UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Anton Prokopov

Moving object detection and tracking using dynamic background and foreground separation for the purpose of traffic analysis on mobile devices

Bachelor's Thesis (9 ECTS)

Supervisor: Priit Salumaa, MSc
Supervisor: Assoc. Prof. Gholamreza Anbarjafari

Tartu 2016

Liikuvate objektide tuvastamine ja jälgimine kasutades dünaamilise esiplaani ja tagaplaani eraldamist liikluse analüüsimiseks mobiilsete seadmetega

Lühikokkuvõte:

See töö kirjeldab automaatset esiplaani-tagaplaani tuvastamist edasise liikluse analüüsimise eesmärgil. Selle projekti põhieesmärgiks on tuvastada ja rakendada tagaplaani eemaldamise algoritmi, mis tugineb Gaussi segumudelil selleks, et sooritada robustset tagaplaani lahutamist reaalajas. Taoline süsteem on võimeline ära tundma liikuvaid objekte videojadas kasutades ainult tavalist kaamerat. Rakendus oli esialgu teostatud arvutil C++ OpenCV teegi abil. Järgnevalt algoritmi kood oli tõlgitud selleks, et see oleks käivitatav Android seadmel.

Võtmesõnad:

Gaussian Mixture Model, tagaplaani lahutamine, esiplaani segmentatsioon, Android NDK, OpenCV, kämpide avastamine, liikluse analüüs

CERCS: Pilditehnika (T111)

Moving object detection and tracking using dynamic background and foreground separation for the purpose of traffic analysis on mobile devices

Abstract:

This work describes an automatic background-foreground detection for the purpose of further analysing of traffic. The aim of this project was to investigate and implement the background removal algorithm that uses Gaussian Mixture Model to perform robust background subtraction in real time. This kind of a system can recognize the moving objects in the video sequence using only camera. It finds its application in traffic analysis. The implementation has been done firstly on a computer using C++ OpenCV library followed by translating the code for being executable on Android device.

Keywords:

Gaussian Mixture Model, Background subtraction, Foreground segmentation, Android NDK, OpenCV, Blob Detection, traffic analysis

CERCS: Imaging, image processing (T111)

Contents

Chapter 1

Introduction

Due to various events, police or municipality may need to divert the traffic and while doing that they need to monitor the traffic [KMIS00], so they can provide citizens a better service. In this thesis a computer vision based automatic traffic-analysis system is proposed. Video streams of the road will be captured via mobile devices and then by counting number of moving vehicles in specific duration of time, some statue for the level of traffic will be assigned. In this work, drones are being used in order to obtain video streams from the road. This thesis is dealing with several important challenges such as dynamic background registration, shadow removal, denoising and moving object detection. The proposed system will work in real-time scenarios, hence there is a challenge of optimizing the algorithms in a way that it will be runnable in real-time.

In order to make a clear understanding of the proposed thesis work, it is required to overview some fundamental definitions. The proposed algorithm requires to detect any changes caused by moving objects. The problem is that movement in objects such as trees and bushes which are not desired, exist, which means the background isn't static all the time. This is why the system has to implement algorithm allowing to determine, which movements are caused by actually moving objects we are interested in and which movements are caused by background objects. All the static areas that do not change over the time jointly with the areas, which are recognized as dynamic background using probabilistic model, form the background layer. All the rest is defined as foreground. Performing this kind of separation makes it possible to have actually moving objects and the background separately in different layouts. In order to accomplish our aims the following tasks need to be conducted properly:

1. Drone-based traffic analysis means a camera is attached to a flying over streets drone. Thus, we need an algorithm that determines, what is an actually moving object and what are the changes caused by the camera movement.

2. Traffic analysis, obviously, can be performed only outside, which means we need to consider weather changes as an influencing factor. For example, rain can cause additional noise, wind may perform movement in the objects such as trees and bushes that we would like to recognize as background.

3. Light conditions might as well affect object detection mechanism, so we need to take that into account too.

4. Real-time detection means high computational abilities. In our case it means the algorithm needs to be very well optimized as it is supposed to be run on a raspberry pi or an Android device attached to the drone. These devices have their hardware limitations, so the algorithm has to correspond to them.

The availability of all the aforementioned challenges makes this work interesting both in theory and application. Current image processing tools have proven it is possible to perform object detection and tracking precisely. For example, determining, what object is moving between video frames is possible by simple background subtraction, which can be enhanced by applying probabilistic models, such as Bayesian rule, to better recognize, what is foreground, what is background.

Illumination changes can be solved by applying illumination enhancement techniques such as contrast stretching and singular value equalization, noise caused by rain or wind can be eliminated using blurring followed by applying morphological operations such as opening. Shadow removal can be performed after modelling the background and applying morphological operations using thresholding.

Although within this thesis we are aiming to find a solution for traffic analysis of the road, but the developed algorithm can be used for any other purpose, in which dynamic background and foreground separation is the main challenge.

Chapter 2

Background

This chapter explains, how the established main goal can be achieved. It gives an overview of existing image processing techniques and algorithms suitable for detecting moving objects. Also some technical details are explained in here.

2.1 Background and foreground separation techniques

Background and foreground separation refers to the process of detecting the parts of input image that belong to the foreground and the background and separating them to different layers, so they can be accessed separately later [HJLB98, YYS+15, SCP+15]. There are various methods that allow to perform background or foreground detection, which are two sides of the same problem. Solving one of them will result in solution of another. In paper [QWX11] some of the basic methods are described. Here is the list of them:

1. Multi-Frame average method [WJW04] - takes average from the frame sequence. To store this sequence it requires memory. In addition, a threshold value must be defined and adjusted depending on the degree of the environmental change.

2. Selection method [LYT+03][RC78] - sudden change in a pixel means this pixel belongs to foreground. Also requires setting up a threshold, which is hard to determine. Threshold has the impact, whether the background will be sensitive or insensitive to changes.

3. Selection-Average method [SFD94] - combination of the previous two, which is meant to fix the disadvantages of one another.

4. Kalman-Filter based adaptive background update method [RMK95] - pixel-wise estimates the background and adapts to catch the light and weather changes, so if any noise appears it is considered moving objects. Requires much calculation time.

5. Another adaptive background update method can be found in [MMP05] - estimates the background using current frame and a binary mask of foreground detected so far.

