VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MAPPING THE MOTION OF PEOPLE BY A STATIONARY CAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH BARTL

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MAPOVÁNÍ POHYBU OSOB STACIONÁRNÍ KAMEROU
MAPPING THE MOTION OF PEOPLE BY A STATIONARY CAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. VOJTĚCH BARTL
AUTHOR

VEDOUCÍ PRÁCE Doc. Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této práce je získat informace o pohybu osob ve scéně ze záznamu ze stacionární kamery. Byl navržen postup, jak detekovat výjimečné události ve scéně. Výjiměčné události mohou být rychle se pohybující osoby nebo osoby pohybující se jinde, než všichni ostatní ve scéně. Pro sledování pohybu osob byly použity a testovány dva algoritmy – Optical flow a CAMSHIFT. Analýza výsledných pohybů je prováděna sledováním průběhu pohybu a jeho porovnáním s ostatními pohyby ve scéně. Výsledkem analýzy jsou detekované výjimečné pohyby ve videu. Rovněž jsou označený oblasti, kde se ve scéně vyskytuje pohyb, oblasti kde je pohyb nejčastější a je provedena analýza směrů pohybů. Hlavním výsledkem jsou extrahované části videa, kde nastal výjimečný pohyb.

Abstract

The aim of this diploma thesis is to obtain information on the motion of people in a scene from the record of the stationary camera. The procedure to detect exceptional events in the scene was designed. Exceptional events can be fast-moving persons, or persons moving in different places than everyone else in the scene. To trace the motion of persons, two algorithms were applied and tested – Optical flow and CAMSHIFT. The analysis of the resulting motions is performed by monitoring the progress of motion, and its comparison with the other motions in the scene. The analysis result is represented by detected exceptional motions that can be found in the video. The areas where the motion occurs in the scene, and where the motion is the most common are also described together with the motion direction analysis. The exceptional motion parts extracted from the video represent the main result of the work.

Klíčová slova

detekce osob, detekce pohybu, sledování objektu, analýza pohybu, detekce výjimečných událostí

Keywords

people detection, motion detection, object tracking, trajectory analysis, anomaly events detection

Citace

Vojtěch Bartl: Mapping the Motion of People by a Stationary Camera, diplomová práce, Brno, FIT VUT v Brně, 2015

Mapping the Motion of People by a Stationary Camera

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Adama Herouta, Ph.D.

....
Vojtěch Bartl
May 25, 2015

Poděkování

Rád bych poděkoval vedoucímu mé práce, panu Doc. Ing. Adamu Heroutovi, Ph.D. za hodnotné rady, poskytnuté zdroje a odborné vedení během mé práce.

Contents

Chapter 1

Introduction

The whole process of mapping the motion is based on object motion trajectories. The main goal is to detect an object in a scene and track the object in the video sequence. Afterwards, the analysis of trajectories is made to understand the scene.

To realize object tracking, it is firstly necessary to detect the object in a frame. For this work purpose the goal is people detection. After people detection, people tracking to get trajectories of people motions can be done. Finally, the analysis of trajectories with the found people trajectories can be done.

In chapter 2 methods for people detection in frame, and methods used for object tracking are described. In chapter 3 the propose how to detect moving objects in image, and how to detect people in these areas is described. A technique how to realize people tracking is described. Techniques how to detect typical motion regions in a scene, analyse motion direction and detect anomaly events in scene are proposed.

Chapter 4 describes how the proposed techniques are implemented, and the final application testing is described. The results of different scene types are discussed and processing times are measured. The results of the proposed technique for anomaly events detection in a scene are shown.

Chapter 2

Related Techniques for Motion Mapping

In this chapter techniques which can be used for motion mapping are discussed. The whole motion mapping problem is very complex. The context of this work focuses on people motion detection, people tracking and consequently gets some semantic scene information by the analysis of trajectories.

Firstly, approaches to people detection by stationary camera are discussed in the section 2.1. Secondly, some techniques typically used for object tracking are mentioned in the section 2.2. Finally, some approaches to trajectory analysis are mentioned in the last section 2.3.

2.1 People Detection by Stationary Camera

At first, it is necessary to detect where the motion in the video is, and if it is a motion which is interesting for this task. In a typical scene a lot of different motions can occur. But moving trees in wind, cars or birds are not interesting for this task. Thus the distinction between a motion of a person and a different kind of motion is necessary.

People detection is one of the typical problems solved by computer vision. The goal is to detect a person in an image. This is complicated task due to variable appearance of people and a lot of different poses that people can reach. The main approaches to people detection at the level state-of-the-art are discussed in this section. Some information about used features, and the main classification methods are mentioned.

2.1.1 Motion History Image

From the task is clear that input is from stationary camera. It is an advantage because it is possible to use simple methods based on frame subtraction for motion detection. The basic idea is to subtract an actual frame from another one and the motion changes are saved in the difference image.

Background subtraction is one of these methods. The actual frame is subtracted from some background model. After subtraction and threshold is captured the mask with the motion in video. A naive approach could be to select one frame as the background model. This could potentially work but there could not be any change in illumination in future frames. Background subtraction, therefore, actualizes background mask more frequently. The update of background mask by some distribution function is often used in every frame.

This approach is much more stable with respect to illumination changes. For example, in the OpenCV library the implementation of background subtraction as described in the article [27] is used. This method is based on Gaussian mixture model (GMM) distribution function for background update.

Another method is motion history image (MHI). In this method the actual frame is not subtracted from some background model but directly from the previous frame or more frames. Again there exist more variants how to compute motion history image dependent for example on the number of previous frames, which is the actual frame subtracted from. Different variants of motion history image as well as information about motion energy image can be found in the paper [1].

The example of motion history image computation is shown in figures. In figure 2.1 there is the previous frame, and in figure 2.2 there is the actual frame. In figure 2.3 there is the frame difference and in figure 2.4 there is the thresholded difference, where the motion is evident.

