GPU-ENABLED REAL-TIME PANORAMIC BACKGROUND SUBTRACTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERDAR BÜYÜKSARAÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2015

Approval of the thesis:

GPU-ENABLED REAL-TIME PANORAMIC BACKGROUND SUBTRACTION

submitted by **SERDAR BÜYÜKSARAÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Eng.**

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Assoc. Prof. Dr. Alptekin Temizel
Co-supervisor, **Modeling and Simulation Dept., METU**

Examining Committee Members:

Prof. Dr. Uğur Halıcı
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Alptekin Temizel
Modeling and Simulation Dept., METU

Assoc. Prof. Dr. Cüneyt Fehmi Bazlamaçcı
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Hakkı Alparslan Ilgın
Electrical and Electronics Engineering Dept., Ankara Uni.

Date:

ABSTRACT

GPU-ENABLED REAL-TIME PANORAMIC BACKGROUND SUBTRACTION

Büyüksaraç, Serdar
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Gözde Bozdağı Akar
Co-Supervisor: Assoc. Prof. Dr. Alptekin Temizel

September 2015, 103 pages

Extraction of foreground objects using a Pan-Tilt camera is a challenging task for various video surveillance applications. It requires several steps such as camera motion extraction, image registration, panorama generation and background subtraction. All these steps require significant computing power. While achieving this by using only Central Processing Unit (CPU) is a challenging task, it might be enabled by efficient parallelization of the algorithms to run on Graphics Processing Unit (GPU). In this thesis an adaptive panoramic background generation and foreground object detection algorithm is implemented on GPU/CPU to run in real-time.

Keywords: Background Subtraction, Panorama Extraction, Pan-Tilt Camera, Graphics Processing Unit (GPU), Compute Unified Device Architecture (CUDA)

ÖZ

GPU KULLANILARAK GERÇEK ZAMANLI PANORAMİK ARKAPLAN ÇIKARMA

Büyüksaraç, Serdar
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Gözde Bozdağı Akar
Ortak Tez Yöneticisi : Assoc. Prof. Dr. Alptekin Temizel

Eylül 2015, 103 sayfa

Yatayda ve düşeyde hareket edebilen bir kamera kullanarak ön plandaki nesnelerin çıkarımı çeşitli video gözetim uygulamaları için zorlu bir görevdir. Bu iş kamera hareketinin tespiti, görüntüde yapılan düzeltme, panorama oluşturulması ve arka plan çıkarılması gibi birçok adım gerektirir. Bu adımlar önemli işlem gücü gerektirir. Sadece Merkezi İşlem Birimi (CPU) kullanarak bu hedefe ulaşmak zor bir görev olsa da, algoritmaların etkin bir biçimde paralelize edilmesiyle Grafik İşleme Birimi (GPU) üzerinde çalışması sağlanabilir. Bu tezde uyarlanabilir panoramik arka plan oluşturan ve ön plan nesnelerini çıkartan bir algoritma GPU ve CPU üzerinde gerçek zamanlı olarak çalışması için uygulanmıştır.

Anahtar Kelimeler: Arka Plan Çıkartma, Panorama Oluşturma, Yatayda ve Düşeyde Hareket Edebilen Kamera, Grafik İşleme Birimi, Birleşik Hesap Cihazı Mimarisi

To My Family

ACKNOWLEDGMENTS

I express my sincere appreciation to my thesis supervisor Prof. Dr. Gözde Bozdağı Akar and co-supervisor Assoc. Prof. Dr. Alptekin Temizel for their guidance, insight and elegant attitude throughout the research.

I wish to thank my parents Hatice and Metin Büyüksaraç and my sister Ayşenur Büyüksaraç for their support, encouragement and confidence throughout the years of my education.

I would like to thank to my company ASELSAN and my colleagues for their understanding.

And of course I would like to express my greatest thanks to Buket for her love, invaluable support, encouragement and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AMP	: Accelerated Massive Parallelism
BRIEF	: Binary Robust Independent Elementary Features
CB	: Codebook
CPU	: Central Processing Unit
CUDA	: Compute Unified Device Architecture
CW	: Code Word
CWSTR	: Code Word Structure
DoF	: Degrees of Freedom
DoG	: Difference of Gaussians
EM	: Expectation Maximization
FAST	: Features from Accelerated Segment Test
FLANN	: Fast Library for Approximate Nearest Neighbors
Fm	: F-measure
FMA	: The Fused Multiply-Add
FN	: False Negative
FNR	: False Negative Rate
FOV	: Field of View
FP	: False Positive
FPGA	: Field Programmable Gate Array
FPR	: False Positive Rate
FTSG	: Flux Tensor with Split Gaussian
GMM	: Gaussian Mixture Model
GPU	: Graphics Processing Unit
IAGMM	: Improved Adaptive Gaussian Mixture Model
IUTIS-3	: In Unity There Is Strength 3
KLT	: Kanade Lucas Tomasi

LBSP	: Local Binary Similarity Pattern
LMedS	: Least Median of Squares
MBS	: Multimode Background Subtraction
MRF	: Markov Random Field
ORB	: Oriented FAST & Rotated BRIEF
Pr	: Precision
PWC	: Percentage of Wrong Classifications
RANSAC	: Random Sample Consensus
Re	: Recall
RGB	: Red-Green-Blue
SFU	: Special Function Unit
SIFT	: Scale Invariant Feature Transform
SOM	: Self-Organizing Map
Sp	: Specificity
SuBSENSE	: Self-Balanced Sensitivity Segmenter
SURF	: Speeded Up Robust Features
TN	: True Negative
TP	: True Positive
ViBe	: Visual Background Extractor

CHAPTER 1

INTRODUCTION

1.1 Motivation

Detection of the crime before occurring and interfering with it, are important for the security of the society. Therefore, surveillance cameras are used in various places such as metros, streets, borders, coasts etc. as a monitoring tool. However, in order to analyze these videos, officers watch cameras all day and it requires enormous amount of labor. In addition, the possibility of missing important details in a scene with naked eye is very high, so surveillance systems should be able to detect suspicious events in the scene automatically by using complex and efficient algorithms.

Most of the researches in literature, on moving object detection are based on static cameras. However, Pan-Tilt cameras are also an important necessity in order to increase coverage. Unfortunately, the algorithms that are designed for static cameras do not perform well on Pan-Tilt cameras due to the motion. Pan-Tilt camera background subtraction systems must have parts such as motion compensation, image registration, panorama generation etc. in addition to static camera algorithms.

Due to the complexity of algorithms, successful background subtraction for the scenes from Pan-Tilt cameras needs significant computing power. Using Central Processing Unit (CPU) is not a sufficient way to deal with the tasks which requires to work in real-time. By using Graphics Processing Unit (GPU) with efficient parallelization, this problem can be solved.

There are studies in literature about background subtraction of a Pan-Tilt camera. However, they are far from real-time working. We proposed a robust real-time background subtraction on frames from the output of a Pan-Tilt camera by inspiring [76].

1.2 Scope

In this thesis work, a method for moving object detection from Pan-Tilt cameras is presented. The method is based on [76]. We assumed that objects in the scene are sufficiently far away from the camera to be sure that the motion of all them does not depend on the distance to camera. In order to make it work in real-time, the algorithm is implemented on GPU and CPU.

The algorithm is partitioned in such a way that the algorithm parts that require much computational power run on the GPU. Selection of these parts and transitions between GPU and CPU are done wisely, which is explained in detail in Chapter 6.

1.3 Outline

In Chapter 2, previous studies on the background subtraction algorithms for both static and Pan-Tilt cameras are explained. In Chapter 3, the methods used for feature detection and matching are explained. Image registration and blending are examined in Chapter 4. The third step in this study is background subtraction and it is explained in Chapter 5. Chapter 6 provides methods, considerations and difficulties while porting the algorithms to the GPU. In Chapter 7, the overall success rates of the system and test techniques to evaluate the success of the system are explained. Finally in Chapter 8, the overall system is concluded and some future works to improve the system are commented.

CHAPTER 2

LITERATURE REVIEW

Background Subtraction is one of the well-studied areas of computer vision over the years. It is part of many video surveillance applications. Generating and keeping a background model is the essential part of the process. Background subtraction of the video sequences from static camera reached a steady state in different manners, but foreground object detection of the frames from the Pan-Tilt camera is moderately a new issue in literature [15], [24], [30], [48], [60], [76], [82] and [90] Traditional background subtraction methods should be merged with compensation of camera motion and registration operations for this purpose.

2.1 Background Subtraction with Static Camera

Background subtraction algorithms found in the literature for static camera can be divided in two groups: parametric methods and non-parametric methods.

2.1.1 Parametric Methods

Parametric methods start with an assumption. The background does not have any kind of movement; the reason of small movements in the scene is camera noise. Therefore, the intensity value of each pixel can be modeled with a parametric distribution.

Gaussian distribution is one of these distributions. Wren et al. [80] proposed an algorithm that uses Gaussian in a simple way. Mean and variance values are calculated from the last *n* intensity value of pixels. They are updated with each new frame as follows.

$$P(X_t) = \frac{1}{\sigma_t \sqrt{2\pi}} * e^{-\frac{(X_t - \mu)^2}{2\,\sigma^2}} \qquad \textbf{(1)}$$

$$\mu_{t+1} = (1-\alpha)\,\mu_t + \alpha\, X_{t+1} \qquad \textbf{(2)}$$

$$\sigma_{t+1}^2 = (1-\alpha)\,\sigma_t^2 + \alpha\,(X_{t+1} - \mu_{t+1}) * (X_{t+1} - \mu_{t+1})^T \qquad \textbf{(3)}$$

Eq. (1) shows Gaussian probability. Eq. (2) and Eq. (3) are valid where X_{t+1} is the intensity value of related pixel, μ and σ are mean and variance values respectively. Moreover, α is learning rate.

The decision of whether the pixel is in background or foreground is made by using the following Eq. **(4)**. If it is below threshold, the pixel is in background, and vice versa.

$$|\mu_{t+1} - X_{t+1}| < T \qquad \textbf{(4)}$$

Although this model is quite fast and simple, it has difficulties with adapting to quick changes such as fluctuation of illumination. Also, choosing correct α is crucial, because wrongly chosen learning rate, causes unnecessary quick updates.

Kalman Filter based background detection algorithm [59] is another parametric method. It is more robust to illumination change. It is based on thresholding the value, which is result of the difference between background model and current frame intensity. The algorithm calculates new prediction by using the difference between current intensity and old prediction of the background model. This difference is added or subtracted to model by weighing. Therefore, values that are matched with model earn higher weight, but the others do not. After all, algorithm performance is limited because it can represent the outer world by using only one model. Many patterns cannot be modeled.

Real world cannot be modeled by using only one model. In order to go beyond this limitation, Friedman and Russel propose an algorithm in the study which is about traffic surveillance [26]. Algorithm includes three Gaussian models for each pixel. One model that is the darkest one for shadows, another model that has largest variance for cars and another one is for roads. It uses the Expectation Maximization (EM) algorithm for initialization operation and EM is also used for decision mechanism.

Stauffer and Grimson [70] took these three Gaussian models and made it a more generalized mixture of K Gaussians algorithm. Each pixel is modeled with K Gaussians that may be foreground or background. The probability of each pixel belongs to whether the background or foreground can be calculated by using Eq. (5) and Eq. (6), where covariance matrix is in the form (7).

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \rho(X_t, \mu_{i,t}, \Sigma_{i,t}) \qquad \textbf{(5)}$$

$$p(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}\ |\Sigma|^{1/2}} * e^{-\frac{1}{2}(X_t - \mu)\Sigma^{-1}(X_t - \mu)} \qquad \textbf{(6)}$$

$$\Sigma_{i,t} = \sigma_{i,t}^2 I \qquad \textbf{(7)}$$

K is the number of Gaussian; it is defined in the initialization step, according to memory and computational power. It is defined between 3 and 5 in [70]. Equations indicate the values for each pixel X_t. Weight and mean are shown with ω and μ respectively. P is probability density function, Σ is standard deviation and p is Gaussian probability density function.

Evaluation of new pixel value starts with decision of corresponding Gaussians. First b Gaussians weights are added to each other until they reach the T threshold value. These b Gaussians are classified as background distributions and the remaining ones are foreground. Eq. (8) is used for comparison.

$$B = argmin_b \left(\sum_{i=1}^{b} \omega_{i,t} > T \right) \qquad \textbf{(8)}$$

Then, matching Gaussian of new pixel value is obtained by using Eq. (9) where k is a constant.

$$sqrt\left(\left(X_{t+1} - \mu_{i,t} \right)^{T} . \ \Sigma_{i,t}^{-1} . \left(X_{t+1} - \mu_{i,t} \right) \right) < k\sigma_{i,t} \qquad \textbf{(9)}$$

There are two possibilities for new pixel:

- If new pixel is matched with one of the Gaussians, it is classified according to the matched Gaussian model. Weight, mean and variance of matching Gaussian are updated by using the Eq. (10), (11) and (12), respectively. On the other hand, mean and variance of unmatched Gaussians stay the same, only their weights are re-arranged.

$$\omega_{i,t+1} = (1 - \alpha)\, \omega_{i,t} + \alpha, \qquad \text{where} \qquad \alpha \text{ is learning rate} \qquad \textbf{(10)}$$

$$\mu_{t+1} = (1-\rho)\,\mu_{i,t} + \rho\, X_{t+1} \quad \textbf{(11)}$$

$$\sigma^2_{i,t+1} = (1-\rho)\,\sigma^2_{i,t} + \rho\,(X_{t+1} - \mu_{i,t+1})(X_{t+1} - \mu_{i,t+1})^T \quad \textbf{(12)}$$

where

$$\rho = \alpha \,.\, \eta\,(X_{t+1}, \mu_i, \Sigma_i) \quad \textbf{(13)}$$

- If there is no match, new pixel is classified as foreground pixel and Gaussian that has the least weight is replaced with a new one. Variance of the new member is initialized to a high value.

Gaussian Mixture Model (GMM) that is proposed by Stauffer and Grimson [70] is one of the most important milestones for background subtraction algorithms. Because of the fact that it has more than one model, the algorithm could deal with changing or moving backgrounds and gradual illumination changes. Despite of this success, it has some disadvantages unfortunately. Number of Gaussians is not adjustable on the fly. This requires more computing power and memory.

In sixteen years, hundreds of studies are conducted in order to improve GMM. Original paper proposed constant number for Gaussians. In order to improve performance against a dynamic background, [16], [68] and [89] suggest variable number for the number of Gaussians. This approach not only increases the performance, but also decreases the computation time. [38] and [47] change the initialization mechanism. [4] and [39] allow moving foreground items during training part. [79] and [85] modify the learning rate. [43] are not contented with new learning rate and they make it adaptive with time. [72] change the decision mechanism of foreground. While the original paper uses the pixels, [56] use blocks and [10] uses clusters. Instead of intensity value of Red-Green-Blue (RGB); [71], [81] prefer different color spaces or features such as edge [32], [33], texture [75]. Even more than one feature similar to brightness, neighborhood relation can be integrated in [86].

Furthermore, extra feature, which is Markov Random Fields, [65], can be added to the original algorithm. Instead of taking data regularly, hierarchical approaches, [54] or multi-level approaches [18], [33] are preferred. Result of the original paper is modified in the final by some post-processing operations [55], [77].

The concern of the authors is not only to increase the performance, but also to decrease the computation time. Studies were also conducted in this manner. Region of Interest (ROI) concept, [6], [84], is applied in order to reduce burden. Instead of taking all data, sampling strategies [40], [51], [66] are used. Another way of increasing speed is to use the hardware implementation [5], [35] on powerful hardware.

Another aspect is that improving foreground detection with external support. Statistical background disturbance [3] and color segmentation [23] are some examples. Finding motion externally helps the algorithm. Motion can be found with optical flow [88], block matching [31], texture models [41], [58] or consecutive frame difference [83].

Guler et al. designed a real-time multi-camera video analytic system [27] which is composed of four main parts. Background Subtraction is the first part; Camera Sabotage Detection, Abandoned Object Detection and Object Tracking are the following algorithm parts. In Background Subtraction part, GPU version of the IAGMM [89] is used. When performance of both CPU and GPU implementations are measured, there is a 75.00 speedup for the images with resolution of 1024 X 768. There is a high performance increase with GPU usage because IAGMM [89] is very appropriate for parallelization. Efficient parallelization and performance measurements of a basic background subtraction algorithm can be examined in [73].

2.1.2 Non-parametric Methods

Parametric methods have a good success in modeling real life’s complex scenes, but they have difficulties in adapting to changing environments [22]. This requires much more attention for choosing parameters and leads to loose of generalization. In order to deal with this problem, non-parametric methods can be used.

Elgammal et al. proposed an algorithm [22] that estimates probability density function of each pixel. By using the Eq. (14), kernel Estimator "K" function can be used as Gaussian function. After replacing of "K" with Gaussian Equation, it becomes Eq. (15). In this structure x_t represents consecutive intensity values

$$P(x_t) = \frac{1}{N}\sum_{i=1}^{N} K(x_t - x_i) \quad \textbf{(14)}$$

$$P(x_t) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{(2\pi)^{d/2}|\sum|^{1/2}} e^{-\frac{1}{2}*(x_t-x_i)^T \Sigma^{-1}(x_t-x_i)} \quad \textbf{(15)}$$

Background or foreground decision is made by comparing probability density function with a threshold. If it is below threshold, it is classified as foreground, and vice versa. After decision, an update of the model should be completed. Elgammal et al. [22] designs the method with two background models, which are short term and long term. As can be understood from the name, short term background model has a narrow sample rate. Generally, this model consists of foreground objects. On the other hand, the long term background model has a slow update mechanism so it usually hosts background objects. Using two models at the same time, gives a chance to understand that detection is really foreground or dynamic background object like a leaf on the wind.

Support Vector Machine (SVM) is a classification method. Lin et al. [42] proposed a method that uses SVM for background subtraction. Frames that do not have moving objects are used for training. During training period, blocks are extracted from image and some features like optical flow are calculated for each block. While new frames are coming, new blocks are extracted and distances are calculated between old blocks and new blocks. Using a threshold on these distances, a background / foreground decision is done.

Maddalena and Petrosino [46] proposed Neural Network Background Modeling. Background and foreground distribution is expressed by using weights of the neural network. Neural network learns how to classify each pixel as background or foreground. The background model generates a Self-Organizing Map (SOM) and this map decides that a pixel is background or foreground.

The Codebook (CB) algorithm [37] implements a clustering method to construct the background model. During the learning period each pixel is sampled and these values are clustered into a set of Code Words (CW). The background is encoded pixel by pixel. Codebook algorithm is based on keeping CWs for each pixel. Each CW has seven data about this pixel. One data is RGB value and it is held in (V) and six data in Code Word Structure (CWSTR).

Table 1: Elements of Code Word

Elements	Meanings
v_i	$(R_i + G_i + B_i)$
$I_{i_{min}}, I_{i_{max}}$	Minimum and maximum brightness, respectively, which the CW has occurred.
n_i	The number of the CWs has occurred.
λ_i	The maximum negative run-length defined as the longest interval during the training period that the CW has NOT recurred
p_i, q_i	The first and last access times, respectively, that the CW has
$CWSTR_i$	$(I_{i_{min}}, I_{i_{max}}, n_i, \lambda_i, p_i, q_i)$

In the learning period each pixel is sampled with the six tuple format given in the Table 1, and the results are stored in the CB. In the CB, each pixel can have maximum five CWs. If a pixel in the CB has the maximum number of CWs, before creating new CW, the least used CW is deleted from the CB.

After the learning period, the background model is constructed and the rest of the operation is straightforward. Each input pixel value is compared with the CWs of that pixel and if the brightness and color distance are lower than the predetermined threshold, that pixel is marked as foreground and the related variables of the matched CW are updated.

Barnich and Droogenbroeck [8] proposed the algorithm VIsual Background Extractor (ViBe) which is a universal background subtraction algorithm. Algorithm stores the values of pixels over the frames. When a new frame comes, it compares the new value and model for the pixels then, decides that pixel is background or foreground. If there is a match, it updates the model. Many algorithms prefer deleting the oldest element from the model in order to add new one but ViBe [8] chooses an element to remove randomly.