6. Frame differencing [JS04] - compares current frame with the previous one, assuming the background stays the same. If any changes appear, they are caused by moving objects

7. Optical flow method [GTCS+01] - where traditional background subtraction algorithm fails, optical flow performs well, for example in crowded places. Traditional subtraction is faster and simpler, but depends on the availability of the background image.

All of the aforementioned algorithms without improvements are meant for static background, which means there has to be no camera movement, while obtaining the real-time video sequence. The goal is to make an algorithm that is able to work with dynamic background, therefore, those methods are not suitable.

Another solution for background detection is introduced in [ZYY13]. Authors call their method DECOLOR - Detecting Contiguous Outliers in the Low-Rank Representation. DECOLOR is not dependant of the static background as well as it does not need any training sequences. It avoids complicated motion computations because of the outlier detection and low-rank modeling. It is assumed that the background images are in linear correlation. "Thus, the matrix composed of vectorized video frames can be approximated by a low-rank matrix, and the moving objects can be detected as outliers in this low-rank representation" [citation from 3]. Object detection and background estimation are performed simultaneously. Objects are segmented based on motion information. Motion segmentation is the process of decomposing the image into motion layers with an assumption that the optical-flow field is smooth in every layer. Alternative approach is analyzing point trajectories. Background subtraction is used in this method, when training sequence doesn't contain foreground objects and the camera is static. DECOLOR works poorly, if the camera moves a lot or if the background is 3D. As stated in [ZYY13] the DECOLOR is not suitable for real-time object detection.

One more way for foreground-background segmentation is using the codebook model [KCHD05]. Background model in case of this algorithm is constructed in a way that each pixel of the background is encoded using one or more codewords. This allows creating compact background model with continuous updates, so it can fit for stationary and dynamic backgrounds. Algorithm makes a decision whether the pixel belongs to the background or the foreground depending on pixel's color

distortion and brightness. If distortion is less than the detection threshold and brightness lies within the brightness of the codeword, then the pixel belongs to background, otherwise - foreground. Despite the fact that the algorithm is fast and cheap, it has undesirable disadvantages:

1. It needs training.

2. Optimizations proposed in [KCHD05] are required to achieve a competitive level.

3. It works good with low-quality videos.

In [SJK09] the next approach can be found. To differentiate between objects in motion and objects at rest geometric constraints can be used at sparse locations. These sparse locations can then be used to build foreground and background appearance models, which basically means segmenting the background and foreground. RANSAC in this algorithm is used for estimating the trajectories of the objects presented in the scene. If the object's actual trajectory is different than the estimated one, it means the moving object is detected. Unfortunately, tracking first and separating after approach requires high computational power, so it cannot be suitable as well, even though the recognition is highly accurate.

2.2 General Gaussian Mixture Model (GMM)

Mixture of Gaussians (MOG) or known as Gaussian mixture model (GMM) is a background modeling method that detects moving objects from static cameras. In this section the basic MOG algorithm is described, so the principle is understandable. In the context of traffic analysis in [FR97] every background pixel is proposed to be represented using mixture of three Gaussians corresponding to road, vehicle and shadows. First step is to initialize the model using EM algorithm [DLR77]. Then manual labeling of the Gaussians is performed as follows: the darkest component is always a shadow, in remaining two components the one with largest variance is vehicle and the other is road. This is fixed for all the process, which causes lack of adaptation to changes over time. Foreground detection is performed in a way that every pixel is compared with each Gaussian and is classified accordingly. For real-time consideration method is maintained using incremental EM algorithm. Described logic of the MOG algorithm is generalized in [SG99] by modeling the recent history of the color features of each pixel $\{X_1, ..., X_t\}$ by a mixture of K Gaussians.

2.2.1 Principle

As described in [BEBV08], each pixel should be characterized by its intensity in the RGB color space. Then current pixel's probability is calculated using following formula:

$$P(X_t) =_{i=1}^{K} w_{i,t} h(X_t, m_{i,t}, \sigma_{i,t}), \tag{2.1}$$

where K is the number of distributions, $w_{i,t}$ is a weight associated to the i^{th} Gaussian at time t with mean $m_{i,t}$ and standard deviation $\sigma_{i,t}$. h is a Gaussian probability density function:

$$h(X_t, m, \sigma) = \frac{1}{(2p)^{n/2}|\sigma|^{1/2}} e^{-\frac{1}{2}(X_t - m)^{-1}(X_t - m)}, \tag{2.2}$$

RGB color components are assumed independent, so they have the same variances. This way covariance matrix is of the form:

$$\sigma_{i,t} = s_{i,t}^2 I \tag{2.3}$$

So, mixture of K Gaussians characterizes every pixel. After defining the background model, different MOG parameters, such as number of Gaussians K, the weight $w_{i,t}$ associated to the i^{th} Gaussian at time t, the mean $m_{i,t}$ and the covariance matrix $\sigma_{i,t}$ must be initialized. Once the parameters are set, a first foreground detection can be made, after that parameters are updated. When new frame incomes at times $t+1$, a match test is made for each pixel. A pixel matches Gaussian distribution, if the Mahalanobis distance is:

$$\sqrt{(X_{t+1} - m_{i,t})^T * \sigma_{i,t}^{-1} * (X_{t+1} - m_{i,t})} < k s_{i,t}, \tag{2.4}$$

where k is a constant threshold 2.5. Then, two cases are possible: a match is found or match is not found. In case 1, if the Gaussian distribution is identified as background, the pixel is considered background as well, otherwise the pixel is classified as foreground. In case 2 pixel is always classified as foreground. Now the background mask is obtained. Updating parameters described in [BEBV08] is required for the next foreground detection.

2.3 GMM alternatives

tocsectionGMM alternatives

Adaptive background estimation and foreground detection can be performed using Kalman-Filtering, as stated in [RMK95]. However, this approach has some critical limitations, such as it cannot assume continuously moving foreground objects and can be applied only to grayscale video input with static background. Also it requires significantly more computational power than GMM. Paper [LHGT04] proposes a method for static modeling of complex backgrounds. This one could be

applied for both static and dynamic backgrounds and performs even with better accuracy than GMM. High accuracy is achieved by operating in all three domains: spectral, spatial and temporal. Spectral features to model the background, spatial features to be robust to illumination changes, temporal features to describe dynamic background pixels. Foreground object detection is performed following these steps:

1. change detection - filter out pixels that do not change (background) using simple image differencing,
2. change classification - change that occurs at a static or a dynamic point is classified as background or foreground using Bayes decision rule and the statistics of the corresponding principal features,
3. foreground object segmentation - by post processing on a segment remaining change points are combined into foreground regions (a pair of opening and closing, filling the holes, deleting small regions, applying AND operation to remove false foreground regions),
4. background maintenance - update the background using the feedback from the previous segmentation.

