Department of Information Technology
National Institute of Technology Srinagar
Jammu and Kashmir, India – 190006

June 2018

Project Report on

Minimum Delay Moving Object Detection in Realtime

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Information Technology

Submitted by

Samim Zahoor **IT/03/13-14**

Under the supervision of

Dr. Shabir Ahmad Sofi

Department of Information Technology
National Institute of Technology Srinagar
Jammu and Kashmir, India – 190006

Certificate

This is to certify that the project titled **Minimum Delay Moving Object Detection in Realtime** has been completed by the below mentioned student(s) under my supervision in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology**. It is also certified that the project has not been submitted or produced for the award of any other degree. .

Enrollment Number	Name of Student
IT/03/13-14	**Samim Zahoor**

Dr. Shabir Ahmad Sofi
Project Coordinator
Department of Information Technology
NIT Srinagar, J&K

Department of Information Technology
National Institute of Technology Srinagar
Jammu and Kashmir, India – 190006

Students's Declaration

I, hereby declare that the work, which is being presented in the project entitled **Minimum Delay Moving Object Detection in Realtime** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology in the session 2018, is an authentic record of my own work carried out under the supervision of **Dr. Shabir Ahmad Sofi, Department of Information Technology, National Institute of Technology, Srinagar**. The matter embodied in this project has not been submitted by us for the award of any other degree.

Samim Zahoor

Dated: _____

Signature: _____

Acknowledgements

I express special gratitude to my project advisor Dr. Shabir Ahmad Sofi who provided me the necessary requirements and timely guidance to understand the conceptual underpinnings of the project and paved the way for its completion. I am thankful to all the faculty members of Information Technology Department for imparting knowledge and wisdom to us during our course here and my friends who are always supportive and never fail to provide deep insights and helpful critique of my work. And most importantly I am thankful to Allah without whose will none of this would have been possible and for blessing me with supportive and caring parents who are always there for me.

Samim Zahoor

Autumn 2017
National Institute of Technology Srinagar

Abstract

Motion segmentation aims at decomposing a video in moving objects and background. Detection and segmentation of moving objects is a fundamental problem in computer vision. As such there are extensive works that tackle the problem of segmentation of moving objects. It is an essential building block for applications such as robotics, inspection, metrology, video surveillance, video indexing, traffic monitoring and many other applications and also forms the basis for further processing. In these applications, an object can move or come into the field of the camera at an unknown instant of time. Our contributions are to propose a method based on weighted clustering of pixel values, dense motion estimates and spatial cordinates for the problem of detection and segmentation of moving objects in video sequences while running at interactive frame-rates. This makes our method more suitable for online and real-time applications such as video surveillance. We test our method on the change detection benchmarking dataset Change Detection 2014[11] consisting of nearly 90,000 frames in 31 video sequences. We evaluate and rank our method on seven different metrics according to the [11]. We also provide a public implementation of the algorithm available at [21].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the tremendous amount of available video data, it is important to maintain the efficiency of video based applications to process only relevant information. Most video files contain redundant information such as background scenery, which costs a huge amount of storage and computing resources. Hence, it is necessary to extract the meaningful information, e.g. vehicles or pedestrians, to deploy those resources more efficiently. Focus has been on developing intelligent visual surveillance for many scientific fields including computer vision, transportation, healthcare, security and millatary. Moving object detection is a classification task that assigns each pixel in a video sequence with a label, for either belonging to the background or to an object in the foreground. Thus moving object detetion can be thought of as a version of background subtraction where along with identifiying the foreground, we also differentiate each of the objects.

Moving object detection, is applied to many advanced video applications as a pre-processing step to remove redundant data, for instance in tracking or automated video surveillance. In addition, for real-time applications, like tracking, the algorithm should be capable of processing the video frames in real-time and in an online manner.

One simple example of the application of a moving object detection method is the pixel-wise subtraction of a video frame from its corresponding background image. After being compared with the difference threshold, pixels with a larger difference than a certain threshold value are labeled as foreground pixels, otherwise as background pixels. Unfortunately, this strategy will yield poor segmentation due to the dynamic nature of the background, that is induced by noise or illumination changes. For example, due to lighting changes, it is common that even pixels belonging to the background scene can have intensities very different from their other pixels in the background image and they will be falsely classified as foreground pixels as a consequence. Thus, sophisticated moving object detection algorithms that assure robust moving object detection under various con-

ditions must be employed. In the following sections, the difficulties in this area, our proposed solution for moving object detection and our contributions will be illustrated.

1.1 Challenges

The main difficulties that complicate the moving object detection process are:

Illumination changes: When scene lighting changes gradually (e.g. moving clouds in the sky) or instantly (e.g. when the light in a room is switched on), the background model usually has a illumination different from the current video frame and therefore yields false classification.

Dynamic background: The background scene is rarely static due to movement in the background (e.g. waves, swaying tree leaves), especially in outdoor scenes. As a consequence, parts of the background in the video frame do not overlap with the corresponding parts in the background image, hence, the pixel-wise correspondence between image and background is no longer existent.

Camera jitter: In some cases, instead of being static, it is possible that the camera itself is frequently in movement due to physical influence. Similar to the dynamic background case, the pixel locations between the video and background frame do not overlap anymore. The difference in this case is that it also applies to non-moving background regions.

Camouflage: Most moving object detection algorithms work with pixel or color intensities. When foreground objects and background scene have a similar color, the foreground objects are more likely to be (falsely) labeled as background.

Night Videos: As most pixels have a similar color in a night scene, recognition of foreground objects and their contours is difficult, especially when color information is the only feature in use for segmentation.

Ghosts/intermittent object motion: Foreground objects that are embedded into the background scene and start moving after background initialization are the so-called *ghosts*. The exposed scene, that was covered by the ghost, should be considered as background. In contrast, foreground objects that stop moving for a certain amount of time, fall into the category of intermittent object motion. Whether the object should be labeled as foreground or background is strongly application dependent. As an example, in the automated video surveillance case, abandoned objects should

be labeled as foreground.

Hard shadows: Dark, moving shadows that do not fall under the illumination change category should not be labeled as foreground.

The goal of our project is to use the data captured in the form of image sequences from video cameras to detect motion and if there is motion, determine the moving objects thereby the seperating the background and the foreground. When there is motion in the sequence of images, we also want to minimize the delay in the detection of motion. This is acheived by finding groups of pixels in each frame of the video sequence that "go together". By "going together" we mean the pixels that belong to the same moving object. Our method of grouping the pixels is based on the hypothesis that *"Pixels of the same object move with approximately the same motion and thus motion can provide more accurate results for grouping the pixels of the same object"*.

Chapter 2

Review of Motion Segmentation Methods

In this section common issues are described, and a description of the main attributes that should be considered for studying the algorithms for motion segmentation is presented. In order to obtain an automatic motion segmentation algorithm that can work with real images there are several issues that need to be solved, particularly important are:noise, missing data and lack of a priori knowledge. One of the main problem is the presence of noise. For some applications the noise level can become critical. For instance, inunderwater imaging there are some specific sub-sea phenomena like water turbidity, marine snow, rapid light attenuation, strong reflections, back-scattering, non-uniform lightingand dynamic lighting that dramatically degrade the quality of the images. Blurring is also a common issue especially when motion is involved. Another common problem is caused by the fact that moving objects can create occlusions, or even worst, the whole object can disappear and reappear in the scene. Finally, it is important to take into account that not always it is possible to have prior knowledge about the objects or about the number of objects in the scene.

The main attributes of a motion segmentation algorithm can be summarized as follows.

- Feature-based or Dense-based: In feature-based methods, the objects are represented by a limited number of points like corners or salient points, whereas dense methods compute a pixel-wise motion.
- Occlusions: it is the ability to deal with occlusions.
- Multiple objects: it is the ability to deal with more than one object in the scene.
- Spatial continuity: it is the ability to exploit spatial continuity.

- Temporary stopping: it is the ability to deal with temporary stop of the objects.
- Robustness: it is the ability to deal with noisy images (in case of feature based methods it is the position of the point to be affected by noise but not the data association).
- Sequentiality: it is the ability to work incrementally, this means for example that the algorithm is able to exploit information that was not present at the beginning of the sequence.
- Missing data: it is the ability to deal with missing data.
- Non-rigid object: it is the ability to deal with non-rigid objects.
- Camera model: if it is required, which camera model is used (orthographic, paraperspective or perspective).

Furthermore, if the aim is to develop a generic algorithm able to deal in many unpredictable situations there are some features that may be considered as a drawback, specifically:

- Prior knowledge: any form of prior knowledge that may be required.
- Training: some algorithms require a training step.

Image difference is one of the simplest and most used technique for detecting changes. It consists in thresholding the intensity difference of two frames pixel by pixel. The result is a coarse map of the temporal changes. An example of an image sequence and the image difference result is shown in figure 1. Despite its simplicity, this technique cannot be used in its basic version because it is really sensitive to noise. Moreover, when the camera is moving the whole image is changing and, if the frame rate is not high enough, the result would not provide any useful information. However, there are a few techniques based on this idea. The key point is to compute a rough map of the changing areas and for each blob to extract spatial or temporal information in order to track the region. Usually different strategies to make the algorithm more robust against noise and light changes are also used. Examples of this technique can be found in [4, 5, 7, 6].

Statistic theory is widely used in the motion segmentation field. In fact, motion segmentation can be seen as a classification problem where each pixel has to be classified as background or foreground. Statistical approaches can be further divided depending on the framework used. Common frameworks are Maximum A posteriori Probability(MAP), Particle Filter (PF) and Expectation Maximization (EM). Statistical approaches provide a general tool that can be used in a very different way depending on the specific technique.

MAP is based on Bayes rule:

$$P(w_j, x) = \frac{p(x|w_j)P(w_j)}{\sum_{i=1}^{C} p(x|w_i)P(w_i)} \tag{2.1}$$

where x is the object to be classified (usually the pixel), $w_1, ...w_c$ are the c classes (usually the background or foreground), $P(w_j, x)$ is the "a posteriori probability", $p(x|w_j)$ is the conditional density, $P(w_j)$ is the "a priori probability" and $\sum_{i=1}^{C} p(x|w_i)P(w_i)$ is the "density function". MAP classifies x as belonging to the class w which maximizes the "a posteriori probability". MAP is often used in combination with other techniques like Probabilistic Data Association Filter and level sets for incorporating motion information.

Another widely used statistical method is PF. The main aim of PF is to track the evolution of a variable over time. The basis of the method is to construct a sample-based representation of the probability density function. Basically, a series of actions are taken, each of them modifying the state of the variable according to some model. Multiple copies of the variable state (particles) are kept, each one with a weight that signifies the quality of that specific particle. An estimation of the variable can be computed as a weighted sum of all the particles. PF is an iterative algorithm, each iteration is composed by prediction and update. After each action the particles are modified according to the model (prediction), then each particle weight is re-evaluated according to the information extracted from an observation (update). At every iteration particles with small weights are eliminated . An example of PF used in segmentation can be found in [8] where some well known algorithms for object segmentation using spatial information, such as geometric active contours and level sets, are unified within a PF framework.