Wang et al. [78] proposed an algorithm Flux Tensor with Split Gaussian models (FTSG) that combines spatio-temporal tensor formulation, foreground/background modeling and multi-cue appearance comparison. The system is formed of three modules that are detection, fusion and classification. In detection module, algorithm finds the foreground objects by using flux tensor based motion detection and Gaussian models at the same time. Then, the results of two different detections are combined. In the final part, edges of present foreground results and edges acquired from the original image are compared. Classification operation is done by using edge matching results.

St-Charles et al. [69] proposed an approach based on the adaptation and integration of Local Binary Similarity Pattern (LBSP). Name of the algorithm is Self-Balanced SENsitivity SEgmenter (SuBSENSE). Firstly, spatio temporal information is extracted from each pixel by using RGB values and LBSP features. Then, a sample consensus approach like ViBe [8] is used for classification of binary features. Both algorithms FTSG [78] and SuBSENSE [69] have satisfactory results. However, because of the computation load of algorithms, making them to run in real-time seams hard.

2.2 Background Subtraction with Pan-Tilt Camera

Background subtraction with Pan-Tilt camera algorithms, which are based on image information only, can be grouped under 2 approaches according to usage of offline generated correspondence layer.

The algorithms of first group generate a layer map offline. Before an operation mode, Pan-Tilt camera scans all areas by panning and tilting and key frames of the scene are generated. During the operation mode algorithm uses these key frames for image registration.

Second group includes algorithms that do not make any preparation operation offline. Algorithms register images by using only the knowledge acquired during runtime.

The algorithms [48] and [82] are members of the first group. They generate key frames at the beginning and take advantage of them during the image registration.

Xue et al. [82] introduced a method for background subtraction of Pan-Tilt-Zoom (PTZ) cameras. In the beginning PTZ camera scans all area for all focal lengths. During scanning, algorithm finds feature points by using Scale Invariant Feature Transform (SIFT) [45] and then by using these feature points panorama is build up. Panoramic frames are used to generate GMM of background. All these operations are done offline. After completion of preliminary part; new frames are registered to panorama by using SIFT and Speeded Up Robust Features (SURF) [9] algorithms. Then, GMM operation is done in the normal fashion. There are some disadvantages of this approach. First of all too much preliminary work have to be done before operation. Taking frames for all areas and for all focal length requires too much time. Also, working on videos taken without these constraints is not possible.

Monari and Pollok [48] proposed a method for background subtraction. This method also uses offline training similar to the method [82]. Camera makes a full scan of the area during initialization mode and generates a key frame map. Frame to frame homography estimation may cause a little error. After some working time, these little errors are getting bigger like an avalanche. Monari and Pollok [48] produced this key frame structure in order to beat this weakness. Key frames are generated in initialization mode. Then, in working mode new frames are put in the panorama with the help of key frames. Later, background subtraction is done. When a new frame comes, a search is started in key frames in order to match to the new frame. There may be some overlap between key frames. These overlapping areas make it harder to find a correct key frame. In order to speed search and make the result more reliable, heuristic neighborhood ranking model is used for searching. Although, key frame structure increases the performance of panorama generating, similar to [82], it requires challenging initialization part. Moreover, the background subtraction part is implemented for only proofing of concept. The attitude of the algorithm against challenging circumstances, such as illumination change, dynamic background is uncertain.

On the other hand, algorithms [15], [24], [30], [60], [76] and [90] prefer the second approach. They register new frames without any prior knowledge.

Zou et al. [90] developed an algorithm that subtracts background for free moving camera. It applies Harris Corner Detector [28] in order to find features for each frame. Then, points that are in consecutive frames are matched by using a simple algorithm. Candidate points are searched in the neighborhood of each point and absolute error is calculated for all pairs. Result with a minimum error is admitted as matching. Later, by using RANdom SAmple Consensus (RANSAC) [25] algorithm, homography is estimated. The panorama is eliminated from foreground objects with a frame skipping topology. Finally, background is subtracted with a technique based on Markov Random Field (MRF) [1]. Assumption of small movement between successive frames makes the approach to be unfeasible for real life applications. Other drawbacks of the algorithm is because of the panorama generating style: The background model cannot adapt itself according to changes in the outer world.

Ferone and Maddelena [24] take the algorithm [46] that examined before and modify it in order to feed it with Pan-Tilt camera output. Decision of background or foreground is based on weights of the neural network. Different to [46], SOM also defuses the motion of the camera.

Ivanov described a background subtraction algorithm [30] for Pan-Tilt cameras mounted on a mobile platform. Instead of generating a panorama, Ivanov made all operations for current frame in its coordinate plane. There are two GMM [70] for background. These background models and motion compensation part are coupled to each other. First model is used for detection and the second one is for elimination of errors. Feature points are extracted by using Features from Accelerated Segment Test (FAST) [61] and descriptors are calculated by using Binary Robust Independent Elementary Features (BRIEF) [14] for each frame. After matching of feature points, homography is calculated by using RANSAC [128]. Background models are shifted according to the found homography matrix. Even though the algorithm can find a foreground object for static camera case by using high learning rates, it fails in foreground subtraction during camera motion because algorithm [30] does not generate panoramic model.

Rodriguez introduced a method [60] for background subtraction. It uses SURF [9] similar to [82] in order to find feature points and calculate descriptors. Some changes are made in the original SURF algorithm. The Brute Force Matching algorithm is preferred for matching the feature points. Using these matches, homography is found with the help of RANSAC [25] algorithm. Background model is found like a static camera case then, it is shifted by using homography matrix. Instead of using panorama, the author prefers to apply transformation to background model like [30]. It looks like more simple way, but accuracy of this method is unfortunately low. Also, it generates blank pixels. In order to fill empty pixels, Rodriguez prefers the interpolation operation. Interpolation takes extra computation time and does not create sufficient background subtraction results.

Chen proposed a method [15] for background subtraction of Pan-Tilt cameras. Points are matched by using Kanade Lucas Tomasi (KLT) tracker [67]. Then RANSAC [25] is applied to determine inliers. Finally, background is subtracted with Graph Cut algorithm which is based on ViBe [8]. Author prefers it due to having fast initialization and strength to noise. Regular points are used in [15] for matching but usage of some feature points such as blobs or corners gives more accurate matching result. It also causes a more reliable foreground mask. Furthermore, Chen [15] used affine transformation instead of projective transformation. Affine transformation is insufficient technique to model the real world. It will be explained in detail in Chapter 4.

Nguyen and Jeon described an application range limited Genetic Algorithm Search [52] for background compensation by using GPU. Genetic Algorithm is a probabilistic search method in continuous space to capture camera motion. Then, projection histograms are used to determine backgrounds. However, computational power requirement of the algorithm is too much. That brings solution to use the GPU. The algorithm finds general camera motion and subtracts results over the histogram. It may find objects instantly, but finding foreground objects over the time, adaptation of the background model and marking dynamic background objects cannot be possible with this system.

Doyle et al. developed a process [19] by using GPU. Optical flow background estimation algorithm, which is especially created for unmanned air vehicles, finds a foreground object and tracks them without generating background model or panorama image. Camera motion is detected by using GPU based optical flow operation. Moving objects are located by using the result of subtraction operation, which is done between the result of the optical flow and background estimation. The Kalman filter is also applied to filter the results. Although this algorithm detects and tracks some object in the foreground, it cannot understand the changes or the movement in the scene totally, because it does not have any background model.

Usage of omni-directional image instead of static camera may be another option to increase the coverage area of camera. In [36], authors state that a parallel algorithm to generate panoramic image by using omni-directional image. They create a significant speedup by parallelization. After panorama generalization, operations such as background subtraction, object tracking can be applied.

Tsinko introduced a method [76], which gets frame from Pan-Tilt camera and subtracts the background from the panorama. The algorithm starts with finding feature points by using SIFT [45], then descriptors are calculated. Matching of these feature points is found in a straight way. An assumption is made that background objects have the same speed as the camera makes pan and tilt; because the distance between them and camera is sufficiently high. Considering this assumption, Hough Transform [7] is applied to matching in order to eliminate points come from foreground object. Then, homography matrix is created and a new frame is added to mosaic with new calculated homography. Finally, background subtraction operation is done by using GMM [70], KDE [22] and Codebook [37] algorithms separately. Most important qualification of this algorithm is that it does not find feature points in mosaic every time because finding feature points every time from panorama has two main drawbacks. First, it requires extra computation time. Second, extracting feature points from processed image is inaccurate. After homography operation, characteristic property of feature points is damaged. A list of points is kept for mosaic. After each matching process, feature points that do not match from new frame are added to the list. Before the addition, their coordinates are adjusted according to homograph. On the other hand, the algorithm has some drawbacks. First seen areas in panorama are wrongly classified as foreground. In order to eliminate these false alarms, creation time of each pixel in panorama can be kept and these results can be used for filtering.

We prefer second type method that does not need any prior knowledge for image registration because it is more convenient. System is ready to detect moving objects without any prior work. Also, it can be tested with any Pan-Tilt camera video because algorithms from first group can generate correct outputs with only compatible videos, which start with scanning of all area.

Motion detection from a Pan-Tilt camera is formed of three main parts as shown in Figure 1. It starts with extraction and matching of feature points. Then, registration and blending of new images to the panorama comes. Finally, background subtraction operation is done.

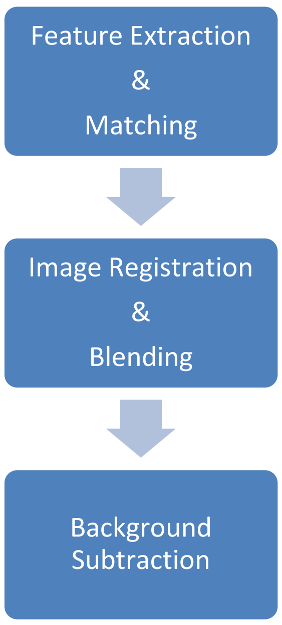
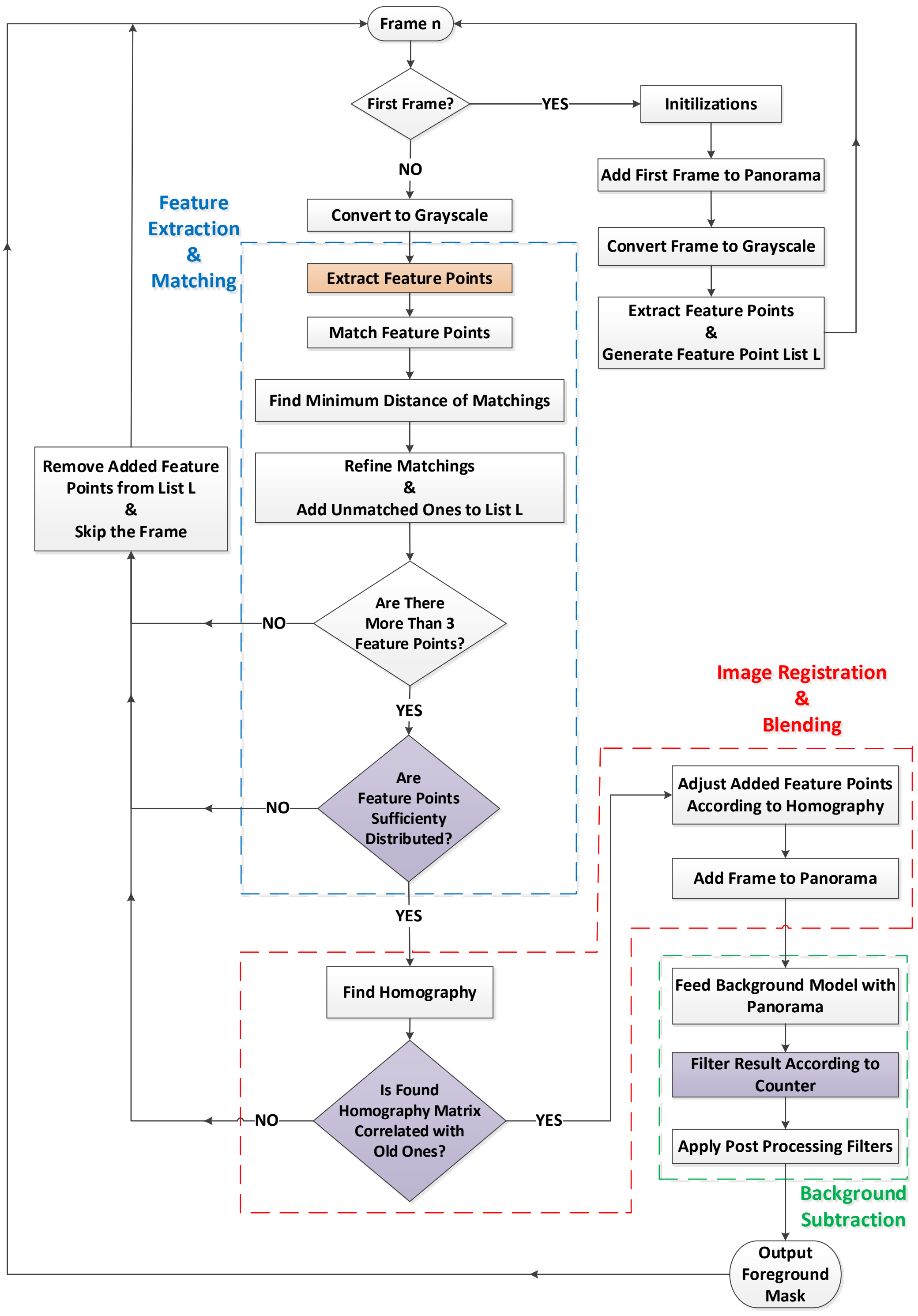


Figure 1: Steps for Pan-Tilt Camera Motion Detection System

In the following chapters, each part will be explained in detail. Before starting to first chapter, feature extraction and matching, algorithm state chart of the whole process can be seen in Figure 2.

In this state chart, three main parts of the algorithm is indicated with different colored dashed lines. Moreover, algorithm part which is different from [76] is shown with orange color. Finally, parts that only our algorithm has are shown with purple color.



Frame n
First Frame?
YES
Initilizations
Add First Frame to Panorama
Convert Frame to Grayscale
Extract Feature Points & Generate Feature Point List L
NO
Convert to Grayscale
Feature Extraction & Matching
Extract Feature Points
Match Feature Points
Find Minimum Distance of Matchings
Refine Matchings & Add Unmatched Ones to List L
Remove Added Feature Points from List L & Skip the Frame
Are There More Than 3 Feature Points?
NO
YES
Are Feature Points Sufficienty Distributed?
NO
YES
Image Registration & Blending
Adjust Added Feature Points According to Homography
Add Frame to Panorama
Find Homography
Is Found Homography Matrix Correlated with Old Ones?
NO
YES
Feed Background Model with Panorama
Filter Result According to Counter
Apply Post Processing Filters
Background Subtraction
Output Foreground Mask

Figure 2: State Chart of the Proposed Algorithm

CHAPTER 3

FEATURE EXTRACTION AND MATCHING

Feature extraction and matching are the first parts of the system. The algorithm starts with the first frame. Feature points of the image are extracted and all of them are put in the feature point list L. Then, the image is located in the middle of the panorama.

Every time when new frame comes, these steps are repeated:

- Feature points are extracted from frame and matching is done with the feature points from list L.
- The smallest value of distance is found.
- Distances of matching are compared with dynamic threshold to refine matching.
- Unmatched feature points from the new frame are added to list L. The coordinates of the feature points will be changed after the homography matrix is calculated.
- Distribution of feature points is checked.
    - If the result is false, added feature points are removed from list L and this frame is skipped.
    - If the result is true, the process continues.

Details of each part will we explained in following sub-sections.

3.1 Feature Extraction Algorithm

Feature extraction algorithms found in the literature can be grouped in three. These are edge based, corner based and blob based. In image registration part, we need at least 4 matching points to calculate homography matrix. Due to this necessity, edge based algorithms are not appropriate.

Harris Corner Detection [28] and Features from Accelerated Segment Test (FAST) [61] are corner based feature extraction algorithms. Harris [28] is based on gradient computation; it looks for significant gradient change in all directions because it is possible only on corners. Similarly, FAST [61] searches changes in all directions, but it uses templates instead of gradients. Algorithm placed a template around the candidate pixel and controls the changes between center pixel and around ones. If there are changes in all directions, this pixel is classified as corner.

Scale Invariant Feature Transform (SIFT) [45], Speeded Up Robust Features (SURF) [9], Binary Robust Independent Elementary Features (BRIEF) [14] and Oriented FAST & Rotated BRIEF (ORB) [63] are the most preferred feature extraction methods based on blob detection in literature.

SIFT [45] is one of the most reliable and robust feature detection methods over the years. Lowe proposed a method that starts with taking Difference of Gaussians (DoG) for different pyramid sizes to find stable feature points. This process gives feature points that are invariant to scale. All pixels in DoG images are compared with eight neighbors. If the value of it is minimum or maximum, there is a candidate point. There are many candidates, but some of them are non-reliable. In order to eliminate weak ones and locate strong ones, interpolation is done with the Taylor expansion of the DoG function as can be seen in Eq. (16). Then keypoints which have low contrast are eliminated according to second order of Taylor expansion in Eq. (17).

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \quad \textbf{(16)}$$

$$D(\hat{x}) = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x} \quad \textbf{(17)}$$

Bins are created around the each candidate point. According to the result of histogram, rotation invariant descriptors are calculated.

Bay et al. proposed a method [9] that can find feature points with Hessian Matrix. It is inspired from SIFT [45] but it is faster than SIFT [45]. Hessian Matrix is given in Eq. (18), where L is the convolution of the Gaussian second order derivative.

$$H = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \quad \textbf{(18)}$$

The authors stated that instead of calculating Gaussians, box filters which are created by approximating Gaussians can be used. This approximation increases the speed of the operation. In Figure 3, original and approximate filters can be seen. Furthermore, these approximate filters are suitable for integral image that removes the dependency on image size.

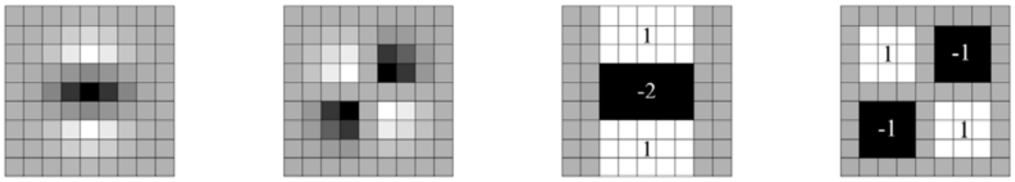


Figure 3: Original and Approximate Filters of SURF [9]

SURF [9] creates rotation invariant descriptors. Haar Wavelet responses in both directions are calculated and weighted for each direction that is defined before. Then, rotation invariant descriptors are calculated for each interest point.

Also up-righted version of the algorithm is available and it is faster than normal version, but it is not invariant to rotate. SURF is faster than the SIFT due to the usage of integral image that makes also operation speed invariant to image size. Moreover, it is more appropriate to parallel operation because of Hessian image independency. Unfortunately, some accuracy losses are possible, but the authors claim that it is minimal and negligible.

BRIEF [14] is a feature descriptor calculation method that is very fast. Traditional methods use a lot of memory for each feature vector. However, memory, which is used by BRIEF [14], is nearly 4 % of the memory usage of SIFT [45] Instead of using float numbers, it uses a binary descriptor. Low memory usage increases the speed of the algorithm. The descriptors are created according to similarity of interest point with neighbors. It takes a patch around the feature point and compares the value of center pixel and the others. If the first value is smaller than the second one, it writes 1 on the corresponding area in the descriptor, else it writes 0. Any extra operation is not done against rotation. Therefore, the algorithm is not rotation invariant.

ORB [63] is an algorithm that is based on BRIEF [14] but it is rotation invariant. It starts with finding feature points by using FAST [61]. Then, BRIEF [14] descriptors are calculated. Both of the algorithms are not invariant to rotation. In order to beat this weakness, ORB [63] weighted FAST [61] templates about rotation in order to detect dominant orientation. Then rotation information is added to descriptors. The algorithm does not require too much computation and it is very fast compared to SIFT [45] or SURF [9]. Also, the authors claimed that it is successful as much as them.

In this study, feature points are required for matching to generate a panorama image. Therefore, robustness is the first concern. If there is a matching error, panorama also has this error. Then, background subtraction algorithm, classifies false matching part wrongly. These false positive or false negative markings decrease the performance unfortunately.

