mixture model is explained in chapter 2. The weighted average value of previous pixels $\mu(x, y, t)$ and the weighted standard deviation $\sigma(x, y, t)$ are computed as below

$$\mu(x, y, t) = \sum_{i=1}^{K} w(i, x, y, t) \cdot \mu(i, x, y, t) \tag{5.10}$$

$$\sigma(x, y, t) = \sqrt{\sum_{i=1}^{K} w(i, x, y, t) \cdot \sigma(i, x, y, t)^2} \tag{5.11}$$

To build the data model, we use, as before:

$$D_p(0) = |I(x, y, t) - \mu(x, y, t)| \tag{5.12}$$

$$D_p(1) = k\sigma(x, y, t) \tag{5.13}$$

The capacity of $t - links$ is also set in the same way.

5.2.2 Color Separation Term

The color separation term used in this work is based on previous work of [31]. They choose to use a binned color histogram with different bin sizes to handle the color space. They use a uniform binning of the color space, using separate binning in each color channel. Each pixel is

assigned to the bin that its color falls into. Bin size has to be carefully chosen. If the bin size is too small, the histogram may be too sparse and unreliable. If the bin size is too large, then many pixels from foreground and background may fall into the same bin, and thus encouraging a color bin to be either fully assigned to the foreground or background is not helpful.

To obtain a smaller size histogram with more reliable bin counts, one can cluster the color feature using clustering algorithms such as kmeans. Using k-means has an advantage over uniform color binning as it adapts to the color space more closely. Unlike uniform binning, color clustering based on kmeans does not produce many empty bins [33]. In this thesis, we evaluate three different algorithms to partition the color space into reliable clusters. K-means [26], mean shift [9] and an efficient graph-based image segmentation algorithm [11] are performed. Notice that we also add the coordinate feature to image pixels when performing color clustering using the image segmentation algorithm in [11]. This enriched space is more helpful for separating moving objects from the background, since moving objects have coherence in space.

Color quantization for all pixels is achieved from the clustering results by these techniques. If we have $1, 2, ..., k$ clusters (or bins) for an image, each pixel from that image is quantized to its corresponding bin from $bin_1, bin_2, ...bin_k$ as a result. Different color clustering methods may bring different distributions between the pixel and corresponding color bin.

Considering that we are dealing with the background subtraction problem, adding the motion information in the clustering process should also help to improve the performance. We should prefer to get clusters that contain pixels moving similarly, since these clusters are more likely to contain only the foreground pixels (i.e. with large motion) or the background pixels (i.e. with small motion). Thus we also include motion information as a feature during the clustering step.

For the k-means clustering algorithm, including the motion information is straightforward. The k-means clustering is a general algorithm that clusters k-dimensional feature vectors. Color is a 3 dimensional feature based on LAB space. In our case, motion information is a three dimensional feature in the LAB space. More specifically, our motion features is the difference between the current frame and the background model, taken in each channel independently. Thus we simply add our 3D motion feature to the 3D color feature getting a 6D combined feature vector. Let $p(x, y)$ be a 3-dimensional color feature vector of an image at position (x, y), we denote the motion information at (x, y) is $m(x, y)$, which is also a 3-dimensional feature. Then for each image the k-means clustering algorithm is performed on a set of 6-dimensional feature vectors $n(x, y) = (p(x, y), m(x, y))$. The kmeans algorithm was explained in Chapter 3. Note that our motion feature is only sensitive to the presence of motion. It does not estimate neither direction nor the magnitude of motion.

For the efficient graph-based image segmentation algorithm, as described in Chapter 3, there are two dissimilarity mesurements, internal difference and external difference. As shown in Equation 3.6 and 3.7, both of them are based on the weight between two neighboring pixels. Therefore, the weight is measured by using the color information and motion information of two neighboring pixels instead of color information only. We compute the weight between two vertices $w(v_i, v_j)$ as

$$w(v_i, v_j) = \sqrt{||p_i - p_j||^2 + ||m_i - m_j||^2} \tag{5.14}$$

where p_i and p_j denote 3-dimensional color features at pixel i and j, m_i and m_j denote 3-dimentional motion features at pixel i and j. Because all the measurements are computed from the edge weight, all the steps to perform this method are exactly as in Algorithm 1.

The clustering results are now more reliable to enhance background subtraction performance. We will show the impact of clustering methods towards the final background subtraction results in the next chapter. It is obvious that the color separation term encourages pixels from the same color bin assigned to the same label. The L_1 color separation term used in this work is the one introduced by Tang et al. [31], which is

$$E_{L_1}(\theta^f, \theta^{\bar{f}}) = -|\theta^f - \theta^{\bar{f}}|_{L_1} \tag{5.15}$$

where θ^f and $\theta^{\bar{f}}$ denote the color histograms of the foreground and background respectively.

For more intuition, we can further rewrite the L_1 term as,

$$E_{L_1}(\theta^f, \theta^{\bar{f}}) = \sum_{k=1}^{K} min(n_k^f, n_k^{\bar{f}}) - \frac{|\mathcal{P}|}{2} \tag{5.16}$$

where K means the number of clusters produced by the clustering algorithm. Here n_k^f and $n_k^{\bar{f}}$ are the number of foreground and background pixels in cluster k respectively. It is easy to see that L_1 color separation term gives penalty for pixels in the same cluster with different labels. That is to say, labeling continuity is encouraged among pixels in the same cluster. If the pixels in cluster k are separated into foreground and background with n_k^f and $n_k^{\bar{f}}$ pixels inside two parts respectively, $min(n_k^f, n_k^{\bar{f}})$ is the number of links that need to be cut in graph cut. The L_1 color separation is encouraged in this way for the labeling achieving the minimum energy. Figure 5.8 shows the details of graph construction for the color separation term when optimizing the L_1 color separation term in one graph cut. The capacity of $a-links$ between the auxiliary node and the pixels is set as the weight of color separation term β.

5.2.3 Smoothness Term

The smoothness term in energy function in Equation 4.2 is the same as the traditional smoothness term proposed in [5]. It gives penalties for discontinuities between pixels p and q. The penalities all lie on the segmentation boundary, so the length of the boundary is encouraged to be small in energy minimization. More specifically, the more similar the two neighboring pixels, the more penalty cost is given if they are assigned to different labels in segmentation. Otherwise, if the two neighboring pixels are different in appearance, it is more possible that they belong to different labels, so the penalty cost for assigning different labels to them is low.

The similarity between pixels can be measured by intensity, color, texture and etc. In this work, smoothness term is constructed by comparing the color information in Lab space of two neighboring pixels. It is a non-increasing function of the color difference in Lab space between neighboring pixels. The smoothness term is:

$$E(f) = \sum_{pq \in \mathcal{N}} w_{pq} \cdot \delta(f_p \neq f_q) \quad = \sum_{pq \in \mathcal{N}} exp(-\tfrac{\triangle I^2}{2\sigma^2}) \cdot \delta(f_p \neq f_q) \tag{5.17}$$

In four-nighborhood system, the distance of all pixel pairs is equal. Therefore we do not need to consider the term $\frac{1}{dist(p,q)}$ in Equation 5.17. In Equation 5.17 $\triangle I$ denotes the color

difference of two neighboring pixels in Lab space, $\triangle I^2 = \|I_p - I_q\|^2$, where $I_p = (L_p, a_p, b_p)$ and $I_q = (L_q, a_q, b_q)$ are three-dimensional feature vectors in Lab space, when L stands for intensities and A, B stand for color opponent dimensions. Parameter σ^2 is the variance of color difference of pixels in Lab space. Intuitively, in this function, if the color difference between neighboring pixels is smaller than average $|I_p - I_q| < \sigma$, high penalty is given for assigning different labels to them. In contrast, if the color difference between neighboring pixels is larger than average $|I_p - I_q| > \sigma$, the function gives low penalty for labeling inconsistency between them.

Function δ is defined as:

$$\delta(f_p \neq f_q) = \begin{cases} 1 & \text{if } f_p \neq f_q \\ 0 & \text{if } f_p = f_q \end{cases} \tag{5.18}$$

It means that the function does not penalize the case when two pxiels are assigned the same label. In contrast, when the labels of two neighboring pixels are different, penalty is given as w_{pq}. The details of the graph construction for the smoothness term are shown in Figure 5.8. The capacity of $n-link$ between pixel p and pixel q is set as $w_{pq} \cdot \delta(f_p \neq f_q)$.