Another group of motion segmentation algorithms are the ones based on wavelets. These methods exploit the ability of wavelets to perform analysis of the different frequency components of the images, and then study each component with a resolution matched to its scale. Usually wavelet multi-scale decomposition is used in order to reduce the noise and in conjunction with other approaches, such as optical flow, applied at different scales. For instance, in [2] Wiskott combines Optical Flow with Gabor-wavelets in order to overcome the aperture problem. Furthermore, he extracts and tracks edges using the information provided by the Mallat-wavelets. Finally, the results are merged in order to obtain a robust segmentation. A different approach is presented by Kong et al, where the motion segmentation algorithm is based on Galilean wavelets. These wavelets behave as matched filters and perform minimum mean-squared error estimations of velocity, orientation, scale and spatio-temporal positions. This information is finally used for tracking and segmenting the objects.

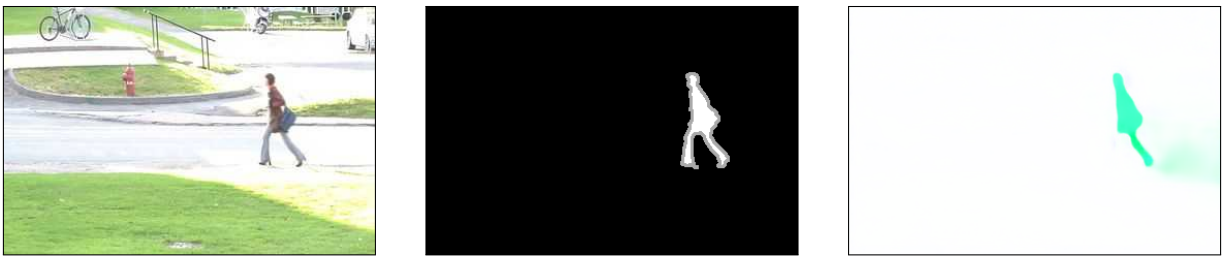
Chapter 3

Problem Statement and Objectives

Moving object detection is among the most frequently used preprocessing tasks in a wide variety of applications such as people counting, traffic video analysis, object tracking, mobility monitoring and video retrieval to name just a few [20]. This is the case because moving objects are generally what we are interested in. For example in case of object tracking for survelliance, we are interested in recognizing the moving objects. Once we have the moving objects we can then determine what they actually are and how to deal with them in various real world scenarios. So determining these objects becomes an important preprocessing step for complex automated applications. The goal of our project is to use the data captured in the form of image sequences from video cameras to detect motion and if there is motion, determine the moving objects. When there is motion in the sequence of images, we also want to minimize the delay in the detection of motion. This is an important factor in applications where fast and accurate response to critical situations is required. In the next chapter we define and explain all the terms and objectives in a more formal way. The objectives of the project can be stated as

- Detection and segmentation of moving objects in video based on apparent motion.
- Minimize the delay in detection.

This is acheived by finding groups of pixels in each frame of the video sequence that "go together". By "going together" we mean the pixels that belong to the same moving object. Our method of grouping the pixels is based on the hypothesis that *"Pixels of the same object move with approximately the same motion and thus motion can provide more accurate results for grouping the pixels of the same object"*. This accomplished by using motion cues provided by optical flow at each of the pixel position in the frame which can also be used to detect moving objects in the entire frame. This can be seen from the Figure 3.1.



(a) Input sequence (b) Ground truth (c) Color coded optical flow

Figure 3.1: One frame from the pedestrian sequence of Change Detection 2014 dataset.

The input image is shown in the first part of Figure 3.1a. The object of interest here is the walking person. As can be seen from the frame, using only pixel values from the RGB channels to distinguish between the background and the object of interest would be quite difficult and require complex background and foreground estimation techniques. This is because the background is not uniform. It is divided into many parts like the grass, road, trees and parts of other objects like cars, fire hydrants and bicycles. The third part of the Figure 3.1c shows the color coded optical flow (Hue in the color coded flow represents the direction of motion and intensity represents the magnitude). As is evident from this depiction, optical flow presents a much more desirable representation. Based on this discussion and the depiction shown in Figure 3.1, we can conclude that motion cues provided by optical flow at each of the pixel position in the frame can prove to be more accurate at detecting and segmenting moving objects.

Chapter 4

Methodology

4.1 Models for object detection

In this section, we present our statistical models for the image sequence and moving object(s) to frame minimum delay object detection. Let $\Omega \subset R^2$ be the domain of images and let $I_t : \Omega \rightarrow R^2, t \geq 1$ where t denotes the frame number be the sequence of images, $k = 3$ denotes the number of color channels. We denote n regions of the image I_t with $R_t^i \subset \Omega$ for $i = 0, 1, 2, ...n-1$. R_t^0 will correspond to the background. Each of the n regions form the segmentation of I_t into different objects. We denote O_t^i as the occlusion of the region R_t^i induced by change of viewpoint or camera or a self-occlusion between time t and $t+1$. Figure 4.1 represents a simplified case of the above explained models for two regions. R_t^0 represents the background and R_t^1 represents the foreground with $n = 2$. All the moving objects are incorporated into the region R_t^1. The displacement between adjacent frames for region i is $v_t^i : R_t^i \ / O_t^i \rightarrow \mathbb{R}^2$. Although such displacements are not defined in the occluded parts of the domain, they will be smoothly extended into the occlusion and the entire domain Ω. Also, $w_{t,t+1}(x)$ is the warp between consecutive frames and is defined in the next section in equation 4.2.

According to the model explained above, the problem of detecting changes and segmenting moving objects in video sequences can be summarised as:

1. Solve for displacements v_t between the frames I_t and I_{t+1} and calculate the warps $w_{t,t+1}(x)$.
2. Check if a change (moving object) is detected.
3. If there is a change, solve for the regions R_t^i for $i = 0, 1, 2, ...n-1$ and each of these regions represent a segmentation of the image with R_t^0 representing the background and R_t^i with $i = 1, 2, ...n-1$ representing the different moving objects.

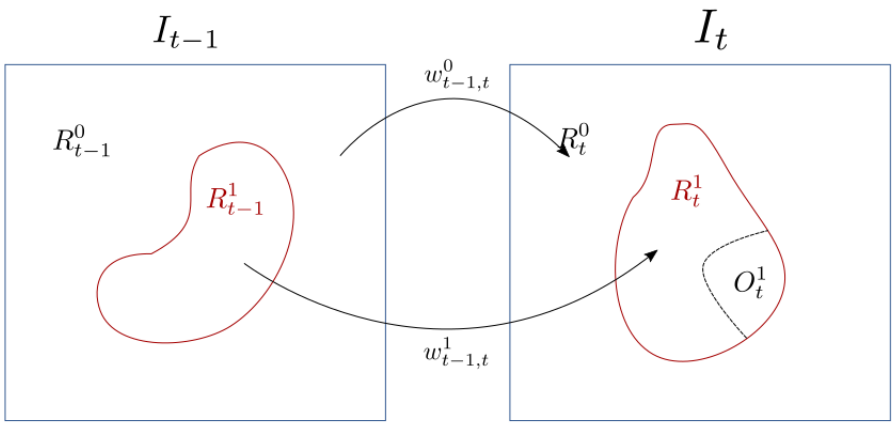


Figure 4.1: Schematic of region models

The rest of the chapter is structured as follows: Section 4.2 begins with explaining the methods and techniques used to solve for the displacements. Then we explain the tests for detecting changes caused by relative motion of objects in an online method and making the decision to move on to the next frame or to declare a detection and produce a segmentation in Section 4.3. Next we define our method of clustering based on weighted clustering of pixel values and dense motion estimates and optionally spatial cordinates to solve for the regions and produce segmentations in Section 4.4.

4.2 Solving for Displacements

The most general (and challenging) version of motion estimation is to compute an independent estimate of motion at each pixel, which is known as optical flow. Optical flow is theoretically a good clue in order to segment motion. However, optical flow alone is not enough since it cannot help to solve occlusions and temporal stopping [9]. Moreover, these methods are sensitive to noise and light changes. Computing optical flow generally involved minimizing the brightness or color difference between corresponding pixels summed over the image. The problem can be formulated

as an energy functional using a brightness constancy constraint and a regularization constraint,

$$E_{HS} = \int [(I_x u + I_y v + I_t)^2 + \alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2)] dxdy \tag{4.1}$$

Solving this equation in variatonal framework is time consuming for inputs comparable to the size of normal images. Recently, convolutional neural network based methods have proven to be successful at predicting dense probability distributions.

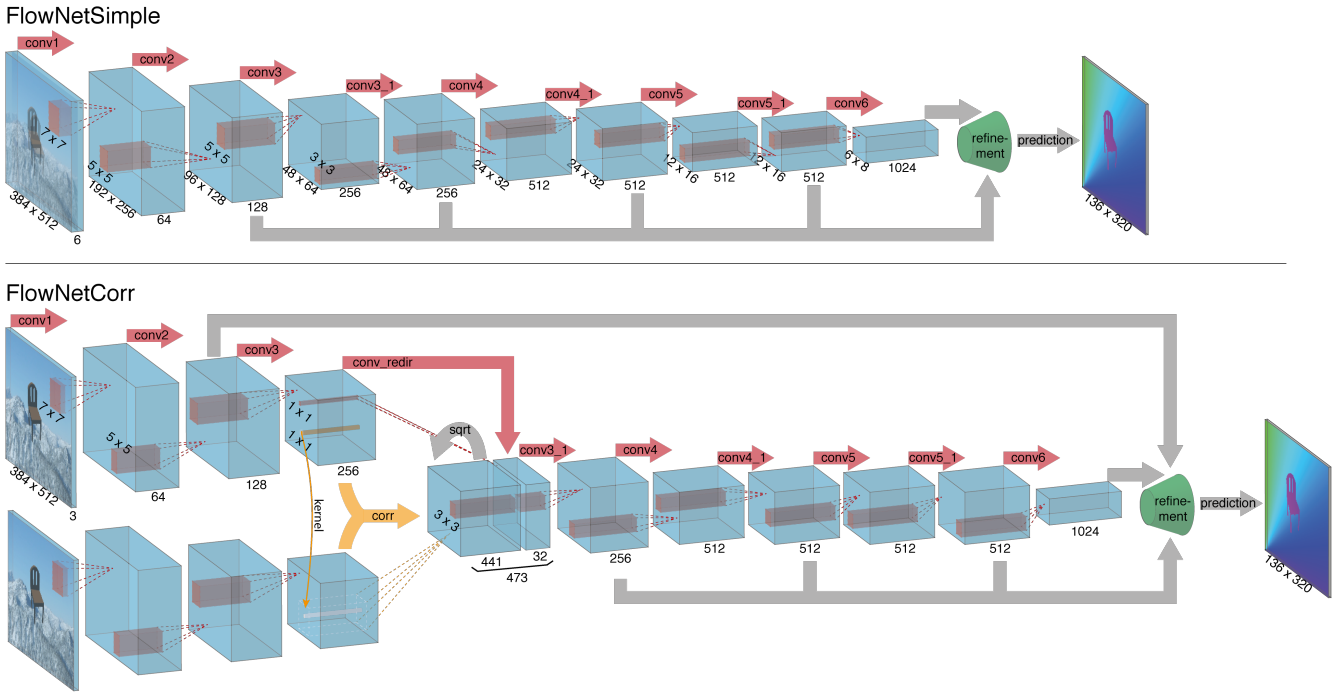


Figure 4.2: Architecture of FlowNetSimple (FlownetS) and FlowNetCorr (FlownetC)

Convolutional neural networks are known to be very good at learning input–output relations given enough labeled data. Dosovitskiy et al [18] take an end-to-end learning approach to predicting optical flow: given a dataset consisting of image pairs and ground truth flows, we train a network to predict the x–y flow fields directly from the images.