Floating point feature extractors like SIFT [45] and SURF [9] generate more robust results. In order to speed up the process, methods with binary descriptors like FAST [61], BRIEF [14], and ORB [63] ignore some performance loss. Therefore, their result is less robust than floating point feature extractor algorithms' result.

El-gayar et al. made a comparison [21] between feature detection algorithms. According to the study [21], SIFT [45] and SURF [9] give better results than the others. However, time consumption of the algorithms should be considered. Algorithms with binary descriptor are faster 6 or 7 times with performance loss. If there is not enough computation power these algorithms should be used, but if there is sufficient computational power like the GPU, Field Programmable Gate Array (FPGA); using robust and more reliable floating point feature extraction algorithms are preferable.

In our study, implementations are done for GPU so computational power is not a bottleneck. In order to get desired results, always more robust solutions are used. Therefore, using SIFT [45] or SURF [9] is reasonable.

Tsinko [76] preferred using SIFT [45] for feature extraction. Author stated the reason behind this selection of the algorithm that durability against scale and rotation distinctively. Moreover, SIFT algorithm has low sensitivity to affine changes less than 30° [45].

Xue et al. [82] used SIFT [45] and SURF [9] at the same time. Authors prefer SIFT [45] during the initialization part in order to get more robust features. Then, SURF [9] is used on-line registration part because of speed and robustness.

Rodriguez [60] stated some comparison results between SIFT [45] and SURF [9]. Both algorithms are examined against change in rotation, scaling, illumination and Field of View (FOV). Although SIFT [45] generate more feature points with a little bit better matching results, SURF [9] algorithm is much faster and the result of it is robust enough. Therefore, Rodriguez chose SURF [9] for feature extraction.

Principle of the SIFT [45] and SURF [9] algorithms are examined in detail. They both follow the same pattern basically but SURF takes advantage by using an integral image and Gaussian approximation. Panchal et al. proposed a comparative study [53] in order to compare SIFT [45] and SURF [9]. According to its study, they give similar results. Performance of SURF [9] is slightly below. However, SURF [9] is faster than SIFT [45]. Most important feature of SURF [9] for our case is that it is more appropriate to parallelize. Terriberry et al. [74] defined a translation for SURF [9] algorithm to the GPU. SURF implementation [74] on this study runs real-time and 4 times faster than SIFT implementation [74] on the GPU. Moreover, 82% of the matches that are created by SURF [9] are classified as inliers by RANSAC algorithm [25]. SURF [9] is chosen in the light of these results.

3.2 Feature Matching Algorithm

The feature matching process is the next step after feature extraction. Feature matching can be done in 2 ways. First method is trying all possible matches. The second and faster method is making this search operation by following some rules.

Brute-Force Matching uses the method that tries all possibilities. It is a very robust method, but it consumes a high amount of time if the computation capability is limited.

On the other hand, there is a Fast Library for Approximate Nearest Neighbors (FLANN) to speed up the operation. Tree Search is most common search technique. Muja and Lowe generated FLANN [49], [50], [92] that can fast approximate nearest neighbor search in high dimensional spaces. It chooses the best algorithm with optimum parameters for data type and size. Most appropriate algorithm is chosen from the algorithms that are K-Dimensional Tree, Randomized K-Dimensional Tree and Hierarchical K-Means Tree. Then, best parameters are assigned by a method that finds parameters with the grid search roughly and tune them with Nelder-Mead Downhill Simplex method [34]. Unfortunately, it does not guarantee to get global minimum, but experiments showed that it is very close to optimum. FLANN [92] gives faster results than the direct linear search, with a little performance loss.

Both of the methods are used in studies. Their outputs are nearly the same. The difference between them is the speed. FLANN [92] may be more appropriate for the system that has the low computation capability. Zhang et al. [87] applied a Random Tree algorithm for matching and Liu et al. [44] also employed KD-Tree algorithm from FLANN [92] On the other hand, Rodriguez [60] preferred Brute-Force Matcher because of its certainty.

In our study, robustness is the first concern as stated before. Because of the usage of GPU, computational power is sufficient. Moreover, Brute-Force matching is more appropriate for parallelization. Therefore, Brute-Force Matcher is employed for feature matching process.

3.3 Panorama Feature Extraction Method

In order to create homography matrix, matching feature points are required between each new frame and panorama. Finding feature points of the panorama is possible in two ways. First one is finding feature points from background panorama repeatedly for each cycle. Secondly, keeping a list of feature points in the panorama and updating this list with each frame. Both methods have advantages and disadvantages.

The first method is extracting feature points from background panorama image after every update of background mosaic. Most important advantage of this approach is that, a number of feature points does not increase too much. Total count increases if only new scenes are started to be visible. Moreover, after some working time, it stays steady. Algorithm with this property works for long time period in limited memory. Another advantage of this method is that it prevents developing of outliers. Algorithm subtracts feature points from the background panorama image that contains only background subjects. Therefore, features must be from real background and they will match with the background object in the new frame.

The method has two shortcomings. Firstly, background panorama is generated from warped, stretched and skewed images. Some parts of the background mosaic are comprised of scenes with viewpoint angle that is larger than 30°. After this angle, SIFT [45] feature points cannot match implicitly. Lowe stated that in the algorithm [45] bin size is 30°. According to the author, this wide range makes possible matching of feature points even geometric distortion occurs during the 3D viewpoint change up to 30°. Therefore, matching results with the angle of viewpoint change is greater than the limit, are not robust. Another shortage of the method, background panorama has to be generated for each cycle except normal panorama. It brings extra memory requirement and extra operations. Furthermore, method couples the feature detection and background subtraction parts. It obstructs evaluation of algorithm parts separately.

The second method is maintaining a list of feature points. All feature points of panorama are kept in a list. When a new frame comes, feature points of it, are extracted and added to the list. During this addition, the coordinates of the feature points are changed according to homography.

There are different styles for addition:

- First, all feature points of new frame can be added to the list. It makes the size of list enormous. Many unnecessary points, which are from background and foreground objects, live in the list and these points decrease the performance by creating outliers. Therefore, it is not useful.
- Another option is adding feature points that are matched with panorama. This style is also not practical. All feature points come from scene of the first frame. Feature points of the unseen parts of the panorama cannot be added.
- Last style is adding feature points that did not match with panorama. With this style new scenes can be represented in a feature point list and size of the list is acceptable.

Keeping a list of feature points and adding unmatched points, is the alternative method to acquire panorama feature points. It solves both problems of feature point extraction from panorama for every cycle. Even angle change is more than 30°; the algorithm gives a good matching performance. Also, there is no need to sustain extra background panorama.

However, the second method has also disadvantages. Firstly, unmatched feature points are added to list without background/foreground knowledge. They may increase outlier ratio a little. Secondly, addition never stops and causes enlargement in the list.

Tsinko [76] conducted an experiment to compare both methods. According to his result, panorama image of the recalculation method contains visible artifacts. The reason behind this artifact is excessive change in angle of viewpoint as stated before. Author also examined a number of the feature points [76]. The number stays the same for recalculation method, but increases with negative acceleration for list method.

Even though, a number of feature points for list method increases, speed of increase is getting smaller with decreasing unseen part. Moreover, list method has better matching performance. In order to get robust background subtraction, error free registration is necessary and it is possible with successful matching. Therefore, feature points list with addition of the unmatched points is chosen for this study.

3.4 Refining of Matching Results

Brute-Force Matcher is an exhaustive search. In other words, it passes over all possibilities. The algorithm takes one element of the first set and finds the closest element from the second set. Also, it generates a distance value for each pair to show the success of the matching. The algorithm does not compare distances with any threshold value. Every element of the first set is matched with the most similar element from the second set even their similarity is too weak. Furthermore, two different elements of first set can be paired with the same element of the second set. Barely, one of that pairs or maybe both are wrong matches.

In order to eliminate wrong matches and increase the performance, refining operation is necessary. Refining operation is based on comparison of matching distances. If the distance of the matching is less than threshold, this pair should be eliminated. However, choosing an appropriate threshold is a challenging task. Defining a constant threshold value for every frame is not practical. A little change such as contrast, focus between consecutive images impress all distances in the same way or changes in external factors like illumination, increases all distances. Therefore, choosing a threshold based on characteristic of each frame is more reasonable.

$$Distance < N\ x\ Minimum\ Distance\ of\ All\ Matches \qquad \textbf{(19)}$$

In our study, matches are filtered by using the Eq. (19). If the distance is higher than the threshold, pair is removed. Minimum distance of all matches is found and comparison is done with N times of it. Using this value as a reference adjusts the threshold automatically. If most of the distance is small, threshold is also small, and vice versa.

Choosing correct N multiplying factor is another issue. Choosing N is relevant that how many percent of matches are meaningful. Rodriguez [60] conducted some experiments on choosing N value. After trials between 2 to 7, authors stated that small N values are more preferable and 2 is the most appropriate value for N because of following results:

- Multiplying factor does not affect performance ratio of matching significantly, it affects final panorama registration [60].
- Even for no filter case, registration is done properly except some artifacts because homography is generated by using RANSAC [25] that is also a good filter for outliers [60].
- If N number increases, also matching count increases. However, a small number of well distributed feature points are better than a high number of feature points [60].
- Although filtering does not affect too much matching performance, it influences the time requirement of RANSAC [25]. Small N values eliminate most of the outliers so RANSAC [25] needs less time. Using 4 for N instead of no filter case, speeded up the RANSAC [25] more than 3 times in Rodriguez test case [60].

3.5 Effect of Feature Points Distribution

After eliminating wrong matches, reliable feature points of the new frame are obtained. Before calculating homography, last control should be done. The distribution of the feature points should be controlled.

The distribution of the feature points over image has an influence on calculated homography matrix. If feature points are equally distributed over the image, they can give more accurate a homography matrix, but if they are not, the calculated homography matrix has errors.

The main reason behind this problem is based on small misalignment of feature points. When matched points are too close to each other, the small error of their position leads to the significant error of homography matrix. On the other hand, if points are far away from each other; small errors are negligible and it is still possible to get accurate homography matrix.

In order to be sure that the distribution of the feature points over the image is sufficient, there is a control mechanism in the system. For each frame, the variance of the feature points’ locations is calculated and this variance is compared with a threshold. If the value is lower than the threshold, this frame is skipped and homography matrix is not calculated for it. On the other hand, if the variance of points is higher than the threshold, process continues.

CHAPTER 4

IMAGE REGISTRATION AND BLENDING

Feature points of the images are extracted, points are matched and refined. After this point, a transformation matrix that represents one frame by using another frame can be generated. Homography matrix between the new image and panorama can be estimated by using isometry transformation, similarity transformation, affine transformation or projective transformation [29]. The main difference of them is the number of the Degrees of Freedom (DoF) that takes in the consideration. Homography matrix, which corresponds to the transformation type, is calculated by using the Direct Linear Transformation (DLT) [2] algorithm.

State chart of the algorithm for only this part can be seen in Figure 4. This part starts with the calculation of homography by using the matching of feature points.

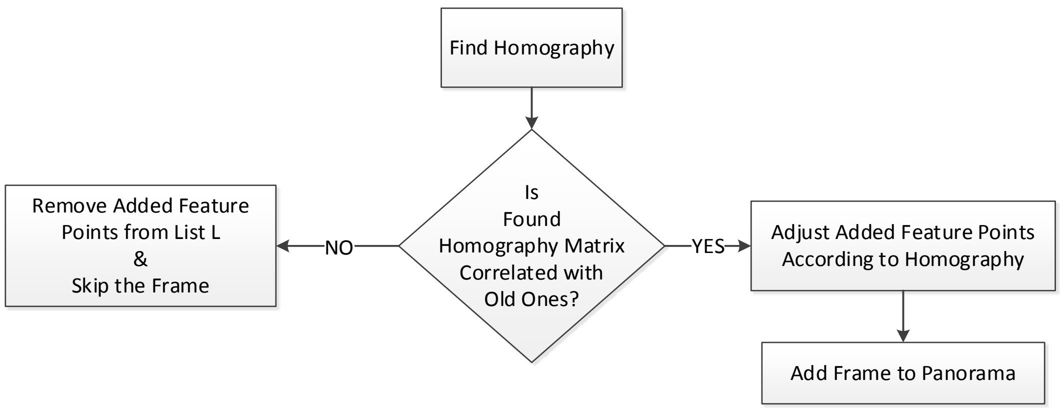

Find Homography
Is Found Homography Matrix Correlated with Old Ones?
NO
YES
Remove Added Feature Points from List L & Skip the Frame
Adjust Added Feature Points According to Homography
Add Frame to Panorama

Figure 4: Algorithm State Chart of Image Registration and Blending Part

As stated before, feature points of panorama are kept in a list L. When there is a mismatch between the feature point of new frame and any feature point in the list, this feature point is added to list but its coordinates is changed according to homography. Additions have already done to the list L. However, their coordinates cannot be changed because there is no homography matrix. At this point, homography matrix is calculated.

The calculated homography matrix is compared with the mean of last five homography matrices. Change in each eight element of the matrix is examined and they are compared with own threshold.

- If even one of changes is more than own threshold, added feature points are removed from list L and this frame is skipped.
- If it is not, the process continues. All newly added feature points are adjusted to be compatible with the coordinate system of the panorama by using homography matrix H.

The pixels from the new image are transformed to panorama. The operation is not done directly. Pixels are placed with blending.

Moreover, a counter for each pixel is increased if this pixel is updated. This counter image is used to filter background subtraction results. Details of it will be explained in Chapter 5.

Panorama image is obtained after these steps and now it is ready for background subtraction.

4.1 Transformation Type

There are many different types of transformations with different features. Even though all transformations can use 3x3 transformation matrices with nine elements, only projective transformation has eight DoF. Each element represents one of freedom and last element is always equal to 1. On the other hand, other transformations make some elements to equal to 0 for simplification. It reduces the number of DoF.

Isometry transformation is Euclidean transformation. In other words, it is a form of translation and rotation. Isometry transformation has got three DoF. These are rotation, translation in vertical direction and translation in horizontal direction. Transformation is invariant to length and angle. Matrix representation of transformation can be seen in the Eq. (20).

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon\cos\theta & -\sin\theta & t_x \\ \epsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{20}$$

Similarity transformation is combining of isometry transformation and scaling. “s” in Eq. (21) represents the scaling factor. Similarity transformation has got four DoF. Three of them is coming from isometry and scaling is added. This extra DoF makes the transformation invariant to scale. Matrix representation of transformation can be seen in the Eq. (21).

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\ \sin\theta & t_x \\ s\sin\theta & \mathrm{s}\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{21}$$

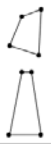
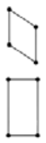
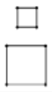
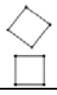
Affine Transformation has two degrees of rotation and two degrees of scaling in contradistinction to similarity transformation with one of each as a degree of freedom. Transformation can preserve the ratio of the scaling. Unfortunately, it lost the preservation of distance ratio and angle between lines but new preservations are gained. Parallel lines will be still parallel with correct length ratio and area ratio, after transformation. Affine transformation has got six DoF. Each one is represented by one element in a matrix than can be seen in the Eq. (22).

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{22}$$

Projective Transformation is a non-singular linear transformation of homogeneous coordinates. Transformation adds two more DoF. Its invariance is different from an affine transform. The cross ratio of lengths of a line is invariant instead of a ratio of lengths. Four point matches are required to compute projective transformation. Matrix representation of transformation can be seen in the Eq. (23). Each element represents a DoF. All transformation types can be seen in Table 2 which is retrieved from [29].

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} Focal\ (Size)\ X & Skew\ X & Moving\ X \\ Skew\ Y & Focal\ (Size)\ Y & Moving\ Y \\ Stretch\ X & Stretch\ Y & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{23}$$

Table 2: Properties of Transformations [29]

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, **I**, **J** (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Isometry and similarity transformations are too inadequate to model the real world. Isometry has three and similarity has four DoF. They do not have invariance in parallelism and ratio of the areas. Therefore, using affine or projective transform is more rational.

If affine and projective transformations are reviewed, it is clear that a projective transformation has two more DoF. Transformation conserves ratio of ratios for lengths instead of ratios. Moreover, there is one more big difference between affine and projective transformations.

Hartley et al. defined key difference between these two transformations in the book "Multiple View Geometry in Computer Vision" [29]. Affine transformation can be seen in Eq. (24) and projective in Eq. (25). Ideal point is modeled to ideal point again for Eq. (24). In other words, ideal point is at infinity. However, Eq. (25) showed that model of ideal point can be located to a finite point. Therefore, projective transformations can model vanishing point that is the intersection point of the parallel lines.

$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ 0 \end{pmatrix} \quad \textbf{(24)}$$

$$\begin{bmatrix} A & t \\ V^T & U \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ U_1 x_1 + U_2 x_2 \end{pmatrix} \quad \textbf{(25)}$$

Projective transform has the ability to model ideal points. Also, it has two more DoF. The invariance concept of projective transformation is better than affine transform to model the real world. In the light of these results, the projective transformation model is chosen for this study.

Homography with eight DoF between images can be estimated by using four pairs [29]. DLT [2] is a simple linear algorithm to calculate homography matrix. There is a pair X_i and X_i'. In order to find an H homography matrix Eq. (26) can be written. Eq. (27) is also full version of it.

$$X_i' = H.X_i \quad \textbf{(26)}$$

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \end{pmatrix} = H \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \qquad \text{where} \qquad H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad \textbf{(27)}$$

If there are four points, homography matrix can be generated by using a DLT algorithm [2] but in most cases there are more than four pairs. Also, some of these pairs are mismatched that are outliers. There are algorithms such as RANSAC [25] in literature to eliminate outliers and get inliers.

4.2 Elimination of Outliers

Four pairs are enough to construct a homography between two scenes, but for the most cases there are too many pairs. Unfortunately, some of them are mismatches. This circumstance causes a necessity to eliminate false matches. RANSAC [25], Least Median of Squares (LMeds) [62] are the most preferred algorithms for elimination of outliers and calculation of homography.

RANdom SAmple Consensus (RANSAC) [25] prefers using a small set of initial data instead of using much of the data possible. The algorithm starts with picking up four pairs randomly and calculating homography H matrix by using them. Then, each other pair is classified as inlier or outlier by using the H matrix. This operation is done may times. Finally, iteration that has the largest number of inliers is selected. Then, homography matrix H is recalculated according to all inliers from this iteration. There are two important issues about this algorithm. First one is the threshold to decide if it is outlier or inlier. Second thing is the iteration count of the algorithm. If it is small, the algorithm may not find all inliers. If it is high, computation time takes much. Deciding on iteration count is based on probabilistic calculation.

LMedS [62] is another method. Algorithm deals with outliers as a minimization problem. It finds distances for all matches. Least median of the distances for all data is selected. Without ant settings or pre-work elimination is done. Then, transformation can be found by using DLT.

RANSAC [25] is the most popular solution for the homography problem. It is the milestone on robust estimation. LMedS [62] is the second popular algorithm. RANSAC [25] dominates the homography estimation area and also LMedS [62] can be seen in some studies. The most important advantage of the RANSAC [25] over the

LMedS [62] is the performance for the case that has outliers more than half [20] because outliers more than 50% affects median in a bad way. Hartley et al. stated that "The RANSAC algorithm is able to cope with a large proportion of outliers." [29]. RANSAC [25] finds inliers that are even a minority. Because of its robustness, [15], [30], [60] [76], [82] and [90] all prefer RANSAC [25]. Therefore, we also prefer RANSAC [25] algorithm for homography estimation in this study.

4.3 Homography Matrix Control

Homography matrices of each frame should be compatible with each other. Elements of the matrix are related to corresponding elements of the other matrices. The reason behind this relation is the nature of the Pan-Tilt camera because jumping from one scene to another scene is not possible. The camera should be rotated to a new position via other scenes.

In order to detect weak homography matrix, a comparison between elements of the matrix and reference matrix is held. Each element of the homography matrix has different meanings as stated in Eq. (23) according to the projective transform. These are moving, size, skew and stretch. The value of each one is in different intervals.

First of all, a reference homography matrix should be generated by using recent homography matrices. The average value of each element of the matrix is calculated. Reference matrix is necessary for comparison operations.

Change in homography matrix from one frame to another frame is so predictable because changes between corresponding elements are in apparent interval. For example, change in "size" value is in the neighborhood 10^{-1}. It maybe be $2x10^{-1}$ or $7x10^{-1}$, but not around 10^{-2}. Similarly, this range is 10^{-2} for "skew" and 10^{-3} for "stretch". New homography matrix and reference homography matrix are compared and even if one of the results are higher than the corresponding ranges, frame is skipped and the process goes on with new frame. The comparison operation starts from base values that are 10^{-1}, 10^{-2} and 10^{-3} for size, skew and stretch respectively. Then, they are increased with each skipped frame until they reach the upper limit.