The graph construction for our energy in Equation 5.1 is shown in Figure 5.8. Each pixel is represented as a node in a four connected graph. There are also two terminal nodes in the graph, source node s and sink node t. The auxiliary nodes are added into the graph, one for each color cluster. Each of the auxiliary node represents a corresponding color bin produced by color clustering. All the nodes in the graph are connected by links. All the pixel nodes are connected to terminal nodes through $t-links$. The way of setting $t-links$ value is through motion information of foreground and background in this work. Foreground has larger motion value and background has smaller motion value. The $t-links$ here corresponds to construction the data term. The capacity of the $t-links$ between sink and pixels is set as $D_p(1)$ and the capacity of $t-links$ between source and pixels is set as $D_p(0)$. For different background modeling methods, $D_p(1)$ and $D_p(0)$ are set to different values, as we described before in section 5.2.1. Each pixel is linked to its neighboring pixels through $n-links$ in the four neighborhood system. The weight of $n-links$ can be contrast-sensitive. If the pair of pixels are similar, the $n-links$ value between them is strong and vice versa. The similarity is measured by color information in this work. $n-links$ here corresponds to the construction of smoothness term. Each $n-link$ between a pair of neighboring pixel p and q is set as $w_{pq} \cdot \delta(f_p \neq f_q)$, where w_{pq} is evaluated by color difference between two pixels, f_p and f_q are the labels of two pixels. The specific value is illustrated before. Each pixel in the graph is connected to its corresponding auxiliary node by $a-links$. Each auxiliary node represents a color bin. These $a-links$ connect the auxiliary node to all pixels that belong to certain color bin. It is obvious that $a-links$ helps labeling consistency among pixels in the same color bin. The capacity of these links is the weight of appearance overlap term. The $a-links$ here corresponds to the construction of L_1 color separation term. Each $a-link$ is set as the weight of color separation term β.

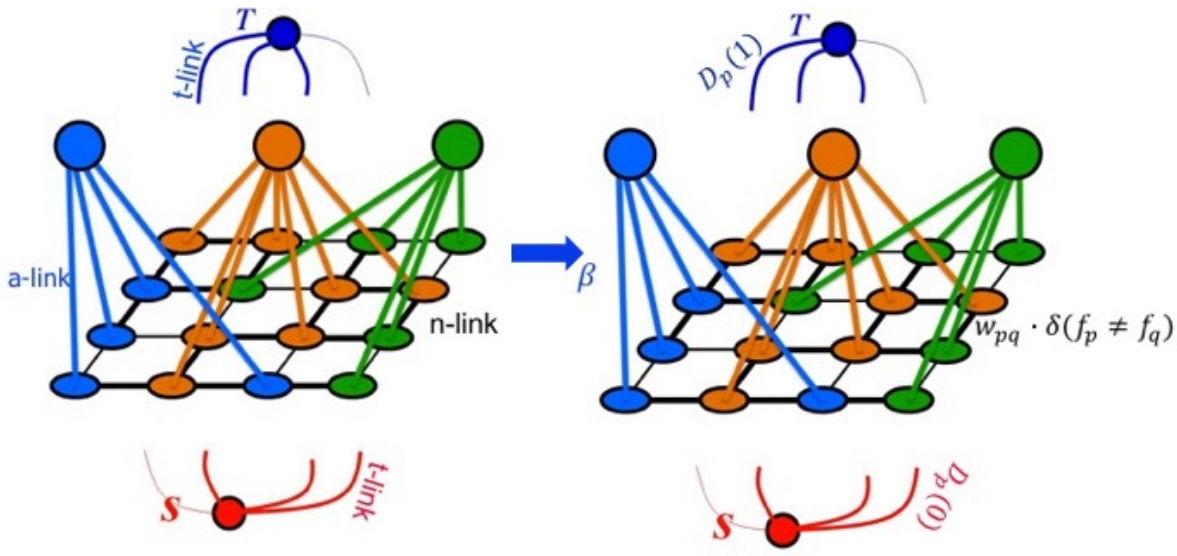


Figure 5.8: Overall graph construction for energy with L_1 color separation term.

Chapter 6

Experimental Results

As explained in Chapter 5, our approach to background subtraction depends on a particular choice for background modeling and color clustering. In this chapter, we evaluate several methods for background modeling and color clustering. In addition, we discuss the effect of adding motion information in color clustering and its influence on background subtraction results. We also provide implementation details and experimental results of our approach including some failure cases.

6.1 Image Database

There are many available datasets for background subtraction taking under conditions of various complexity. In order to be able to handle more complex background subtraction settings, we would need to consider more complex background models. However, the main goal of this thesis is to show the usefulness of the color separation term. Thus we restrict the background modeling methods to the relatively simple models discussed in Chapter 2 and evaluate our method in only moderately complex settings. Our approach can be extended to more complex background subtraction settings by incorporating more complex background modeling techniques.

In order to build a dataset that can convincingly prove the usefulness of color separation and make wide comparisions, we build our own dataset by selecting sequences from other datasets and also recording a test video by ourselves. We select sequences with distinct colorful moving objects. First, one general sequence is selected from the object detection dataset used in [7]. This dataset includes four different sequences that contain challenging phenomena such as cast shadows, color and depth camouflage. For each sequence hand-labeled ground truth is provided in order to test background subtraction algorithms. We choose the basic indoor environment sequence from [7].

Second, one of the most popular datasets to test background subtraction algorithms is the wallflower dataset [32]. It contains seven different test sequences that represent a different, potentially problematic scenario for background subtraction. Each sequence has a hand-segmented evaluation image. We test our algorithm on two sequences of this dataset. Also, we record a video stream and use it as another sequence in our dataset. Each frame has a hand-segmented ground truth constructed by ourselves. Because the ground truth is obtained by

using photoshop, there may be small labeling errors. Figure 6.1 shows several sample images and the corresponding ground truth from all the sequences in our dataset.

(a) (b) (c) (d)

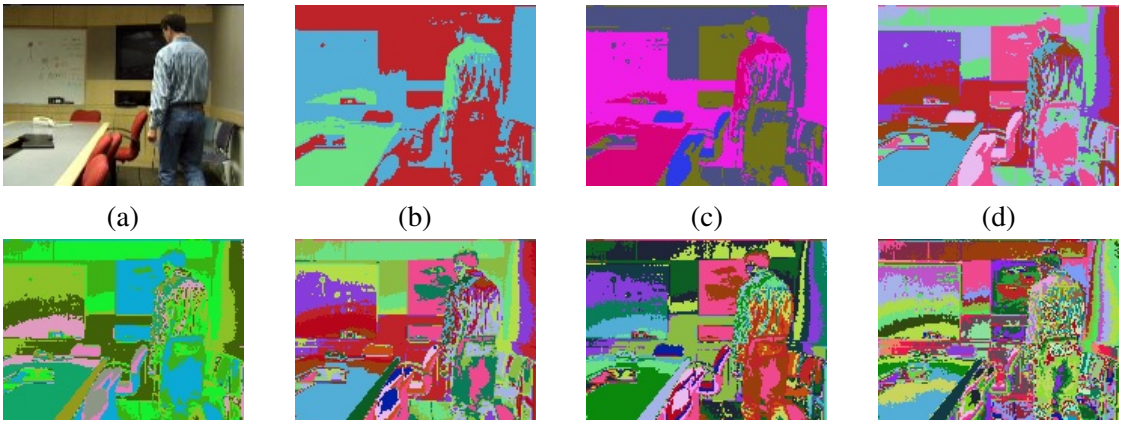
Figure 6.1: Sample frames (top) and the corresponding ground truth (bottom) from our dataset: (a) a general sequence from object detection dataset used in [7], (b)-(c) sequences from wallflower dataset [32], (d) sequence obtained by ourselves.

6.2 Parameter Selection

In order to implement a high accuracy background subtraction approach, besides algorithm design, parameter selection is another challenge. There are multiple parameters to be set both at the background modeling and color clustering stages. In addition, there are the weights in the energy function that control the relative strengths of the data and color separation terms. We use a grid search over parameters to find a setting that works best for all the sequences. Below we list the parameters that need to be searched for.

Parameters in Kmeans

There is only one parameter in Kmeans, namely the number of clusters K. The number of clusters determines how many disjoint clusters the image colors are divided into. If the number of clusters K is too large, then there may be too many color bins, and therefore, too few pixels for each bin. This means that the moving object is broken into too many small clusters, and thus the color clustering term will have no significant influence for the background subtraction. On the other hand, if the number of clusters is too small, a large portion of pixels from the background and object may fall into the same color bin. In such a case, including a color separation term will worsen the performance, as it will encourage pixels that should not have the same label (i.e. the pixels from background and foreground that fall into the same color cluster) to have the same label. So in the final algorithm, we need to set cluster number K to an appropriate value. A grid search is performed for all the possible values of K. Figure 6.2 shows the Kmeans clustering results with different number of clusters.