A simple choice for the architecture of the network is to stack both input images together and feed them through a rather generic network, allowing the network to decide itself how to process the image pair to extract the motion information. This is illustrated in Fig. 4.2 (top). This architecture consisting only of convolutional layers is referred to as 'FlowNetSimple' or 'FlownetS' in brief.

In principle, if this network is large enough, it could learn to predict optical flow. However, one can never be sure that a local gradient optimization like stochastic gradient descent can get

the network to this point. Therefore, it could be beneficial to hand-design an architecture which is less generic, but may perform better with the given data and optimization techniques.

A straightforward step is to create two separate, yet identical processing streams for the two images and to combine them at a later stage as shown in Fig. 4.2 (bottom). With this architecture the network is constrained to first produce meaningful representations of the two images separately and then combine them on a higher level. This roughly resembles the standard matching approach when one first extracts features from patches of both images and then compares those feature vectors.

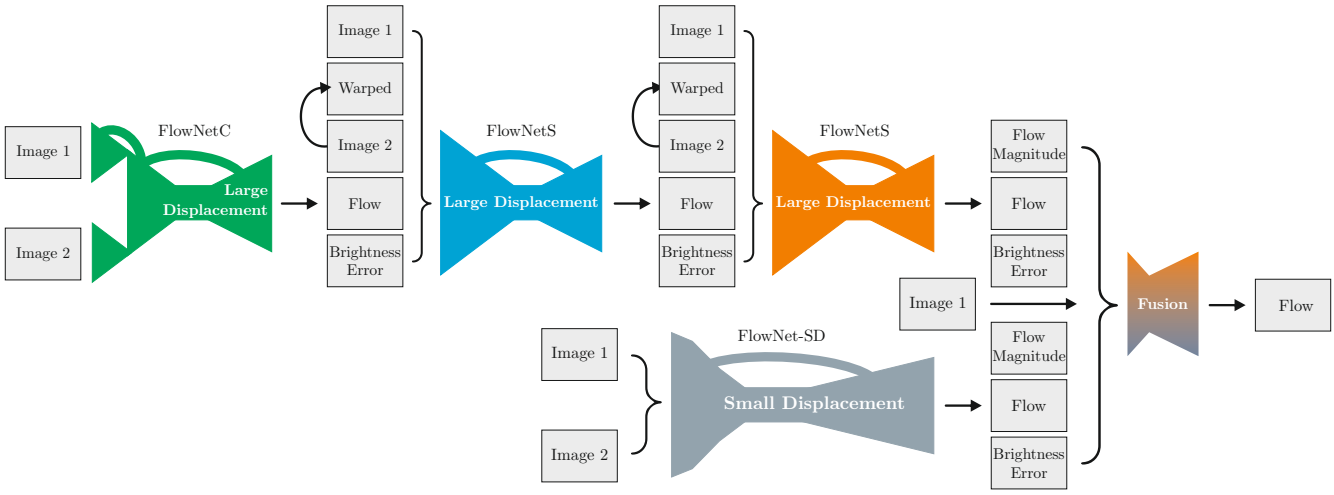


Figure 4.3: Schematic view of complete architecture of the Flownet2 model.

Ilg et al [19] stack multiple FlowNetS and FlowNetC models and come up with a new architecture that they call Flownet2 and also present different ways to stack networks. The first network in the stack always gets the images I_1 and I_2 as input. Subsequent networks get I_1, I_2, and the previous flow estimate $w_i = (u_i, v_i)^\top$, where i denotes the index of the network in the stack.

To make assessment of the previous error and computing an incremental update easier for the network, the second image $I_2(x, y)$ is optionally warped via the flow w_i and bilinear interpolation to $\tilde{I}_{2,i}(x, y) = I_2(x+u_i, y+v_i)$. This way, the next network in the stack can focus on the remaining increment between I_1 and $\tilde{I}_{2,i}$. See Figure 4.3. Thanks to bilinear interpolation, the derivatives of the warping operation can be computed. This enables training of stacked networks end-to-end.

Figure 4.4 shows runtime vs. endpoint error comparisons of various optical flow estimation methods on two datasets: Sintel and KITTI2012. In both cases models of the FlowNet 2.0 family offer an excellent speed/accuracy trade-off and they also provide an opensource implementation of their methods along with the weights of the trained models. Thus we incorporate their code of calculating the displacements into our method.

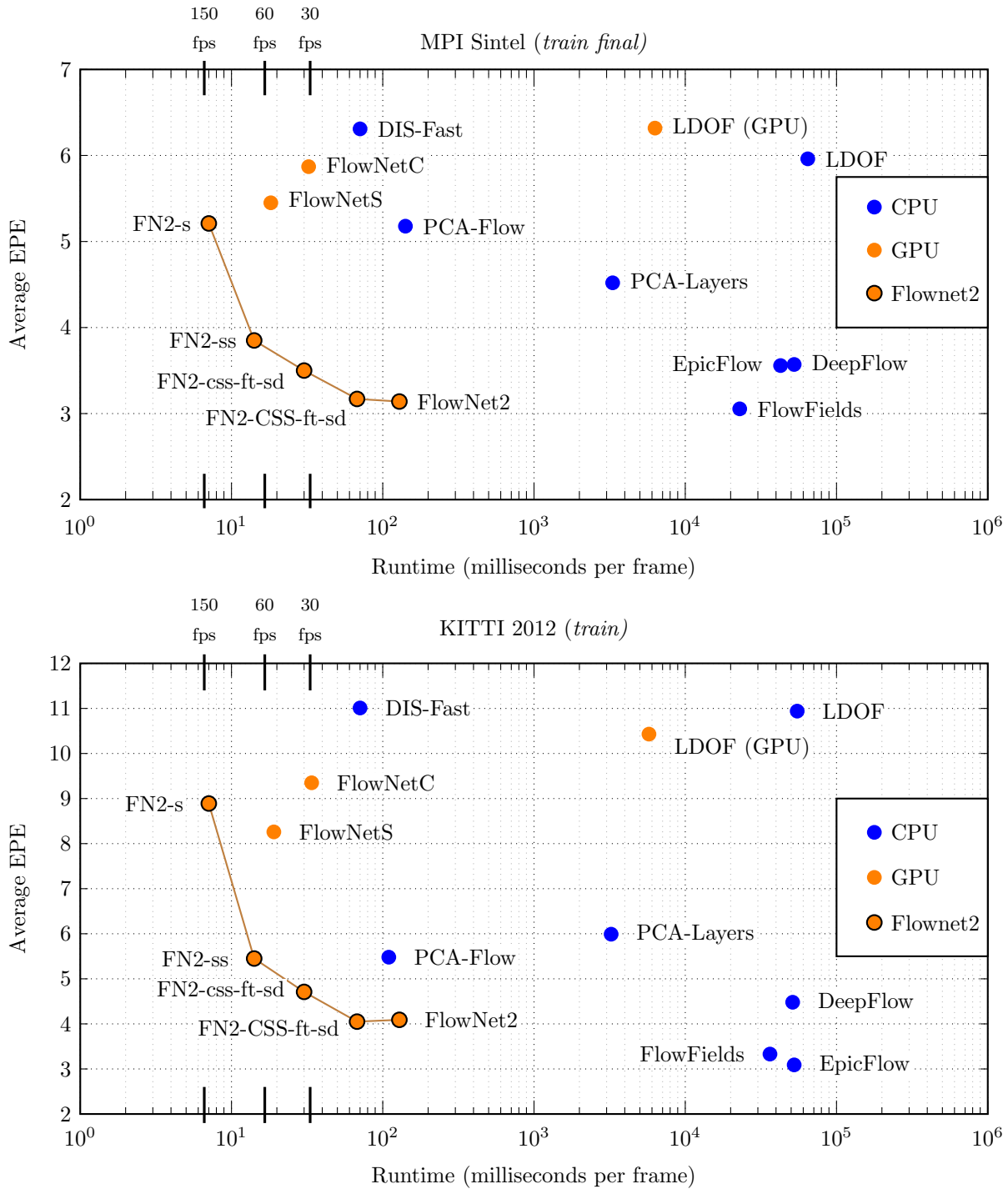


Figure 4.4: Runtime vs. endpoint error comparison to the fastest existing methods with available code. The FlowNet2 family outperforms other methods by a large margin.

The residual from frame I_t to I_{t+1} is given by

$$Res_t = \rho(I_t(w_{t,t+1}(x)) - I_{t+1}(x)) \tag{4.2}$$

Here, $w_{t,t+1}(x)$ is the warp between consecutive frames and is given by

$$w_{t,t+1}(x) = x + v_t(x) \tag{4.3}$$

where $v_t(x)$ is the displacement at x. Also $\rho(y)$ which is a truncated quadratic, is a robust norm that eliminates the explicit estimation of occlusions [10]. It is defined as

$$\rho(y) = min\{|y|, \beta\}(\beta > 0) \tag{4.4}$$

4.3 Detecting Changes

Probable Change time (t_c^*) is defined as the instance at which the probability of a change is maximum. Here a change is induced by the relative motion of objects between two frames. For example a change can be a car coming into the field of view of camera, a stationary pedestrian that start to walk, etc. Inorder to calculate the probable change time a window of size d is considered(see figure 4.5). Determining the probable change time t_c^* from a window starting at $t - d$ (d denotes the size of the window) to t is involves two tests.