This property is a little different for move element of the homography matrix. Comparing with only value is not sufficient because it varies in high range. When the camera rotates slowly it can be 3 pixels. When the camera rotates faster, 30 pixels are possible. Therefore, its threshold is not only a constant value. It is a multiplication of average reference value and a constant.

4.4 Blending Type

After homography calculation, the position of the new scene is defined. Putting of new pixels over the interested area in the panorama is another step. Without any extra process, changing old pixels with new pixels is possible but better results can be made with little effort or complex algorithms. Also, blending covers up registration errors moderately.

Blending techniques can be simple or complex as stated before. Alpha Blending [57] is the most appropriate one to get good result in brief time. Also more methods are created with the help of signal domain. Pyramid Laplacian Blending [13] and 2-band Blending [12] are the most famous and successful ones. However, they require much time.

In order to use signal domain based blending, an extra transformation label should be created. Each pixel from new image is taken and it has to be placed in the blank transformation image. Then, blending operation can be done between panorama and transformation image. As can be seen clearly, methods require extra an image, also extra time and memory. Another disadvantage of these method for our case, parallelization of them is not as easy as Alpha Blending [57]. Extra libraries and operations are required.

On the other hand, Alpha Blending [57] is a simple method that does not require any extra label. Pixels from new image can be put in directly into a new position on the panorama after some addition and division operations. It is naturally appropriate for parallelization. While all pixels are being passed in panorama, blending can also be applied. Because of the reasons above, Alpha Blending [57] is used in this study.

All channels of a pixel can be calculated by using the Eq. (28) from [57]. It is applied to channels red, green and blue, respectively. The value of the α is set to 0.5. The reason behind the blending is mostly visualization. Therefore, 0.5 is a good constant because half comes from history and the other half comes from the new image.

$$C_{Final\ Pixel} = \alpha \,.\, C_{Panorama\ Pixel} + (1 - \alpha).\, C_{New\ Image\ Pixel} \qquad \textbf{(28)}$$

CHAPTER 5

BACKGROUND SUBTRACTION

Background subtraction is the last part of the main process. After new frame addition to panorama, panorama is ready for background subtraction. Background subtraction is done on an area that is much larger than the image size. The panorama and the background model of panorama cover the whole area in the FOV of the camera.

In our study, background model is generated by using panorama. In other words, background is subtracted in the normal fashion from mosaic. Camera motion is compensated and new image is registered to panorama. The situation can be examined like a static camera case similar to [15], [48], [60], [76] and [82].

The process will be explained in detail for background subtraction part. Moreover, state chart of the algorithm for only this part can be seen in Figure 5.

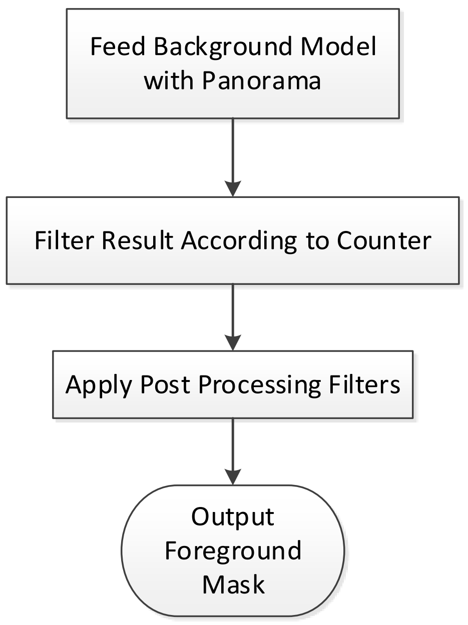

Feed Background Model with Panorama
Filter Result According to Counter
Apply Post Processing Filters
Output Foreground Mask

Figure 5: State Chart of Background Subtraction Part

The new frame is added to the panorama and it is ready for background subtraction. The background subtraction algorithm is fed with panorama image. It starts with the first image and continues until the last image.

The background subtraction algorithm generates black and white foreground mask image. In order to remove false positive labels because of first seen parts, each pixel in the foreground mask image is modified according to the comparison result between corresponding pixels in counter image and threshold. If it is lower than the threshold, value of the foreground mask pixel is set to zero, else no modification is done.

After the filter operation morphological part should be completed. In order to get rid of pepper and salt noises consecutive one closing and one opening operation are applied.

5.1 Background Subtraction Algorithm

There are different kinds of algorithms to subtract background. Basically, it can be grouped in two categories: Parametric methods and non-parametric methods. Parametric ones try to fit the background model for parametric distribution. On the other hand, non-parametric methods follow counts and statistics about pixels.

In order to model background, usage of GMM is the most preferred method in parametric ones and also in all methods. Wren et al. [80] suggested using Gaussians for background modeling in 1997. Friedman and Russel [26] improved this idea one point more and they use three Gaussians for modeling. Then, Stauffer and Grimson [70] generalized the method. Algorithm transformed to the K Gaussians algorithm. K is the number of Gaussians which is defined in initialization step.

Study of Stauffer and Grimson [70] is the very big step for background subtraction because the algorithm is successful to handle dynamic backgrounds, gradual illumination changes and many things. However, the constant Gaussian count is a drawback. It can waste computational power and time by using many Gaussian models for very basic patterns than can be modeled with only one Gaussian. Many variants have been designed for sixteen years. Algorithms have taken [70] as a base and improved it in many ways.

Zivkovic proposed another GMM algorithm [89]. It is one of the successful ones. Algorithm is named with Improved Adaptive Gaussian Mixture Model (IAGMM). The algorithm uses a variable number of Gaussians. The number is defined and changed automatically, similar to [16], [68]. A distinguishable feature that [89] has but [16], [68] do not have is about OpenCV [98]. OpenCV has both CPU and GPU implementations of [89].

On the other hand, there are non-parametric methods. SuBSENSE [69] is an algorithm based on LBSP. The performance of the algorithm is high, but because of much work load, it works slowly. Moreover, Flux-Tensor method [78] is also good subtracter but its speed should be attended. Codebook [37] background subtraction algorithm is a clustering method. It samples the pixels. Then, data from pixels are clustered into code words. Every code word contains seven data. Data usage of each pixel is very high. Unfortunately, this data load affects negatively the parallelism of the algorithm. It decreases Shared Memory usage that will be explained in Chapter 6 in detail.

Parametric methods, especially GMM are appropriate for parallelization because in parametric methods each pixel is taken and it is fitted to a distribution. In other words, the same operation is done for all pixels. Also, the data size of each pixel is in acceptable size so parallelization optimizations such as Shared Memory usage can be applied.

In order to get robust background subtraction results with highly efficient parallelization, a variant of GMM algorithm is used similar to [30], [48], [60], [76] and [82] in this study. [89] is preferred from many GMM algorithms because of its useful contribution to original paper [70]. Furthermore, the presence of the algorithm in OpenCV [98] affects our decision.

5.2 Filtering According to First Seen Part

The background subtraction operation is applied to panorama after each new frame is added to the panorama. When the camera is turned to an area that has not been seen before, pixels which correspond to that area in the panorama are changed from black to a new value. Because of this dramatic change, background subtraction algorithm classifies this kind of areas as foreground object, even they are part of the background. In order to cope with this problem a filtering can be applied to result of background subtraction.

A counter is kept for each pixel and this counter image is updated, when new frame is added to the panorama. Size of counter image is the same with a panorama in width and height. Counter image is initialized to 0 at the beginning. After first frame is located in the middle of panorama, pixels in the middle of the counter image is set to 1. Then, value of counters is updated with each new frame added to the panorama.

Consecutive frames are taken while the camera is being rotated to the left. As you can see in the Figure 6, left side of the output marked with a red label as foreground. For this case, any filtering operation is not applied. Also filter applied result can be seen in Figure 7.

After background subtraction completes its work, filtering can be applied to results of the background subtraction algorithm. The value of the counter is compared with threshold for each pixel of foreground objects. If the value is lower than the threshold, that pixel is removed from the foreground by setting its value to 0. Else, nothing is done about it.

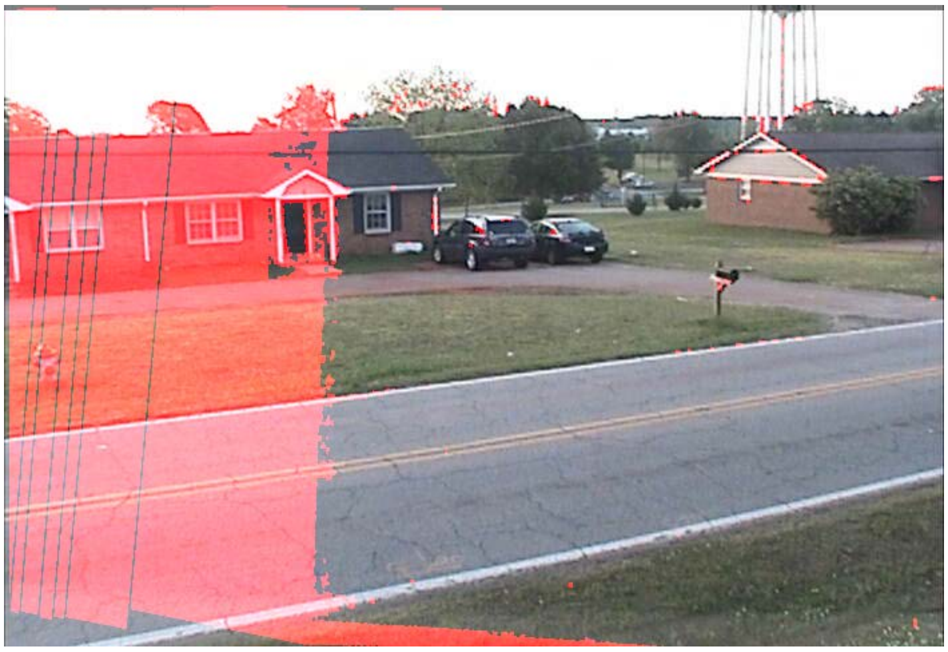


Figure 6: Foreground Mask without Counter Filtering

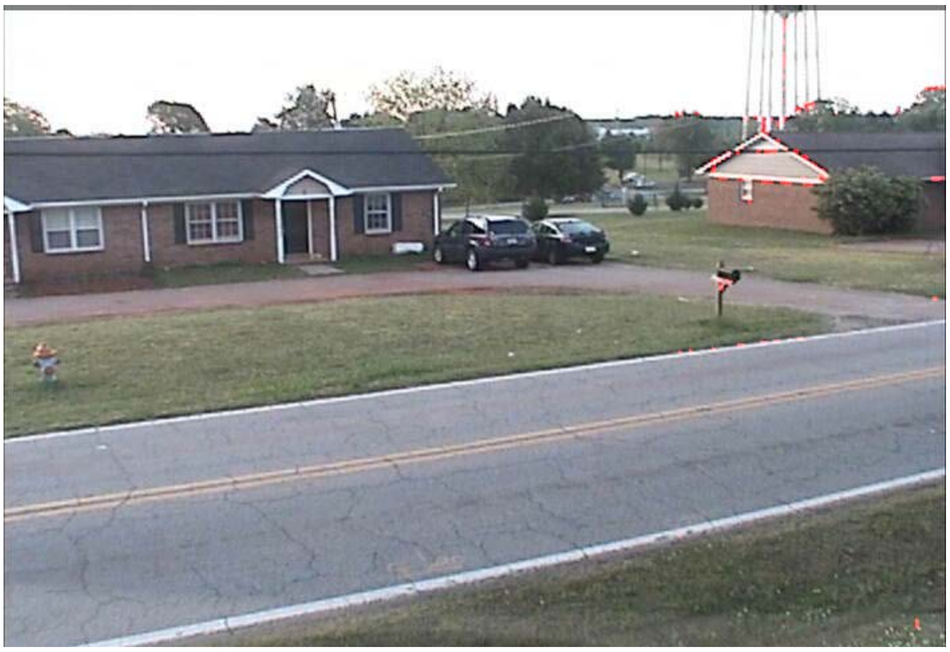


Figure 7: Foreground Mask with Counter Filtering

The value of the threshold is connected with learning rate of the background subtraction algorithm. Small learning rates require higher thresholds, and vice versa. There is an inverse proportion between them.

5.3 Post Processing Operations

The result of the background subtraction algorithm is not perfect. Some false positive and false negative classifications are possible because of many reasons. Camera noise, registration errors are extrinsic reasons. Performance defects of the background subtraction algorithm are intrinsic reasons. Performance defects may occur because of challenging circumstances, such as dynamic background, illumination changes.

The image may contain pepper and salt type noises. These noises may be bigger than one pixel. There might be some unrelated objects in the background. Another possibility is foreground objects have black impurities inside. In order to remove them and increase the performance of the algorithm, morphological operations should be applied.

First of all closing operation should be applied to foreground mask image in order to remove noise. 3x3 square kernel is enough for low resolution images and 5x5 is appropriate for images with higher resolution. Closing is a dilation followed by an erosion with the same kernel. The boundary of the image is enlarged with dilation. Holes, which are named as pepper noise, smaller than kernel are filled. After erosion, boundaries take their old forms but holes still stay as filled.

Secondly, opening is applied to the image which is the result of the closing. 3x3 square kernel is used. Again 5x5 is better for high resolution images. Opening is an erosion followed by a dilation with the same kernel. Opening removes foreground pixels smaller than the kernel. Erosion clears undesirable foreground objects, which are named as salt noise then dilation repairs the other parts of the image.

CHAPTER 6

PORTING ALGORITHMS TO GPU

6.1 Introduction

Real-time background subtraction of images from Pan-Tilt cameras is a challenging task. The algorithm has to deal with camera motion, image registration, panorama generation, background subtraction and final supplementary operations. In order to achieve this task, significant computation power is necessary. Supplying this computational power is possible by using not only the CPU but also the GPU.

Background subtraction of the scenes from Pan-Tilt camera history may not be long, but background subtraction on static cameras is one of the well-studied areas of computer vision. Moreover, many algorithms have been parallelized on GPU and shown to have promising performance results. Feature extraction, feature matching, and image registration are also have been accelerated using GPUs.

CPU and GPU have different architectures; while CPUs are good at executing serial code and branching, GPUs are good at highly parallel and throughput oriented parts. On the other hand, some problems are suitable for parallelization and the others are not. Therefore, in order to get efficient, heterogeneous solution, both CPU and GPU should be used. In such a heterogeneous solution, parts of the algorithm are run on CPU and GPU.

6.2 GPU Architecture

GPUs were initially designed for graphics rendering. It has started with 2D followed by 3D graphics rendering. Because of their multi-thread multiprocessors, GPUs process images, videos, and graphics efficiently in parallel.

The high performance of the GPUs received attention from the programmers and GPUs have started to be used for general purpose computation. Then, Ian Buck and his team generated a programming model by extending the C with data-parallel constructs in 2003. Afterwards, NVIDIA hired Ian Buck and started developing Compute Unified Device Architecture (CUDA) to provide a convenient solution for general purpose programming on GPUs. The first solution for general computing on GPUs was unveiled in 2006 [99].

One of the anchor points of GPU performance is its floating point capability. The GPU is specialized for compute-intensive, highly parallel computation so this architecture contains more transistors for data processing rather than data caching and flow control [94]. This architecture difference is illustrated in Figure 8 [94]. Corresponding parts are showed with the same color for both architecture.

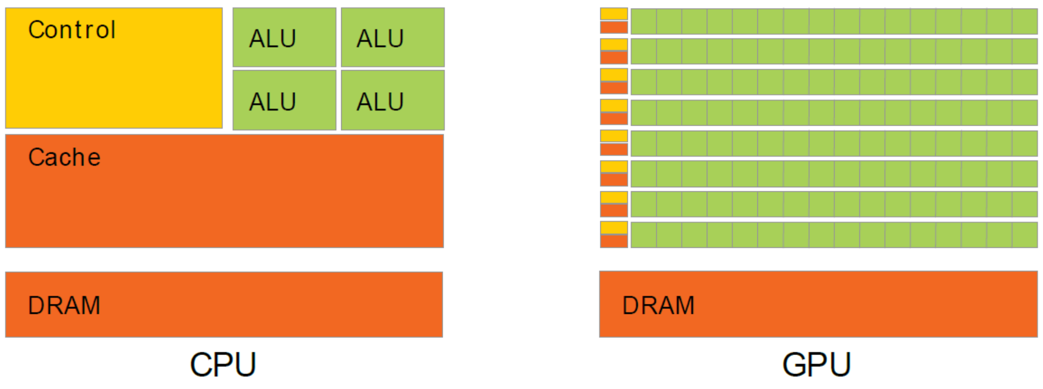


Figure 8: Architecture Difference between CPU and GPU [94]

The same program runs in parallel to apply the same operation on many data. Instead of focusing on the flow control, GPUs are designed to increase arithmetic intensity and memory bandwidth.

GPU architectures evolve in type and NVIDIA's GPUs are named according to their architecture. The architectures are called Tesla (2006), Fermi (2010), Kepler (2012) and Maxwell (2014) in historical order. Maxwell is the latest and the most advanced one. Moreover, new architecture Pascal is announced by NVIDIA. In Figure 9 [95], Maxwell Streaming Multiprocessor from GeForce GTX 980 can be seen. There are 16 stream multiprocessors in GTX 980. Each multiprocessor has 128 cores partitioned into four distinct 32-CUDA core processing blocks [95]. 32 LD/ST units are used for loading and storing operations in each multiprocessor. 32 Special Function Units (SFU) are used for acceleration of the functions like $\sin()$, $\cos()$, $\log()$. Furthermore, there are different types of memory blocks in the multiprocessor.

While there are a number of different parallel computing frameworks for general purpose computation on the GPU, the most prominent ones are CUDA and OpenCL [97]. CUDA has been developed by NVIDIA and it is specific to NVIDIA GPUs. CUDA uses C based language with some extensions [94]. On the other hand, OpenCL is an open source parallel programming platform and it was designed to allow programming different GPUs as well as other platforms like FPGA. It was initially developed by Apple and now maintained by AMD, INTEL, IBM and NVIDIA [97]. Its progress is moderately slow because it is a general framework and designed to run on a variety of different hardware. They both have different advantages and disadvantages over each other. Performance of the CUDA more convincing than the OpenCL because compatibility of hardware and software is higher for CUDA. Also, CUDA has a user-friendly language. However, CUDA obligates the user to use NVIDIA cards. On the other hand, any GPU card can be used with OpenCL. In this study, CUDA is preferred for GPU programming because performance of CUDA more satisfactory and it provides an easy to use C based language.