(a) (b) (c) (d)

(e) (f) (g) (h)

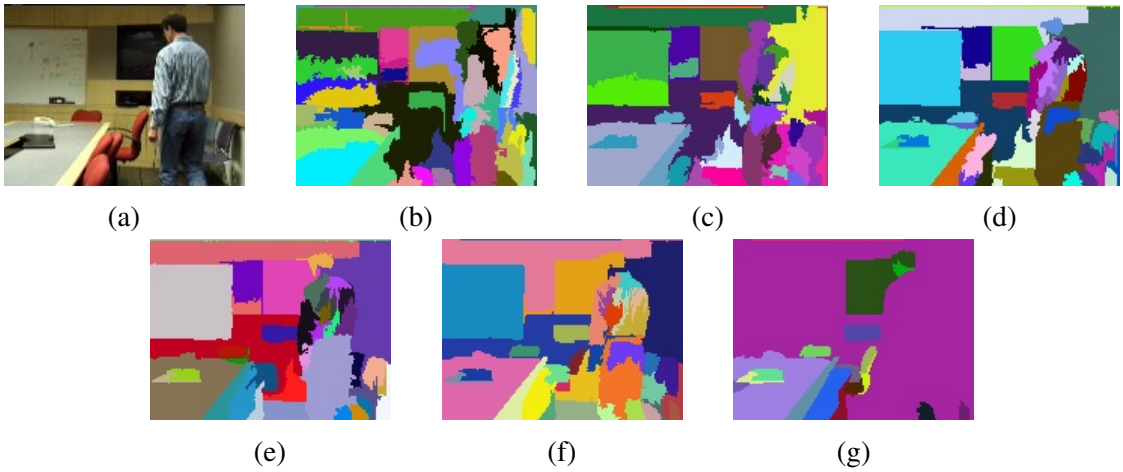
Figure 6.2: Example of kmeans clustering results with different number of clusters: (a) original image, (b)-(g) kmeans color clustering with number of clusters 3, 5, 8, 10, 15, 20 and 50 respectively.

Parameters in Meanshift

In mean shift clustering, there are two main parameters, namely the spatial bandwidth (h_s) and the color bandwidth (h_c). These two parameters in the algorithm have a large effect on the clustering results. By controlling the bandwidth parameter $h = (h_s, h_c)$, we can determine the resolution of the mode detection. This mode represents a cluster. In this work, we use the LAB color space, and these two parameters are different width in LAB (color feature) and XY (spatial feature) parts of the space. Adding spatial features helps coherence in the image domain. The bandwidth parameter selection is critical. They indirectly control the number of clusters and thus have an effect on the construction of color separation term in background subtraction. They cannot be too small or too large. Figure 6.3 shows examples of mean shift clustering with different color bandwidth (h_c). Slight changes in color bandwidth parameter can make large differences in the clustering results. We do grid search for all the combinations of possible h_s and h_c and select the settings with best performance. Figure 6.3(a) shows the original image. When the color bandwidth is too small, we get an over-clustered image in Figure 6.3(b). As we increase the value of color bandwidth, we get reasonably intuitive clustering results with clear objects boundary in Figure 6.3(e). But Figure 6.3(g) is under-clustered so that the foreground and background objects are merged together in one cluster, making such result unsuitable for the use in the color separation term.

Parameters in Efficient Graph-based Image Segmentation Algorithm

There are three parameters in the graph-based image segmentation algorithm [11]. One parameter σ is for Gaussian filter used first to remove noise and smooth the image before segmentation. Another parameter is a threshold parameter *thresh* which determines the size of clusters. Bigger threshold means bigger clusters. There is one more parameter that sets the minimum allowed cluster size. Any cluster less than the allowed size is merged to a neighboring cluster.

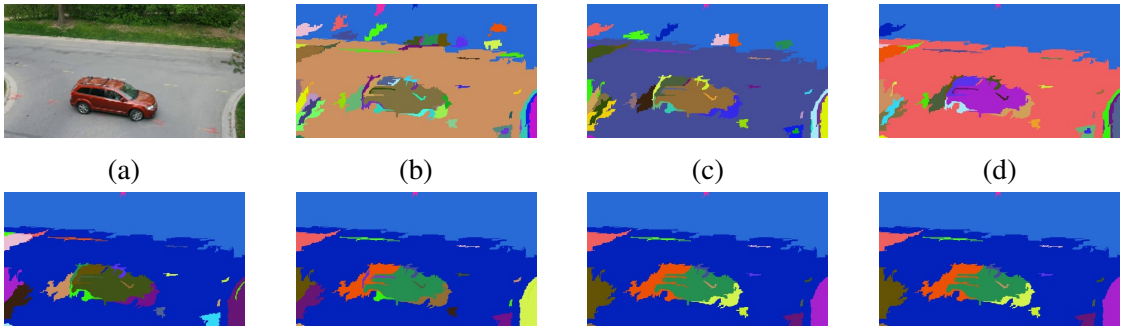


(a) (b) (c) (d)

(e) (f) (g)

Figure 6.3: Example of meanshift clustering results with different color bandwidths: (a) original image, (b)-(g) mean shift clustering using scale bandwidth 10 and color bandwidths 3, 5, 6, 6.5, 8 and 10 respectively.

The minimum segment size and σ do not have a significant effect on the clustering results. Therefore we set them to default recommended values, namely $\sigma = 0.8$ and minimum segment size set to 50. The threshold parameter *thresh* has a significant effect, and therefore this is the paramter we search over. Figure 6.4 shows different clustering results with different cluster threshold values.



(a) (b) (c) (d)

(e) (f) (g) (h)

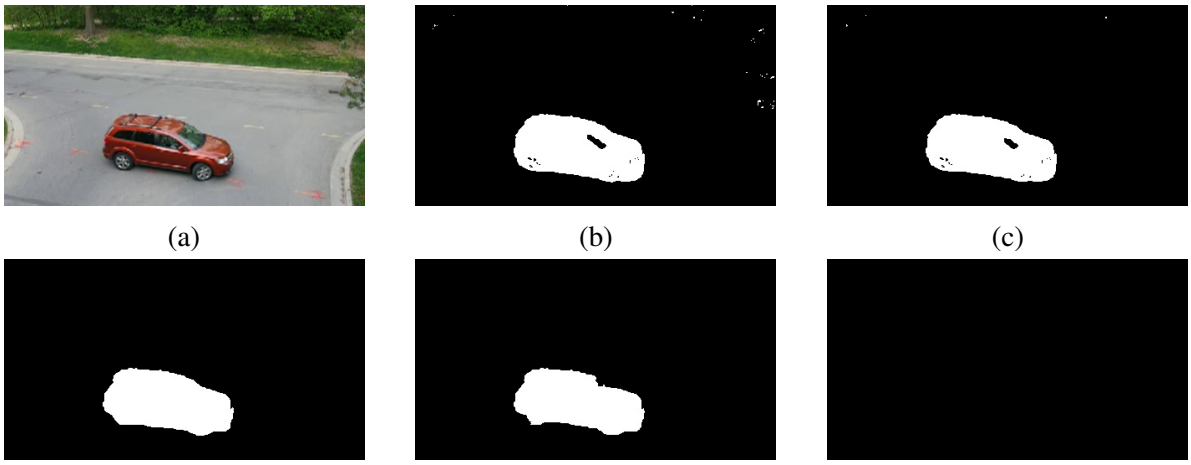
Figure 6.4: Example of an efficient graph-based clustering results with different color features: (a) Original image, (b)-(g) graph-based clustering using cluster threshold 100, 150, 200, 300, 500, 1000 and 1500 respectively.

Parameters in Energy Function

The parameters λ and β are the relative weights of the smoothness and color separation terms with respect to the data term. They determine how much each of these energy terms influence the resulting foreground segmentation. In Figure 6.5, the background subtraction results for

different λ are illustrated. Small λ will cause background subtraction result to be noisy as shown in Figure 6.5(b), while when λ is too large, the result is too smooth. In fact, for a very large λ the object is completely smoothed out from the result, see Figure 6.5(f).

In Figure 6.6, the background subtraction results for different β are illustrated. A large β will cause the background subtraction result to miss pieces of the object, as shown in Figure 6.6(d). There are many small color clusters that straddle both the background and the object along the object boundary, and these get segmented as the background, thus the object boundary is not smooth enough. While when β is too small, the color consistency term is loose, and some color clusters inside the car object get split between the foreground and background, resulting in the object that misses pixels, as in Figure 6.6(b).



(a) (b) (c)

(d) (e) (f)

Figure 6.5: Example of background subtraction results with different weight of smoothness term using mean filter background modeling techniques without the color separation term: (a) original image, (b)-(d) background subtraction results with $\lambda = 10, 50, 500, 2500, 3000$ respectively.

For different images in one video stream, the best λ and β are likely to be different. However, for an automatic system, we have to use the same parameter setting for each sequence. We perform a grid search to find the combination of parameters that work well, on average, for all frames. For the sequence shown in Figure 6.6 we choose $\lambda = 100$ and $\beta = 100$ for all the frames.