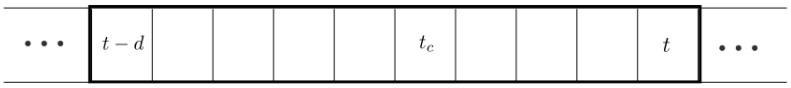


Figure 4.5: Sliding window approach to calculating probable change time

The first test is based on the spatial average of the optical flows (or residuals). For each frame in the window, we calculate the statistic defined by

$$F_{t_c,t} = (t - t_c + 1)(\hat{\mu}_{t-d:t_c-1} - \hat{\mu}_{t_c:t}) \tag{4.5}$$

Here, d denotes the size of the window and $\hat{\mu}_{t_1:t_2}$ is the average of the spatial average of flows from t_1 to t_2. Upon the acquisition of a new frame at time t, $F_{t_c,t}$ is calculated. The maximizer of $F_{t_c,t}$ over the entire window, gives us the probable change (t_c^*).

Next, we perform the standard deviation test. Now that we have (t_c^*), we calculate the standard deviation of the optical flow at (t_c^*). A large standard deviation indicates the presence of moving object. Typically if the standard deviation $\sigma_{t_c}(flow)$ exceeds a value of 5.0, the test is passed. Using optical flow for the standard deviation test instead of the residuals is more robust and generalizes well for different settings.

4.4 Motion Segmentation Scheme

Clustering based on the K-Means algorithm, originally proposed by McQueen [1], is a widely used region segmentation method for still images. [3]. These methods rely only upon intensity values of an image, spatial information and other features like texture and work only on one image at a time. In the case of videos, we propose that motion between consecutive frames would provide data that can be incorporated into the K-Means algorithm and for them basis of producing accurate and reliable segmentation of moving objects. We propose a variant of the K-Means algorithm in which we combine motion information with intensity and povide an efficient method of detecting and segmenting moving objects in video sequences.

The K-Meanns algorithm based on the motion and intensity information consist of the following steps.

1. The region number and region clusters are initialized. The simplest method based ont the parameters and testing conditions of the data-set is to set the region number to 2 and randomly initialize cluster centers.

2. for every pixel $\mathbf{p}$, the distance between $\mathbf{p}$ and all the cluster centers is calculated. The pixel is then assigned to the region for which the distance is minimized. The distance between the pixel $\mathbf{p}$ and cluster s_k is defined as follows:

$$D(\mathbf{p}, s_k) = \|U(\mathbf{p} - U(s_k)\| + \lambda \|I(\mathbf{p} - I(s_k)\| \tag{4.6}$$

In the above equation,

$$\|U(\mathbf{p} - U(s_k)\| = \sqrt{(U_u(\mathbf{p}) - U_u(s_k))^2 + (U_v(\mathbf{p}) - U_v(s_k))^2} \tag{4.7}$$

where $U_u(\mathbf{p})$ is the normalized motion at point $\mathbf{p}(x, y)$ in the x-direction and $U_v(\mathbf{p})$ is the normalized motion at point $\mathbf{p}(x, y)$ in the y-direction and

$$\|I(\mathbf{p} - I(s_k)\| = \sqrt{(I_l(\mathbf{p}) - I_l(s_k))^2 + (I_a(\mathbf{p}) - I_a(s_k))^2 + (I_a(\mathbf{p}) - I_a(s_k))^2} \tag{4.8}$$

where $I_l(\mathbf{p}), I_a(\mathbf{p}), I_a(\mathbf{p})$ are the intensity values at the position $\mathbf{p}(x, y)$ in the L,a,b color space.

λ is the weighting parameter that ensures that the pixels are assigned to a region primarily based on the similarity in motion.

3. Region centers are recalculated. If the difference between the new centers and the corrosponding centers calculated in the previous iteration is smaller than a pre-defined threshold, the algorithm has converged and is stopped returning the labels(which are the cluster centers to which each pixel is assigned)

The entire method of change detection explained above is summarised in Algorithm 1.

Algorithm 1: The main subroutine

```
Result: Segmentation and Change Detection
Get frame_1 from the sequence.;
while TRUE do
    Get frame_2 from the sequence.;
    Determine optical flow (flow) from frame_1 to frame_2;
    Determine residual (Res) from frame_1 to frame_2 using flow;
    Determine the probable change time t_c^*;
    if t_c^* is updated then
        Do the standard deviation test.;
        if standard deviation test is passed then
            Declare a detection.;
            Produce the segmentation to the screen.;
        end
    end
end
Set frame_1 to frame_2 and continue;
```

Chapter 5

Implementation

The objectives for implementation were to code the ideas defined in the previous chapters to develop a fast and lightweight method of moving object detection and segmentation. Based on this we built upon the existing tools and libraries developed for performing general purpose image processing and vector manipulation tasks. For image processing OpenCV version 2.4.11 was used for retrieving, manipulating and saving the images. Our implementation relies on heavy use of vector operations for which we used the Numpy and Scipy libraries of Python. In the succeeding sections of this chapters the Python implementation of our models is explained. Only the most important and relevant parts of the code are explained. The enitire code base is made available opensource at hosted as a github repository[21].

5.1 Implementation of Section 4.3

To implement the sliding window acrchitecture of Section 4.3 we use the class `FlowHolder`. In the code snippet below `self.flow_size` represents the size of the sliding window. A size of 15 provides a good tradeoff between accuaracy and speed. `self.flow_list` and `self.residual_list` are lists that hold the calculated optical flow and residual at each instance of the sliding window.

```
class FlowHolder:
    def __init__(self, flow_size):
        ...
        # flow_size is the maximum frames we take into consideration
        #
        self.flow_size = flow_size
        self.flow_list = []
```

```
self.residual_list = []
# Standard Deviation test threshold
# For residue based
self.std_thres = 21.0
# For optical flow based
self.std_thres_flows = 3.0

self.likelyhood_threshold = 2.5
# Occlusion threshold
self.beta = 30
# self.means containing means at each instance.
self.means = np.zeros(self.flow_size)
        ...
```

The main operation of the sliding window is to calculate the probable change time t_c^* based on the spatial average of the optical flow. If the value of probable change time t_c^* is updated, i.e it changes to some other value within the sliding window and not just the most recent one then the first test of change detection is declared to have passed. We then proceed to the second test based on the standard deviation of flows. If this standard deviation is greater than the threshold value `std_thres_flows`, then a change is detected and we proceed to the next section which is to produce a segmentation of the current frame. All this is acheived using the `processFlow` method.

```
def processFlow(self, flow):
m = flow.mean()
if self.flow_count < self.flow_size:
    self.flow_list.append(flow)
    self.flow_count += 1

    self.means[self.flow_count - 1] = m
    # Update fstat
    self.fstat[self.flow_count -1] = (self.flow_count - ...
    print(self.tc)
else:
    self.flow_list.pop(0)
```

```
            self.flow_list.append(flow)
            #Shif values to the left
            self.means = shift(self.means, -1, cval=0.0)
            self.means[self.flow_count - 1] = m
            self.fstat = shift(self.fstat, -1, cval=0.0)
            self.fstat[self.flow_count - 1] = (self.flow_count - ...
            # Check if tc is updated
            # print(self.tc)
            if self.tc != self.fstat.argmax():
                # Do standard deviation test.
                std = flow.std()
                print("Standard Deviation = " + str(std))
                # print("Change time = " + str(self.tc))
                if self.std_thres_flows < std:
                    self.tc = self.fstat.argmax()
                    # print("Change time = " + str(self.tc))
                    # Compose the flows.
                    return True
        return False
```

5.2 Implementation of Section 4.2 (Calculating Displacement)

This is a critical part of the implementation. The accuracy of our method is directly related to the accuracy of the displacement estimates between two consecutive frames. This is also the most computationally expensive step. The framerate that can be acheived is also directly dependent on the speed of the method we use for optaining the displacement estimates. For the End Point Error vs Framerate graph of the various optical flow methods shown in the Figure 4.4, it is evident that Flownet2 [19] is the best choice available at the time of writing this report. Flownet2 is implemented as a fork of the popular deep learning library Caffe [12]. Various pretrained models and weights are provided by the authors of Flownet2. They have also made their methods opensource. The code snippet below shows the important sections related to Flownet2 in our implementation.

```
...

caffemodel='FlowNet2-css-ft-sd_weights.caffemodel.h5'
deployproto='FlowNet2-css-ft-sd_deploy.prototxt.template'

caffe.set_device(gpu)
caffe.set_mode_gpu()
net = caffe.Net(tmp.name, caffemodel, caffe.TEST)

net.forward(**input_dict)

...
```

This snippet shows parts of code required for the initialization of model implemented in caffe. This is done using the `caffe.Net()` method which returns the caffe neural network object initialized with the appropriate parameters. These include the architecture defined by the prototxt file and the set of weights of the predefined model. The method `net.forward()` is used for doing a forward pass and stores the reuslts (nxnx2 channel matrix containing displacement values at each pixel) in `net.blobs` member of the object.

Figure 5.1 shows that Caffe runs much faster on GPUs compared to the CPUs. For this reason we use the gpu mode of caffe by setting the `caffe.set_device(gpu)` and `caffe.set_mode_gpu()`.

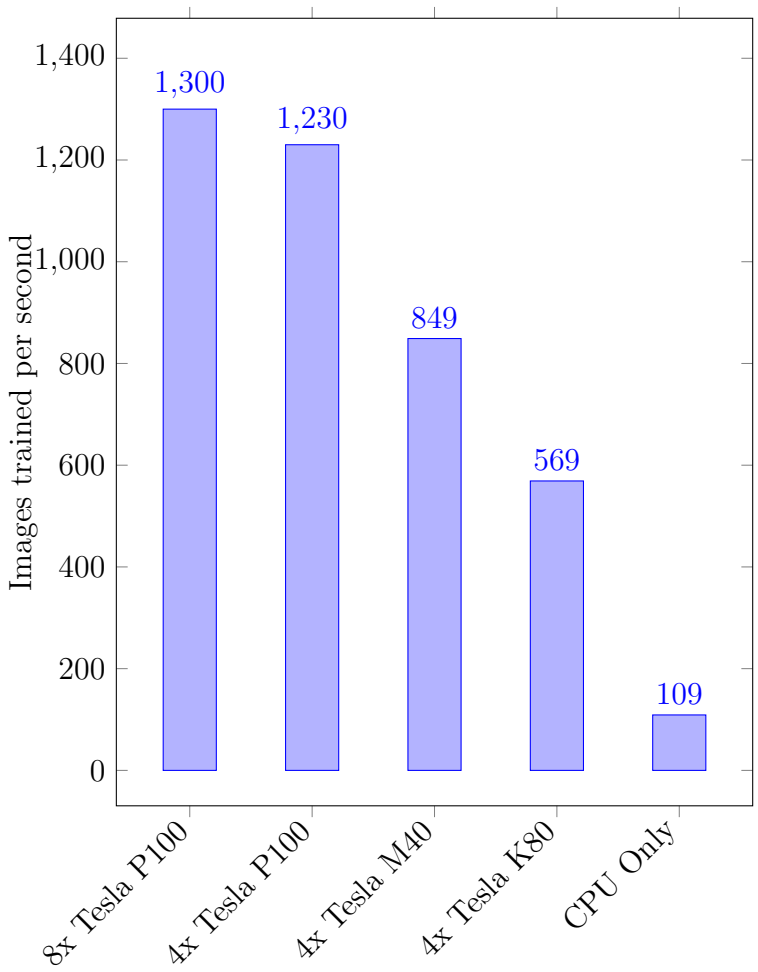


Figure 5.1: Image Classification Training Performance (GPU vs CPU)

Chapter 6

Design Parameters

Based on all the things dicussed so far in the report, a lot of design parameters that need to be set have come up in the form of type of models to be used for solving the displacements, threshold values for tests of declaring change and the motion segmentation method for the modified KMeans algorithm. Starting from the simplest paramater $\sigma_{t_c}(flow)$ from Section 4.3, we set the value of this parameter to 5.0, i.e

$$\sigma_{t_c}(flow) := 5.0 \tag{6.1}$$

We set this value based on the qualitative assesment by looking at where the changes occur in the video sequence and what the value of $\sigma_{t_c}(flow)$ is. In general, the value ranges from $0.25 - 1$ for no changes and goes very high for when there is a change say a moving car or pedestrian enters the frame. In the following sections, we discuss the setting of various other parameters.