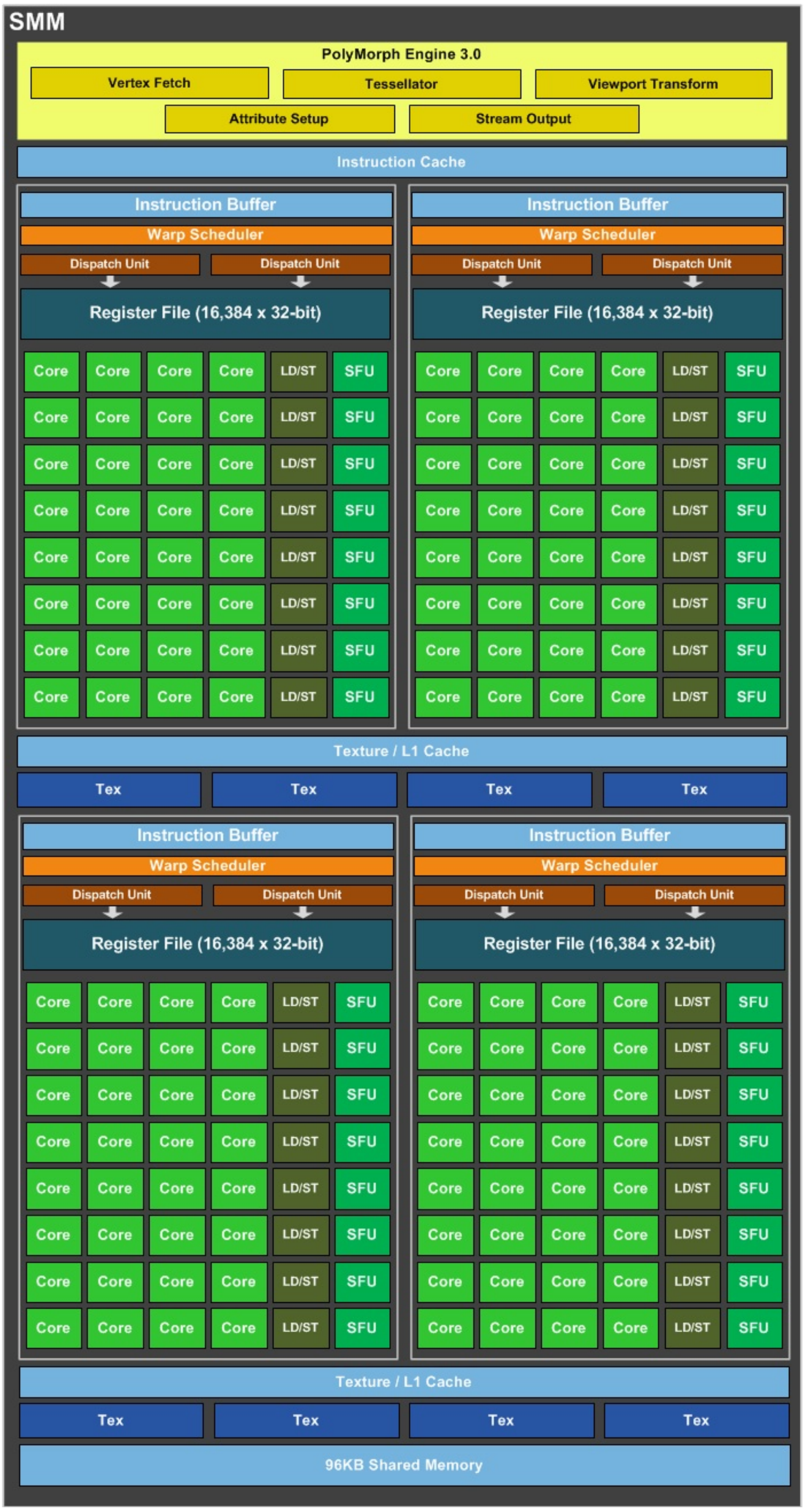


Figure 9: Maxwell Streaming Multiprocessor [95]

6.2.1 Compute Unified Device Architecture (CUDA)

The GPU is named as “device” and the CPU is named as “host” in CUDA programming. “Kernel” is the main CUDA function that is called from the host and it works on device in parallel. Each thread executes this kernel. Threads are organized in blocks and blocks are organized in a grid. Threads in the same block work in coherence. They can cooperate by using shared memory [94]. Thread organization can be seen in Figure 10.

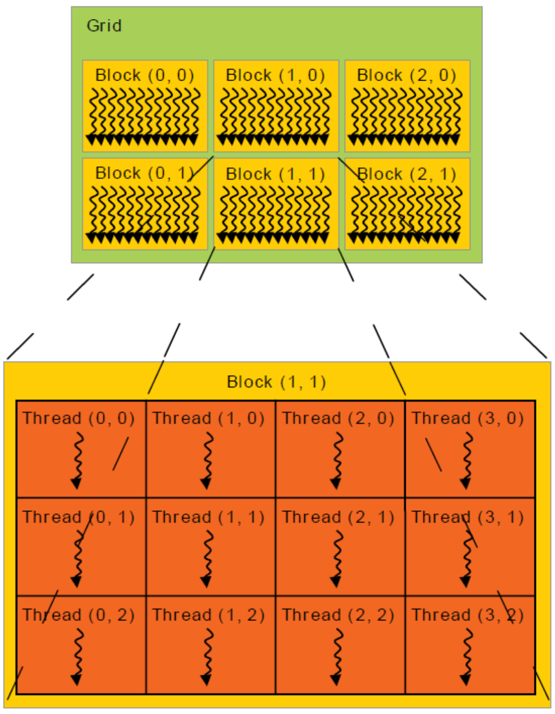


Figure 10: Thread Organization [94]

The number of blocks in a grid and the number of threads in a block are not constant. They are defined by users at kernel execution time. In order to achieve the optimal performance, they should be adjusted carefully for the specific hardware. There is no certain rule to choose number of threads in a block. It is usually chosen by tuning. Values are tried in order to maximize occupancy. During this trial, hardware constraints should be taken into account. When the same program run on different hardware, same experiments and definitions should be repeated to get actual performance of that hardware.

CUDA allows asynchronous operations on a heterogeneous CPU-GPU architecture. It means that during kernel execution on the GPU, the rest of the program can execute on the CPU as can be seen in Figure 11. Moreover, memories of the device and the host are separate, and the device makes its own allocations. Data transfers between host and device needs to be designed to allow asynchronous operation [94].

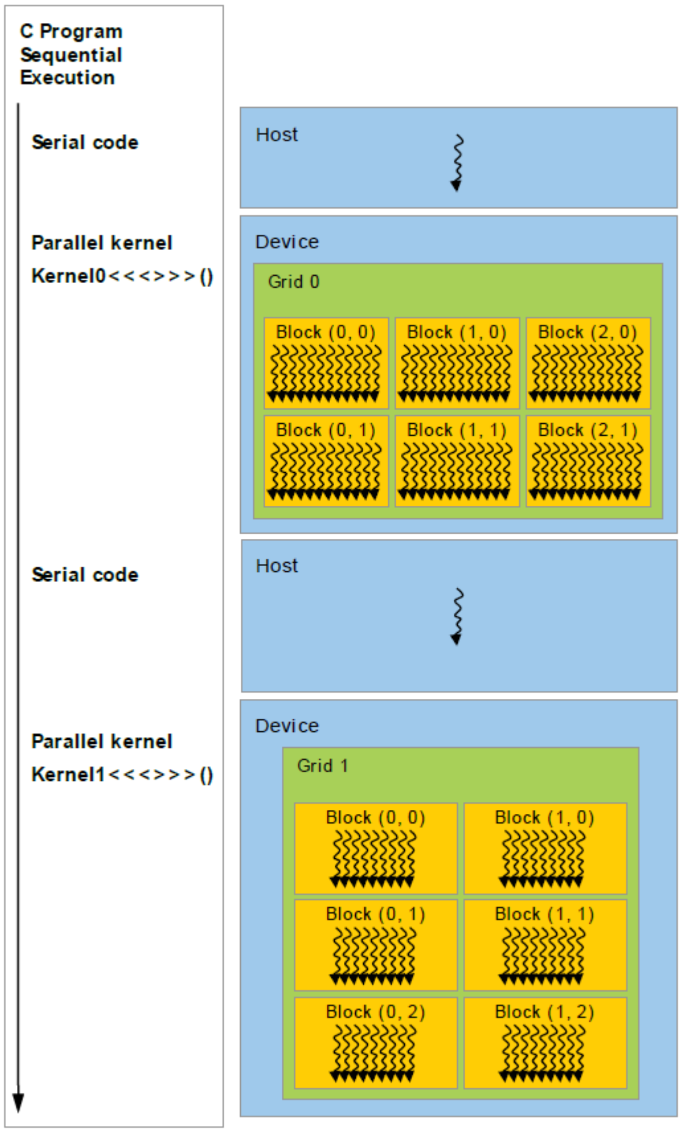


Figure 11: Heterogeneous Programming [94]

6.2.2 CUDA Memory Hierarchy

CUDA device has its own memory as mentioned before. Different kinds of memory types exist. These are register, shared, global, constant and texture. Their hierarchy can be seen in Figure 12.

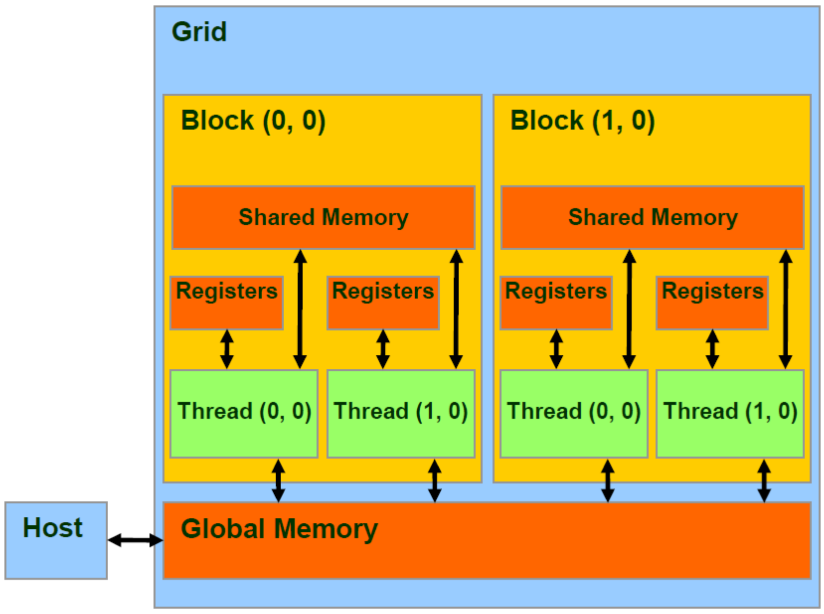


Figure 12: CUDA Memory Hierarchy [17]

- Registers belong to a thread. Only one thread can access one register and life time of the data is limited with one thread execution.
- Shared memory is matched with a block. All threads from a block can access to it. Threads can share data with the other threads from the same block via shared memory. Accessing to shared memory is as fast as accessing to the registers. Application can be speeded up with efficient shared memory usage. Its lifetime is related to the block. It is deallocated when the block ends.
- Global memory can be accessed from both the host and the device. It is explicitly allocated and exists until explicit deallocation. Its access speed is significantly low compared to shared memory and registers.
- Constant memory is a read only memory. It is optimized for broadcasting. It is also accessible from both the host and the device until the termination of the code.

- Texture memory is similar to global memory and constant memory. However, it is optimized for 2D data. Address calculation is done externally on hardware. Therefore, access to texture memory is faster than access to global memory.

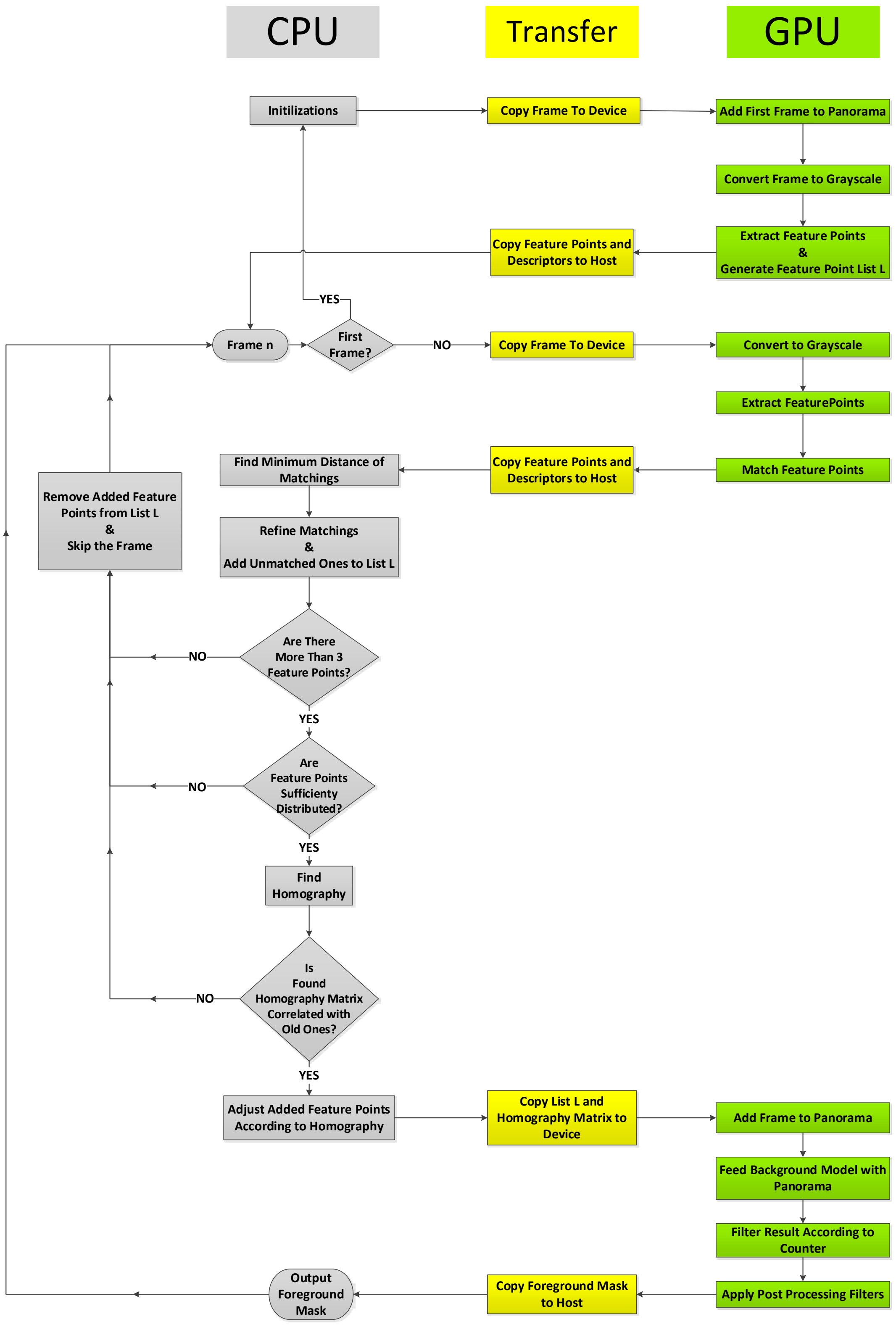
6.3 CUDA Implementation

CPUs and GPUs have different architectures so they both have different advantages and disadvantages over each other. In order to get an effective solution they must be used together because the GPU is good for some kind of problems and the CPU is good for the others.

In this study, heterogeneous programming is followed. Some algorithm steps work on the CPU, and the other parts run on the GPU. Firstly, CPU implementation of the whole study was completed and performance of each part was analyzed. Then, proper parts for parallelization are identified.

GPU parts of the process are handled in two ways. GPU versions of algorithms like SURF [9], Brute-Force Matching, IAGMM [89] are already available in OpenCV [98]. These functions have been used from library directly. On the other hand, algorithm parts such as panorama generation, blending, filtering according to counter have been designed and implemented for this study.

State chart of the whole process can be seen in Figure 13. In this figure, left side shows the CPU part, right side is for the GPU part and the middle area is for transfers. Moreover, colors show the group of each part.



CPU
Transfer
GPU
Initilizations
Copy Frame To Device
Add First Frame to Panorama
Convert Frame to Grayscale
Extract Feature Points & Generate Feature Point List L
Copy Feature Points and Descriptors to Host
YES
Frame n
First Frame?
NO
Copy Frame To Device
Convert to Grayscale
Extract FeaturePoints
Match Feature Points
Copy Feature Points and Descriptors to Host
Find Minimum Distance of Matchings
Refine Matchings & Add Unmatched Ones to List L
Remove Added Feature Points from List L & Skip the Frame
NO
Are There More Than 3 Feature Points?
YES
NO
Are Feature Points Sufficienty Distributed?
YES
Find Homography
NO
Is Found Homography Matrix Correlated with Old Ones?
YES
Adjust Added Feature Points According to Homography
Copy List L and Homography Matrix to Device
Add Frame to Panorama
Feed Background Model with Panorama
Filter Result According to Counter
Apply Post Processing Filters
Copy Foreground Mask to Host
Output Foreground Mask

Figure 13: Partitions of the Proposed Algorithm on GPU and CPU

6.3.1 Initialization

When algorithm starts, firstly memory is allocated on both CPU and GPU. After memory allocations, a new frame is taken and it is copied into the device memory. Then, the processing is passed onto the GPU as shown in Figure 13.

Adding the first frame to panorama operation is done on the GPU because panorama is kept in the GPU memory and background subtraction is also done on the GPU.

Feature points of first frame are extracted and panorama features point list is generated. Method of feature point will be explained in detail in next chapter.

6.3.2 Feature Extraction and Matching Part

Every time a new frame comes, it is copied into the device memory with “cudaMemcpy” function. Then, features are extracted. The process can be seen in Figure 13.

SURF feature points are extracted from the frame by using OpenCV “gpu::SURF_GPU” function. Processing this part on the GPU is very important for an effective implementation because finding feature points of the image with size 704x480 takes 106 ms on the CPU. However, on GPU it takes only 2.87 ms with 39.93 speedup.

The matching operation of new feature points and feature points from the list is completed on the GPU side with the OpenCV function “gpu:: BruteForceMatcher_GPU_base”. Similar to SURF case there is a nearly 48 times speedup for GPU. Execution times on CPU, GPU and speedup values in are summarized Table 4.

Following parts (refining feature points and controlling distribution of feature points) take less than 1 ms so these operations are done on the CPU side. Feature points and descriptors are copied to CPU to complete these steps.

6.3.3 Image Registration

When feature points are matched and refined, it is time for homography calculation. Finding homography by using RANSAC is also not challenging for CPU. It takes 0.72 ms in average so it is decided to complete RANSAC on CPU side.

The new frame is added to the panorama on the GPU side. As stated before, panorama image is kept in GPU memory. The operation is done by a custom kernel function developed for this study. Each thread of this kernel takes one pixel from the new image and adds to the panorama after blending it with the current value of the panorama. There is a 174 times speedup for GPU. In this operation each task has low complexity, but number of tasks is too many. Therefore, it is suitable for parallelization.

6.3.4 Background Subtraction

After panorama update, background subtraction is applied by using OpenCV “gpu::MOG2_GPU” function. For this case speedup is equal to 42.77 times for GPU. On the CPU side, the corresponding CPU OpenCV function is used.

Filtering according to first seen part is the responsibility of the kernel which is developed for this study. Each thread of kernel compares the counter value with a threshold and according to result corresponded pixel from the foreground mask is modified. Execution time of this operation is 29.61 ms for CPU and 0.9 ms for GPU.

Morphological operations (opening and closing) are also available in OpenCV: "gpu::dilate" and "gpu::erode" functions. These functions are more than 20 times faster than their CPU OpenCV counterparts.

6.4 Precision Difference between GPU and CPU

Execution time of GPU and CPU are different because of their computational power. Moreover, they do not produce the same results. There are three main reasons behind this difference.

The handling of floating point numbers is different for CPU and GPU. The GPU has a better approach about floating point numbers [96]. GPU uses The Fused Multiply-Add (FMA) approach [96]. The basic principle of the approach is decreasing the number of rounding operations by merging operations. FMA usage may increase the accuracy. Therefore, GPU floating point operations may be more accurate than CPU operations [96].

The second reason is about conservation of performance. Some parts of the algorithms that are not appropriate for parallel processing. It is not possible to transfer them to the GPU side directly without any modifications. Unfortunately, these modifications cause some differences between CPU and GPU results. Insisting to port algorithms exactly may generate performance penalty and a balance should be sustained between accuracy and performance.

For example, descriptors that are calculated by using SURF [9] with CPU and GPU are different even if their feature points are the same. “SURF” function uses inter area interpolation on the CPU. On the other hand, “gpu:: SURF_GPU” function uses bilinear interpolation for better performance as it is more suitable for parallel computation.

Lastly, optimizations in GPU codes may cause different outputs for each runtime. Return point or exit point of the code may differ for different runs. Because of this reason, there may be a difference between not only GPU and CPU results, but also in two GPU runs. For example “gpu::SURF_GPU” calculates different descriptors for the same set of feature points at consecutive runs.

CHAPTER 7

TEST RESULTS AND APPLICATIONS OF THE THEORY

7.1 Introduction

Steps of Pan-Tilt camera background subtraction algorithm are described in Chapter 3, Chapter 4 and Chapter 5. Then, porting this system to the GPU is explained in Chapter 6. Both CPU and GPU - CPU implementations of the system have been completed and GPU and CPU results have been compared with the groundtruths. In addition, GPU results have been compared with the results of Change Detection [93] algorithms and Tsinko’s result [76] respectively.

Two frame sources are used for the tests. First one is Change Detection [93] and the second one is Tsinko [76]. ContinuousPan frame sequence is used from Change Detection [93]. Resolution of the frames is 704 x 480. Six different videos are listed in Tsinko’s web page [91]. Resolution of them is 160 x 131.