Disadvantages of this method are:

1. If a moving object stops and stays still for a long time, it becomes a background.
2. Features of foreground objects can be determined wrongly, if the scene is crowded.
3. Memory requirement for each pixel is 1.78 KB. If video is of the size 160x120 pixels, then maximum rate is 15 fps. This is not real-time sampling.

Another paper about foreground object detection from videos with complex backgrounds is proposed in [LHGT03]. As the previous one, this method allows extracting foreground from complex background, which contains stationary and moving objects, includes learning strategies for sudden "once-off" changes and makes decision whether the pixel belongs to background or foreground using Bayes decision rule. To model the complex background two types of features are employed: stationary parts are described by statistics of most significant colors, motion parts are described by most significant color cooccurences. This method uses general Bayesian framework, which allows integrating more than one feature, unlike almost all existing methods that use only one type of feature (color or optical flow) to model the background. The algorithm steps are the same, as in previous method. It also performs with higher accuracy than the GMM, however it has all the same disadvantages, as the previous method.

2.4 Learning algorithms

Besides the naive approaches, such as frame differencing, and statistical approaches, such as GMM, there are also algorithms for detecting foreground objects using learning mechanisms. In [LXG+13] the integration of statistical background-foreground extraction and SVM classifier for pedestrian detection and tracking is explained. In this case background-foreground extraction algorithm is used to assist training and detection phases of SVM classifier that is based on the Histogram of Oriented Gradients (HOG) feature [DT05]. For background representation GMM is used. Tracking is realised by using Camshift tracker and Kalman-Filter. Foreground extraction speed for a frame of 704 576 is about 140 ms, background learning speed for the same frame is 120 ms, tracking is about 20 ms per pedestrian. This makes it possible to be used as a real-time algorithm. However, every learning algorithm requires training before it can be used. This is the reason adjusting GMM was preferred over the learning.

Chapter 3

Methodology

In order to achieve such a high level goal it requires use of combination of various image processing techniques, so as a result the final goal can be accomplished. *Figure 3.1* illustrates a flowchart of the implemented algorithm. Firstly it is required to obtain a real-time video sequence streams acquired by the camera. After the video is acquired the processing can be started. The proposed algorithm is applied to each frame of the video sequence. The details of the proposed algorithm is given below.

3.1 Gaussian Mixture Model. Shadows

To separate input frame into 2 layers, namely, background and foreground, the enhanced Gaussian mixture model (GMM) method proposed by Zivcovic [Ziv04] is applied. It is a pixel-wise operation, which means the probabilistic estimation for determining, whether the pixel belongs to the moving part of the image (foreground) or to the static one (background), is build for every pixel. Foreground layer is built from pixels that are considered moving objects presented in the scene. All the other pixels that are recognized as static pixels, i.e. no significant change was detected in them, form the background. As a result there are 2 layers: one contains pixels that most likely belong to the foreground and the other contains pixels that most likely belong to the background. This kind of a separation allows performing further processing on the foreground separately, without including pixels which are in the background cluster. Output of the GMM is a binary mask, where 0's are the background and 1's are the foreground as shown in *Figure 3.2*. Probabilistic approach allows to discard sudden changes, because they are not persistent in the scene, thus the probability that it is an actually moving object is very low.

One of the problems, while separating the foreground from the background, is risen to the fact that objects in real world have shadows. When object is moving, its

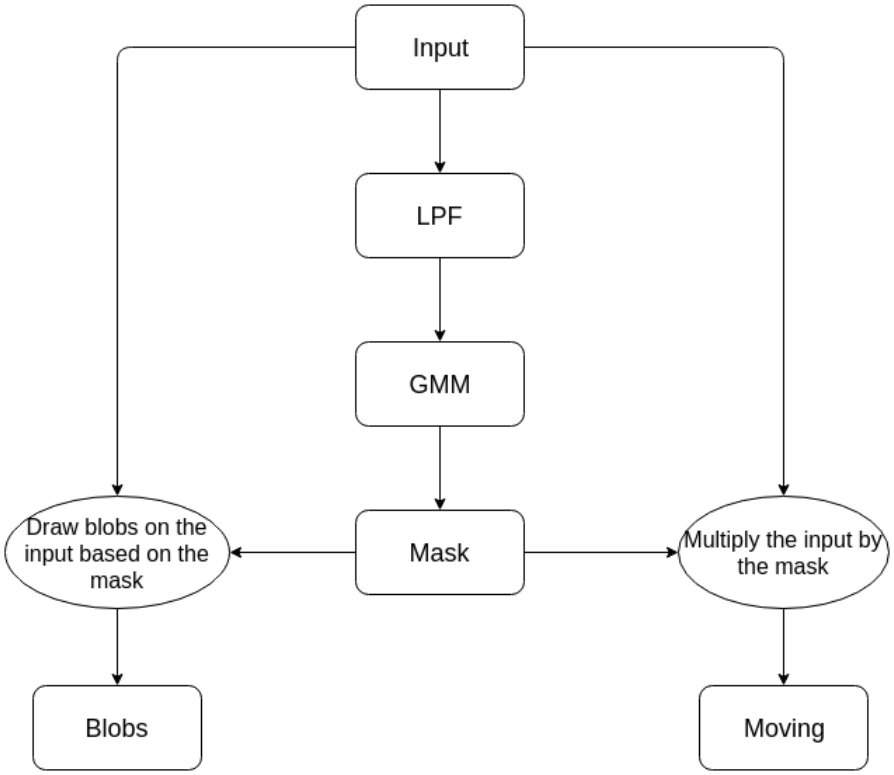


Figure 3.1: Flow-chart of the algorithm. Each frame of the video sequence is an input. After input is received, low-pass filter (LPF) is applied. Next step is to separate forefround from background using Gaussian Mixture Model (GMM). Result of GMM is a binary mask that is used for displaying only moving parts and drawing blobs around moving objects. Further details on every step can be found below

shadow is also moving causing significant changes of illumination of background in the scene that might be falsely considered as foreground. However algorithm is meant to detect moving cars, not their shadows. This is the reason shadows are considered as noise. GMM allows the algorithm to detect shadows and returns mask containing 3 layers: background, foreground and detected shadows, as described earlier in [inner ref to GMM section from background]. Shadow is represented as a gray layer on a mask as shown in *Figure 3.3*. By manipulating GMM parameters it is really simple to discard all the detected shadows stored in the gray layer, so eventually the binary mask containing only foreground and background is obtained.