6.3 Evaluation Methods

To evaluate the performance of our background subtraction algorithm, we choose two commonly used metrics, error rate and F-measure. Error rate is a simple widely used evaluation method in image segmentation, which computes the ratio of pixels that are wrongly labeled and the total number pixels. The equation for the error rate is:

$$error = \frac{\text{FP} + \text{FN}}{\text{number of pixels}} \tag{6.1}$$

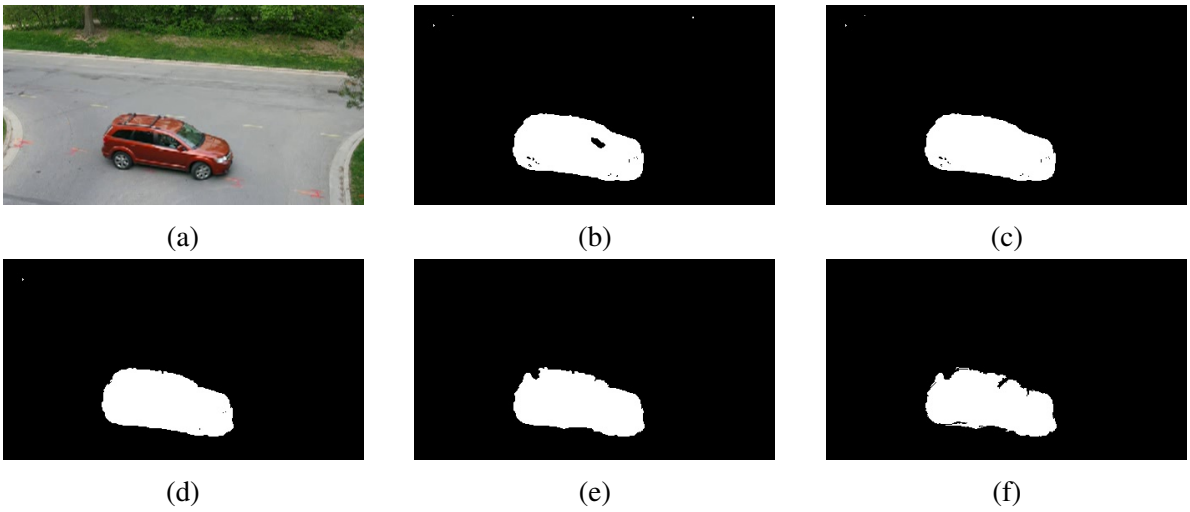


Figure 6.6: Example of background subtraction results with different weight of color separation term using mean filter background modeling techniques without motion information: (a) original image, (b)-(d) background subtraction results with $\lambda = 50$ and $\beta = 1$, 10, 100, 200, 1000 respectively.

where FP is the false positive number, which is the number of background pixels falsely labeled as foreground. FN is the false negative number, which is the number of foreground pixels falsely labeled as background in this case.

Another standard evaluation metric used in this thesis is the F-measure, which is calculated based on the precision and recall. Precision is defined as the fraction of elements correctly classified as positive out of all the elements the algorithm classifies as positive, whereas recall is the fraction of elements correctly classified as positive out of all the positive elements. The calculation of precision and recall are as follows,

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

where TP is the number of true positives, FP the number of false positives and FN the number of false negatives. Then, the F-measure is computed as below,

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{6.4}$$

where a positive real β is used to set the relative importance between precision and recall. In this work, we choose $\beta = 1$ to give recall as much importance as precision. This balanced F-measure metric is called F1 score. Notice that TN does not play a role in computing the F-measure.

6.4 Evaluation of the Results

Chapter 5 explains our approach based on color separation term for improving the background subtraction. Many background modeling techniques and color clustering algorithms have been tried in our experiments. They have a different effect on the performance of our approach. Thus, we have various combinations of these methods. In addition, the influence of motion information added in the color clustering process also needs to be evaluated. We compare the F-measure averaged on all frames in one image sequence for the results in this section.

Results without Color Separation and without Motion Information

We first discuss the results using different background modeling methods without the color separation and motion features. The comparison of the average error rate and F-measure are listed in Table 6.1. Gaussian mixture model performs better than the other two. Mean filter performs the worst. This is because the background model built by Gaussian mixture model is the most reliable out of the three background subtraction methods we evaluate.

method	error rate	precision	recall	F-measure
mean filter	0.02713	0.76141	0.986969	0.85964
Gaussian average	0.01694	0.837478	0.991106	0.907838
Gaussian mixture model	0.01582	0.851693	0.983295	**0.912775**

Table 6.1: Average error rate, recall, precision and F-measure of different methods in backgound modeling.

Results with Color Separation and without Motion Information

Second, the effect of color separation term is discussed. For clustering, we use the kmeans, meanshift, and the graph-based algorithm [11]. For background modeling, we use the same methods as previously. For each clustering algorithm, we compare the results with different background modeling techniques. Also, for each background modeling technique, we compare the results with different color clustering algorithms. Notice that all the parameters are set to those that produce the best results. The average error rate, precision, recall and F-measure results of all possible combinations between background modeling techniques and color clustering algorithms are listed in the following tables. Color appearance constraints help to get significantly better result compared to background subtraction results without color separation term in Table 6.1. When there is no color appearance constraint, the pairwise constraint can easily cause oversmoothing.

The results of using kmeans, meanshift and efficient graph-based color clustering algorithm for different background modeling methods are shown separately in Table 6.2, Table 6.3 and Table 6.4.

We can see from the three tables that no matter which color clustering algorithms is performed, the Gaussian mixture model still outperforms other two background modeling techniques in general. And the mean filter works the worst as well. Comparing the results from each row in the three tables, we can see the accuracy of different color clustering algorithms.

method	error rate	precision	recall	F-measure
mean filter	0.01300	0.889879	0.964928	0.925885
Gaussian average	0.008900	0.930622	0.96632	0.948135
Gaussian mixture model	0.006826	0.939391	0.98229	**0.960361**

Table 6.2: Average error rate, recall, precision and F-measure of different backgound modeling techniques using kmeans color clustering. This table shows the effectiveness of color separation.

method	error rate	precision	recall	F-measure
mean filter	0.018255	0.828367	0.987819	0.901093
Gaussian average	0.013988	0.863839	0.989869	0.92257
Gaussian mixture model	0.013236	0.873292	0.985809	**0.926226**

Table 6.3: Average error rate, recall, precision and F-measure of different backgound modeling techniques using meanshift color clustering. This table shows the effectiveness of color separation.

method	error rate	precision	recall	F-measure
mean filter	0.013330	0.878987	0.976026	0.924968
Gaussian average	0.011426	0.890111	0.986002	0.935606
Gaussian mixture model	0.009489	0.909311	0.985577	**0.945909**

Table 6.4: Average error rate, recall, precision and F-measure of different backgound modeling techniques using an efficient graph-based color clustering. This table shows the effectiveness of color separation.

Among the three algorithms, kmeans works the best, while meanshift is not that accurate compared to the other two.

Results with Color Separation and with Motion Information

Table 6.5 and Table 6.6 show the results of different background modeling techniques with color separation and motion information, using kmeans and an efficient graph-based color clustering methods separately. We did not perform experiments with adding motion to the meanshift algorithm as the meanshift performs the worst out of color clustering algorithms. Incorporating motion information in the color clustering process improves the result, because the motion information provides relevant information to clustering.

method	error rate	precision	recall	F-measure
mean filter	0.012653	0.889997	0.969529	0.928062
Gaussian average	0.003978	0.977778	0.974904	**0.976339**
Gaussian mixture model	0.005166	0.977872	0.960365	**0.96904**

Table 6.5: Average error rate, recall, precision and F-measure of different backgound modeling techniques using kmeans color clustering with motion information. This table shows the benefits of motion information.

method	error rate	precision	recall	F-measure
mean filter	0.012585	0.889003	0.97185	0.928528
Gaussian average	0.010443	0.897491	0.988902	**0.940982**
Gaussian mixture model	0.009440	0.91015	0.985113	**0.946149**

Table 6.6: Average error rate, recall, precision and F-measure of different backgound modeling techniques using an efficient graph-based color clustering with motion information. This table shows the benefits of motion information.

Comparing the F-measure results of color separation without motion information in Table 6.2 and Table 6.4 with the results in Table 6.5 and Table 6.6, we can see color separation with motion information can help background subtration to get a better performance. Generally speaking, kmeans color clustering can help to get the best performance both with and without motion features. Gaussian mixture model to build the background model tends to work the best in most cases. However sometimes the Gaussian average model is slightly better. Our best overall F-measure is 0.976339 building background model using Gaussian average, color clustering by kmeans with motion information, corresponding to the second row in Table 6.5.

6.5 Results Comparison

In this section, we present comparisons among all the combinations of different background modeling techniques and color clustering algorithms, including with and without motion information. We display just some sample video frames. Note that all images are generated from

the best parameter settings of different methods respectively. First, three different kinds of background modeling techniques, mean filter, Gaussian average and Gaussian mixture model will be compared. Second, the advantages of color separation will be discussed. In the end, the benefits of motion information included in color clustering will be explained. Since we have too many combinations, there are a lot of comparisons that could be done. Hence, we only show the representative ones as example. Some failure cases are shown in the next section.