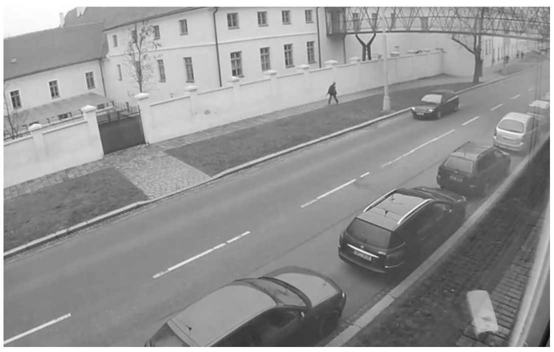


Figure 2.1: MHI previous frame

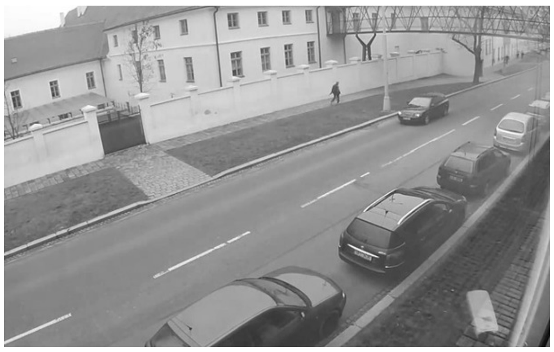


Figure 2.2: MHI actual frame

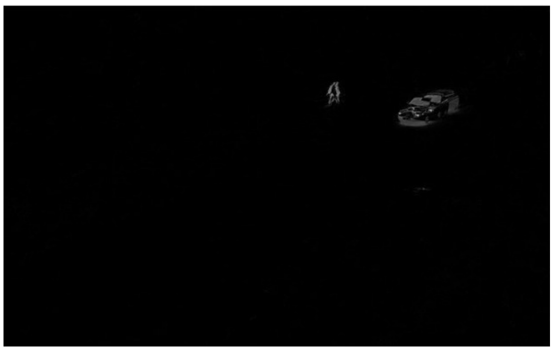


Figure 2.3: MHI difference image

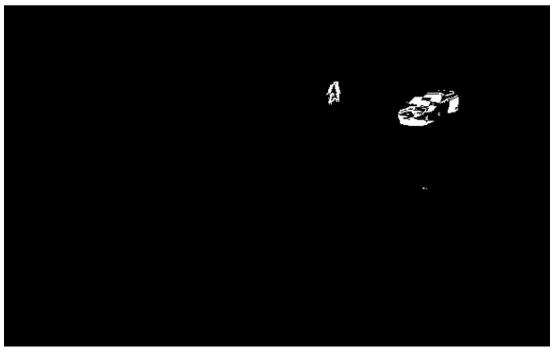


Figure 2.4: MHI thresholded difference image

The advantage of motion history image is in lower noise than with background subtraction method, as there is not so complicated background model. Motion history image is suitable for detect areas where some motion in video sequence occurs.

2.1.2 Morphological Operations

In figure 2.4 is shown, that the difference output contains a lot of details and motion area is not continuous. Better output should be as continuous as possible. For this task is possible to use morphological operators. The basic morphological operators are erosion and dilation. Both of these operators are implemented in OpenCV library.

Erosion of the image A by the structuring element B is defined as

$$A \ominus B = \bigcap_{b \in B} A_{-b} \tag{2.1}$$

Erosion removes border pixels from all binary elements in an image. If there are only small elements in an image, erosion can cause remove of these elements. In case of noise, erosion can cause its removal.

Dilation of the image A by the structuring element B is defined as

$$A \oplus B = \bigcup_{b \in B} A_{b} \tag{2.2}$$

Dilation adds border to all elements in binary image. This can be done if it is necessary to highlight elements in an image.

If the goal is to highlight whole area and remove unnecessary details, the dilation can be used for this task. After dilation, erosion can be used for getting more accurate result. Dilation followed by erosion is called closing and it is defined as

$$A \bullet B = (A \oplus B) \ominus B \tag{2.3}$$

In figure 2.5 the result of closing operator applied to difference output from figure 2.4 can be seen. Some more formal information about morphological operators can be found for example in the book [17].

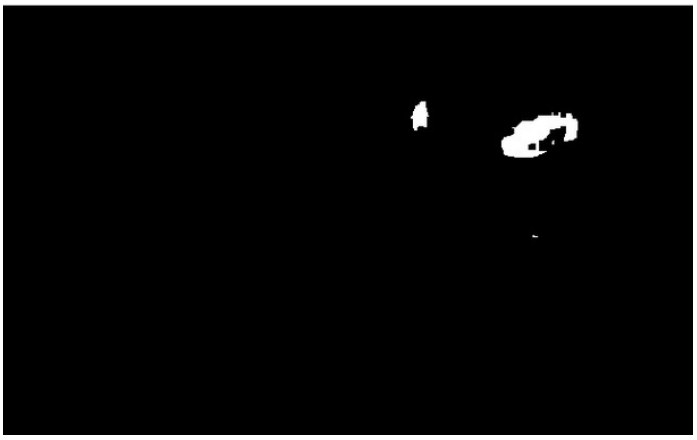


Figure 2.5: Closing operator applied on difference output

2.1.3 Contours

Contours serves to find continuous borders of elements in a binary image. It fits as good as possible to borders of objects in the binary image, thus it can be used for locating objects in binary image. If a bounding rectangle for every computed contour is made, it describes a region, where some object is located. The example of contour computation is in figures 2.6, 2.7 and 2.8. In figure 2.6 there is a binary input image. In figure 2.7 there are computed contours, and in figure 2.8 there are marked bounding rectangles for the found contours. Details about contours can be found in article [18].

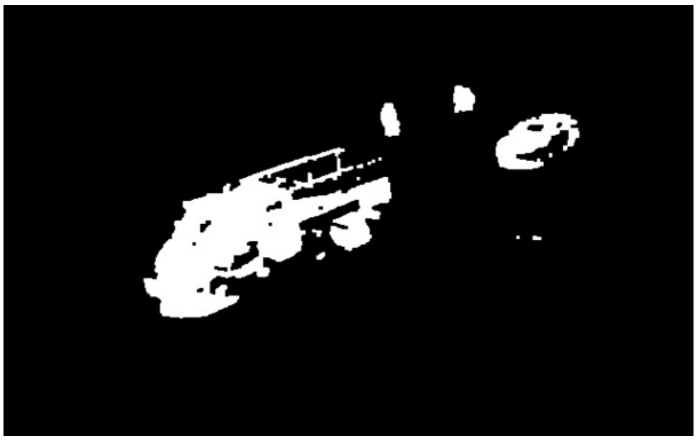


Figure 2.6: Binary input image

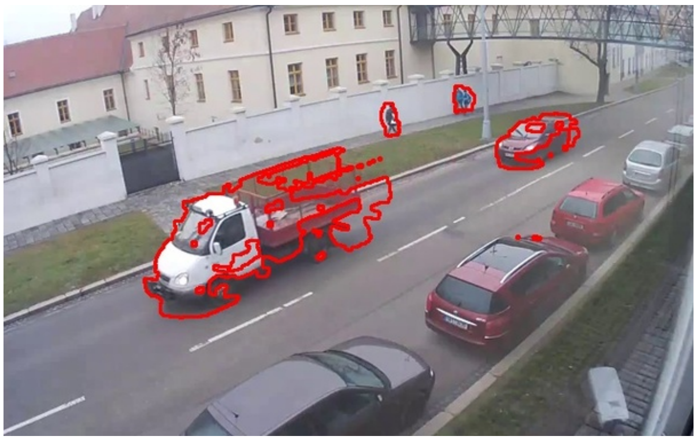


Figure 2.7: Computed contours

2.1.4 Features

If objects detection in an image is required, it is necessary to establish some features. Features serve to object description. These features are then look for in an input image. If the features are found, the object is detected. There exist a lot of different types of features. Features concept is complex and choice of features is dependent on the solved problem.

One of the most used features for people detection is histogram of oriented gradients (HOG). It was developed mainly for people detection in 2005 and was presented in the article [7]. The HOG method is based on computing gradients in an image and accumulating them to histograms, which describe gradient orientations in specific parts of the image.

Although HOG reaches really good results in pedestrian detection, there are still tendencies to find better features for people detection. Another step in finding the best features for people detection was to combine HOG features with a different type of features. In the paper [4] more than 40 different methods for pedestrian detection are tested, and HOG features or HOG plus some other feature type takes place in more than 60 % of these methods.