All experiments have been done on a PC with Intel i5 3.3 GHz. CPU. There are two GPUs used for the test. First one is NVIDIA GeForce GTX 560 (GPU 1) and the other one is NVIDIA Tesla K40 (GPU 2).

Although, our CPU implementation runs on quad-core processer, it is single thread application. Its performance can be increased by using Accelerated Massive Parallelism (AMP). All performance comparison between GPU and CPU in our study should be evaluated by considering that our CPU implementation runs on only one core of the Intel i5 processer.

7.2 Evaluation Metrics

In order to build a reliable and robust real-time system, the speed of the algorithm is considered to be the most important performance criteria. Therefore the speed is the first evaluation metric to evaluate the system. In this thesis work, the execution times of each individual part of the algorithm and the overall processing time measured in milliseconds.

Measuring the ability to extract foreground pixels accurately and moving object detection have been challenging tasks over the years. Change Detection [93] deals with this problem by bringing seven different performance metrics in literature together. Before explaining them, four detection types should be examined. True Positive (TP) is used for the pixels that are correctly identified. False Positive (FP) means that the pixel is incorrectly identified. True Negative (TN) is suitable for correctly rejected pixels. Finally, False Negative (FN) is used for incorrectly rejected pixels. Evaluation metric of Change Detection [93] can be seen in Table 3.

Table 3: Evaluation Metrics

Number	Name	Abbrev.	Equation
1	Recall	Re	$\frac{TP}{TP+FN}$
2	Specificity	Sp	$\frac{TN}{TN+FP}$
3	False Positive Rate	FPR	$\frac{FP}{\text{FP} + \text{TN}}$
4	False Negative Rate	FNR	$\frac{FN}{TP+FN}$
5	Percentage of Wrong Classifications	PWC	$100 * \frac{FN+FP}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$
6	Precision	Pr	$\frac{TP}{TP+FP}$
7	F-measure	Fm	$2 * \frac{Pr * Re}{Pr + Re}$

7.3 Comparison of CPU and GPU Results

The results are compared with each other in two ways. Firstly, the speeds of the implementations are compared for each part and the whole process. Secondly, foreground masks of the implementations are compared with the groundtruths.

ContinuousPan frame sequence from Change Detection [93] is used for this test. Resolution of the image is 704 x 480. In this frame sequence, the area is very wide so the camera is rotated too much horizontally. In order to get the overall area, size of the panorama is defined as 11 x 3 times of the frames. Therefore, resolution of the panorama is 7744 x 1440.

The video has a very good texture, therefore Hessian threshold is adjusted to more than 5000, in order to reduce the number of feature points and increase their quality. 6000 is used for this test. Some frames of the image sequence can be seen in Figure 14.

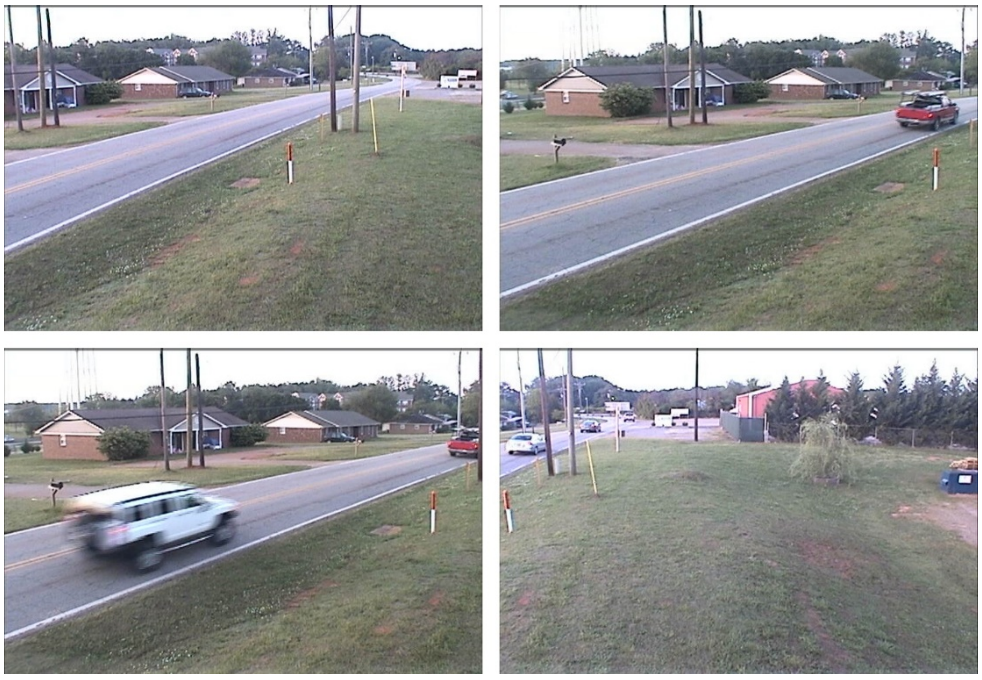


Figure 14: Change Detection ContinuousPan Image Sequence Samples

Parameters of the background subtraction algorithm are used mostly the same with default values. However, history is set to 100 instead of 500 because motion starts before reaching frame 500. Background ratio is changed from 0.9 to 0.8 and minimum variance value is increased from 0.4 to 0.5 in order to increase sensitivity of the algorithm. Also, learning rate is set to 0.01 similar to [76]. 60 is chosen for threshold to filter according to first seen part. The threshold has an inverse relation with learning rate; small threshold value can be used for higher learning rate, and vice versa.

Both CPU and GPU implementations are processed with this video and execution time of each algorithm is measured separately. Also, time of the whole process is measured. Measurements are done for each frame and averaged. In order to get reliable results, the operation is repeated 10 times using the same data and averaged. The results can be compared as in Table 4.

Operations like “Get Frame” and “Find Homography” are completed by the CPU. Moreover, there are 2 copy operations between CPU and GPU. These operations are done by the CPU. The CPU writes data to GPU memory or reads from there. For only CPU case, there is no corresponding operation. It is shown with a hyphen on the table.

Table 4: Time Comparison

	CPU (ms)	GPU 1 (ms)	GPU 2 (ms)	Speed Up GPU 1	Speed Up GPU 2
Get Frame	0.57 (by CPU)			-	-
Copy Frame to GPU	-	0.32 (by CPU)		-	-
Feature Extraction	106	15.89	2.87	6.67	36.93
Feature Matching	41.59	2.21	0.85	18.82	48.93
Copy Feature Points and Descriptors to CPU	-	0.15 (by CPU)		-	-
Refine Matching	0.01 (by CPU)			-	-
Feature Point Distribution Control	0.07 (by CPU)			-	-
Find Homography	0.72 (by CPU)			-	-
Homography Matrix Control	0.01 (by CPU)			-	-
Arrange Added Points	0.01 (by CPU)			-	-
Copy List L and Homography Matrix to GPU	-	1.27 (by CPU)		-	-
Addition of Frame to Panorama	95.71	0.68	0.55	140.75	174.02
Background Subtraction	200.17	12.53	4.68	15.98	42.77
Filtering According to First Seen Part	29.61	0.94	0.91	31.5	32.54
Post Processing	149.59	7.45	5.52	20.08	27.10
Copy Results to CPU	-	0.35 (by CPU)		-	-
TOTAL	624.06	43.18	18.86	14.45	33.09

The CPU reads the frame in less than 1 ms. The frame is ready for CPU to process, but for GPU to process, it has to be copied to the GPU. Copying is completed by CPU in less than 1 ms. Feature extraction and feature matching are the next steps where the advantage of the GPU can be observed. The CPU needs 106 ms but GPU 2 requires only 2.87 ms to extract the features. As stated before SURF [9] is a robust algorithm, but it is slower than binary feature extractors. By using GPU disadvantage of the SURF [9] is defused. There are 6.67 and 36.93 speedups for GPU 1 and GPU 2 respectively.

After extraction, feature matching operation is applied. It has better speedup results of 18.82 and 48.93 respectively for GPU 1 and GPU 2. There is an important issue about feature matching. As stated in Chapter 3, feature points of panorama are kept in the list L. When there is no match between a feature point of the new frame and feature points from L, unmatched feature point is added to L. Therefore, the size of the list increases with new frames. Because of this increase in size, feature matching time also increases. For example, GPU 2 requires 0.01 ms at the beginning of the sequence for feature matching but at the end 1.65 ms is necessary for this operation. Feature matching time consumption values in the table are average values that are calculated with the values from all the frames.

Results of the feature extraction and matching are transferred to the CPU in 0.15 ms. Then, Refine Matching, Feature Point Distribution Control, Find Homography and Homography Matrix Control operations are completed respectively in nearly 1 ms. All of them combined takes less than 1ms so porting them to the GPU is unnecessary. Then, updated feature points list L is copied to GPU for the next matching operation and also found homography matrix is copied to GPU. Both require 1.27 ms. Similar to feature matching, duration of this copy depends on the size of list L. It is increasing with time. In order to prevent this increase, list L can be kept in both sides and only newly added feature points can be transferred.

Addition of frame to the panorama has the highest speedup with 140.75 and 174.02 because of the nature of the operation. The computational load of each task is low and the number of tasks is high. These properties make it convenient to be parallelized. The pixels are read from the new frame and placed into new positions according to the homography matrix in the panorama. Each thread processes 1 pixel and the operation completed in a very brief time.

Filtering According to First Seen Part is very similar to addition of frames. Each thread makes comparison of 1 pixel. The operation takes less than 1ms and speedup is approximately 30 for both GPUs.

Background subtraction is the most time consuming part of CPU side with 200.17 ms duration. GPUs reduce this processing time to 12.53 and 4.68 ms. Speedups are 15.98 and 42.77 for GPU 1 and GPU 2 respectively. Panorama image has more than 10 million pixels, so background subtraction on this image requires significant computational power. Therefore, GPU usage has results in significant speedup.

Post processing is consecutive closing and opening operations. They take 149.59 ms on the CPU side. It is also the second most time consuming part of the CPU side. GPU usage accelerates the operation to 20.08 and 27.10 times for GPU 1 and GPU 2, respectively.

Finally, the foreground mask result is transferred to the CPU and the process is completed. The total process takes 624.06, 43.18 and 18.86 ms for CPU, GPU 1 and GPU 2 respectively. There are 14.45 speedup for GPU 1 and 33.09 speedup for GPU 2. Implementation of background subtraction with panorama model to run in real-time with only CPU is not possible with 624.06 ms process time for each frame. However, GPU makes possible running in real-time with high frame rates. Tsinko [76] stated that Phyton implementation of his algorithm requires 40 seconds to process one frame with resolution of 160 X 131 on Intel i5 2.66 GHz. CPU. Our C++ implementation works on frames with higher resolution with higher performance on both CPU and GPU.

If the results of GPU 1 and GPU 2 are compared, it can be seen that GPU 2 is faster than GPU 1 nearly three times. GPU 1 is NVIDIA GeForce GTX 560. It has 336 CUDA cores and Fermi architecture with CUDA Compute Capability 2.1. On the other hand, GPU 2 is NVIDIA Tesla K40. It has 2880 CUDA cores and Kepler architecture with CUDA Compute Capability 3.5. GPU 2 has higher number of CUDA cores [100] and most of the speed difference can be attributed to this. It has also a more optimized architecture. Moreover, CUDA Compute Capability of GPU 2 is higher than GPU 1. Compute capability is an indicator of GPU's abilities.

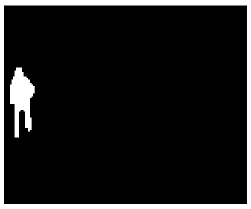
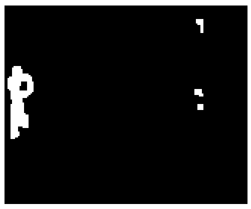
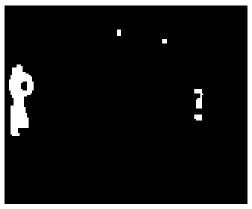
Outputs of the CPU and GPU are also compared. Foreground masks of them are compared to the groundtruths and their similarities are calculated by checking each pixel.

As stated before in Chapter 6.4, GPU results of the same software on the same device may differ from one runtime to another. Therefore, tests on the GPU are completed more than one times and then averaged.

Both CPU and GPU implementations were run on six different videos from videos of Tsinko [76] and one video from Change Detection [93] for 10 times and their outputs are compared with groundtruths. Comparison was done by comparing each pixel and calculating F-measure (Fm), Fm creates a chance to understand their similarity in an objective way because Fm is composed of Precision (Pr) and Recall (Re). In other words, Fm is ratio of the similarities to all detections.

Average similarity between GPU outputs and groundtruths is 0.73. This value is 0.71 for the CPU comparison. Values are very close to each other. CPU, GPU outputs and Groundtruth of Frame 442 in Sequence 2 from [76] can be seen in Table 5. Outputs look like very similar. GPU usage does not cause any notable performance loss. Small differences seem unimportant beside a contribution of the GPU. Without considerable loss, GPU usage increases speed by using parallel high computational power.

Table 5: CPU & GPU Outputs and Groundtruth

	Average Fm Values		
	1.00	0.73	0.71
			
Example Original Frame	Groundtruth	GPU Result	CPU Result

7.4 Comparison with the Change Detection Algorithms

In [93], a dataset for Change Detection challenge organized by IEEE Computer Society. The dataset includes videos and their groundtruths under 11 different categories. The best performing algorithm results are also published in website of the challenge.

Although, algorithms in the competition are designed for the sequences of static camera, they made their experiments by adjusting some parameters to detect motion of the scenes from Pan-Tilt camera and their results are published on the web page [93].

We conduct two experiments on this data. First, only background subtraction part [89] of our study is applied to ContinuousPan frame sequence similar to other implementations. IAGMM [89] is preferred for background subtraction part of our study as stated before. Then, full version of our study is applied to the video and then performances are compared. Comparison is done between five results. These are background subtraction part [89] of our study, our normal study and the first three rank algorithm of the PTZ category. Results can be seen in Table 6.

GPU/CPU comparison is given in the previous section, so only GPU results are used for this comparison. Performance ratios are calculated for each frame and their average is taken. In order to get more stable measurements, this operation is repeated 10 times and the average is taken again.

Table 6: Comparison with the Change Detection Algorithms

		Algorithms				
		Our Study	IAGMM [89]	MBS [64]	IUTIS-3 [11]	SuBSENSE [69]
Metrics	Re	0.6074	0.6697	0.5973	0.6644	**0.8306**
	Sp	0.9575	0.9238	**0.9963**	0.9868	0.9629
	FPR	0.0425	0.0762	**0.0037**	0.0132	0.0371
	FNR	0.3224	0.2065	0.4027	0.3356	**0.1694**
	PWC	6.0918	7.9980	**0.5850**	1.5649	3.8159
	Pr	0.4534	0.1623	**0.5400**	0.3474	0.2840
	Fm	0.5088	0.1978	**0.5520**	0.3921	0.3476

When Recall (Re) values are compared, our study has similar performance with the other algorithms. The reason behind that the static camera algorithms have high Re performance is about definition of Re. It is not related to moving or static camera. When the camera moves, background model of static camera algorithm becomes incompatible and algorithm classifies many pixels as foreground. This increases true positive (TP) and false positive (FP) at the same time. This is the reason of static camera algorithms have high Re performance, especially [69].

Specificity (Sp) result of our algorithm is relatively lower. However, it is still very close to the other results. The reason of small differences between our both results and the other results is shadow. We do not do anything to prevent the classification of shadows as a foreground but they do. Shadows cause an increase in FP and a decrease in TN. Therefore, Sp has lower value.

Similarly, Our False Positive Rate (FPR) results are higher than the others because of shadows and registration errors which cause some FPs. Another reason of higher FPs is modifications in other algorithms [11], [64] to decrease FPs. These modifications will be explained in the next section.

False Negative Rate (FNR) of the Change Detection [93] algorithms is high because the algorithms are designed for static camera. However, in order to process Pan-Tilt video [11] and [64] made some modifications on the parameters. The purpose of these modifications is decreasing the FP because normally when the camera moves, background model becomes incompatible and the algorithm classifies everywhere as foreground. After some modifications like increasing learning rate, the algorithms prevent these FPs and FPRs. As stated before, they have very low FPR. Disadvantage of these modifications is high False Negative (FN) because real foreground objects are also cannot be labeled. Only some parts of them classified as foreground. On the other hand, IAGMM [89] and SuBSENSE [69] give lower result because there is not any modifications to decrease FPs. Therefore, when the camera moves, [69] and [89] classify everywhere as foreground and count of FNs is small. This is also the reason for highest FPR.

Percentage of Wrong Classifications (PWC) shows the percentage of errors to all detections. The reason behind high PWC of our study is again shadow and registration errors. RANSAC [25] algorithm creates a homography with one or two pixel error. These errors cause a misalignment between panorama and background model. Therefore, all registration errors are classified as foreground. Reason of the high PWC of IAGMM [89] is different. When the camera moves, it classifies everywhere as foreground with a high FPR, therefore its PWC is also high.

Last two metrics are Precision (Pr) and F-measure (Fm). They are very important indicators of the performance. Our algorithm gets the second best place among the algorithms. In our study, we subtract background after eliminating the effects of Pan-Tilt camera so we do not need an extra effort to clear FPs because of camera motion. [11] and [64] make some modifications to reduce FPs so TPs are also decreasing. Because of this reason, [11] and [64] have lower TP so their Re and Pr are also low. On the contrary, our study has high Re and Pr. Furthermore, IAGMM [89] has the lowest Pr and Fm because of labeling everywhere as foreground.

To sum up, our algorithm has persuasive performance with high Re, Pr and Fm rates. Performance of IAGMM [89] is increased with our camera motion elimination process. On the other hand, FPR should be decreased. The main reason of high FPR is classification of shadows and registration errors as a foreground. Moreover, if our background subtraction part is replaced with a more successful algorithm, performance of our study will increase automatically. If all scores are considered, our algorithm has convincing performance among Change Detection [93] algorithms.

7.5 Comparison with [76]

As stated before, the scope of our study and study of Tsinko [76] are very similar to each other. Same approaches are followed in both theses. Our implementation is operated with the videos of Tsinko [76]. The videos are accessible on his web page [91].

In study [76], three different background subtraction algorithms are given. One of these algorithms is GMM [70]. We compare our algorithm with GMM implementation of [76] in this section. Since GPU/CPU comparison is given before, we only use GPU results in this section.

The outputs were taken from six videos in [76]. Half of the videos are outdoors and the others are indoor videos. Four frames are chosen from each video and groundtruths are available for these frames. Number of FN and FP were counted on each of these four frames. Also, ratio of these calculated FN and FP values to the number of all pixels in the frame was calculated. Our algorithm was also operated with two videos. One video is from outdoor videos and the other one is from indoor videos. These videos are available on the web page [91].

Used parameters are mostly the same with the previous part. Only Hessian threshold of SURF [9] is adjusted to a small value such as 30, because videos are in low resolution 160 x 131 and their textures are not so much. Therefore, in order to increase the number of feature points, a small Hessian threshold value is used.

7.5.1 Frame Sequence 1

Sequence 1 is formed of 1090 frames. It is also named as Sequence 1 in [76]. The recorded area is a lawn with a small pathway. There are buildings behind. Some trees are on the right. They are swaying with the wind. It is a good moving background challenge. There is a significant change in the illumination on the scene. This is also a good challenge. During camera motion, a person comes from the right in frame 584 and after a small walk he stops in frame 657. He stays there for a while. Later, he continues in frame 800 and leaves the camera’s field of view in frame 981. Some frames of the image sequence can be seen in Figure 15.

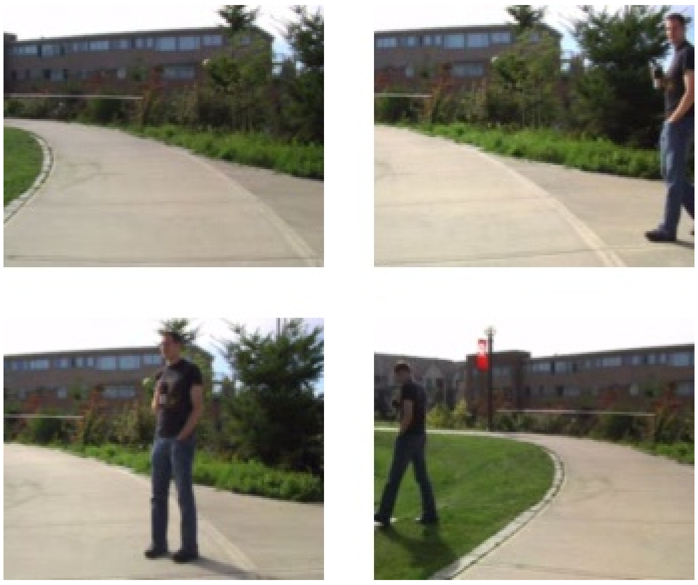


Figure 15: Some Frames of Sequence 1

Comparison results are gathered in two tables. Table 7 shows the foreground labels. On the other hand, Table 8 contains numbers and ratios of FP, FN and their total.