6.1 Selecting the correct Flownet2 model

Flownet2 as discussed earlier provides various models for solving the optical flow. The most accurate model is the FlowNet2 4.4. How ever this is also the slowest and memory intensive model (with the caffe weights filesize of around 700MB). This model runs at around 5FPS which is too slow for our implementation. Besides using this model results in frequent out of memory errors thrown by caffe. Next is the FlowNet2-CSS 4.4 which is slightly less accurate but runs at around 10FPS. The CD2014 dataset contains images of two sizes 1) 320x240 and 2) 768x360. The sequences with the larger size result in out of memory errors thrown by caffe so we discarded this model. Finally the Flownet2-css-ft-sd 4.4 which is the finetuned version of the smaller class of models made available by flownet results in a good balance between accuracy, speed and compatability. This model runs at around 30FPS and is small in size (around 60MB). There are no out of memory errors thrown by caffe.

6.2 Setting segmentation threshold

In the KMeans scheme explained in section 4.4, we use the cluster centers $s_k, k = 0, 1$to determine the foreground and background. The cluster with a larger value corrosponds to the objects that are actually in motion as the clusters are primarily based on motion information. The cluster with the larger value of s_k is assigned the label of fore ground which is 255 and the cluster with the smaller value of s_k is assigned the label of background which is 0. However, this scheme results in unnecessary artifacts in case of no or very little motion in the frame. This is shown in the figure. To overcome this problem, we use a threshold value called the `std_threshold`. The graph

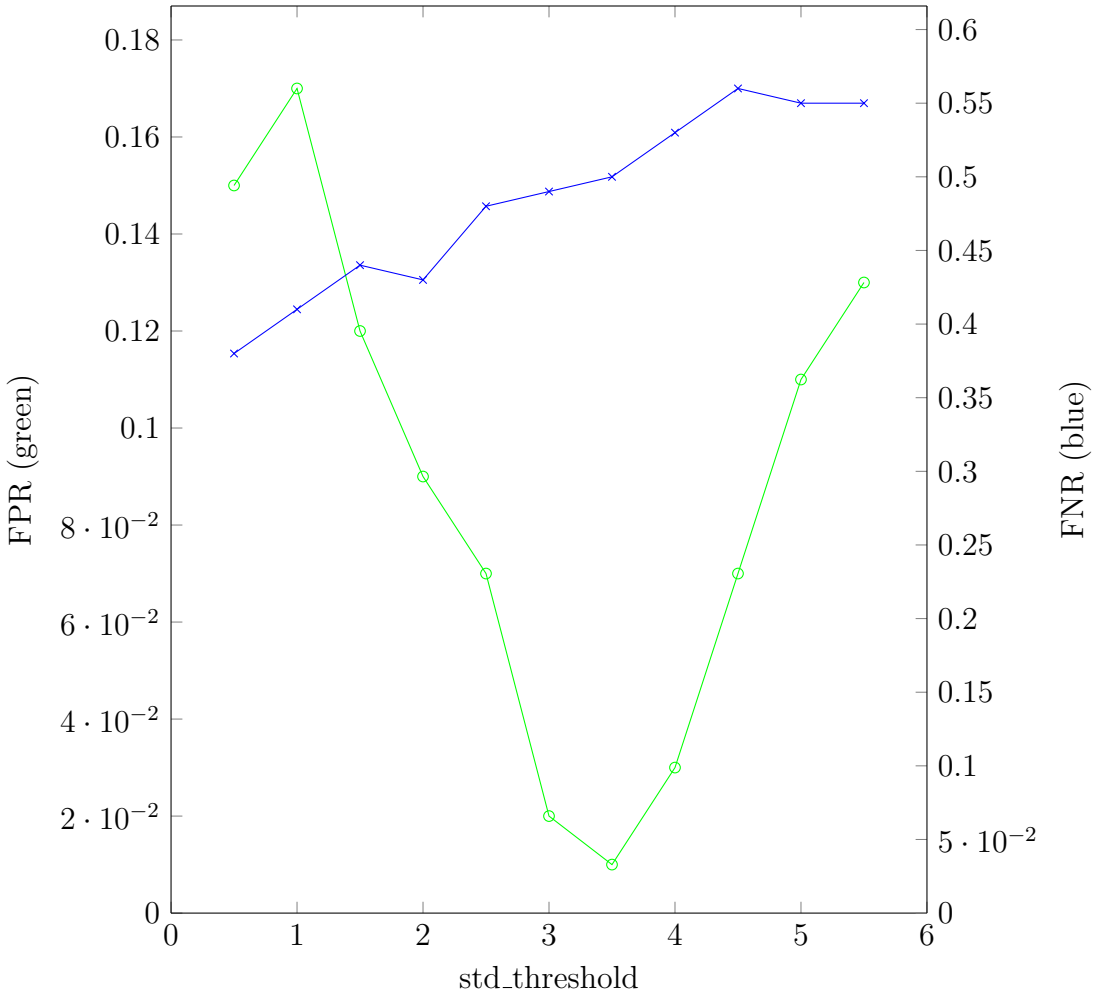


Figure 6.1: Tradeoff between FPR and FNR for selecting s_1 threshold (Average values for the Baseline category are shown)

in Figure 6.1 shows that the optimum value of the `std_threshold`. The FPR decreases when we use the `std_threshold` and reaches its minimum at a value in the range of 3 and 4. This is becuase of the decrease in the artifacts shown in the Figure 6.2. The false negative rate increases

with the increase in the value of `std_threshold`. This is because more and more pixels are left out and labelled as 0 as the value of `std_threshold` increases. An optimum value for the value of `std_threshold` for minimum FPR and FNR is reached in the range of 3 to 4. Using this reasoning we set the value at 3.5 i.e,

$$std\_threshold := 3.5 \tag{6.2}$$

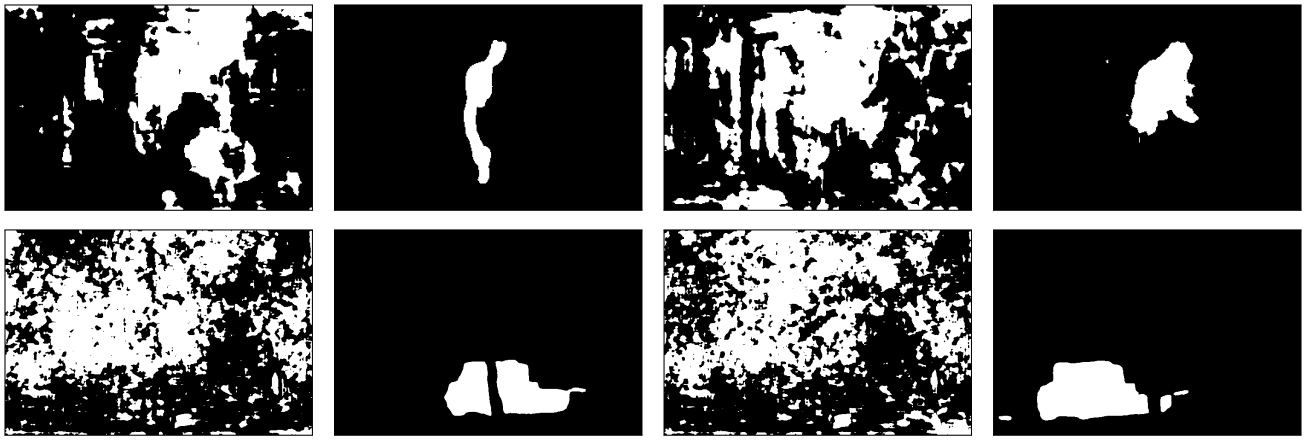


Figure 6.2: Removal of artifacts by using std_threshold.

Chapter 7

Results and Analysis

7.1 CDNet Dataset

To evaluate our method, we conduct experiments on the CDnet 2014 dataset [11].CDnet, presented at the IEEE Change Detection Workshop [11] consists of 53 videos depicting indoor and outdoor scenes with boats, cars, trucks, and pedestrians that have been captured in different scenarios and contain a range of challenges. The videos have been obtained with different cameras ranging from low-resolution IP cameras,through mid-resolution camcorders and PTZ cameras, to thermal cameras. As a consequence, spatial resolutions of the videos in CDnet vary from 320x240 to 720x576. Also, due to diverse lighting conditions present and compression parameters used, the level of noise and compression artifacts varies from one video to another. The length of the videos also varies from 1,000 to 8,000 frames and the videos shot by low-end IP cameras suffer from noticeable radial distortion. Different cameras may have different hue bias (due to different white balancing algorithms employed) and some cameras apply automatic exposure adjustment resulting in global brightness fluctuations in time. We believe that the fact that our videos have been captured under a range of settings will help prevent this dataset from favoring a certain family of change detection methods over others.

The videos are grouped into eleven categories according to the type of challenge each represents. The videos have been selected so that the challenge in one category is unique to that category. For example, only videos in the “Shadows” category contain strong shadows and only those in the “Dynamic Background” category contain strong parasitic background motion. Such a grouping is essential for a clear identification of the strengths and weaknesses of different change detection methods. With the exception of one video in the “Baseline” category, that comes from the PETS 2006 dataset, all the videos have been captured by the authors.There are 5 new categories a. Similarly to the 2012 dataset, the change detetion challenge in a category is unique to that category. Such a grouping is essential for an unbiased and clear identification of the strengths and weaknesses

of different methods.