In 2009 a new approach named Integral Channel Features, described in work [9], was presented. The goal was to describe better features for people detection with the use of existing approaches. One of features used in this work is gradient histogram, because of

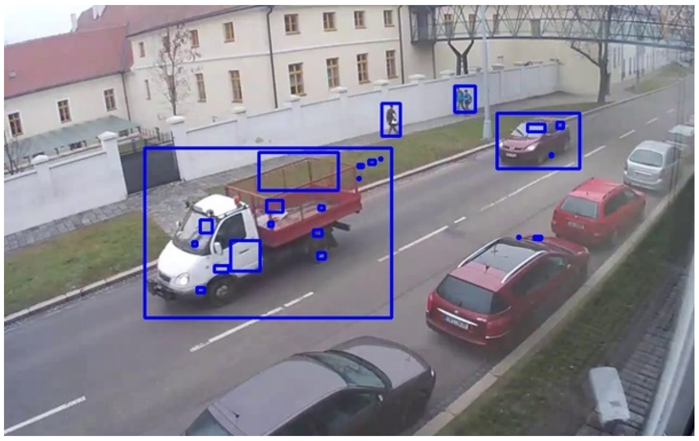


Figure 2.8: Contours bounding rectangles

possible approximation of HOG with the usage of integral image as described in article [26]. In work [9] numerous experiments were done, and the best results were reached with the combination of gradient histogram, gradient magnitude and LUV color channels. These 3 types of features can be used to describe an object. As seen in figure 2.9, in the sum 10 channels for image description are used – 6 for the gradient histogram, 1 for the gradient magnitude and 3 for the LUV color channels.

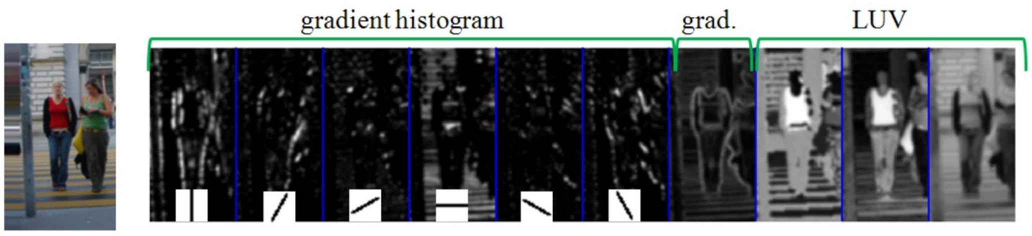


Figure 2.9: Computed integral channel features of the image (from [9])

2.1.5 Classification

After some features from the image are extracted, a technique based on the features which decides if the searched object is present in the image, is needed. Classificator learns information from training data set with training images when extracting features and learning, which images meet the condition of required object, and which do not meet this condition. Classificator output is often a decision about the presence of detected object in the image. Thus the actual features are sent to classificator and it decides if the features look like learned object or not.

There exist many types of classification methods. It is important how many classes a classificator contains. It can contain only two classes, the first with positive examples of an object and the second with negative examples of an object. Or, it can contain more classes, and than the output of classificator is the class, which meets most of given conditions. This multiclass type of classification can be used for example in character recognition, where decision which character is in the input is required. In the case of people detection the classification contains the selection of two classes whether the object is detected or not.

Support Vector Machines

A typical example of classificator is Support Vector Machine (SVM). It was presented in the article [6] in 1995. But until these days it is often used for its good results. The main idea is to separate high-dimensional space with training data by hyperplane or set of hyperplanes. The hyperplane should has the largest possible distance to the nearest training data point of any class. The margin of the hyperplane should be the largest possible. Example is in figure 2.10, where L_1 and L_2 separates classes, but with a small margin. The goal is to separate classes with the largest possible margin, so L_3 is the best solution.

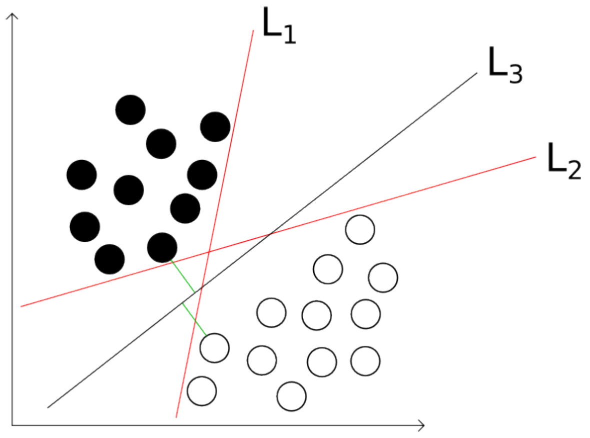


Figure 2.10: Support Vector Machine example

The largest possible margin is requested for the minimizing of possible error. If the margin would be small, the separating hyperplane is than close to classes, and a possible wrong decision could occur. The largest possible margin provides a large distance between classes, and there is more space for edge values. It means there is less space for wrong decision.

This type of classification is called linear SVM, due to linear separation hyperplane. If there are more classes there can be more linear hyperplanes to separate training data. The linear SVM is the basic type and it was used for example as a classificator with HOG features in the work [7]. There also exist another types of SVMs with other types of kernels. Radial Basis Function (RBF) is also often used as a kernel. BRF can be mentioned like a non-linear kernel. According to the paper [4] with extensive comparison of many available methods, there is mentioned that there is no conclusive evidence indicating whether non-linear kernels provides meaningful gains over linear kernels (for pedestrian detection, when using non-trivial features).

Adaptive Boosting

Although Adaptive Boosting, also known as AdaBoost was presented already in work [10] in 1995, it is today one of the most exploited classification algorithms. It is based on boosting, where final classificator is composed from couple of weak classifiers. The final output of AdaBoost classifier is linear combination of weak classifiers. But is is not the linear classifier. Weak classifiers have typically some simple function like threshold. AdaBoost is a general algorithm for training one strong classifier from the sequence of weak classifiers.

AdaBoost is adaptive in the sense that in every iteration of algorithm is recomputed the weight vector for all weak classifiers and the best weak classifier which minimize error is chosen. In every iteration a distribution function is also recomputed. Distribution function serves for balancing the weights of weak classifiers, so there is not selected the same weak classifier in the next iteration.

Algorithm 1 Adaptive Boosting (AdaBoost)

```
Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = \frac{1}{m}$

for $t = 1, 2, \ldots, T$ do
    • Train weak classifier using distribution $D_t$
    • Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error
          $$\epsilon_t = \sum_{i=1}^{m} D_t(i)[y_i \neq h_t(x_i)]$$
    • Choose
          $\alpha_t = \frac{1}{2} ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
    • Update:
          $D_{t+1}(i) = \frac{D_t(i) \cdot exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
      Where $Z_t$ is normalization factor (chosen so that $D_{t+1}$ will be distribution).
end for

Output the final hypothesis
          $$H(x) = sign \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
```

AdaBoost is described in algorithm 1. The samples $x_1, \ldots, x_m$ are the input with incident ratings $y_1, \ldots, y_m$ where $y \in \{-1, +1\}$ due to positive or negative sample. The output of algorithm is classifier composed of weak classifiers. AdaBoost algorithm has an advantage in low overfitting threat, and potentially arbitrary low error on training data set.

AdaBoost was used for example in the work [20] for face detection with Haar-like features. In this work AdaBoost was used for learning classifiers in a cascade classifier. Every classifier in sequence has a decision competence. Each classifier can therefore decide that the sample, which is actually being tested is not a searched object. As is shown in figure 2.11, every classifier $C_1, C_2, C_3, \ldots, C_N$ can decide in a negative way and in that case finish the evaluation. If the sequence of classifiers reaches the last classifier which decides positively, the object was detected. The main gain of this approach is the fast fire of negative detections with some classifier from the beginning of the sequence.