In Frame 625, our algorithm detects the person, but some parts of the upper body are not detected. The reason is the color of the t-shirt. It is very similar to the dark trees behind. Therefore, the algorithm cannot distinguish it from the background. When some parts of a foreground object has the same color with background, IAGMM [89] generally classifies it wrongly because background IAGMM models may match with this object. This drawback of the IAGMM [89] increases the FN. Another point is the shadow of the person. Our algorithm labels it as a foreground because we do not make any operation to detect shadows. This situation increases the FP. The results are nearly same for Tsinko [76]. Due to the morphological filters at the end, the result of our study is more preferable and better than the others, also these filters prevent the noise.

Frame 657 is the start point of the person standing. When the frame count is equal to 720, he has been standing on the same location since 63 frames. Therefore, our algorithm learns it. He is classified as background and he causes 10.73 % FN ratio. Algorithm of the Tsinko [76] responds with 10.4 % similarly. All GMM algorithms except the algorithms with zero learning rate, learn and classify foreground objects that stay in the scene without moving as a background after some time. On the left side, there are some FP points. The reason is the illumination change. The grasses in the frames appears to be brighter now because the sky is clearer.

Frame 850 has a similar result with Frame 625. It finds the person with some missing parts because of t-shirt color. These missing parts cause some FN points. On the left side, there are some illumination change false detections and in the middle, some parts of the shadows are classified as foreground. Both of them cause FP points. However, the result is satisfying with 5.44 % FP + FN ratio. It is nearly half of [76]. There is a problem with background model of [76] because it finds two people. After a long duration of the appearance of the person in scene, algorithm of [76] learned the person and classified as background similar to our study. However, after person leaves from that position, the algorithm did not adapt itself quickly. This time, it classified the absence of the person as a foreground object. This problem may be solved with higher learning rates. In GMM algorithms, when stationary foreground object moves, it is classified again foreground object but this time, absence of it perceived as a change in background for a while. This causes FPs.

In Frame 930, our algorithm works with a very small FN ratio, but a high FP ratio. There are two reasons of high FP ratio. First one is the shadow and the second one is illumination change on the left side as stated before. While the camera is facing to the right side, the sky of the left side comes to a clearer state, but still our performance is better than [76]. There is another problem on [76]. It has no control for first seen areas. All left side is classified as foreground by [76]. In order to hinder this situation our algorithm has a control mechanism.

Table 7: Comparisons of Sequence 1

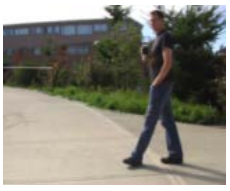
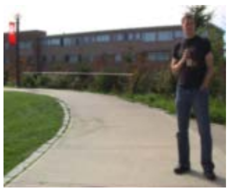
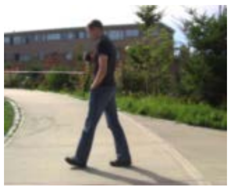
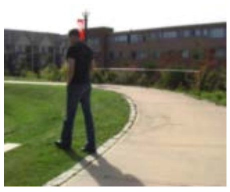
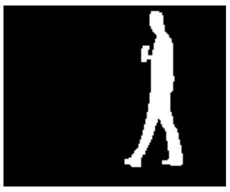
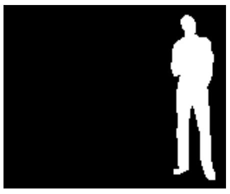
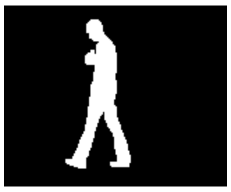
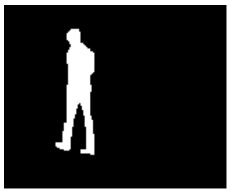
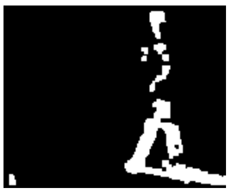
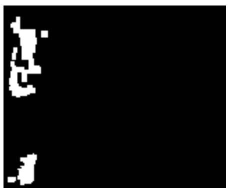
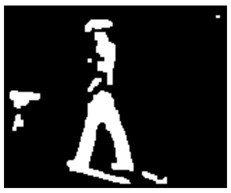
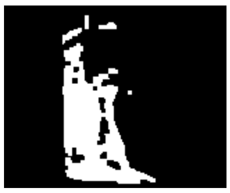
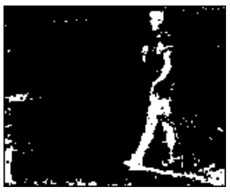
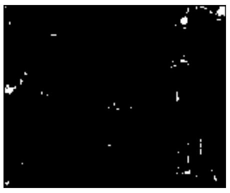
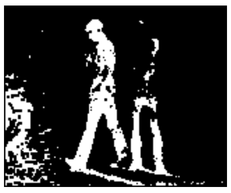
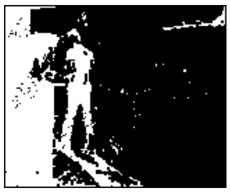
	Frame 625	Frame 720	Frame 850	Frame 930
Frames				
Groundtruths				
Our Results				
Results of Tsinko				

Table 8: Errors for Sequence 1

			Frame 625	Frame 720	Frame 850	Frame 930	Total
Our Results	FP	Count	713	908	723	2220	4564
		Ratio (%)	3.40	4.33	3.45	10.59	5.44
	FN	Count	771	2250	417	314	210
		Ratio	3.68	10.73	1.99	1.5	4.48
	FP + FN	Count	1484	3157	1140	2534	8315
		Ratio (%)	7.08	15.06	5.44	12.09	9.92
Results of Tsinko	FP	Count	667	128	1778	4701	7274
		Ratio (%)	3.18	0.61	8.48	22.43	8.68
	FN	Count	860	2181	485	317	3843
		Ratio (%)	4.10	10.4	2.31	1.51	4.58
	FP + FN	Count	1527	2309	2263	5018	11117
		Ratio (%)	7.28	11.01	10.79	23.94	13.26

Our algorithm gets frame and puts them in panorama before background subtraction process. In the Figure 16 panorama of the Sequence 1 can be seen.

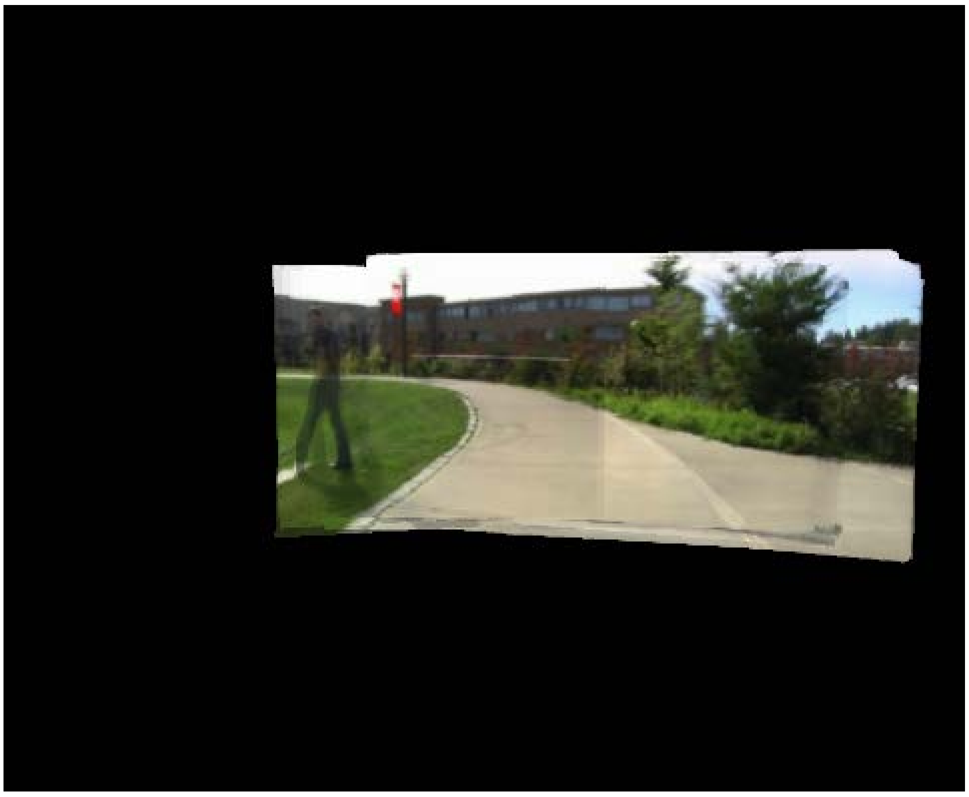


Figure 16: Panorama of Sequence 1

7.5.2 Frame Sequence 2

Sequence 2 is formed of 454 frames. It is named as Sequence 6 in [76]. The recorded area is in the inside of the building. The camera sweeps the area by panning and tilting. There is a plant on the left of the scene. There are walking people in the scene. During camera motion, a person appears on the right side in frame 200 and he starts to walk towards to the camera. Then, another person appears in frame 345 and he also starts to walk towards to the camera. First person leaves the field of view in frame 419. In the end, the camera rotates to right and second person also disappears. Some frames of the image sequence can be seen in Figure 17.

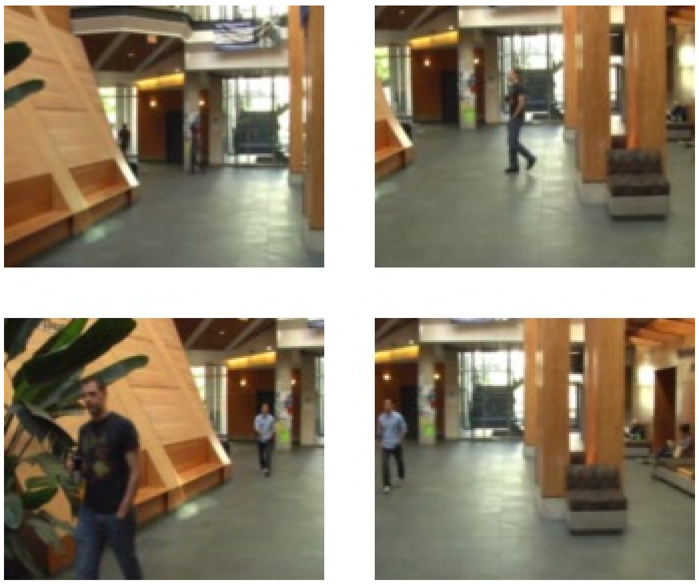


Figure 17: Some Frames of Sequence 2

Comparison results are gathered in two tables. Table 9 shows foreground labels. On the other hand, Table 10 contains numbers and ratios of FP, FN and their total.

In frame 239, our algorithm detects the person with a satisfactory FN ratio that is only 0.28 but FP ratio of the result is a little bit higher. There are two reasons behind the fact that FP ratio is equal to 2.65. First one is that, the area on the left side has sunlight. The algorithm classified it as a foreground. Another reason is the detection of the shadow. As stated before, our system does not cover any operation to understand shadows. These both situations cause false alarms so FP ratio increases. On the other hand, the FP ratio of the Tsinko [76] is very low, but it does not mean that the performance of [76] is better than our performance because the foreground area of frame 239 is already small. Moreover, our study has better FN ratio.

Frame 314 is one of the frames that the person walks through to the camera. Our outputs are better than the results of [76] in both aspects. Walking person is classified as foreground with 2.20 FP + FN ratio. However, [76] has higher error ratios. The first problem of [76] is classification of some body parts as background because of the clothes' color. Another and more important problem is the absence of the control for the first seen areas. Some regions on the left side are classified as foreground by [76]. In order to prevent this kind of false alarms, a counter is kept for each pixel and without reaching a threshold; this pixel is not classified as foreground.

Frame 406 has a similar result with frame 314 because very small area is labeled as a foreground by [76]. Therefore, the FP ratio of [76] is small but the FN ratio of it is nearly two times of our result. On the other hand, our FP ratio is 4.53 because of the presence of the plant. The algorithm classifies some parts of the plant as foreground because of the similarities between its color and the person's t-shirt color.

In Frame 442, [76] finds nearly nothing again, in comparison our algorithm detects the person with satisfactory ratios. On the right side, there are some false alarms because of the registration errors. In order to increase the performance of our algorithm, registration errors of output should be decreased.

Table 9: Comparisons of Sequence 2

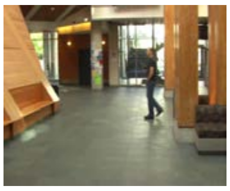
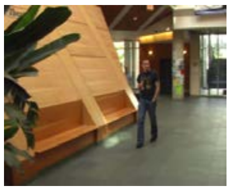
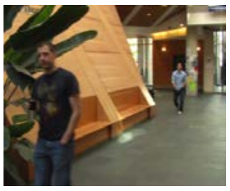
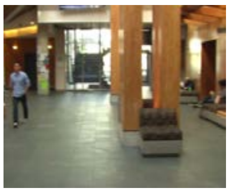
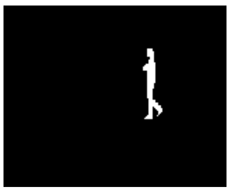
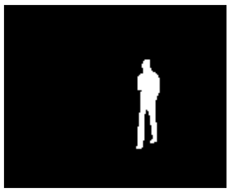
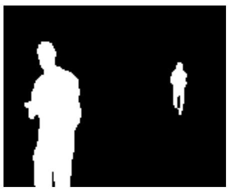
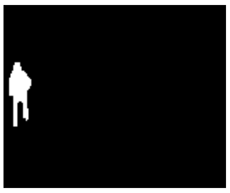
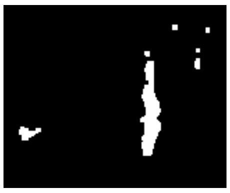
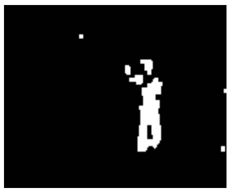
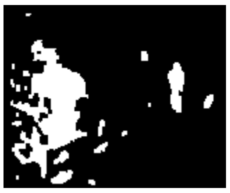
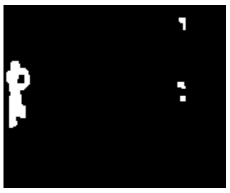
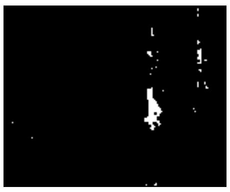
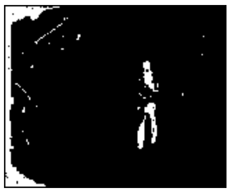
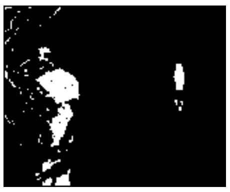
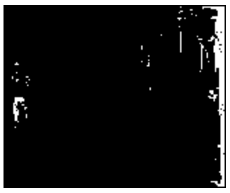
	Frame 239	Frame 314	Frame 406	Frame 442
Frames				
Groundtruths				
Our Results				
Results of Tsinko				

Table 10: Errors for Sequence 2

			Frame 239	Frame 314	Frame 406	Frame 442	Total
Our Results	FP	Count	555	317	950	239	2061
		Ratio (%)	2.65	1.51	4.53	1.14	2.46
	FN	Count	37	145	853	113	1148
		Ratio	0.28	0.69	4.07	0.54	1.40
	FP + FN	Count	592	462	1803	352	3209
		Ratio (%)	2.93	2.20	8.60	1.68	3.85
Results of Tsinko	FP	Count	134	964	99	725	1922
		Ratio (%)	0.64	4.60	0.47	3.46	2.29
	FN	Count	158	341	1833	346	2678
		Ratio (%)	0.75	1.63	8.75	1.65	3.19
	FP + FN	Count	292	1305	1932	1071	4600
		Ratio (%)	1.39	6.23	9.22	5.11	5.48

Overall, our implementation has more satisfactory error ratios. One of the reasons is better adaptation of our algorithm. Frame 850 of Sequence 1 is the proof of it. After a long stay of foreground object, our algorithm can cover his model and classify the object correctly. Second, our study can handle first seen parts similar to Frame 930 of Sequence 1 and Frame 314 of Sequence 2. First seen parts are always the problem of Pan-Tilt camera background subtraction. Because of insufficient time, they are classified as foreground automatically. However, our algorithm can deal this problem.

Our algorithm gets frame and puts them in panorama before background subtraction process. In Figure 18 panorama and foreground mask of the Sequence 2 can be seen.

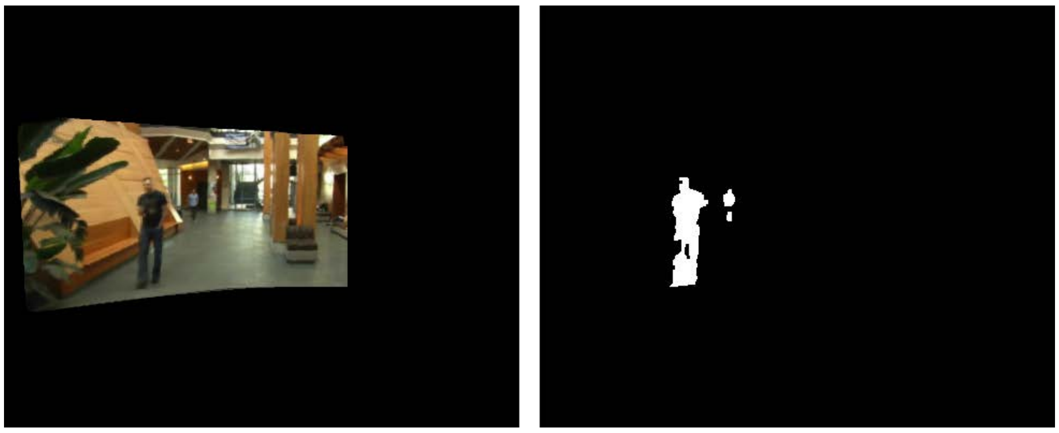


Figure 18: Panorama of Sequence 2

CHAPTER 8

CONCLUSIONS

In this thesis, we proposed a method for background subtraction of frames recorded by Pan-Tilt cameras. In order to compensate camera motion, features in each frame are extracted with SURF [9] and matching is found between features of new frame and features of panorama by using Brute-Force Matching. RANSAC algorithm [25] is used to calculate homography by using these feature pairs. Then new frame is added to panorama according to the homography matrix. Finally, IAGMM [89] is applied to the panorama image to generate the foreground mask.

Robust real-time background subtraction on frames from the output of Pan-Tilt camera requires remarkable computational power. This computational power is provided by not only the CPU, but also GPU by using a heterogeneous programming model in this thesis.

Pan-Tilt camera background subtraction algorithms, which are based on only image information, are divided into two groups according to the usage of offline generated map. First group needs a scanning operation in the beginning but the second group does not need. Our algorithm is from second group, it does not require any prior knowledge. In other words, there is no need to scan all areas before operation mode.

In this study, feature points of the panorama are kept in a list instead of extracting them each time. Keeping a list provides more qualified feature points in shorter time.

The algorithm uses two methods in order to get more accurate homography matrix. First one is the checking the distribution of the feature points. Homography matrix is not created for the feature points that are not distributed sufficiently, because when feature points are close to each other, even small errors cause serious mistakes. The second method is controlling of generated homography matrices. Matrices are compared with reference homography matrix and the matrix, which has excessive difference, is eliminated because the changes in homography matrices must be little.

The whole process of the algorithm is implemented for both only CPU case and GPU-CPU hybrid case. Their performance and outputs are compared. Processing of one frame takes 624.06, 43.18 and 18.86 ms for CPU, GPU 1 and GPU 2, respectively. There are 14.45 speedup for GPU 1 and 33.09 speedup for GPU 2. Our study makes panoramic background subtraction to run in real-time with the help of the GPU.