Not only shadows can cause undesirable changes in the scene. As all the retrieved video sequences are taken outdoors, weather conditions have significant impact on the captured scene. Changes caused by weather is also noise signals for the the proposed algorithm and this is important to note that all the noise has to be considered as part of the background. Examples of noise can be undesired events such as rain, snow, movement in the trees and bushes caused by wind,

Figure 3.2: This figure illustrates, how the output of the GMM algorithm looks like on Android device. It is a black-and-white mask, where black is background and white is foreground containing moving objects. Objects have such smooth borders, because low-pass filter was applied to the image before GMM. Details are in the next section

Figure 3.3: On the left side original image is presented. On the right side the output of the GMM - binary mask is displayed. It has one more layer meant for shadows - gray areas around the objects. Those are detected shadows

light changes during the day. All of those changes have to be discarded for more accurate recognition. Some other moving objects might be present in the scene, such as birds, dogs, cats, pedestrians and so on. As the main objective of this algorithm is to detect cars, all the smaller objects such as humans or animals are undesirable for us and have to be disregarded.

3.2 Gaussian low-pass filter

One of the ways of reducing the noise causing small changes is blurring the image. In the proposed algorithm Gaussian blur is used before applying GMM. When the image is blurred, small changes will be averaged, i.e. the changes will be eliminated. Also borders between the objects become smooth or even get lost, depending on the window size used for blurring that is applied. As a result, only objects with significant changes will be still recognized as moving parts. *Figure 3.5* shows the impact of the blurring on the original image. The challenge is to define the meaning of significant in the aforementioned description. This depends on the size of the matrix that is applied for blurring. Bigger window size of blurring matrix will result into disregarding the bigger moving objects in estimation of background and foreground. However, with limited computational power, matrixes of really big size cannot be applied due to its higher computational complexity. Blurring is performed pixel-wise and requires calculation of new values for every other pixel in the region of applied mask. So the bigger the mask, the bigger the amount of pixels to calculate new values for on every step of blurring. That is why the particular size of the matrix have to be found to boost both recognition rate and the calculation speed. To resolve this problem the size parameter was set to be changeable by the user using SeekBar in the Android application, as illustrated in *Figure 3.5*.

Figure 3.4: Gaussian low-pass filter applied to the input image before further processing for the purpose of noise reduction

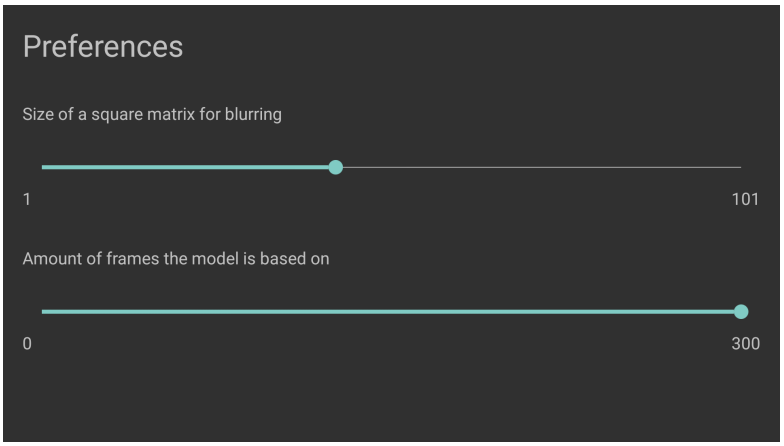


Figure 3.5: Preferences screen on Android. First SeekBar customizes the window size for blurring, allowing it to be from 1x1 to 101x101 matrix. Second SeekBar is meant for adjusting the amount of frames the GMM is based on. It is allowed to make probabilistic decisions using history of frames from 0 to 300

3.3 Binary mask. Opening

Output of the GMM method, as stated before, is a black-and-white image, i.e. it is a binary mask. It is clear the noise presented in the scene cannot be all eliminated only by using Gaussian low-pass filter. Blurring only reduces the noise. For better recognition some morphological operations such as "opening" are applied. Opening is an operation of applying dilation right after erosion. Closing is an operation of erosion applied right after dilation. When matrix A is eroded by matrix B, the minimal pixel value overlapped by B is computed and the pixel under the anchor point with that minimal value is replaced on a the matrix A. An example of erosion is illustrated in *Figure 3.6*. Dilation is the opposite of erosion. Not the minimal, but the maximum value overlapped by matrix B on matrix A is computed and the pixel under the anchor point is changed accordingly. Similarly an example of dilation is shown in *Figure 3.7*. Result of opening is shown in *Figure 3.8*.

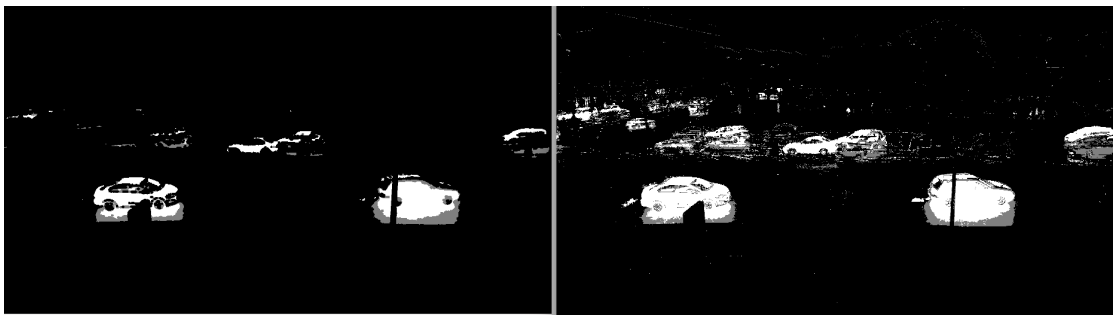


Figure 3.6: Right image shows the mask obtained by applying GMM, left image is the result of the erosion on this mask