2.1.6 Sliding Window

The major part of detectors works with the concept of sliding window. It means that sliding window is moving in the image and in every possible position of the window, is the window content tested for the presence of an object. Basic window size is typically set at the same

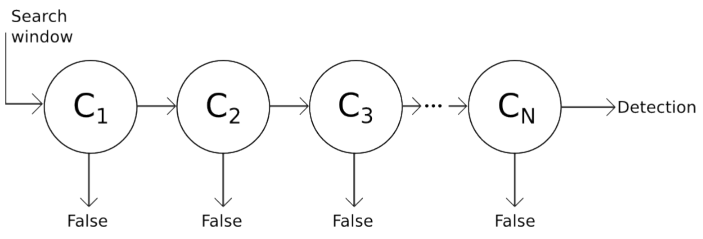


Figure 2.11: Cascade Classifier

size, which was used for test images set. For example in the work [7] the test images set size is 64×128 pixels. This value can be set as a default sliding window size. The sliding window is than moved in the image with specific stride on both dimensions. The example is in figure 2.12.

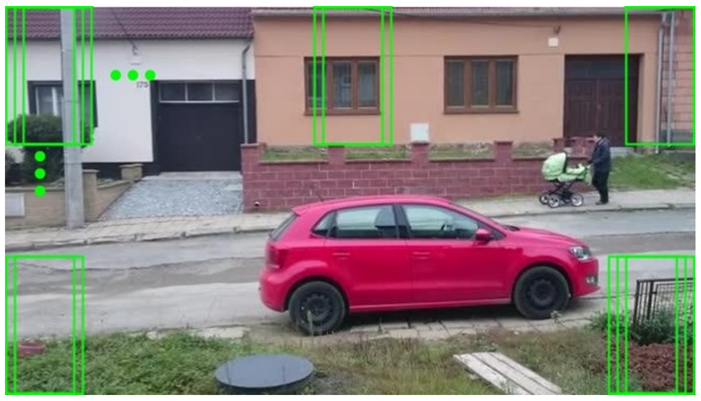


Figure 2.12: Sliding window

Sliding window is tested on lots of possible positions in an image, but it does not assure object detection, because of numerous possible object sizes. Usage of more possible sliding window scales is necessary. Default window is upscaled and downscale many times to cover more possible object sizes in an image. The example of downscaled sliding window can be seen in figure 2.13.

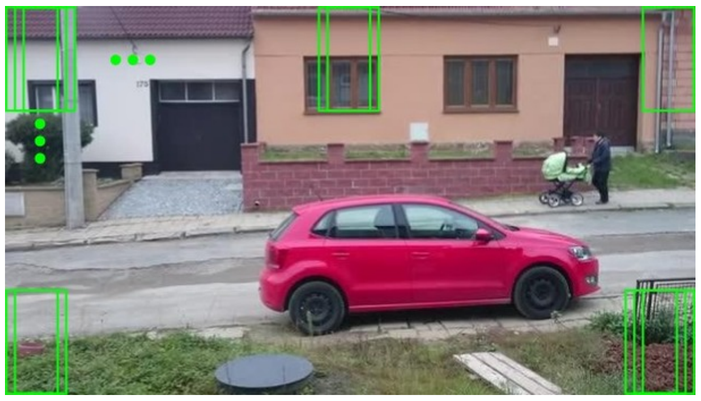


Figure 2.13: Downscaled sliding window

When many sliding window scales are used, there grows the computation cost for detection. So when a number of scales and window strides is chosen, one must be very careful with these values. When high number of scales and small stride in both dimensions is chosen, the computation time would be excessive.

2.1.7 Model and Image Scaling

Typically, it is not satisfactory to detect only one person size in an image but many possible sizes are demanded. However, there comes the problem of pixel discretization. Small scaled people image can suffer with blur effect, which a trained model does not computed with. Likewise large scaled people image can contain many more pixels and can provide more detailed information. Again, the model does not compute with this unusual case. A satisfactory solution is needed to solve this problem.

The naive approach to the problem would be to train model in many different scales. The number of scales is usually something about 50 which should cover majority of possible appearances. The model scale approach can be seen in figure 2.14. With this approach about 50 models should be trained. The training and also detection would be extremely slow with this method.

Another possible solution is not to scale the model but to scale the image where detections are looked for. This approach can be seen in figure 2.15. This method provides quiet good results but there also occur two main problems. The first problem is to select the right scale for a training model. A scale, which covers high-detailed large objects, and also blurred small objects, is necessary. The second problem is a computational time. Every image where the detection process is realized must be scaled about 50 times and every time image features must be recomputed. This approach is again extremely slow.

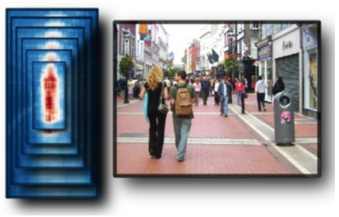


Figure 2.14: Model scale (from [3])

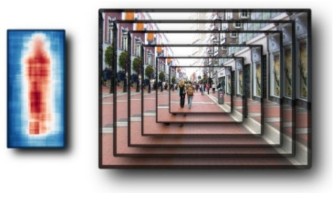


Figure 2.15: Image scale (from [3])

A try to reduce number of scales was presented in work [8]. The goal was to reach same results as with approach, where image was scaled 50 times but with less scaling. The idea is displayed in figure 2.16. An image is scaled just about 5 times. For every scaled image, features are computed, and these image features are used to approximate the features in the remaining 45 scales. The main idea is that the features of nearby scales can be approximated accurately enough, only with a minor error. The whole process of detection is much more faster, and the detection quality is almost comparable due to previous methods with the usage of this knowledge. This method is called `The Fastest Pedestrian Detector in the West` (`FPDW`).

The previously mentioned method was the base for another new method described in paper [3]. The goal is to move scaling from testing to training stage. Actually, the main idea is to take method from [8], and to reverse it. The model is scaled just few times, and approximates the rest of scales from this trained models. The method is shown in figure 2.17. Only 5 models are trained in a training time (scales 0.5, 1, 2, 4, 8), and the rest of 45 model scales is approximated during test time. The features can be computed only once for an image because the image is only in a default scale and does not need to be scaled.

The example of trained models in all 5 scales can be seen in figure 2.18. The scale 1 has size of the training data set – 64×128 pixels.

Although the training time increases due to training 5 model scales, the test time is extremely reduced in comparison with all previous methods. And due to the results described in paper [3], the detection success is better than in all previously mentioned methods. This method, which the authors call `VeryFast`, is nowadays probably the best possible approach for people detection.