Similarity between groundtruths and outputs of GPU - CPU hybrid case and only CPU case are measured as 0.73 and 0.71 respectively. Measurements are done by running the algorithm on many times and averaging the results. Both implementations of our study generate nearly the same result. GPU usage increases the speed of the algorithm without any remarkable performance loss.

In Pan-Tilt camera background subtraction algorithms, there is a possibility to classify the first seen part as a foreground because of the inadequate learning. In order to beat this weakness in our study, a counter is kept for each pixel and foreground detection is prevented until the counter reaches a threshold value.

Our algorithm is compared with Change Detection algorithms [93] by using their performance metrics. It produces impressive performance with high Re, Pr and Fm rates. On the other hand, FPR needs an improvement. Decreasing the misalignment errors of panorama generation will also decrease the FPR. As stated before, parts of our system are not coupled to each other. Each of them can be replaced with another algorithm. If our background subtraction part is changed with another algorithm such as the successful algorithm from Change Detection [93] that has higher performance, overall performance of our system will increase.

Our method is based on [76]. Similar operations are applied in same order with some differences. In order to extract features SURF [9] is preferred instead of SIFT [45] because SURF is more appropriate for parallelization. Furthermore, two extra methods are used to get more robust homography matrix and eliminate weak ones. These methods are checking distribution of feature points and controlling coherence of consecutive homography matrices. Our last contribution is preventing false alarms from first seen parts.

Results of our study and [76] are compared. Phyton implementation of [76] processes a single frame of the sequence with resolution 160 x 131 in 40 seconds. On the other hand, our GPU + CPU hybrid C++ implementation processes a single frame with 704 x 480 resolution in 33.09 ms. There is a huge speedup. Moreover, our implementation generates better FP and FN ratios because of the more robust homography matrices. One of the advantages of our study is that our algorithm adapts itself faster than [76] to change in the background of the scene. Secondly, [76] does not have any control for the first seen areas so it labels them as a foreground. However, our algorithm keeps a count in order to prevent false alarms from the first seen parts.

As a future work, zoom feature can be added to the system. By using interpolation, frames with different FOV can be processed. Another feature which will be useful if it is added to the system is detection of shadows. Now, shadows are classified as foreground. Detection of them will increase the performance by means of decreasing the rate of false positives.

REFERENCES

[1] T. Aach and A. Kaup, "Bayesian algorithms for adaptive change detection in image sequences using markov random fields", *Signal processing: Image Communication*, Vol. 7, pp. 147-160, 1995.

[2] Y. Abdel-Aziz and H. Karara, “Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry”, *Proceedings of the Symposium on Close Range Photogrammetry, American Society of Photogrammetry*, pp. 1-18, 1971.

[3] A. Al-Mazeed, M. Nixon and S. Gunn, “Classifiers Combination for Improved Motion Segmentation”, *ICIAR 2004*, pp. 363-371, 2004.

[4] M. Amintoosi, F. Farbiz, M. Fathy, M. Analoui and N. Mozayani, “QR decomposition-based algorithm for background subtraction”, *ICASSP 2007*, 2007.

[5] K. Appiah and A. Hunter, “Single-Chip FPGA Implementation of Realtime Adaptive Background Model”, *IEEE Conference on Field-Programmable Technology (FPT 2005)*, December 2005.

[6] P. Atrey, V. Kumar, A. Kumar nad M. Kankanhalli, “Experiential sampling based foreground/background segmentation for video surveillance”, *ICME 2006*, pp. 1809-1812, July 2006.

[7] D. H. Ballard, “Generalizing the Hough Transform to Detect Arbitrary Shapes”, *Pattern Recognition*, Vol. 13, No. 2, pp. 111-122, 1981.

[8] O. Barnich and M. Van Droogenbroeck, "ViBe: A Universal Background Subtraction Algorithm for Video Sequences", *IEEE Transactions on Image Processing*, Vol. 20, No. 6, pp. 1709-1724, June 2011.

[9] H. Bay, and T. Tuytelaars and L. Van Gool, “SURF: Speeded Up Robust Features”, *9th European Conference on Computer Vision*, 2006.

[10] H. Bhaskar, L. Mihaylova and S. Maskell, “Automatic Target Detection Based on Background Modeling Using Adaptive Cluster Density Estimation”, *3rd German Workshop on Sensor Data Fusion: Trends, Solutions, Applications*, September 2007.

[11] S. Bianco, G. Ciocca and R. Schettini, "How far can you get by combining change detection algorithms?", *Submitted to IEEE Transactions on Image Processing*, 2015.

[12] M. Brown and D. G. Lowe, "Recognising panoramas”, *International Conference on Computer Vision (ICCV 2003)*, pp. 1218-1225, October 2003.

[13] P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code", *IEEE Transactions on Communications*, Vol. 31, No. 4, pp. 532-540, April 1983.

[14] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features”, *11th European Conference on Computer Vision (ECCV)*, September 2010.

[15] Y. Chen, "A background subtraction algorithm for a pan-tilt camera", M.S. thesis, Dept. Comput. Sci., University Of Alberta, 2014.

[16] J. Cheng, J. Yang, Y. Zhou and Y. Cui, “Flexible background mixture models for foreground segmentation”, *Image and Vision Computing*, Vol. 24, pp. 473-482, 2006.

[17] M. Chouchene, F. E. Sayadi, Y. Said, M. Atri and R. Tourki, “Efficient implementation of Sobel edge detection algorithm on CPU, GPU and FPGA”, *Int. J. Advanced Media and Communication*, Vol. 5, No. 2/3, pp.105-117, 2014.

[18] M. Cristani and V. Murino, “A spatial sampling mechanism for effective background subtraction”, *VISAPP 2007*, Vol. 2, pp. 403-410, March 2007.

[19] D. D. Doyle, A. L. Jennings, J. T. Black, "Optical flow background subtraction for real-time PTZ camera object tracking," *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*, pp. 866-871, May 2013.

[20] E. Dubrofsky, “Homography Estimation”, M.S. thesis, Dept. Comput. Sci., The University Of British Columbia, 2009.

[21] M. M. El-Gayar, H. Soliman and N. Meky “A comparative study of image low level feature extraction algorithms”, *Egyptian Informatics Journal 14*, pp. 175-181, 2013.

[22] A. Elgammal, D. Harwood and L. Davis, “Non-parametric Model for Background Subtraction”, *ECCV 2000*, pp. 751-767, June 2000.

[23] X. Fang, W. Xiong, B. Hu B and L. Wang, “A Moving Object Detection Algorithm Based on Color Information, Journal of Physics, Vol. 48, pp. 384-387, 2006.

[24] A. Ferone, L. Maddalena, "Neural Background Subtraction for Pan-Tilt-Zoom Cameras", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 44, No. 5, pp. 571-579, May 2014.

[25] M. Fischler and R. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography”, *Communications of the ACM*, Vol. 24, No. 6, pp. 381-395, 1981.

[26] N. Friedman and S. Russell, “Image Segmentation in Video Sequences: A Probabilistic Approach”, Proceedings of the Thirteenth conference on Uncertainty in Artificial Intelligence, pp. 175-181, 1997.

[27] P. Guler, D. Emeksiz, M. Teke, A. Temizel and T. Taskaya Temizel, "Real-time Multi-Camera Video Analytics System on GPU", *Journal of Real-Time Image Processing*, March 2013.

[28] C. Harris and M. Stephens, “A combined corner and edge detector”, *Proceedings of the Alvey Vision Conference*, pp. 147-151, 1988.

[29] R. Hartley and A. Zisserman, *“Multiple View Geometry in Computer Vision”*, Cambridge University Press, 2004.

[30] O. N. Ivanov, “Adaptation of Known Background Subtraction Methods in the Case of a Moving PTZ Camera Mounted on a Mobile Platform”, *Pattern Recognition and Image Analysis*, Vol. 24, No. 2, pp. 318-323, June 2014.

[31] P. Jaikumar, A. Singh and S. Mitra, “Background Subtraction in Videos using Bayesian Learning with Motion Information”, *BMVC 2008*, pp. 615-624, September 2008.

[32] V. Jain, B. Kimia and J. Mundy, “Background modelling based on subpixel edges”, *ICIP 2007*, Vol. 6, pp. 321-324, September 2007.

[33] O. Javed, K. Shafique and M. Shah, “A Hierarchical Approach to Robust Background Subtraction using Color and Gradient Information”, *WMVC 2002*, pp. 22, December 2002.

[34] Y. Jia, J. Wang, G. Zeng, H. Zha, and X. S. Hua, “Optimizing kdtrees for scalable visual descriptor indexing,” *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 3392-3399, 2010.

[35] H. Jiang, H. Ardo and V. Owall, “Hardware accelerator design for video segmentation with multi-modal background modeling”, *ISCAS 2005*, Vol. 2, pp. 1142-1145, May 2005.

[36] E. Kepucka, I. Gurcan and A. Temizel, "Fast Omnidirectional Image Unwrapping on GPU", *Euromicro International Conference on Parallel, Distributed and Network-Based Computing (WIP)*, February 2012.

[37] K. Kim, T. Chalidabhongse, D. Harwood and L. Davis, "Background modeling and subtraction by codebook construction" *International Conference on Image Processing*, Vol. 5, pp. 3061-3064, October 2004.

[38] D. Lee, “Online Adaptive Gaussian Mixture Learning for Video Applications”, *ECCV Workshop on Statistical Methods for Video Processing*, May 2004.

[39] A. Lepisk, “The use of Optic Flow within Background Subtraction”, M.S. thesis, Royal Institute of Technology, 2005.

[40] Y. Liang, Z. Wang, X. Xu and X. Cao, “Background Pixel Classification for Motion Segmentation using Mean Shift Algorithm”, *ICMLC 2007*, pp. 1693-1698, 2007.

[41] C. Lien, C. Hua, Y. Jiang and L. Jang, “Large Area Video Surveillance System with Handoff Scheme among Multiple Cameras”, *MVA 2009*, May 2009.

[42] H. Lin, T. Liu and J. Chuang, "A probabilistic SVM approach for background scene initialization", *ICIP 2002*, Vol. 3, pp. 893-896, September 2002.

[43] J. Lindstrom, F. Lindgren, K. Ltrstrom, J. Holst and U. Holst, “Background and Foreground Modeling Using an Online EM Algorithm”, *ECCV 2006*, May 2006.

[44] N. Liu, H. Wu, and L. Lin, "Hierarchical Ensemble of Background Models for PTZ-Based Video Surveillance", *IEEE Transactions on Cybernetics*, Vol. 45, No.1, pp. 89-102, January 2015.

[45] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91-110, 2004.

[46] L. Maddalena and A. Petrosino, “A self organizing approach to background subtraction for visual surveillance applications”, *IEEE Transactions on Image Processing*, Vol. 17, No. 7, pp 1729-1736, 2008.

[47] V. Morellas, L. Pavlidis P. Tsiamyrtzis, “DETER: detection of events for threat evaluation and recognition”, *Machine Vision and Applications*, Vol. 15, pp. 29-45, June 2003.

[48] E. Monari and T. Pollok, "A Real-Time Image-to-Panorama Registration Approach for Background Subtraction Using Pan-Tilt-Cameras", *8th Intern. Conf. AVSS*, pp. 237-242, 2011.

[49] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration”, *International Conference on Computer Vision Theory and Application VISSAPP’09*, pp. 331-340, 2009.

[50] M. Muja, D. G. Lowe, "Fast Matching of Binary Features," *2012 Ninth Conference on Computer and Robot Vision (CRV)*, pp. 404-410, May 2012.

[51] L. Niu and N. Jiang, “Moving Objects Detection Algorithm Based on Improved Background Subtraction”, *ISDA 2008*, Vol. 03, pp. 604-607, 2008.

[52] T. T. Nguyen and J. W. Jeon, “Real-Time Background Compensation for PTZ Cameras Using GPU Accelerated and Range-Limited Genetic Algorithm Search” in Advances in Image and Video Technology, Vol. 7087, Y. S. Ho, Berlin, Springer Berlin Heidelberg, 2011.

[53] P.M. Panchal, S.R. Panchal and S.K. Shah, “A Comparison of SIFT and SURF”, *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 1, No. 2, April 2013.

[54] J. Park, A. Tabb and A. Kak, “Hierarchical Data Structure for Real Time Background Subtraction”, *ICIP 2006*, pp. 1849-1852, October 2006.

[55] D. Parks and S. Fels, “Evaluation of Background Subtraction Algorithms with Post-processing”, *AVSS 2008*, September 2008.

[56] D. Pokrajac and L. Latecki, “Spatiotemporal Blocks-Based Moving Objects Identification and Tracking”, *VS-PETS 2003*, pp. 70-77, October 2003.

[57] T. Porter and T. Duff, “Compositing digital images”, *Acm Siggraph Computer Graphics*, Vol. 18, No. 3, pp. 253-259, July 1984.

[58] Z. Qu, M. Yu and J. Liu, “Real-time traffic vehicle tracking based on improved MoG background extraction and motion segmentation”, *ISSCAA 2010*, pp. 676-680, June 2010.

[59] C. Ridder, O. Munkelt and H. Kirchner, “Adaptive background estimation and foreground detection using kalman-filtering”, *Proceedings of International Conference on recent Advances in Mechatronics*, pp. 193-199, 1995.

[60] R. Rodriguez, “Background Subtraction with PTZ Cameras”, M.S. thesis, Dept. Elect. and Comput. Sci., Technical University of Berlin, 2012.

[61] E. Rosten, R. Porter and T. Drummond, “Faster and better: a machine learning approach to corner detection”, *IEEE Transaction Pattern Analysis and Machine Intelligence*, Vol 32, pp. 105-119, 2010.

[62] P. J. Rousseeuw, “Least median of squares regression”, *Journal of the American Statistical Association*, Vol. 79, No. 388, pp. 871-880, March 2012.

[63] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *IEEE International Conference on Computer Vision (ICCV)*, pp. 2564-2571, Novemberr 2011.

[64] H. Sajid and S. Cheung, "Universal Multimode Background Subtraction", *Submitted to IEEE Transactions on Image Processing*, 2015.

[65] K. Schindler and H. Wang, “Smooth Foreground-Background Segmentation for Video Processing”, *ACCV 2006*, Vol. 3852, pp. 581-590, January 2006.

[66] Z. Sheng and X. Cui, “An adaptive learning rate GMM for background extraction”, *Optoelectronics Letters*, Vol. 4, No. 6, pp. 460-463, November 2008.

[67] H. Shi and C. Tomassi, “Good Features to Track”, *9th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593-600, 1994.

[68] A. Shimada, D. Arita and R. Taniguchi. “Dynamic Control of Adaptive Mixture-of Gaussians Background Model”, *AVSS 2006*, pp. 5, November 2006.

[69] P. L. St-Charles, G. A. Bilodeau and R. Bergevin, R, "SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity", *IEEE Transactions on Image Processing*, Vol. 24, No.1, pp. 359-373, January 2015.

[70] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 246-252, 1999.

[71] G. Stijnman and R. Van den Boomgaard, "Background estimation in video sequences", Technical Report 10, Intelligent Sensory Information Systems Group, University of Amsterdam, January 2000.

[72] Y. Sun, "Better Foreground Segmentation for Static Cameras via New Energy Form and Dynamic Graph-cut", *ICPR 2006*, 2006.

[73] A. Temizel, T. Halici, B. Logoglu, T. Taskaya Temizel, F. Omruuzun and E. Karaman, "Experiences on Image and Video Processing with CUDA and OpenCL" in *NVIDIA GPU Computing Gems*, Vol. 1, W. Hwu, Elsevier, 2011.

[74] T. Terriberry, L. French, and J. Helmsen, "GPU accelerating speeded-up robust features", *4th International Symposium on 3D Data Processing, Visualization and Transmission*, 2008, pp. 1–8.

[75] Y. Tian and A. Hampapur, "Robust Salient Motion Detection with Complex Background for Real-time Video Surveillance", *CVPR 2005*, Vol. 2, pp. 30-35, January 2005.

[76] E. Tsinko, "Background Subtraction with a Pan/Tilt Camera", M.S. thesis, Dept. Comput. Sci., The University Of British Columbia, 2010.

[77] D. Turdu and H. Erdogan, "Improved post-processing for GMM based adaptive background modeling", *ISCIS 2007*, pp. 1-6, November 2007.

[78] R. Wang, F. Bunyak, G. Seetharaman, K. Palaniappan, "Static and Moving Object Detection Using Flux Tensor with Split Gaussian Models", *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference*, pp. 420,424, June 2014.

[79] B. White and M. Shah “Automatically Tuning Background Subtraction Parameters Using Particle Swarm Optimization”, *ICME 2007*, pp. 1826-1829, 2007.

[80] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, “Pfinder: Real-Time Tracking of the Human Body”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 780-785, July1997.

[81] M. Xu and T. Ellis, “Illumination-invariant motion detection using color mixture models, *BMVA 2001*, pp. 163-172, September 2001.

[82] K. Xue, Y. Liu, G. Ogunmakin, J. Chen, J. Zhang, “Panoramic Gaussian Mixture Model and large-scale range background substraction method for PTZ camera-based surveillance systems”, *Machine Vision and Applications*, Vol. 24, No. 3, pp. 477-492, April 2013.

[83] S. Xuehua, C. Yu, G. Jianfeng and C. Jingzhu, “A Robust Moving Objects Detection Algorithm Based on Gaussian Mixture Model”, *ITSC 2009,* Vol. 1, pp. 566-569, 2009.

[84] Q. Yan, Y. Xu, X. Yang nad L. Traversoni, “Real-Time Foreground Detection Based on Tempo-Spatial Consistency Validation and Gaussian Mixture Model”, *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB 2010)*, pp. 1-4, March 2010.

[85] Q. Zang and R. Klette, “Evaluation of an Adaptive Composite Gaussian Model in Video Surveillance”, CITR Technical Report 114, Auckland University, August 2002.

[86] H. Zheng, Z. Liu and X. Wang, “Video Segmentation Method with Integrated Multi-features Based on GMM”, *International Conference on Digital Image Processing (DIP 2009)*, pp. 62-66, March 2009.

[87] J. Zhang, Y. Wang, J. Chen and K. Xue, "A framework of surveillance system using a PTZ camera", *IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, Vol. 1, pp. 658-662, July 2010.

[88] D. Zhou and H. Zhang, "Modified GMM background modeling and optical flow for detection of moving objects", *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2224-2229, October 2005.

[89] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction", *International Conference Pattern Recognition, Vol. 2, pp. 28-31, 2004.*

[90] X. Zou, X. Zhao, Z. Chi, "A robust background subtraction approach with a moving camera", *Computing and Convergence Technology (ICCCT), 2012 7th International Conference*, pp. 1026-1029, December 2012.

[91] Egor Tsinko's Thesis, [Online], Available: http://www.cs.ubc.ca/nest/lci/thesis/etsinko/, [Accessed: Sep. 01, 2015].

[92] FLANN - Fast Library for Approximate Nearest Neighbors: FLANN, [Online], Available: http://www.cs.ubc.ca/research/flann/, [Accessed: Sep. 01, 2015].

[93] IEEE Change Detection Workshop, [Online], Available: http://changedetection.net, [Accessed: Sep. 01, 2015].

[94] Nvidia Corporation, "CUDA C Programming Guide v7.0", [Online], Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide, [Accessed: Sep. 01, 2015].

[95] Nvidia Corporation, "GeForce GTX 980 Whitepaper", [Online], Available: http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce_GTX_980_Whitepaper_FINAL.PDF, [Accessed: Sep. 01, 2015].

[96] Nvidia Corporation, "Precision And Performance: Floating Point And Ieee 754 Compliance for Nvidia Gpus - Nvidia White Paper", [Online], Available: http://docs.nvidia.com/cuda/floating-point/index.html, [Accessed: Sep. 01, 2015].

[97] OpenCL - The open standard for parallel programming of heterogeneous systems, [Online], Available: https://www.khronos.org/opencl/, [Accessed: Sep. 01, 2015].

[98] OpenCV, [Online], Available: http://opencv.org/, [Accessed: Sep. 01, 2015].

[99] Parallel Programming and Computing Platform, [Online], Available: http://www.nvidia.com/object/cuda_home_new.html, [Accessed: Sep. 01, 2015].

[100] Visual Computing Leadership from NVIDIA, [Online], Available: http://www.nvidia.com/page/home.html, [Accessed: Sep. 01, 2015].