7.1.1 Video Categories

31 videos totaling nearly 90,000 frames are grouped into the following 11 categories that have been selected to cover a wide range of change detection challenges that are representative of typical visual data captured today in surveillance, smart environment, and video analytics applications:

1. **Baseline:**This category contains four videos, two indoor and two outdoor. These videos represent a mixture of mild challenges typical of the next 4 categories.Some videos have subtle background motion, others have isolated shadows, some have an abandoned object and others have pedestrians that stop for a short while and then move away. These videos are fairly easy, but not trivial, to process, and are provided mainly as reference.

2. **Dynamic Background:** There are six videos in this category depicting outdoor scenes with strong (parasitic) background motion. Two videos represent boats on shimmering water, two videos show cars passing next to a fountain, and the last two depict pedestrians, cars and trucks passing in front of a tree shaken by the wind.

3. **Camera Jitter:** his category contains one indoor and three outdoor videos captured by unstable (e.g.,vibrating) cameras. The jitter magnitude varies from one video to another.

4. **Shadows:** This category consists of two indoor and four outdoor videos exhibiting strong as well as faint shadows. Some shadows are fairly narrow while others occupy most of the scene. Also, some shadows are cast by moving objects while others are cast by trees and buildings.

5. **Intermittent Object Motion:** This category contains six videos with scenarios known for causing "ghosting" artifacts in the detected motion, i.e., objects move, then stop for a short while, after which they start moving again. Some videos include still objects that suddenly start moving, e.g., a parked vehicle driving away, and also abandoned objects. This category is intended for testing how various algorithms adapt to background changes.

6. **Thermal:** In this category, five videos (three outdoor and two indoor) have been captured by far-infrared cameras. These videos contain typical thermal artifacts such as heat stamps (e.g., bright spots left on a seat after a person gets up and leaves), heat reflection on floors and windows, and camouflage effects, when a moving object has the same temperature as the surrounding regions.

7. **Challenging Weather:** This category contains 4 outdoor videos showing low-visibility winter storm conditions. This includes two traffic scenes in a blizzard, cars and pedestrians at the corner of a street and people skating in the snow. These videos present a double challenge: in addition to snow accumulation, the dark tire tracks left in the snow have potential to cause false positives.

8. **Low Frame-Rate:** In this category 4 videos, all recorded with IP cameras, are included. The frame rate varies from 0.17 fps to 1 fps due to limited transmission bandwidth. By nature, these videos show "erratic motion patterns" of moving objects that are hard (if not impossible) to correlate. Optical flow might be ineffective for these videos. One sequence is particularly challenging (port 0 17fps), which shows boats and people coming in and out of a harbour, as 390 the low framerate accentuates the wavy motion of moored boats causing false detections.

9. **Night:** This category has 6 motor traffic videos. The main challenge is to cope with low-visibility of vehicles yet their very strong headlights that cause over saturation. Headlights cause halos and reflections on the street.

10. **PTZ:** This category includes 4 videos: one video with a slow continuous camera pan, one video with an intermittent pan, one video with a 2-position patrol-mode PTZ, and one video with zoom-in/zoom-out. The PTZ category by itself requires different type of change detection techniques in comparison to static camera videos.

11. **Air Turbulence:** This category contains 4 videos showing moving objects depicted by a near-infrared camera at noon during a hot summer day. Since the scene is filmed at a distance (5 to 15 km) with a telephoto lens, the heat causes constant air turbulence and distortion in frames. This results in false positives. The size of the moving objects also varies significantly from one video to another. The air turbulence category presents very similar challenges to those arising in long-distance remote surveillance applications

7.1.2 Ground-Truth Labels

The current online datasets have been designed mainly for testing tracking and scene understanding algorithms, and thus the ground truth is provided in the form of bounding boxes. Although this can be used to validate change detection methods, a precise validation requires ground truth at pixel resolution. Therefore, ideally, videos should be labeled a number of times by different persons and the results averaged out. This, however, is impractical due to resource and time constraints. Furthermore, it is very difficult for a person to produce uncontroversial binary ground-truth images for camera-captured videos. This is particularly difficult near moving object boundaries and in

semi-transparent areas. Due to motion blur and partially-opaque objects (e.g., sparse bushes, dirty windows, fountains), pixels in these areas may contain both the moving object and background. As a consequence, one cannot reliably classify such pixels as belonging to either Static or Moving class. Since these areas carry a certain level of uncertainty, evaluation metrics should not be computed for pixels in these areas. Therefore, we decided to produce ground-truth images with the following labels:

- **Static:** assigned grayscale value of 0.
- **Shadow:** assigned grayscale value of 50.
- **Non-ROI:** assigned grayscale value of 85.
- **Unknown:** assigned grayscale value of 170.
- **Moving** assigned grayscale value of 255.

The Static and Moving classes are associated with pixels for which the motion status is obvious. The Shadow label is associated with hard and well-defined moving shadows . Hard shadows are among the most difficult artifacts to cope with and we believe that adding this extra information improves the richness and utility of the dataset. Please note that evaluation metrics consider the Shadow pixels as Static pixels. The Unknown label is assigned to pixels that are half-occluded and those corrupted by motion blur. All pixels located close to moving-object boundaries are automatically labeled as Unknown. This prevents evaluation metrics from being corrupted by pixels whose status is unclear.

The Non-ROI (not in region of interest) label serves two purposes. Firstly, since most change detection methods incur a delay before their background model stabilizes, we labeled the first few hundred frames of each video sequence as Non-ROI. This prevents the corruption of evaluation metrics due to errors during initialization. Secondly, the Non-ROI label prevents the metrics from being corrupted by activities unrelated to the category considered.

7.1.3 Evaluation Metrics

Finding the right metric to accurately measure the ability of a method to detect motion or change without producing excessive false positives and false negatives is not trivial. For instance, recall favors methods with a low False Negative Rate. On the contrary, specificity favors methods with a low False Positive Rate. Having the entire precision-recall tradeoff curve or the ROC curve would be ideal, but not all methods have the flexibility to sweep through the complete gamut of

tradeoffs. In addition, one cannot, in general, rank-order methods based on a curve. We deal with these difficulties by reporting the average performance of each method for each video category with respect to 7 different performance metrics each of which has been well-studied in the literature. Specifically, for each method, each video category, and each metric, we report the performance (as measured by the value of the metric) of the method averaged across all the videos of the category.

Let TP = number of true positives, TN = number of true negatives, FN = number of false negatives, and FP = number of false positives. The 7 metrics that we use are:

- Recall ($Re = \frac{TP}{(TP+FN)}$): Recall is used to measure the fraction of positive patterns that are correctly classified
- Specificity ($Sp = \frac{TN}{(TN+FP)}$: This metric is used to measure the fraction of negative patterns that are correctly classified.
- False Positive Rate ($FPR = \frac{FP}{(FP+TN)}$: This metric is used to measure the fraction of positive patterns that are incorrectly classified.
- False Negative Rate ($FNR = \frac{FN}{(TN+FP)}$: This metric is used to measure the fraction of negative patterns that are incorrectly classified.
- Percentage of Wrong Classifications ($PWC = \frac{100(FN+FP)}{(TP+FN+FP+TN)}$): Misclassification error measures the percentage of incorrect predictions over the total number of instances evaluated.
- Precision ($Pr = \frac{TP}{(TP+FP)}$): Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class.
- F-measure ($\frac{2PrRe}{(Pr+Re)}$): This metric represents the harmonic mean between recall and precision values

7.2 Results

The results of the evaluation of our method on the entire dataset of Change Detection 2014 [11] are shown in Table 7.1. The entire evaluation results of our method on the CDnet dataset are shown in Table I. It can be seen that our method performs well for Baseline and Shadow categories, with F-measures at the level of 0.5. Nevertheless, PTZ and Night Videos categories pose heavy challenges to our method, with F-measures less than 0.2. For the Baseline category, our method is affected by a video named "PETS2006", where a person keeps motionless in a subway station for a while, causing a number of false negatives. For the Dynamic Background category, our method is robust

to background motion within a certain range. However, drastic background motion can cause numerous false positives, like in videos "fountain01" and "fall". For the Camera Jitter category, our method performs satisfactorily in general, except on the video "boulevard", in which there is a mixture of severe challenges of camera jitter, camouflage, and camera automatic adjustments. Since our method is a bottom-up method and exploits no object-level cues, Intermittent Object Motion poses a heavy challenge and cause many false classifications. This is because changes in this category are not induced by motion. For example in the "streetLight" sequence, changes are in the form of a stationary traffic signal turning on and off. In general any sequence where changes are induced by something other than relative motion like change in color pose heavy challenges for our method. This is because, as mentioned earlier, our entire method is based on classifying objects based on motion cues so bad results for this kind of frames are expected. For the Shadow category, our method succeeds in eliminating soft shadows, but is not good at handling hard shadows, with FPR-S as high as 0.5749.

For Thermal category, even though the video images are in black and white (without chromaticity information available), our method achieves acceptable performance. It acheives an F-measure of 0.43 with PWC at 8 percent. Again Bad Weather poses heavy difficulty for our method. This is because of limited region of interest and objects being occluded due to artifacts in the video. Low Framerate itself does not become a challenge to our method. However, a video called "port_0_17fps" poses a great challenge due to the mixture of global illumination changes and dynamic background. In Night Videos, our method suffers from vehicle headlight reflections on the road and camouflage. PTZ camera poses the greatest challenge to our method, especially when the camera rotates continuously.

Table 7.1: Evaluation results for each category of the CDNET Dataset

Video Category	Recall	Specificity	FPR	FNR	PWC	Precision	F-measure
Baseline	0.517	0.984	0.015	0.482	3.620	0.562	0.479
PTZ	0.516	0.787	0.212	0.483	21.493	0.106	0.139
Dyn. Background	0.697	0.751	0.248	0.302	24.606	0.103	0.139
Bad Weather	0.549	0.900	0.099	0.450	10.660	0.147	0.210
Thermal	0.428	0.958	0.041	0.571	8.128	0.457	0.430
Inter. Obj. Mtn	0.405	0.817	0.182	0.594	20.093	0.293	0.272
Shadow	0.447	0.986	0.013	0.552	3.799	0.641	0.501
Low Framerate	0.255	0.969	0.030	0.744	5.413	0.446	0.281
Turbulence	0.385	0.718	0.281	0.614	28.339	0.018	0.034
Night Videos	0.383	0.947	0.052	0.616	6.342	0.183	0.227
Camera Jitter	0.518	0.684	0.315	0.481	32.555	0.072	0.122
Overall Avg	0.464	0.864	0.135	0.535	15.004	0.275	0.258

A comparsion with the state of the art methods is shown in Table 7.2.Although not quite in league with the best of the methods like M4CD [20], our runs much faster than most of these method that require on average more than one second to process one frame of the video.