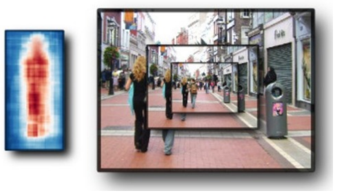


Figure 2.16: Method `FPDW` (from [3])

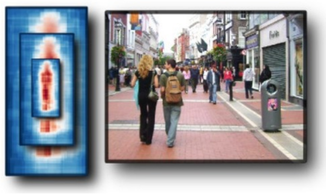


Figure 2.17: Method `VeryFast` (from [3])

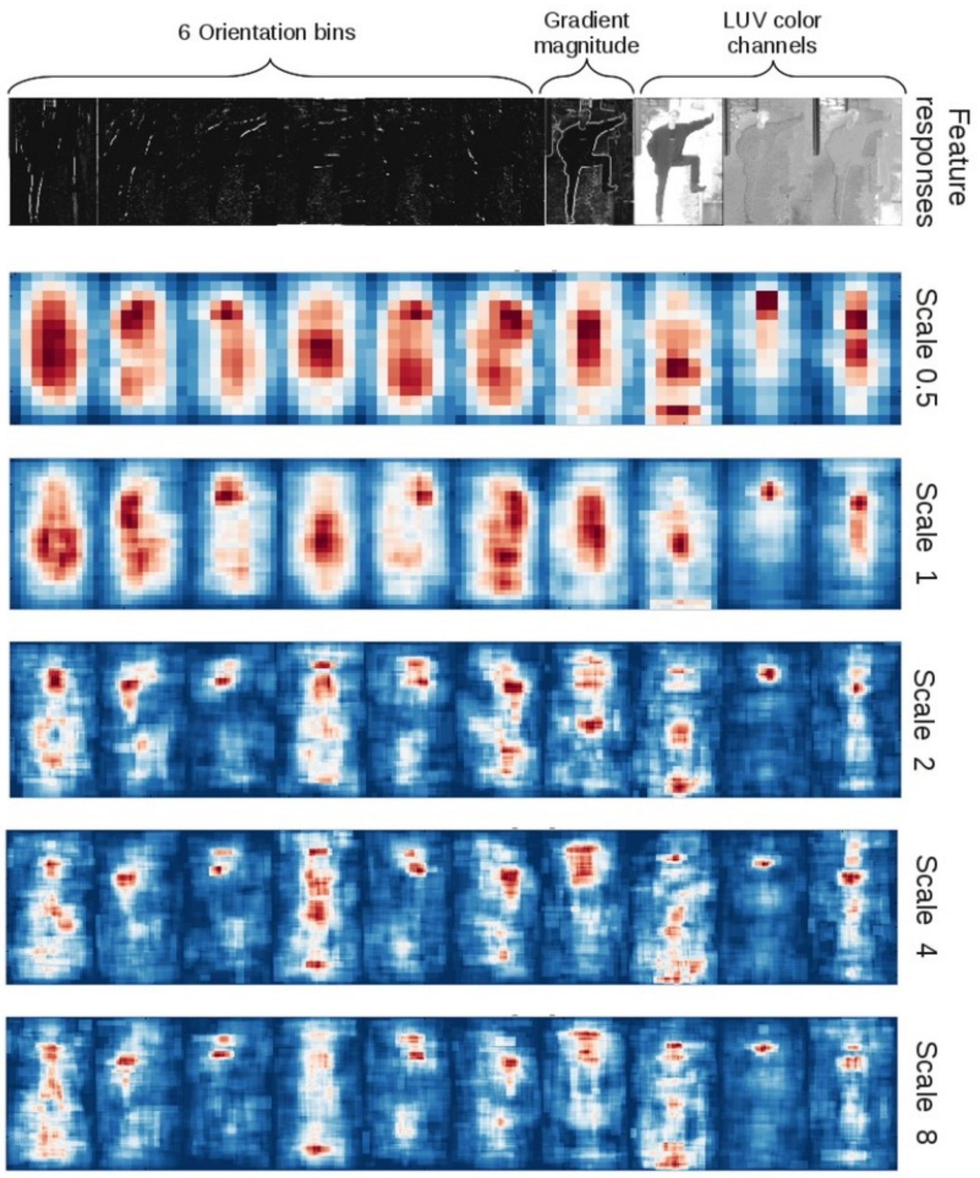


Figure 2.18: Trained multi-scales model (from [3])

2.2 Detected Object Tracking

People detection process provides information about people occurence in an input image. But it does not provide any information about people motion. To get information about people motion, it is necessary to process people tracking. Tracker serves for tracing of any object in video sequence and record step by step object position in each sequence frame. Tracking gives information where motion starts and ends, and what was its progression. Total trajectory of object motion is available.

Tracking is based on finding correspondences of features between single frames from video sequence. Some features to track are established and these features are watched by selected tracking method. There are many different possible options of feature and tracking method selection. The comprehensive overview about object tracking can be found in survey

[23]. The most often used methods for tracking are background subtraction (mentioned in section 2.1.1), Mean-shift (and its variant CAMSHIFT described in section 2.2.4) and Optical flow (section 2.2.2), which are all discussed in book [14].

2.2.1 Harris Corner Detector

For tracking process is convenient to have some features which can be easily followed. Object significant points are often chosen, and over these points tracking is done. Significant corners help to describe object silhouette and this can be followed by some tracking method.

Harris corner detector can be used to find suitable corners in an image. It is based on computation of direction gradients in small rectangle in a grayscale image, and on the basis of these gradients a flat area, an edge or a corner is distinguished. The eigenvalues are computed for guarantee of rotation invariant. Detailed description can be found in paper [12].

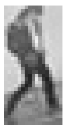


Figure 2.19: Input image

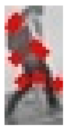


Figure 2.20: Computed corners

In figure 2.19 an input image for Harris corners detection can be seen. In figure 2.20 there are eight best rated detected Harris corners.

2.2.2 Optical Flow

Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image. Computation is based on locating feature motion based on searching the neighbourhood of each pixel in the rectangle which describes a subimage. Estimation of motion features in image is computed iteratively. It is often used for computation the motion of suitably selected features. These can be some texture features, or some significant points in image. After optical flow computation, new positions of features in a new image are set in comparison to positions in previous image. The full description of algorithm can be found in work [13].

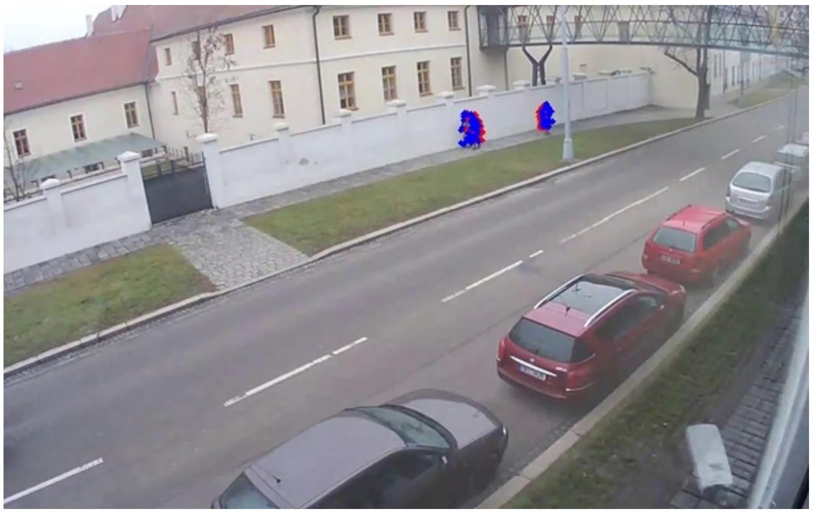


Figure 2.21: Optical flow computation

In figure 2.21 is the example of optical flow computation. Red points mark the features in a previous image and blue points mark the newly computed feature positions in an actual image.

2.2.3 Histogram Back Projection

Histogram back projection, described in article [19], is a method for detecting object in an image. It is based on computing objects histogram, and than checks how well the pixels in the given image fit the distribution of pixels in histogram.