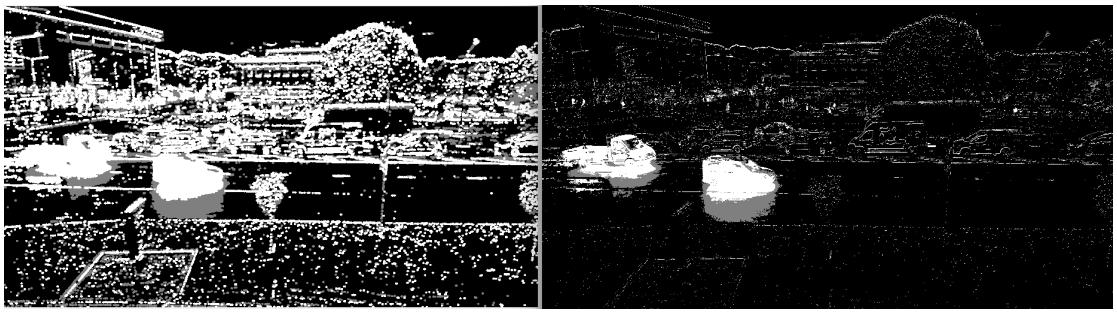


Figure 3.7: Right image shows the mask obtained by applying GMM, left image is the result of the dilation on this mask

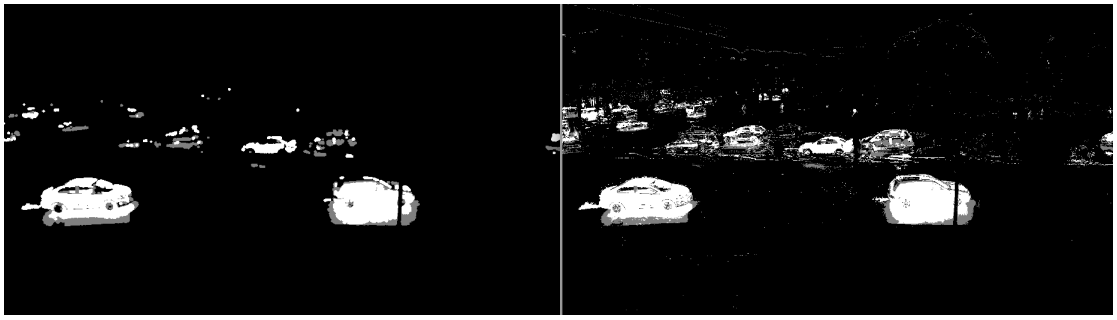


Figure 3.8: Final result of erosion followed by dilation is shown on the left image, right image is a mask obtained by GMM

3.4 Moving part

After erosion and dilation of the mask obtained by GMM algorithm, the detected foreground and background are ready for further process. At this stage by multiplying the binary mask with the input frame element-wise an image that contains only those parts of the original image, where the moving objects were detected, will be obtained. When multiplying an image by binary mask, all the pixels of original image that are multiplied by 0 (black color) are replaced by black pixels and all the pixels that are multiplied by 1 (white) just stay the same. This is why only moving objects can be displayed with a black background as shown in *Figure 3.9*.

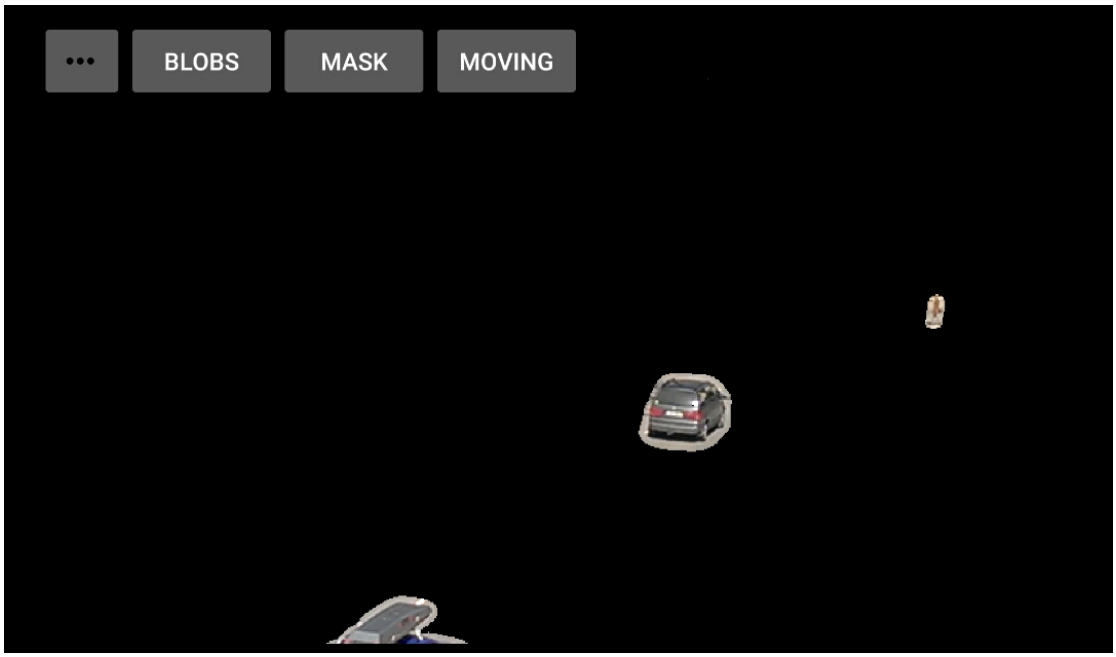


Figure 3.9: Result of multiplication of the original image by GMM mask, displaying only moving objects in the scene, is presented in this figure (Android device screen)

3.5 Blobs

The main output of the proposed algorithm is an image with highlighted moving objects, which in this thesis a rectangle has been drawn around the car. Such a highlighting is usually referred as blobs. Blob is simply a highlighted area of the original image. To return an image with highlighted moving objects 2 inputs are required, namely, original image, on which blobs will be drawn and the binary mask containing separated foreground and background. Firstly the contours of moving objects are extracted and presented in the scene. By using OpenCV library function *findContours* [fin], which takes binary image as an input, it is possible to have an array of vectors representing the contours of every moving object found in the scene as a result. When contours are obtained and stored in a convenient format, further step is required to calculate the smallest bounding rectangles, in which contours of every single object can fit. After that rectangles can be simply drawn on the original frame. Original frame with rectangles drawn around every detected moving object is returned as a result. This is shown in *Figure 5.2.*

Figure 3.10: Original image with blobs drawn around moving objects processed on Android device

Chapter 4

Technologies

Building a prototype requires a specific set of technologies to be used. In this chapter used technologies are described. The explanation, why this specific technology is chosen is also provided.