Background Modeling Techniques Comparison

The average F-measure in Table 6.1 shows that background subtraction with Gaussian mixture model works slightly better than Gaussian average when there is no color separation term in the energy function. Although the background subtraction with mean filter performs, on average, the worst, there are still cases where mean filter works better than the other two background model methods. Figure 6.7, Figure 6.8 and Figure 6.9 show three examples of background subtraction results with different background modeling techniques without color separation. In the sequence as shown in Figure 6.7, Gaussian mixture model achieve best performance among all tree methods. However, Gaussian mixture is not all always the best. As shown in Figure 6.8, Gaussian mixture model works almost the same as Gaussian average. And in the last sequence in Figure 6.9 with a waving tree in the background, mean filter works better than the other two. The Gaussian mixture model tends to model all the moving pixels by different Gaussian distributions, so that the moving tree in the background is most likely modeled by one Gaussian distribution and is labeled as a moving object. In this case, Gaussian mixture model is overly sensitive to the background changes.

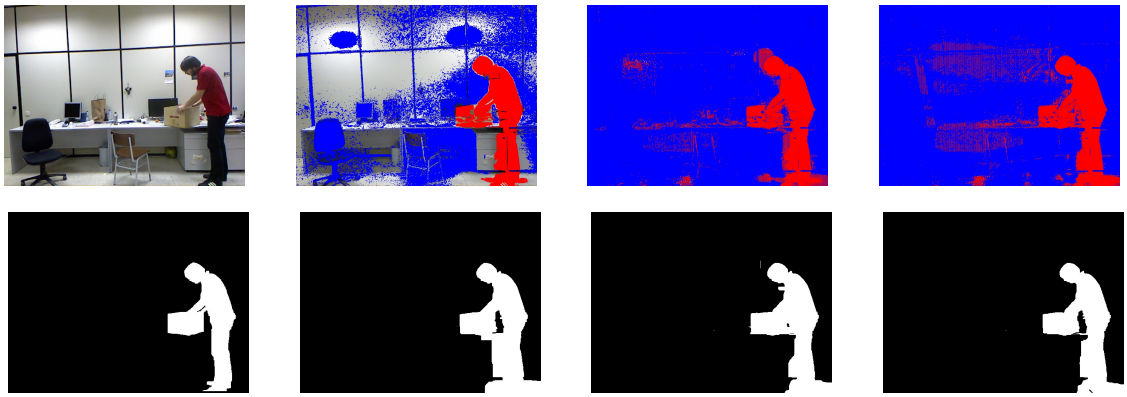


Figure 6.7: Background subtraction results on an indoor scene with different background modeling techniques without color separation term. First column shows the original image (top) and the corresponding ground truth (bottom). The second column is the data term mask (top) and background subtraction result (botton) using mean filter background modeling technique. Red is the foreground and blue is the background masks for the background modeling step. The third column is the data term mask (top) and background subtraction result (bottom) using Gaussian average background modeling technique. The fourth column is the data term mask (top) and background subtraction result (bottom) using Gaussian mixture model background modeling technique.

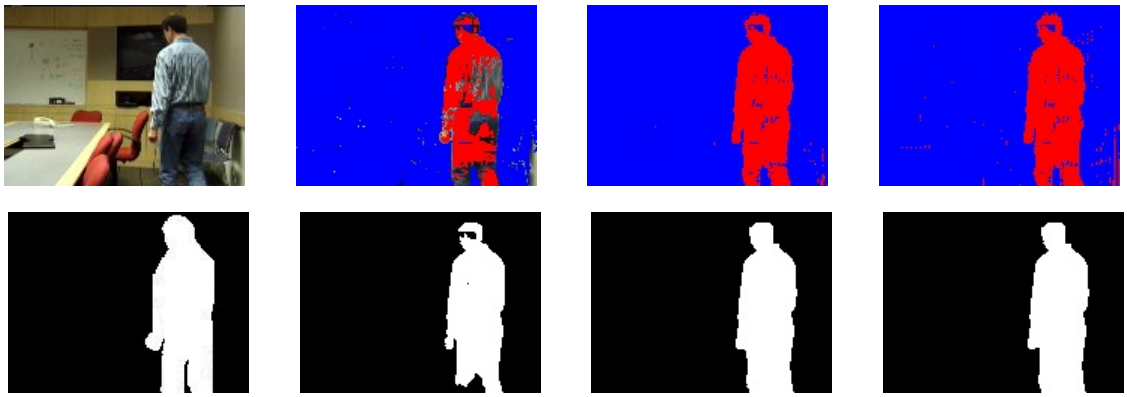


Figure 6.8: Background subtraction results on a moving person scene with different background modeling techniques without color separation term. First column shows the original image (top) and the corresponding ground truth (bottom). The second column is the dataterm mask (top) and background subtraction result (bottom) using mean filter background modeling technique. Red is the priori foreground information from the background modeling process and blue is the priori background information. The third column is the dataterm mask (top) and background subtraction result (bottom) using Gaussian average background modeling technique. The fourth column is the dataterm mask (top) and background subtraction result (bottom) using Gaussian mixture model background modeling technique.

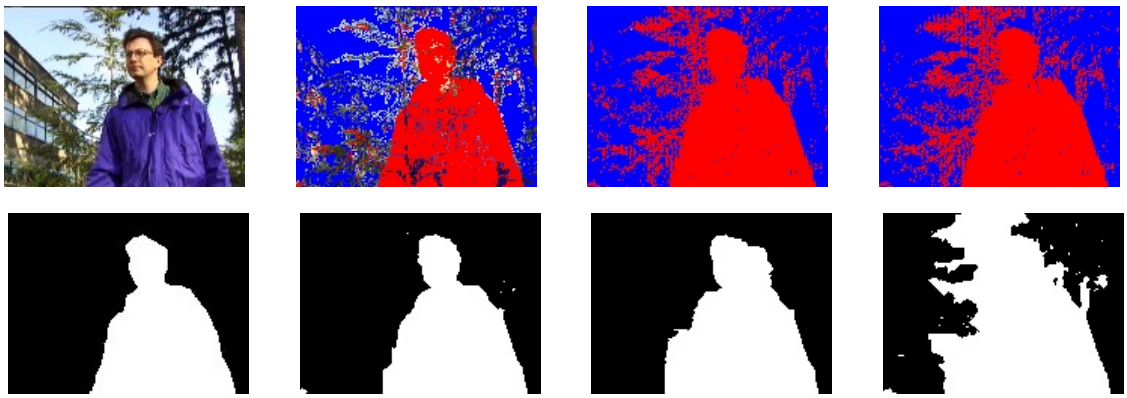


Figure 6.9: Background subtraction results on a waving tree scene with different background modeling techniques without color separation term. First column shows the original image (top) and the corresponding ground truth (bottom). The second column is the dataterm mask (top) and background subtraction result (bottom) using mean filter background modeling technique. Red is the priori foreground information from the background modeling process and blue is the priori background information. The third column is the dataterm mask (top) and background subtraction result (bottom) using Gaussian average background modeling technique. The fourth column is the dataterm mask (top) and background subtraction result (bottom) using Gaussian mixture model background modeling technique.

Comparison with Color Separation

When the color separation term is incorporated, it shows significant improvement over the background subtraction without color separation using the same background modeling technique. However, the efficiency of the color separation term also varies when using different color clustering algorithms. One example of the background subtraction results with different combinations of color clustering algorithms and background modeling techniques are listed in Figure 6.10. Since a relatively accurate result has been achieved even without the color separation term on the second sequence in Figure 6.8, the average F-measure is almost 0.97. Even though including the color separation term still helps, the small enhancement is not easy to notice in the foreground segmentation result. We don't show the result with color separation on this sequence here since we cannot observe the obvious changes easily. Also, a rather accurate result has been achieved without color separation by using mean filter as shown in Figure 6.9. The value of F-measure is 0.979 in this case. In contrast, on the same sequence, the method using Gaussian mixture model without color separation performs poorly with a F-measure of 0.75. The barrier here lies in the background modeling method. It is sensitive to a moving background. While adding color separation term helps and the F-measure increases, the improvement is only marginal. We show the test results on this sequence in Figure 6.11 which is implemented with Gaussian average.

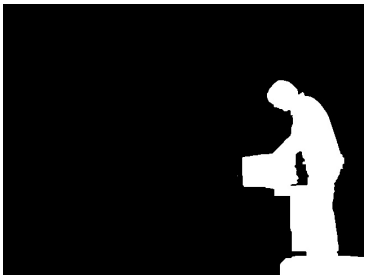
Comparing each row in Figure 6.10, we can see that all the methods with color separation outperform the one without color separation which uses the same background modeling method. And among the three color clustering algorithms, kmeans works the best in improving the performance, while meanshift seems not that powerful compared to the other two. Comparing each column in Figure 6.10, Gaussian mixture model seems more reliable in general.