Figure 2.22: Detection object

Figure 2.23: Corresponding histogram

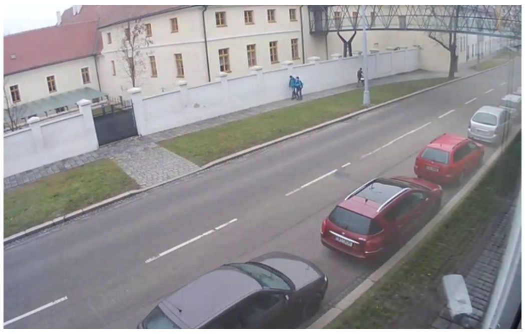


Figure 2.24: Input image

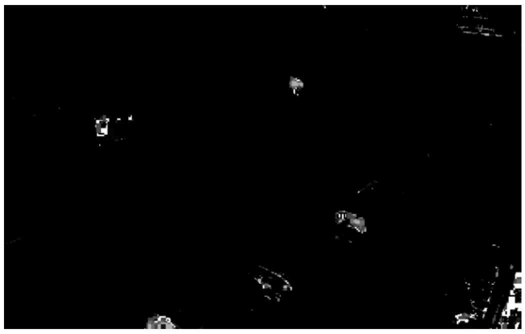


Figure 2.25: Histogram back projection

In figure 2.24 the actual processed frame can be seen. In figure 2.22 there is a part of whole frame, for which histogram is calculated, and this object is than looked for in an image. Histogram computation is made with HSV image (described in section 2.2.4), because mainly color part of the image holds information. Computed histogram of the object is shown in figure 2.23. In this case hue channel histogram is 1D with 64 bins. In figure 2.25 there is a result of histogram back projection.

When computing histogram back projection, every pixel in tested image (converted to hue channel) is selected and its histogram bin value is chosen. This bin value is searched in object histogram and value from object histogram is stored into a new back projection image. All pixels in test image are processed in this way. In back projection result image is stored the probability, that pixels in the test image have the same color as searched object.

From figure 2.25 it is evident, that the areas with higher values (light areas) belong to objects with similar color as wanted object. And the areas with a low value (dark areas) belong to the background. The higher value in back projection signifies the higher probability of object color detection.

2.2.4 CAMSHIFT

CAMSHIFT (Continuously Adaptive Mean Shift) is an algorithm used for the locating of some object in an image. It comes out from Mean Shift algorithm and it uses HSV color model.

HSV Image

HSV (Hue, Saturation, Value) color model is often used in computer vision. Hue channel stores information about color in image. Saturation channel describes how strong color is. Value channel describes strength of brightness. As is described in section 2.2.3, HSV image is often used for histogram back projection because it stores color information only in one channel in comparison with RGB model, which is mainly interesting for object description when color features are used. In figure 2.26 there is original RGB image, in figure 2.27 hue channel is separated, in figure 2.28 saturation channel is separated and in figure 2.29 there is separated value channel.

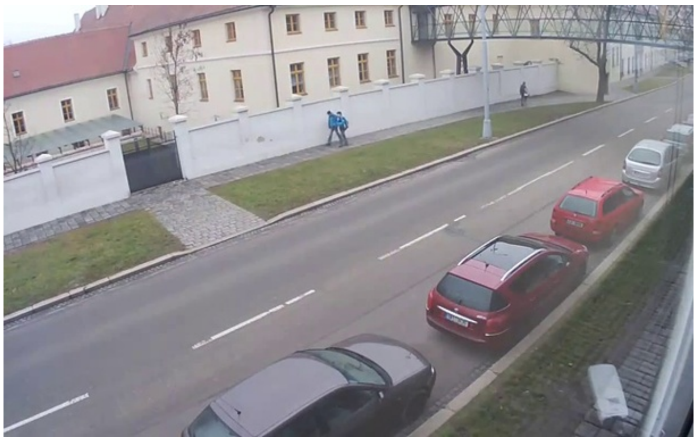


Figure 2.26: Original image

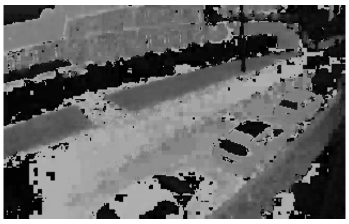


Figure 2.27: Hue

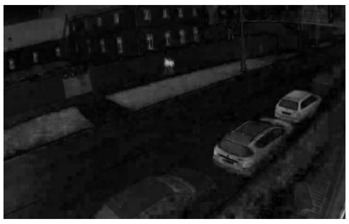


Figure 2.28: Saturation

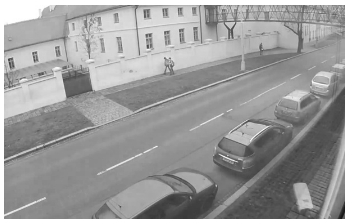


Figure 2.29: Value

Mean Shift

Mean Shift algorithm (described in articles [11] and [24]) is a non-parametric technique that climbs the gradient of a probability distribution to find the peak in feature space. It serves for finding models in a set of data samples with usage of probability density function (PDF). It can be used with arbitrary feature type. It is often used with color space features. The main idea is to look up feature space and locate the maxima with usage of probability density function.

Algorithm 2 Mean Shift

```
1: Choose a search window size
2: Choose the initial location of the search window
3: Compute the mean location in the search window
4: Shift center of search window at the mean location computed in step 3
5: Repeat steps 3 and 4 until predefined number of iterations is reached or the search
   window center change its location (by some predefined divergence)
```

From algorithm 2 it is evident, that algorithm is iterative. An example of locating the maxima in the search window is shown in figure 2.30. Search window is a red square and the mean location is purple circle. In every iteration (b), (c) the search window center is closer to the densest feature location. At the end of algorithm (d) the search window is at the place where the densest features take place.