Table 7.2: Comparison of our method with state of the art in overall metrics

Method ID	Average Ranking	Average Recall	Average Specificity	Average FPR	Average FNR	Average PWC	Average Precision	Average F-measure
[14]	2.71	0.7849	0.9948	0.0052	0.2151	1.1986	0.8087	0.7403
[16]	7.57	0.8124	0.9904	0.0096	0.1876	1.6780	0.7509	0.7408
[17]	7.00	0.8098	0.9912	0.0088	0.1902	1.4996	0.7503	0.7474
[13]	9.14	0.7657	0.9922	0.0078	0.2343	1.3763	0.7696	0.7283
[20]	12.29	0.7885	0.9841	0.0159	0.2115	2.3011	0.7423	0.7038
[15]	10.43	0.7714	0.9914	0.0086	0.2286	1.8969	0.7628	0.7176
*	10.71	0.7416	0.9923	0.0077	0.2584	1.8902	0.7754	0.7129
CwisarDRP	10.57	0.7062	0.9947	0.0053	0.2938	1.7197	0.7880	0.7095
Ours	-	0.4641	0.8642	0.1357	0.5358	15.004	0.2758	0.2580

*Superpixel Strengthen Background Subtraction.

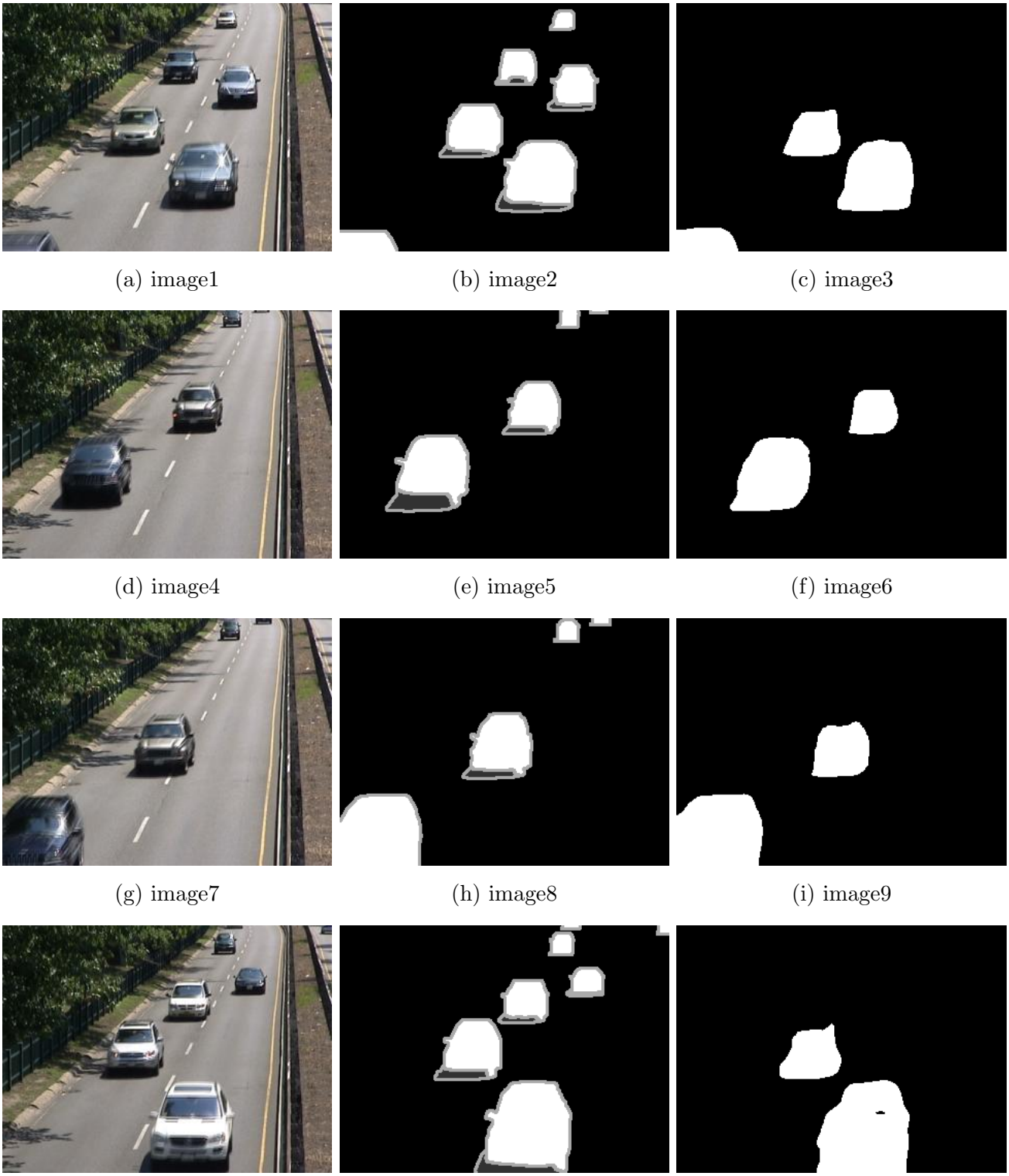
7.2.1 Computational Time

We implemented our method on a laptop computer running Ubuntu 14.04 with 3.1GHz Intel Core i7 CPU and 8GB memory. The GPU we used was the Nvidia Geforce 920M with 4GB video memory. The driver version used was Nvidia 385 and with CUDA 8. The computational time is monitored by `time.time()` function. Table 7.3 shows the average time of processing one frame with two different resolutions 320x240 and 720x480. The cost details of three modules corresponding to Sections are also given. It takes around 0.07 seconds to process one frame, so the system in close to real-time frame rate. In the future, we will use parallel computing or gpu computing platforms to speed up the kmeans clustering method. The cost details of three modules corresponding to Sections are also given. It takes around 0.07 seconds to process one frame, so the system in close to real-time frame rate. In the future, we will use parallel computing or gpu computing platforms to speed up the kmeans clustering method.

Some qualitative results from different video categories are shown below. The first column is the input image, the second column is the ground truth and the third column is the output of our method taken from some some sequences.

Table 7.3: Average time for processing one frame

Operation	Avg Processing time in sec (320x240)	Avg Processing time in sec (720x480)
Optical Flow calculation using Flownet2-css-ft-sd	0.06	0.11
Change detection from Section4.3	0.03	0.03
Segmentation using KMeans from Section4.4	0.05	0.09
Total	0.14	0.23