It is obvious that adding color separation term in the energy function helps to enhance the foreground/background difference. But different color clusterings have different performance. As shown in Figure 6.11, kmeans makes background subtraction particularly accurate in this case. The F-measure reaches the highest 0.9846. Mean shift helps not much than the other two, but still helps a lot in improving accuracy.

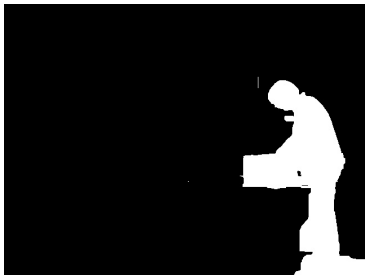
Comparision with Motion Information

To enhance the connection between background subtraction and color separation, we propose to include the motion information in the color clustering step, making it a part of the color separation construcion process. Thus the color clustering can make the foreground and background regions more distinct from each other. Figure 6.12 shows an example of background subtraction results with different background modeling techniques using kmeans color clustering algorithm including motion information. Figure 6.13 shows an example of background subtraction results with different background modeling techniques using efficient graph-based color clustering algorithm including motion information.

Recall that in Table 6.5 and Table 6.6 motion information makes F-measure results better. In Figure 6.12 and Figure 6.13, shows an example. For the mean filter case, it is hard to see a noticeable improvement with motion information. Under Gaussian and mixture of Gaussian, the improvement is more noticeable.



(a) meanfilter without color separation

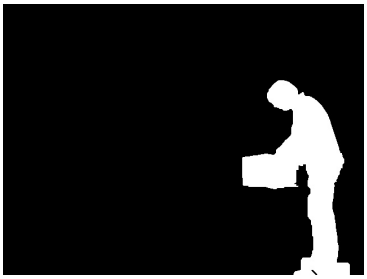


(b) Gaussian average without no color separation

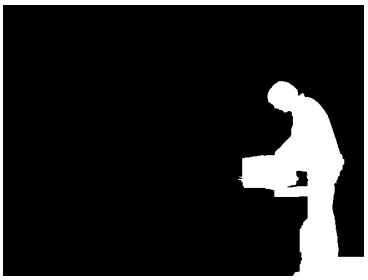
(c) Gaussian mixture model without no color separation

(d) mean filter uing kmeans

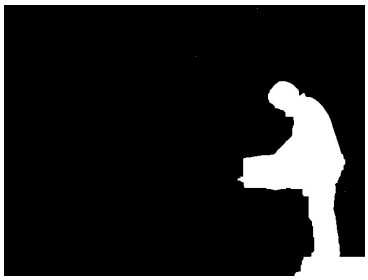
(e) Gaussian average using kmeans



(f) Gaussian mixture using kmeans



(g) mean filter using meanshift

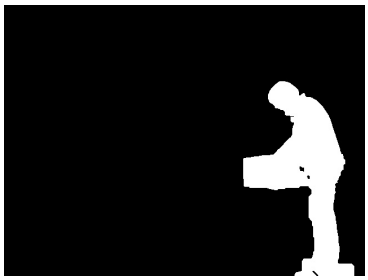


(h) Gaussian average using meanshift

(i) Gaussian mixture model using meanshift

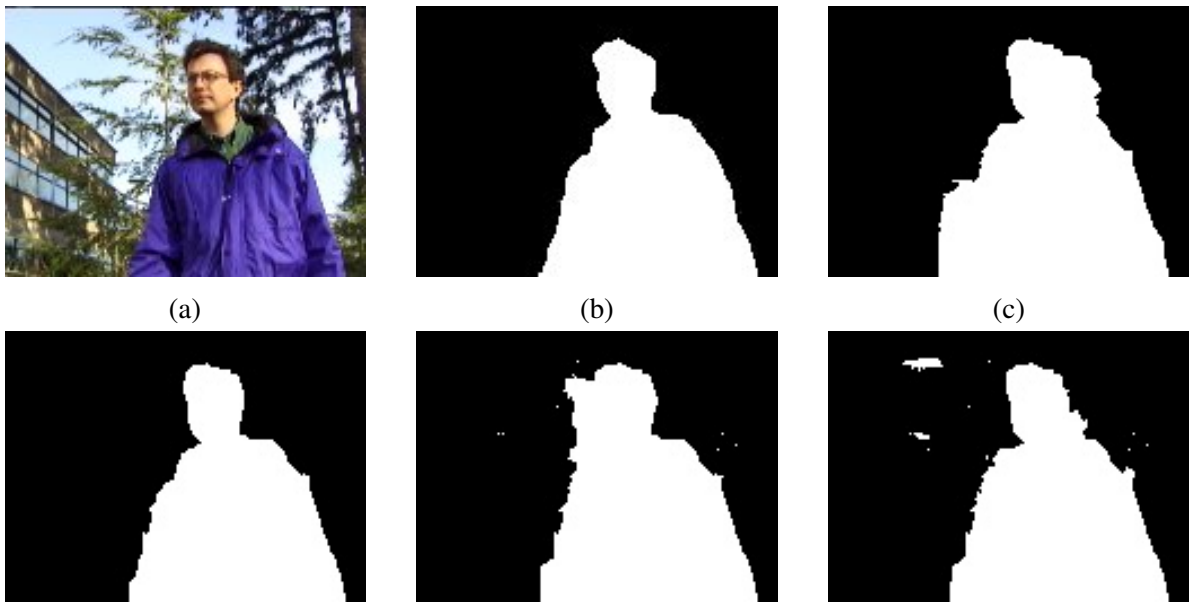
(j) mean filter using graph-based

(k) Gaussian average using graph-based



(l) Gaussian mixture model using graph-based

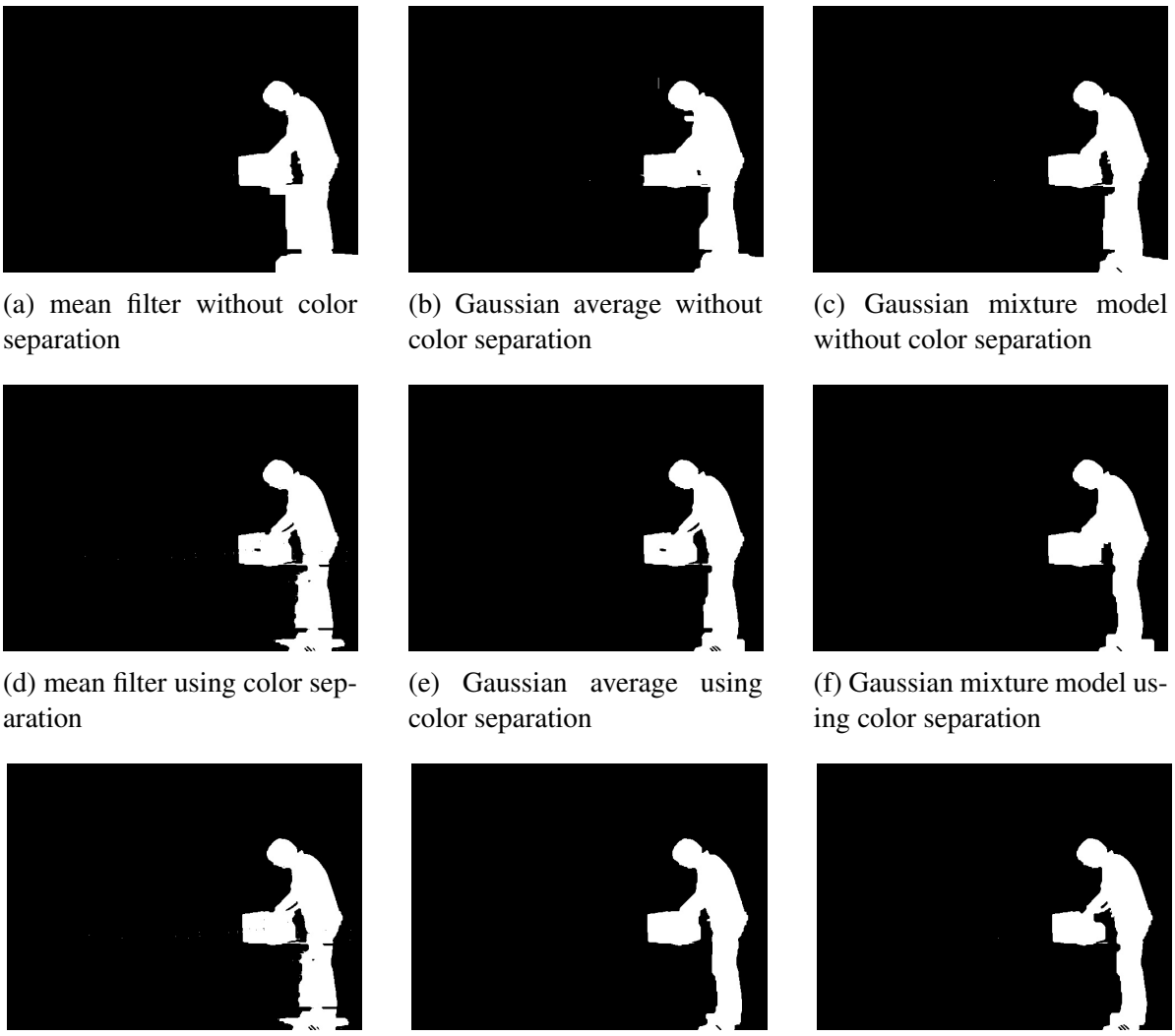
Figure 6.10: An example of background subtraction results on an indoor scene with different combinations of background modeling techniques and color clustering algorithms. Left: background subtraction method using mean filter. Middle: background subtraction method using Gaussian average. Right: background subtraction method using Gaussian mixture model. From top to bottom are background subtraction methods: without color separation, with kmeans color clustering algorithm, with meanshift color clustering algorithm and with efficient graph-based color clustering algorithm.