4.1 Software

4.1.1 Android. Java

Android is selected as a platform for the proposed algorithm to be run. It is a very popular and well-known mobile operating system developed by Google. It is primarily designed for smartphones and tablets, but is used also for cars (Android Auto), watches (Android Wear), televisions (Android TV) [andc]. Android's flexibility and convenience is the reason it is used on so many different devices. User interface is scalable to the variety of the screens, so the same app can be conveniently used on different screens in different modes: portrait or landscape. One more reason to select Android as a development platform is that smartphones and also tablets nowadays have powerful processors, so they can be used for heavy computing, such as image processing. Also camera is required for getting the input data and all the smartphones have it already attached. There is no need to create any custom hardware with camera and high computational power, because modern smartphones have it all. It is a ready to use solution. Next advantage is that developing for Android is easy. To start coding it is only necessary to install Android Studio - Android development environment [andd]. All the documentation is open and free to use [anda]. Code for Android is written in Java language - one of the most popular programming languages in the world [jav]. Language with such popularity has a large community and is well-documented, so in case of a problem it is relatively easy to find a solution.

4.1.2 OpenCV library

OpenCV is an open-source computer vision and machine learning library [ope]. It includes more than 2500 image processing and learning algorithms from the classic ones to the most recent state-of-art algorithms. Proposed algorithm uses the library mainly for detecting and tracking moving objects within the scene, but also for converting the input for every method to a suitable format and reducing the noise. This library is available for many platforms, such as Python, C++, Java, Android specifically, however the content of the library may vary depending on the platform. As the development is meant for Android it is reasonable to use OpenCV for Android, but during the work progress it became clear that Android version lacks some desired methods, so the development of the main algorithm switched to C++. Running C++ with included OpenCV for C++ in it on Android is described in the next section. The most recent stable version of OpenCV - 3.1 is used in the proposed algorithm.

4.1.3 Android NDK. C++

Switching to C++ from Java (Android) has its advantages: C++ works faster, it gives the ability to use OpenCV without any limitations, documentation for OpenCV is written for C++ and Python only. Android allows running C++ code on its platform, if all the C++ codes and used libraries are configured as NDK (Native Development Kit) toolset [andb]. Setting up the environment of the Android app using C++ code with C++ OpenCV library was quite challenging, because there are no proper documentation on how to achieve that, especially considering the fact that this kind of environment is quite specific. Nevertheless, the app runs on Android using C++ codes. One more benefit of this approach is making the code flexible and reusable. Android is for user interface, the whole algorithm is on C++. In such a way the algorithm can be easily ported to any other platform, for example, Raspberry Pi.

4.2 Hardware

A couple of words about the device being used for developing and testing the algorithm. HTC One (m7) was chosen to be the one. Technical specifications [htc]:

CPU speed:	Qualcomm Snapdragon 600, quad-core, 1.7 GHz
RAM:	2GB DDR2
Display:	FullHD 1080p, 468 PPI
Android version:	Android 5.1

Chapter 5

Experimental results

This chapter describes, what results have been achieved, while developing the algorithm. As it can be seen from *figures 5.1, 5.2, 5.3*, the recognition has high accuracy in different scenes with different camera angles. All the video sequences are taken outside, so the environment is not stable i.e background is complex and might contain movement. However, the algorithm detects only desired movement, leaving all the trees and bushes as part of the background. Different light conditions do not affect the algorithm's accuracy as well.

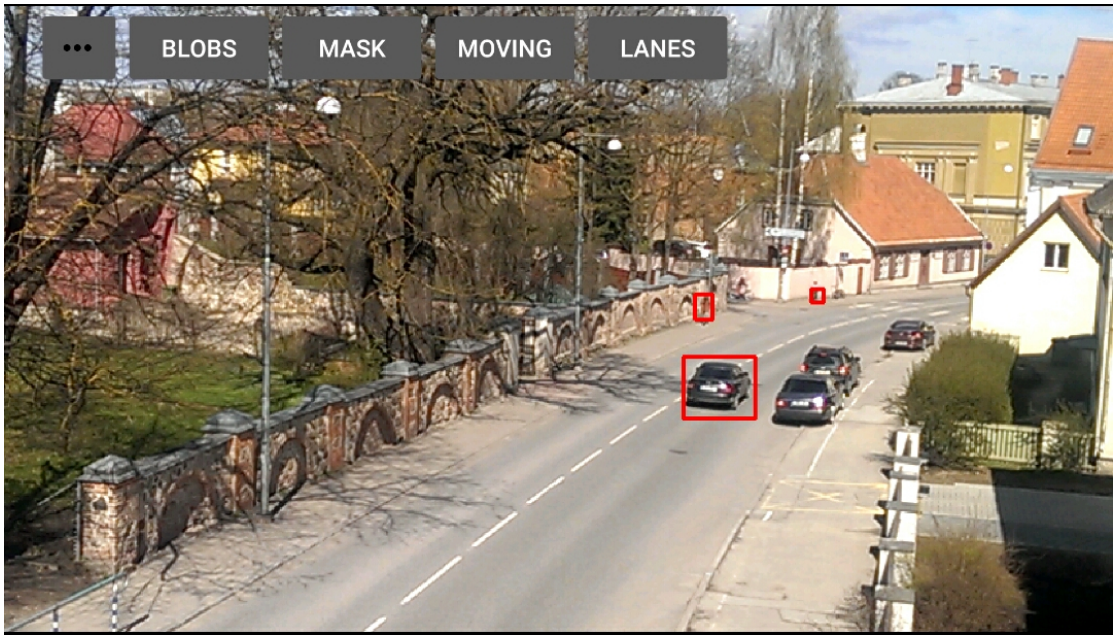


Figure 5.1: Final step of the algorithm - drawing blobs around moving objects: two pedestrians and a car. Screen from Android device is present

Figures 5.4, 5.5, 5.6 show the different steps of the algorithm in a way that processed output is compared to the original input frame, so it is easy to follow each step of the algorithm and see its results. Figures are also ordered accordingly:

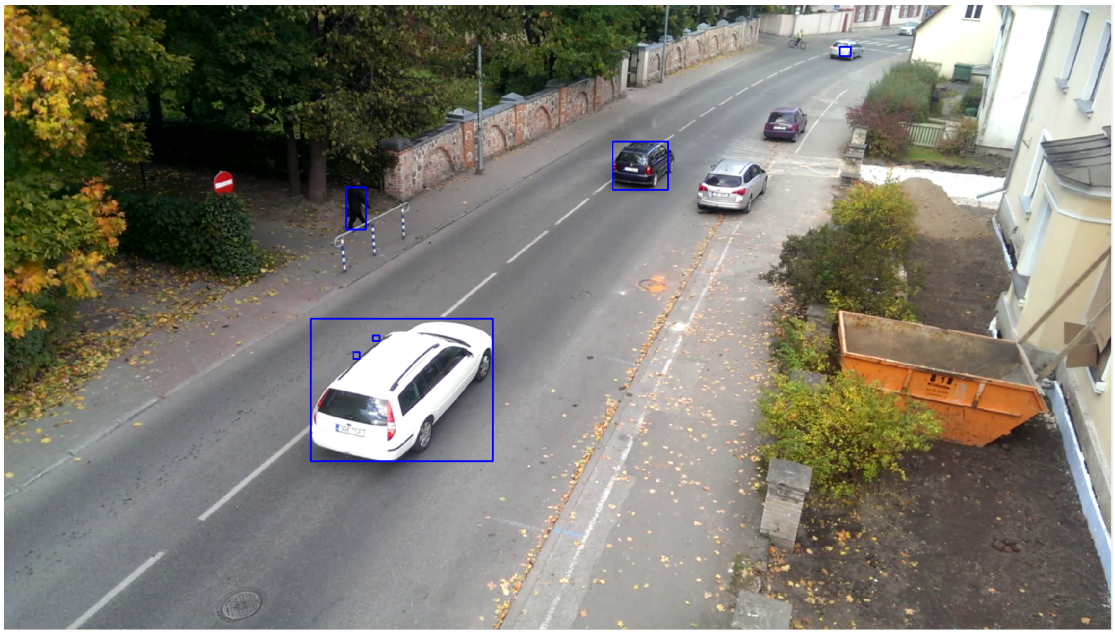


Figure 5.2: Algorithm has correctly detected all the 3 cars moving within the frame and also a pedestrian. All the objects are highlighted by blue rectangles - blobs. Notice that no movement in the trees and bushed is highlighted

Figure 5.3: Another example of the algorithm's detection. The scene is more complicated, but the recognition is still good

input frame is processed by GMM, which returns binary mask, then the noise within the mask is reduced, after that mask can be used to discard all but moving parts from the original frame or to draw blobs around moving objects, as it was seen on previous figures.

(a) Mask

(b) Original

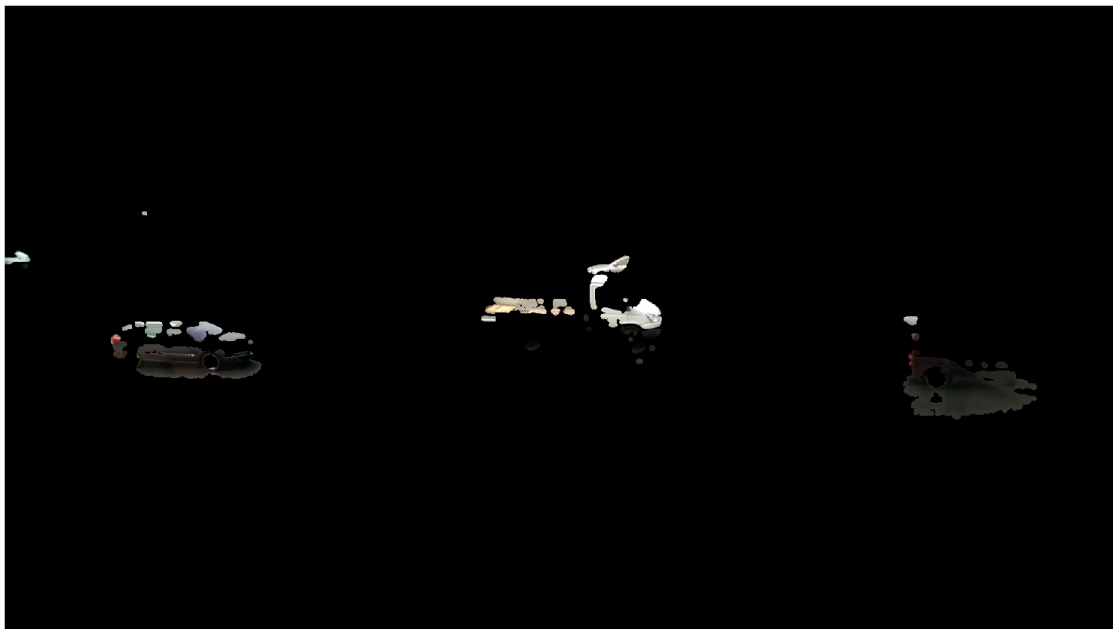
Figure 5.4: Only GMM is applied to the original frame (b) in here to illustrate the amount of noise that needs to be discarded. Mask produced by GMM is in (a)

(a) Mask

(b) Opening

(c) Original

Figure 5.5: Image (a) is a raw mask that is just obtained from the GMM. Image (b) shows the mask after morphological operations such as erosion and dilation. Image (c) is the original input frame



(a) Moving

(b) Original

Figure 5.6: Image (a) contains only moving objects, all the rest is 0. Image (b) is the original input frame

Chapter 6

Summary

Clearly, the algorithm is capable of detecting moving objects from video sequences with decent accuracy. However real-time recognition is not yet achieved. Due to hardware limitations and algorithm's complexity it is impossible to process every frame of a Full HD video sequence in real-time. For that purpose algorithm should process every frame in less than 0.35 seconds. Current implementation allows to process a frame within approximately 2.3 seconds.

Some ideas for improving this fact have occured:

1. Processing can be optimized by using using different threads, where possible. As most of the mobile devices nowadays have multiple cores, it shouldn't be a problem at all.

2. Cloud computing can also be used for optimizing the process. Some tests have to be conducted first to prove that streaming the data over the Internet won't take as much time, as processing on the device directly requires.

3. Performing all the calculations on the GPU. On Android it can be simply done by using RenderScript [ande] inside the app.

4. The most simple solution for improving the processing speed is usage of powerful hardware. As stated before, the code can be easily ported to a different platform, so custom devices using Raspberry Pi can be created for the purpose of the algorithm's optimization.