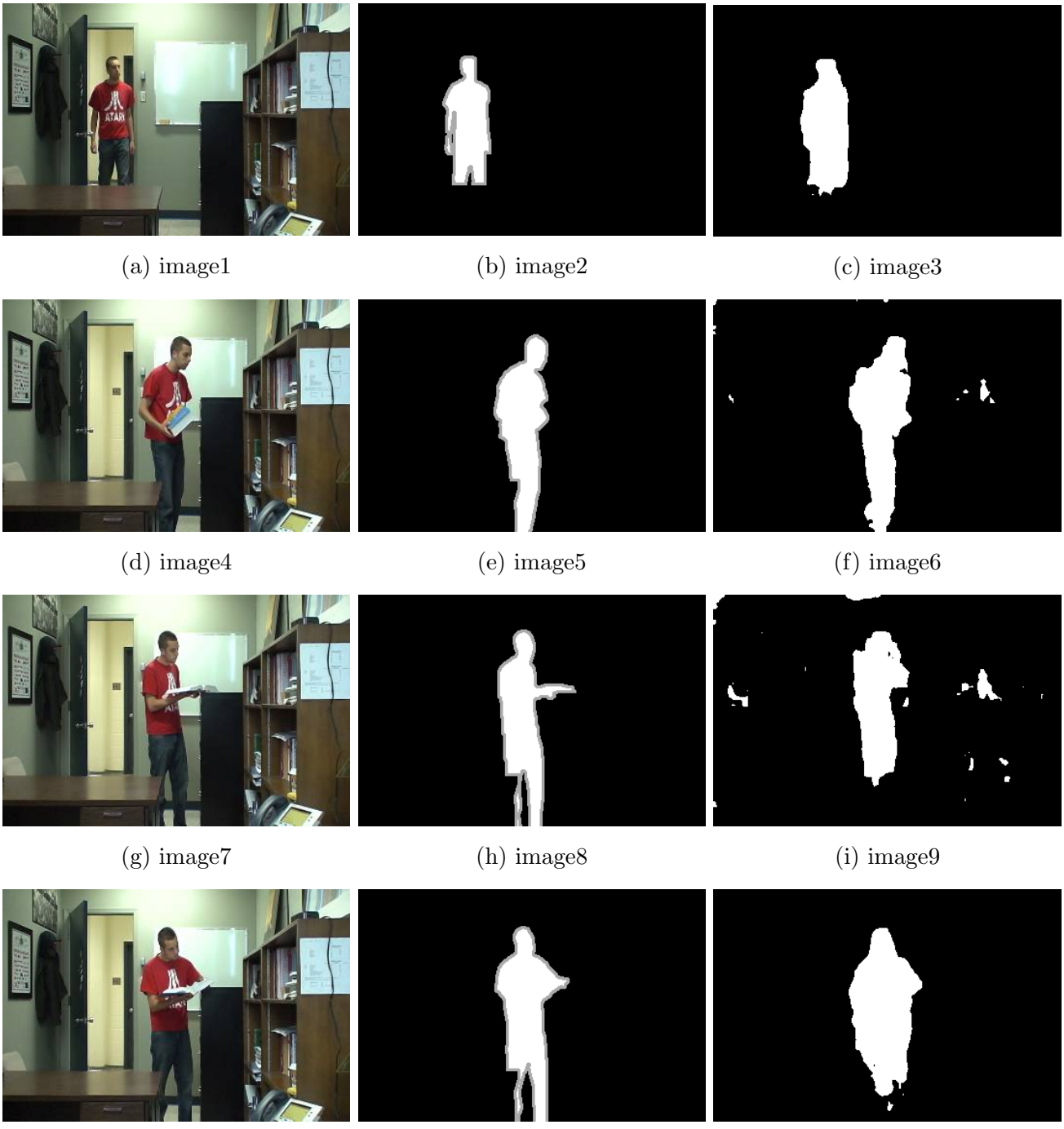
(a) image1 (b) image2 (c) image3

(d) image4 (e) image5 (f) image6

(g) image7 (h) image8 (i) image9

(j) image10 (k) image11 (l) image12

Figure 7.1: Highway sequence from baseline category



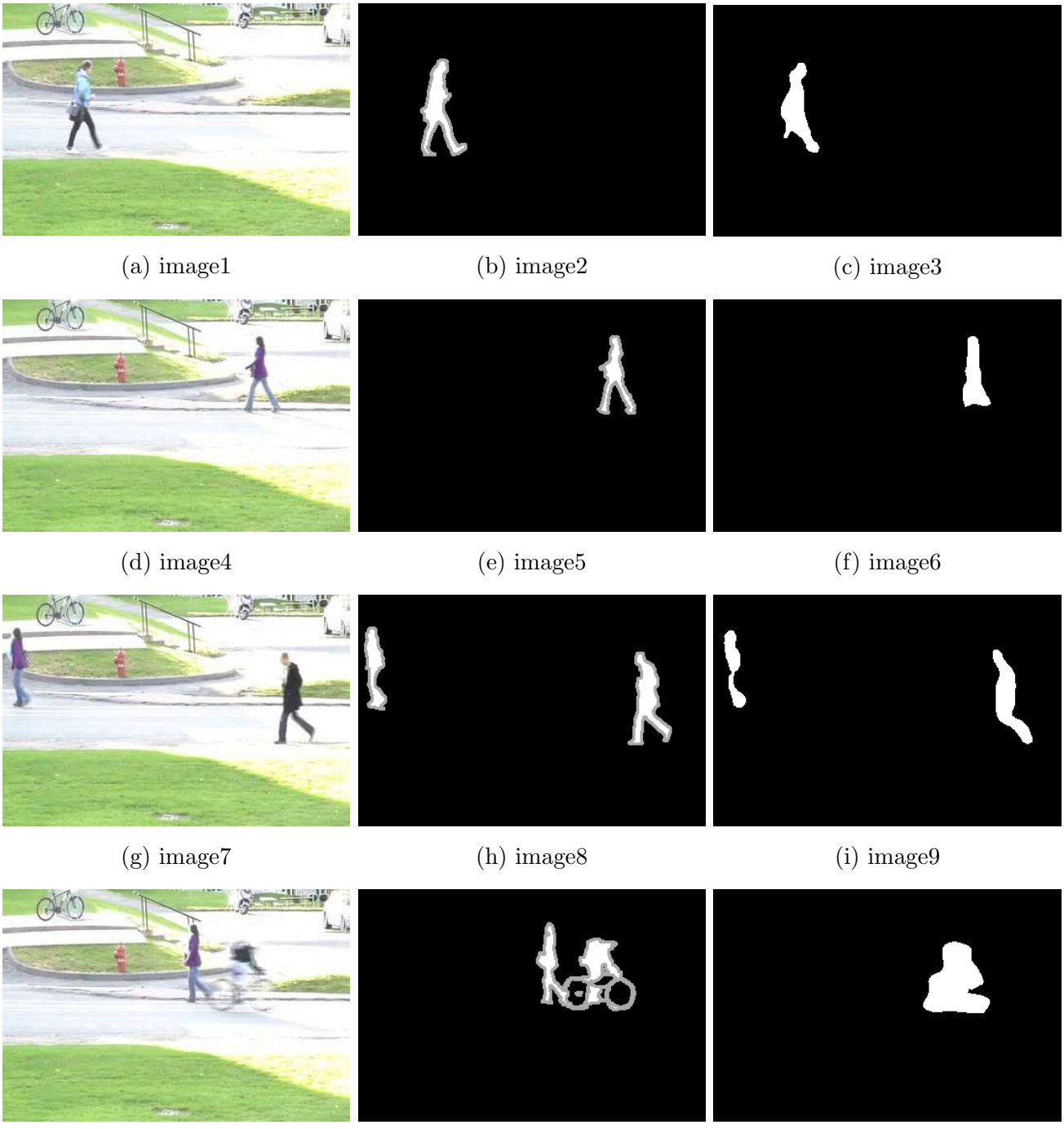
(a) image1 (b) image2 (c) image3

(d) image4 (e) image5 (f) image6

(g) image7 (h) image8 (i) image9

(j) image10 (k) image11 (l) image12

Figure 7.2: Office sequence from baseline category.



(a) image1 (b) image2 (c) image3

(d) image4 (e) image5 (f) image6

(g) image7 (h) image8 (i) image9

(j) image10 (k) image11 (l) image12

Figure 7.3: Pedestrians seqence from baseline category.

Chapter 8

Conclusions

We have devised a framework for moving object detection and background subtraction task. The proposed method is based on three processing steps. These steps are

- Solving for displacements: This step involoves looking at two consecutive frames and determining the displacement of each pixel in the first frame corrosponding to the second frame. This step generates the most important data for our subsequent step and forms the basis of our method. This step is also the most time consuming but since we have adopted a modular approach different methods can be used to replace the method we have used. With the success of CNNs in estimating optical flow and advances in GPU computing, newer used in this step methods will be able to acheive higher accuracy.

- Change Detection: This step involve looking at a sequence of frames in a sliding window approach and determining wheather there is actually a moving object in the current frame. This is an important step that increases the speed of our method by removing the need to perform the next step if there is no change. We simply label the entire frame as background. This step however results in a complication that when objects move but then become stationary after some frames, it is expected that the object will be segmented. No motion in the frame stops the process at this step and hence an incorrect or unexpected output is produced.

- Segmentation: We rely on the unsupervised KMeans clustering method for grouping the pixels of the same image. The fastest and the simplest case is to initialize KMeans with two clusters. We define a distance metric based on the pixel values and the motion information. Other information like the spatial cordinates and texture information can also be incorporated into the distance metric. This however reduces the running time of our algorithm. At the expense of gaining higher accuracy, we thus prefer the simpler method in our implementation.

Finally we test our method on the Change Detection 2014 [11] dataset and benchmarks. The dataset contains video sequences from of categories. Our method performs well in some of the categories like Baseline, shadow etc. These include simpler cases with limited background and camera motions. The images are comparatively less corrupted. On the other hand, more challenging categories like PTZ, Camera Jitter, Dynamic Background Motion prove to be very challenging. The results in these categories reflect a poor performance. However, considering that methods like [20] which are at the top of the benchmarks use complex methods which result in slow performance, our method is optimized for speed and we lean towards making our method as fast as possible in the speed vs accuracy trade off.

Bibliography

[1] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

[2] Laurenz Wiskott. "Segmentation from Motion: Combining Gabor- and Mallat-Wavelets to Overcome Aperture and Correspondence Problem". In: *Proceedings of the 7th International Conference on Computer Analysis of Images and Patterns*. CAIP '97. London, UK, UK: Springer-Verlag, 1997, pp. 329–336. ISBN: 3-540-63460-6. URL: http://dl.acm.org/citation.cfm?id=648241.752692.

[3] Vasileios Mezaris, Ioannis Kompatsiaris, and MichaelG. Strntzis. "Still Image Segmentation Tools for Object-Based Multimedia Applications". In: *International Journal of Pattern Recognition and Artificial Intelligence* 18.04 (2004), pp. 701–725. eprint: https://www.worldscientific.com/doi/pdf/10.1142/S0218001404003393.

[4] A. Cavallaro, O. Steiger, and T. Ebrahimi. "Tracking Video Objects in Cluttered Background". In: *IEEE Trans. Cir. and Sys. for Video Technol.* 15.4 (Apr. 2005), pp. 575–584. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2005.844447. URL: http://dx.doi.org/10.1109/TCSVT.2005.844447.

[5] Fang-Hsuan Cheng and Yu-Liang Chen. "Real Time Multiple Objects Tracking and Identification Based on Discrete Wavelet Transform". In: *Pattern Recogn.* 39.6 (June 2006), pp. 1126–1139. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2005.12.010. URL: http://dx.doi.org/10.1016/j.patcog.2005.12.010.

[6] A. Colombari, A. Fusiello, and V. Murino. "Segmentation and Tracking of Multiple Video Objects". In: *Pattern Recogn.* 40.4 (Apr. 2007), pp. 1307–1317. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.07.008. URL: http://dx.doi.org/10.1016/j.patcog.2006.07.008.

[7] Renjie Li, Songyu Yu, and Xiaokang Yang. "Efficient Spatio-temporal Segmentation for Extracting Moving Objects in Video Sequences". In: *IEEE Trans. on Consum. Electron.* 53.3 (Aug. 2007), pp. 1161–1167. ISSN: 0098-3063. DOI: 10.1109/TCE.2007.4341600. URL: https://doi.org/10.1109/TCE.2007.4341600.

[8] Yogesh Rathi et al. "Tracking Deforming Objects Using Particle Filtering for Geometric Active Contours". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29.8 (Aug. 2007), pp. 1470–1475. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1081. URL: http://dx.doi.org/10.1109/TPAMI.2007.1081.

[9] Luca Zappella, Xavier Lladó, and Joaquim Salvi. "Motion Segmentation: A Review". In: *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2008, pp. 398–407. ISBN: 978-1-58603-925-7. URL: http://dl.acm.org/citation.cfm?id=1566899.1566953.

[10] D. Sun, S. Roth, and M. J. Black. "Secrets of optical flow estimation and their principles". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2010, pp. 2432–2439.

[11] N. Goyette et al. "Changedetection.net: A new change detection benchmark dataset". In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. June 2012, pp. 1–8. DOI: 10.1109/CVPRW.2012.6238919.

[12] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[13] R. Wang et al. "Static and Moving Object Detection Using Flux Tensor with Split Gaussian Models". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. June 2014, pp. 420–424. DOI: 10.1109/CVPRW.2014.68.

[14] Simone Bianco, Gianluigi Ciocca, and Raimondo Schettini. "How Far Can You Get By Combining Change Detection Algorithms?" In: *CoRR* abs/1505.02921 (2015). arXiv: 1505.02921. URL: http://arxiv.org/abs/1505.02921.

[15] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin. "A Self-Adjusting Approach to Change Detection Based on Background Word Consensus". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. Jan. 2015, pp. 990–997. DOI: 10.1109/WACV.2015.137.

[16] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin. "SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity". In: *IEEE Transactions on Image Processing* 24.1 (Jan. 2015), pp. 359–373. ISSN: 1057-7149. DOI: 10.1109/TIP.2014.2378053.

[17] Yingying Chen, Jinqiao Wang, and Hanqing Lu. "Learning sharable models for robust background subtraction". In: *2015 IEEE International Conference on Multimedia and Expo (ICME)*. June 2015, pp. 1–6. DOI: 10.1109/ICME.2015.7177419.

[18] Philipp Fischer et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *CoRR* abs/1504.06852 (2015). arXiv: 1504.06852. URL: http://arxiv.org/abs/1504.06852.

[19] Eddy Ilg et al. *FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*. 2016. eprint: arXiv:1612.01925.

[20] Kunfeng Wang, Chao Gou, and Fei-Yue Wang. "M4CD: A Robust Change Detection Method for Intelligent Visual Surveillance". In: *CoRR* abs/1802.04979 (2018). arXiv: 1802.04979. URL: http://arxiv.org/abs/1802.04979.

[21] Samim Taray. *Minimum Delay Moving Object Detection in Realtime*. URL: https://github.com/zsameem/realtime-mdmod.