(a) (b) (c)

(d) (e) (f)

Figure 6.11: An example of background subtraction results on a waving tree scene with different color clustering algorithms using Gaussian average background modeling technique. (a) original image, (b) ground truth, (c) background subtraction result without color separation, (d) background subtraction result with kmeans color clustering, (e) background subtraction result with meanshift color clustering, (f) background subtraction result with efficient graph-based color clustering.



(a) mean filter without color separation

(b) Gaussian average without color separation

(c) Gaussian mixture model without color separation

(d) mean filter using color separation

(e) Gaussian average using color separation

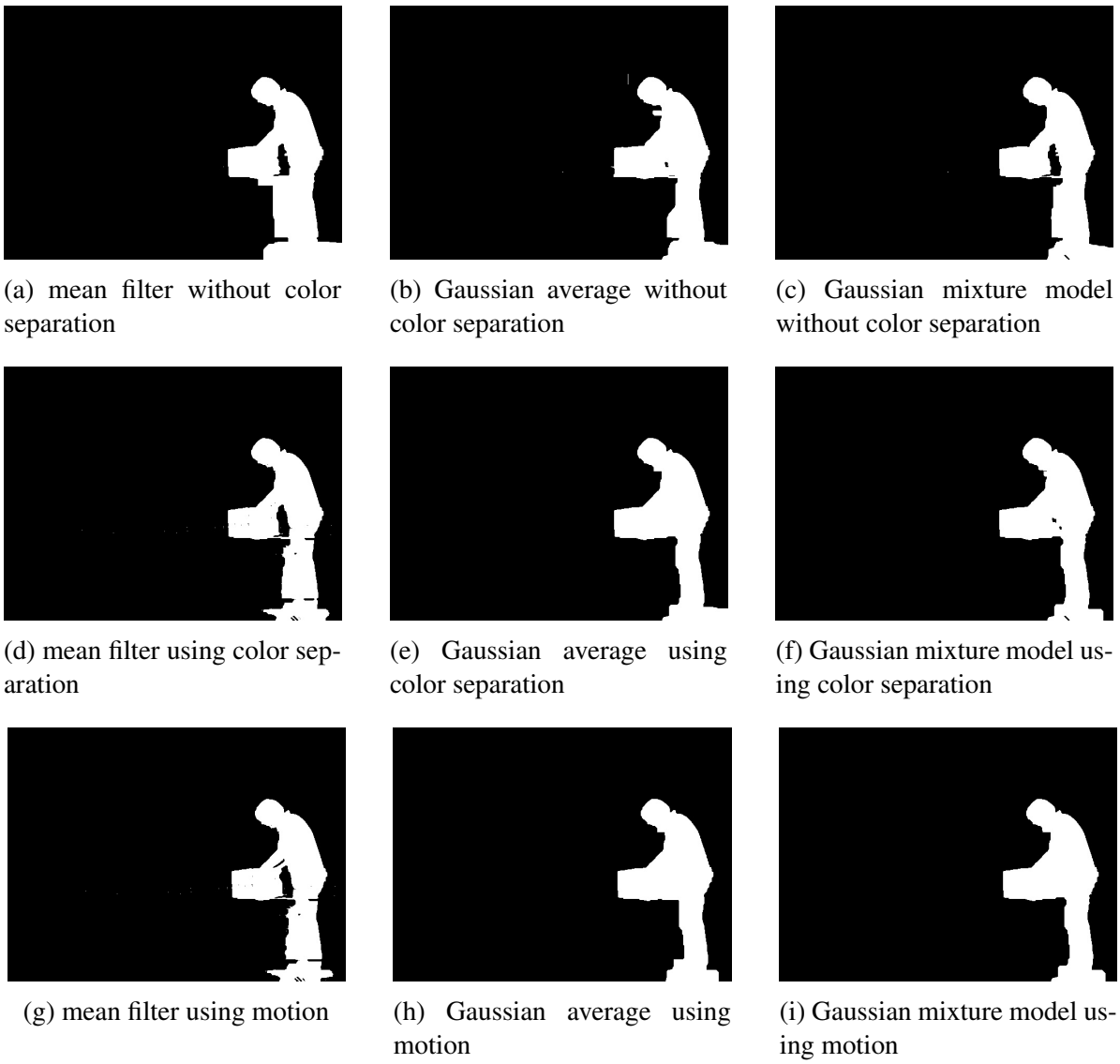
(f) Gaussian mixture model using color separation

(g) mean filter using motion

(h) Gaussian average using motion

(i) Gaussian mixture model using motion

Figure 6.12: An example of background subtraction results on an indoor scene with different background modeling techniques using kmeans color clustering algorithms with motion information. Left: background subtraction method using mean filter. Middle: background subtraction method using Gaussian average. Right: background subtraction method using Gaussian mixture model. From top to bottom are background subtraction methods: without color separation, with kmeans color clustering algorithm without motion, with kmeans color clustering algorithm with motion.



(a) mean filter without color separation

(b) Gaussian average without color separation

(c) Gaussian mixture model without color separation

(d) mean filter using color separation

(e) Gaussian average using color separation

(f) Gaussian mixture model using color separation

(g) mean filter using motion

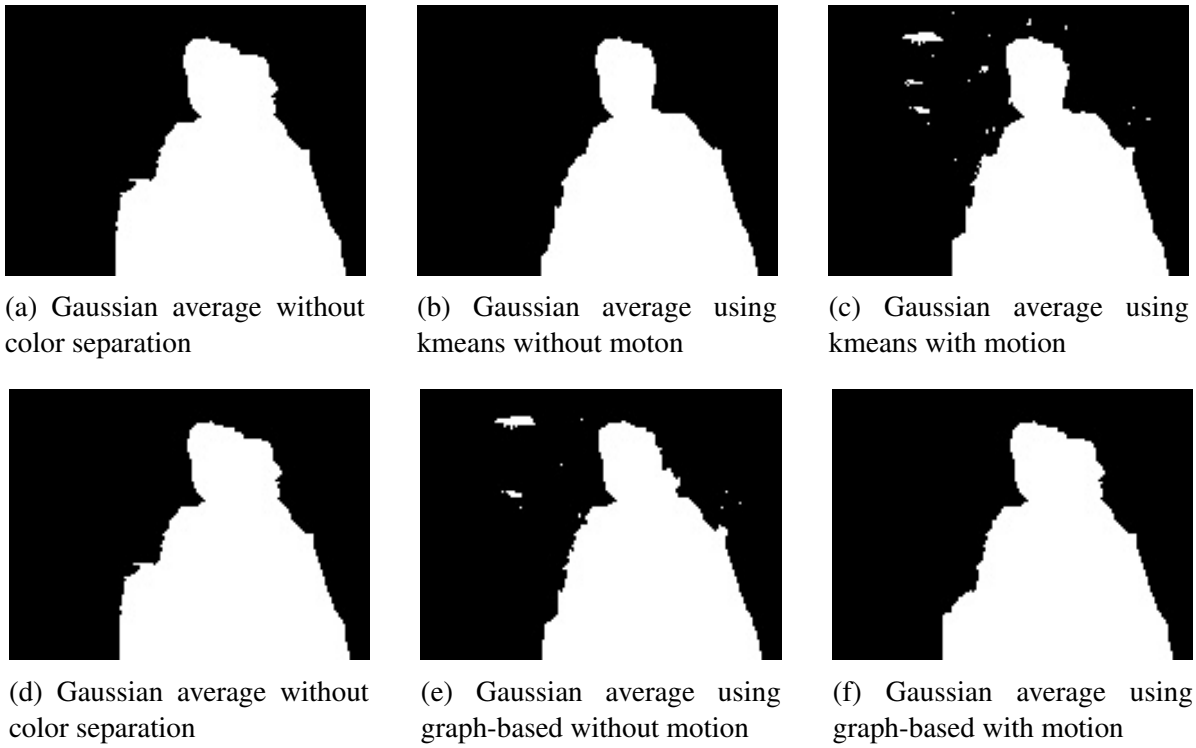
(h) Gaussian average using motion

(i) Gaussian mixture model using motion

Figure 6.13: An example of background subtraction results on an indoor scene with different background modeling techniques using efficient graph-based color clustering algorithms with motion information. Left: background subtraction method using mean filter. Middle: background subtraction method using Gaussian average. Right: background subtraction method using Gaussian mixture model. From top to bottom are background subtraction methods: without color separation, with efficient graph-based color clustering algorithm without motion, with efficient graph-based color clustering algorithm with motion.