5. Implementing learning into the algorithm will require collecting some sample data first and training the algorithm, but further detection in this case will be much faster. This solution is selected to be focused on firstly.

Another disadvantage is that proposed algorithm works for static cameras only, moving ones cause too much noise. Idea was to recreate Gaussian Mixture Model

more frequently, for example every 5 or 10 frames, to reduce the noise caused by camera movement. Every time the GMM is initialized, previous changes are discarded and the algorithm works for a new background. For that purpose the amount of frames used for creating the model is made a dynamic value that can be changed via SeekBar, as explained previously. However, freely moving camera still creates too much noise. With aforementioned limitations the main goal can be considered achieved, but further improvements are required to raise the algorithm to a competitive level.

Bibliography

[anda] Android Development documentation and tutorials. `https://developers.google.com`.

[andb] Android NDK native development kit. `http://developer.android.com/ndk/index.html`.

[andc] Android OS homepage. `https://www.android.com`.

[andd] Android Studio download page. `http://developer.android.com/sdk/index.html`.

[ande] RenderScript homepage. `http://developer.android.com/intl/ru/guide/topics/renderscript/compute.html`.

[BEBV08] Thierry Bouwmans, Fida El Baf, and Bertrand Vachon. Background modeling using mixture of gaussians for foreground detection-a survey. *Recent Patents on Computer Science*, 1(3):219–237, 2008.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[fin] OpenCV 3.1 find contours function. `http://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a`.

[FR97] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 175–181. Morgan Kaufmann Publishers Inc., 1997.

[GTCS+01] Daniel Gutchess, M Trajković, Eric Cohen-Solal, Damian Lyons, and Anil K Jain. A background model initialization algorithm for video surveillance. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 733–740. IEEE, 2001.

[HJLB98] MP Hobson, AW Jones, AN Lasenby, and FR Bouchet. Foreground separation methods for satellite observations of the cosmic microwave background. *Monthly Notices of the Royal Astronomical Society*, 300(1):1–29, 1998.

[htc] HTC One m7 device description. `http://www.htc.com/us/smartphones/htc-one-m7`.

[jav] Redmonk programming language rankings. `http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/`.

[JS04] Boyoon Jung and Gaurav S Sukhatme. Detecting moving objects using a single camera on a mobile robot in an outdoor environment. In *International Conference on Intelligent Autonomous Systems*, pages 980–987, 2004.

[KCHD05] Kyungnam Kim, Thanarat H Chalidabhongse, David Harwood, and Larry Davis. Real-time foreground–background segmentation using codebook model. *Real-time imaging*, 11(3):172–185, 2005.

[KMIS00] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *Intelligent Transportation Systems, IEEE Transactions on*, 1(2):108–118, 2000.

[LHGT03] Liyuan Li, Weimin Huang, Irene YH Gu, and Qi Tian. Foreground object detection from videos containing complex background. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10. ACM, 2003.

[LHGT04] Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian. Statistical modeling of complex backgrounds for foreground object detection. *Image Processing, IEEE Transactions on*, 13(11):1459–1472, 2004.

[LXG+13] Dawei Li, Lihong Xu, Erik D Goodman, Yuan Xu, and Yang Wu. Integrating a statistical background-foreground extraction algorithm and svm classifier for pedestrian detection and tracking. *Integrated Computer-Aided Engineering*, 20(3):201–216, 2013.

[LYT+03] Graham Leedham, Chen Yan, Kalyan Takru, Joie Hadi Nata Tan, and Li Mian. Comparison of some thresholding algorithms for text/background segmentation in difficult document images. In *null*, page 859. IEEE, 2003.

[MMP05] Benjamin Maurin, Osama Masoud, and Nikolaos P Papanikolopoulos. Tracking all traffic: computer vision algorithms for monitoring vehicles, individuals, and crowds. *Robotics & Automation Magazine, IEEE*, 12(1):29–36, 2005.

[ope] OpenCV library about. `http://opencv.org/about.html`.

[QWX11] Huihuan Qian, Xinyu Wu, and Yangsheng Xu. Background/foreground detection. In *Intelligent Surveillance Systems*, pages 7–21. Springer, 2011.

[RC78] TW Ridler and S Calvard. Picture thresholding using an iterative selection method. *IEEE trans syst Man Cybern*, 8(8):630–632, 1978.

[RMK95] Christof Ridder, Olaf Munkelt, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of International Conference on recent Advances in Mechatronics*, pages 193–199. Citeseer, 1995.

[SCP+15] Eran Steinberg, Peter Corcoran, Yury Prilutsky, Petronel Bigioi, Alexei Pososin, Mihai Ciuc, Adrian Zamfir, and Adrian Capata. Foreground/background separation in digital images, August 25 2015. US Patent 9,117,282.

[SFD94] M Yakoob Siyal, M Fathy, and CG Darkin. Image processing algorithms for detecting moving objects. In *International Conference on Automation, Robotics and Computer Vision (3rd: 1994: Singapore). Proceedings: ICARCV'94. Vol. 3*, 1994.

[SG99] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[SJK09] Yaser Sheikh, Omar Javed, and Takeo Kanade. Background subtraction for freely moving cameras. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1219–1225. IEEE, 2009.

[WJW04] Rongrong Wang, Wanjun Jin, and Lide Wu. A novel video caption detection approach using multi-frame integration. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 449–452. IEEE, 2004.

[YYS+15] Xinchen Ye, Jingyu Yang, Xin Sun, Kun Li, Chunping Hou, and Yao Wang. Foreground–background separation from video clips via motion-assisted matrix restoration. *Circuits and Systems for Video Technology, IEEE Transactions on*, 25(11):1721–1734, 2015.

[Ziv04] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[ZYY13] Xiaowei Zhou, Can Yang, and Weichuan Yu. Moving object detection by detecting contiguous outliers in the low-rank representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(3):597–610, 2013.

All the web references were valid on 12.05.2016.

License

Non-exclusive license to reproduce thesis and make thesis public

I, Anton Prokopov (date of birth: 14.03.1993),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   "Real-time moving object detection and tracking using dynamic background and foreground separation for the purpose of moving camera based traffic analysis", supervised by Gholamreza Anbarjafari,

2. am aware of the fact that the author retains these rights.

3. certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2016

Source Code

As this thesis is a project requested by Mooncascade, source code is not publicly available. To receive the source code of the developed Android app, please contact the author via the following e-mail: prokopov@hotmail.com