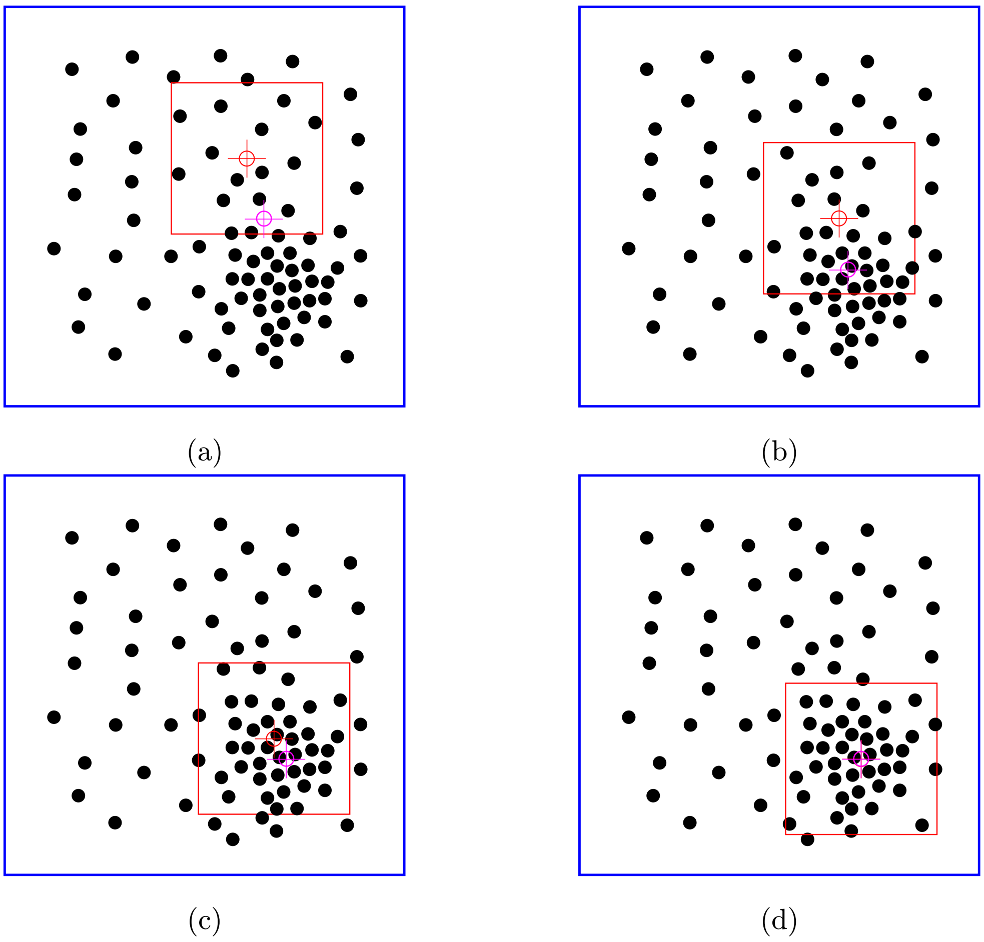

(a) (b) (c) (d)

Figure 2.30: Mean shift algorithm

Window size does not changing during algorithm which can be the problematic part of process. This can be solved with an adaptive window size used for example in CAMSHIFT algorithm.

CAMSHIFT

Mean Shift algorithm works well on static feature space. But if feature space is dynamic, setting the search window size can be complicated. Mean shift can be used for object tracking, but with certain limitations. If an object changes its size, Mean Shift is not ready for this situation, with respect to static size of search window. In a typical video sequence, an object often changes its dimensions. If a pedestrian walks form the background part of the scene to the front part of the scene, it changes its size. Here Mean Shift algorithm meets the problem. The usage of algorithm with adaptive window size provides the solution.

CAMSHIFT (Continuously Adaptive Mean Shift) is an algorithm based on the Mean Shift algorithm, but with an adaptive window size. In comparison to original Mean Shift it changes search window size during iterations, and it is more efficient for object tracking.

From algorithm 3 it is evident, that CAMSHIFT is again iterative. Search window size, orientation and position are set to the best possible fit to searched features. First, CAMSHIFT algorithm was used for face tracking, as described in [5], but it can be used for any object tracking based on color features. It works by tracking the hue of an object.

Algorithm 3 CAMSHIFT

```
1: Choose the initial location of search window
2: Process Mean Shift algorithm (more iterations possible) and store the zeroth moment
3: Set the search window size equal to a function of the zeroth moment found in step 2
4: Repeat steps 2 and 3 until the search window center change its location (by some
   predefined divergence)
```

In figure 2.31 histogram back projection of the object hue channel is marked (as described in section 2.2.3). There can be used CAMSHIFT to look for best possible correspondency of color features in back projection output.

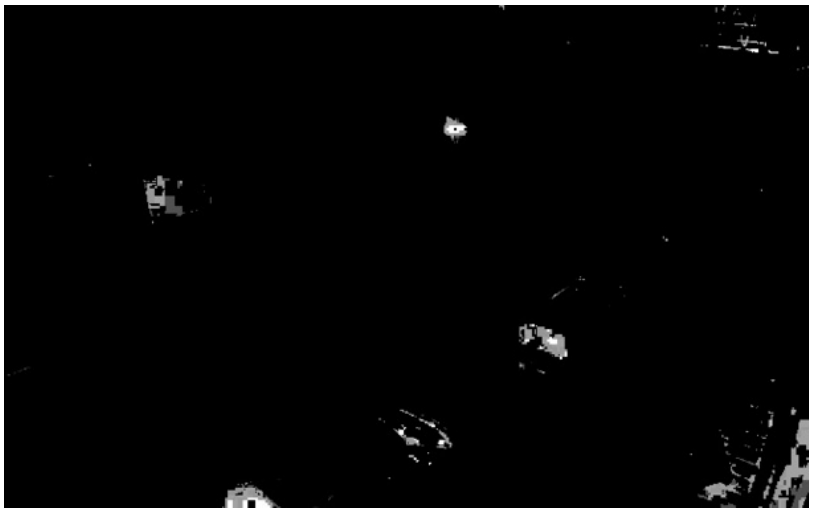


Figure 2.31: Histogram back projection

In figure 2.31 there is histogram back projection of object in figure 2.22. In this histogram back projection image CAMSHIFT algorithm is realized to search the object. The output can be seen in figure 2.32. The output detection rectangle is rotated because CAMSHIFT algorithm makes search window adapting to the best possible fit.

2.3 Trajectory Analysis

After the tracking is processed, certain pieces of information about single motions of people are known. It is known, where every single person started its motion, what was the progression of motion, and where did the motion finish. But the goal is to make a complex motion mapping of all people in the scene. It is necessary to make some analysis of the stored trajectories, to get a complex semantic information, and to understand the scene.

There are many different approaches to a trajectory analysis. First, it is necessary to make clear, which trajectory representation to use. More different types of trajectory representation exist as it is described in book [16]. There is often made some trajectories preprocessing to make analysis only on relevant and suitable trajectories. Different methods of trajectories preprocessing as well as comparison of analysis methods are discussed in paper [15].

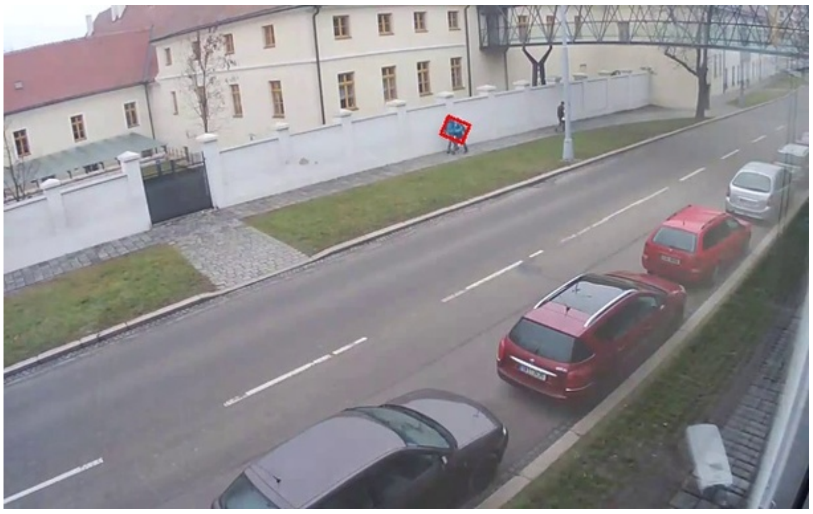


Figure 2.32: Object detection with CAMSHIFT

2.3.1 Heat Map

Heat map is one of the simplest solutions for visual show information about all motions in a scene. All trajectories are saved into heat map matrix, so the places with a higher accumulation of trajectories hold larger value in map and place where with no trajectories hold zero value. These values are then converted into linear color space, where higher values have higher color saturation, lower values have lower color saturation, and zero values are not marked in map.