6.6 Failure Cases

Including the motion features into clustering does not always help. There are also few cases where adding motion information makes the background subtraction worse, as shown in Figure 6.14.



(a) Gaussian average without color separation

(b) Gaussian average using kmeans without moton

(c) Gaussian average using kmeans with motion

(d) Gaussian average without color separation

(e) Gaussian average using graph-based without motion

(f) Gaussian average using graph-based with motion

Figure 6.14: An example of background subtraction results on a waving tree scene using Gaussian average background modeling method with two color clustering algorithms including motion information. From left to right are background subtraction methods: without color separation, with color clustering algorithm without motion, with clustering algorithm with motion. Top: with kmeans color clustering algorithm. Bottom: with efficient graph-based color clustering algorithm.

In this case, adding motion information fails in improving the performance of background subtraction algorithm. Although it is a useful cue for color clustering, for sequences whose background has moving objects, such as a moving tree, the observed motion information contains a lot of noise. If we add these motion information in clustering, the pixels in the background may be clustered in the same color bin as foreground pixels. This can directly deduct the effectiveness of color separation term. As we can see in Figure 6.14, the background subtraction results become worse after adding motion information. In conclustion, it is not appropriate to consider motion information for color clustering when the background has slight moving objects, which can be easily recognized as foreground as a consequence.

Chapter 7

Conclusion and Future Work

In this thesis, we propose to use a color separation term for graph-cut based background subtraction. Our approach is based on the energy funtion with L_1 color separation term proposed by [31]. It turns out that this L_1 color separation term changes many NP-hard energy minimization problems into tractable ones. In this work, we make application of this efficient enery function to the foreground detection process of background subtraction and show how this simple term can help to achieve better performance. Graph cut method is utilized for the energy optimization. A background modeling technique is required first for background maintenance. We tried three kinds of background modeling techniques in our experiments, mean filter, Gaussian average and Gaussian mixture model. Then it is a binary energy optimization problem to perform foreground detection. There are three terms in our background subtraction energy function. Apart from the boundary constraint in the smoothness term, we have also integrated motion information in the data term, which is based the background modeling process. Most imporantly, we include the L_1 color separation term, which gives penalty for the color appearance overlap in foreground and background. Three color clustering algorithms are used in our work to perform color quantization. The resulting clusters respresent the color appearance for each pixel. Finally, we propose to add motion features in the clustering process, so that the effectiveness of color separation term in background subtraction problem is enhanced. With all these improvements, the background subraction accuracy improves.

Although we have achieved significant improvements for the background subtraction, there are still a lot of complex cases to investigate. When the background of a sequence is dynamic, adding motion information deducts the effectiveness of color separation. It turns out that the motion information is not accurate. The main reason lies in the limitation of background modeling methods. We could investigate other more reliable background modeling techniques for improvement.

So far we have done a lot of work in parameter selection, even the images in the same sequence have different optimal parameters, it is quite expensive for us to select parameters that perform the best overall. More work is needed to solve this problem.

Bibliography

[1] R. Achanta, F. Estrada, P. Wils, and S. Ssstrunk. Interactive image segmentation using an adaptive gmmrf model. *European Conference on Computer Vision (ECCV)*, I:428–441, 2004.

[2] M.M. Azab, H.A. Shedeed, and A.S. Hussein. A new technique for background modeling and subtraction for motion detection in real-time videos. *IEEE International Conference on Image Processing (ICIP)*, pages 3453–3456, 2010.

[3] Y. Benezeth, P.M. Jodoin, and B. Emile. Review and evaluation of commonly-implemented background subtraction algorithms. *IEEE International Conference on Pattern Recognition (ICPR)*, 2008.

[4] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *IEEE International Conference on Computer Vison (ICCV)*, 70(2):109–131, 2006.

[5] Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation. *IEEE International Conference on Computer Vison (ICCV)*, pages 105–112, 2001.

[6] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Pattern Analysis and Machine Intelligence*.

[7] M. Camplani and L. Salgado. Background foreground segmentation with rgb-d kinect data: an efficient combination of classifiers. *Journal of Visual Communication and Image Representation*, 25(1):122–136, 2014.

[8] C. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. *International Conference on Pattern Recognition*, 4:40190, 2002.

[9] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms. *MIT Press, Cambridge, MA, USA*.

[11] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IEEE International Journal of Computer Vision*, 59(2), 2004.

[12] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[13] Nicholas R. Howe and Alexandra Deschamps. Better foreground segmentation through graph cuts. *arXiv preprint cs/0401017*, 2004.

[14] V. Jain, B.B. Kimia, and J.L. Mundy. Background modeling based on subpixel edges. *IEEE International Conference on Image Processing (ICIP)*, 2007.

[15] X. Jian, D. Xiaoqing, W. Shengjin, and W. Youshou. Background subtraction based on a combination of texture, color and intensity. *International Conference on Signal Processing (ICSP)*, 2008.

[16] Lester Randolph Ford Jr and Delbert Ray Fulkerson. Flows in networks. *Princeton University Press*, 1962.

[17] Shehroz S. Khan and Amir Ahmad. Cluster centre initialization algorithm for k -means cluster. *Pattern Recognition Letters*, pages 1293–1302, 2004.

[18] Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2008.

[19] D. Koller, J. Weber, T. Huan, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. *International Conference on Pattern Recognition (ICPR)*, pages 126–131, 1994.

[20] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Pattern Analysis and Machine Intelligence*.

[21] A. Lai and N. Yung. A fast and accurate scoreboard algorithm for extimating stationary backgrounds in an image sequence. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 241–244, 1998.

[22] Y. Li, J. Sun, C-K. Tang, and H-Y. Shum. grabcut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3), 2004.

[23] Jiangyu Liu, Jian Sun, and Heung-Yeung Shum. Paint selection. *siggraph*, 2009.

[24] B.P.L Lo and S.A. Velastin. Automatic congestion detection system for underground platforms. *International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 158–161, 2001.

[25] Ekaterina Lobacheva, Olga Veksler, and Yuri Boykov. Joint optimization of segmentation and color clustering. *IEEE International Conference on Computer Vison (ICCV)*, pages 1626–1634, 2015.

[26] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Mathematical Statistics and Probability*, pages 281–297, 1967.

[27] K. A. Abdul Nazeer and M. P. Sebastian. Improving the accuracy and efficiency of the k-means clustering algorithm. *Proceedings of the World Congress on Engineering*, 1, 2009.

[28] Caroline Pantofaru and Martial Hebert. A comparison of image segmentation algorithms. *The Robotics Institute in Carnegie Mellon University*, 2005.

[29] C. Rother, V. Kolmogorov, and A. Blake. grabcut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[30] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 246–252, 1999.

[31] M. Tang, L. Gorelick, O. Veksler, and Y. Boykov. Grabcut in one cut. *IEEE International Conference on Computer Vision (ICCV)*, pages 1769–1776, 2013.

[32] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. *IEEE International Conference on Computer Vision (ICCV)*, pages 255–261, 1999.

[33] S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[34] C. Wren and A. Azarbayejani. Pfinder: real-time tracking of the human body. *IEEE Trans. on Pattern Anal. and Machine Intell*, 19(7):780–785, 1997.

[35] Madhu Yedla, Srinivasa Rao Pathakota, and T. M. Srinivasa. Enhanced k -means clustering algorithm with improved initial center. *International Journal of Science and Information Technologies*, 1(2):121–125, 2010.

[36] H. Zhang and D. Xu. Fusing color and texture features for background model. *International Conference on Fuzzy Systems and Knowledge Discovery*, 2006.

Curriculum Vitae

Name:	Jiaqi Zhou
Post-Secondary Education and Degrees:	University of Western Ontario London, ON, CA M.Sc. in computer science, Sep. 2015 - Dec. 2016 Nankai University Tianjin, China B.E. in Information Security, 2011 - 2015
Honours and Awards:	WGRS, Western University, 2015 -2016 Outstanding Graduate, Nankai University, 2015
Related Work Experience:	Teaching Assistant, Western University,Sep. 2015 -Dec. 2016 Research Assistant, Computer Vision Group, University of Western Ontario, Sep. 2015 - Dec. 2016