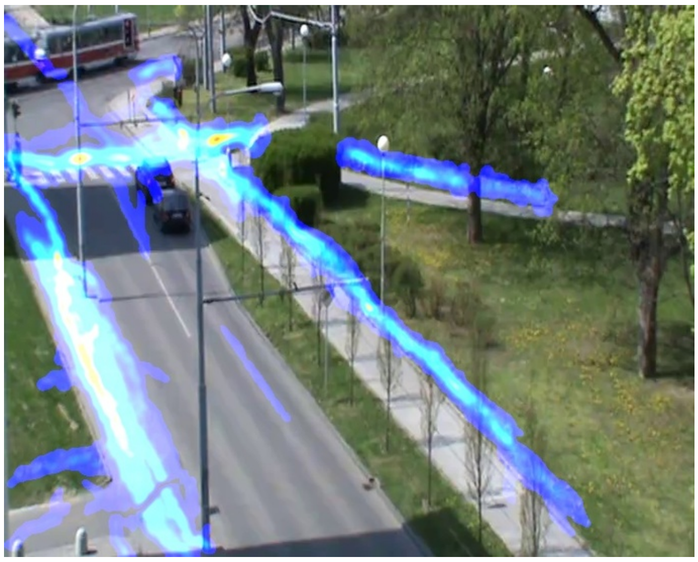


Figure 2.33: Heat map example

In figure 2.33 there is an example of heat map. The lighter tone of blue color signifies a higher motion occurence. Orange and red colors signify the highest occurence of motion in the scene. In this scene are motions evident on walkways and pedestrian crossing. A heat map gives visual information about places where motions take place and frequency of motions in the scene.

2.3.2 Clustering

The goal of clustering is to partition objects into clusters in the way that the objects in the same cluster are more similar to each other than to those in another clusters. Often, the condition of similarity is distance. Points in space are partitioned into k clusters, where each point in the cluster is closer to the cluster center than to any other cluster center.

K-means

K-means is one of the most often used clustering algorithms. Given a number k and a set of n data points in $\mathbb{R}^d$, the goal is to choose k centers so as to minimize the total squared distance between each point and its closest center. K-means algorithm starts with k arbitrary centers chosen randomly from the data points. Each point is then assigned to the nearest center and each center is recomputed as the center of mass of all points assigned to it. Assigning points and centers recomputing is repeated until the process stabilizes.

An integer k is given and a set of n data points $\chi \subset \mathbb{R}^d$. Goal is to to choose k centers $\mathcal{C}$ so as to minimize the potential function,

$$\phi = \sum_{x \in \chi} \min_{c \in \mathcal{C}} ||x - c||^2. \tag{2.4}$$

From these centers, a clustering can be defined by grouping data points according to which center each point is assigned to.

Algorithm 4 K-means

```
1: Arbitrarily choose an initial $k$ centers $\mathcal{C} = c_1, c_2, \dots, c_k$
2: For each $i \in \{1, \dots, k\}$, set the cluster $C_i$ to be the set of points in $\chi$ that are closer to $c_i$ than they are to $c_j$ for all $j \neq i$
3: For each $i \in \{1, \dots, k\}$, set $c_i$ to be the center of mass of all points in $C_i$: $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$
4: Repeat steps 2 and 3 until $\mathcal{C}$ no longer changes
```

K-means is described in algorithm 4. The main idea is that steps 2 and 3 are both decreasing ϕ , so the algorithm makes local improvements to an arbitrary clustering until it is no longer possible to do so.

K-means++

Original K-means algorithm starts with arbitrarily chosen initial centers. This is where some improvements can be done. Algorithm K-means++ is based on a specific way of choosing initial centers for the K-means algorithm. It is described in paper [2]. Let $D(x)$ denote the shortest distance from a data point to the closest center we have already chosen.

K-means++ is described in algorithm 5. Steps 4, 5 and 6 are the same as K-means algorithm. The difference is in choosing initial cluster centers.

The example of K-means++ output can be seen in figures. In figure 2.34 there are data points. In figure 2.35 there are points partitioned into three clusters marked with different colors. Cluster centers are marked with white dots. In figure 2.36 there are the same points partitioned into four clusters.

Algorithm 5 K-means++

```
1: Take one center $c_1$, chosen uniformly at random from $\chi$
2: Take a new center $c_i$, choosing $x \in \chi$ with probability $\frac{D(x)^2}{\sum_{x\in\chi} D(x)^2}$
3: Repeat step 2 until we have taken $k$ centers altogether
4: For each $i \in \{1, \dots, k\}$, set the cluster $C_i$ to be the set of points in $\chi$ that are closer to
   $c_i$ than they are to $c_j$ for all $j \neq i$
5: For each $i \in \{1, \dots, k\}$, set $c_i$ to be the center of mass of all points in $C_i$: $c_i =
   \frac{1}{|C_i|} \sum_{x\in C_i} x$
6: Repeat steps 4 and 5 until $\mathcal{C}$ no longer changes
```

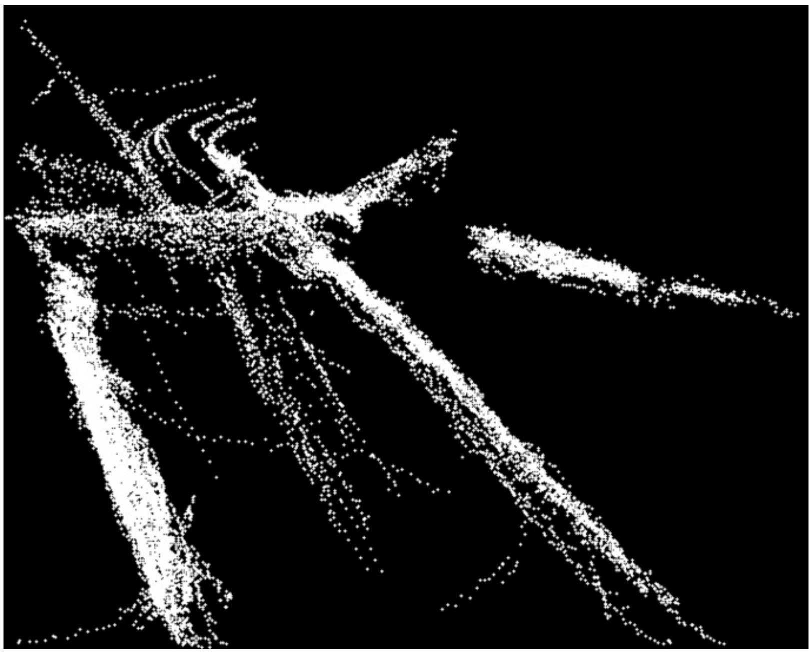


Figure 2.34: Data points

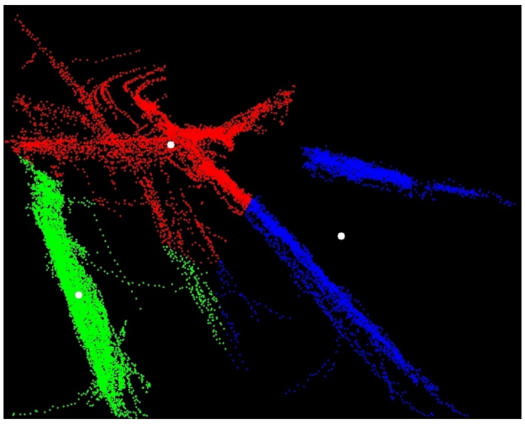


Figure 2.35: Three clusters

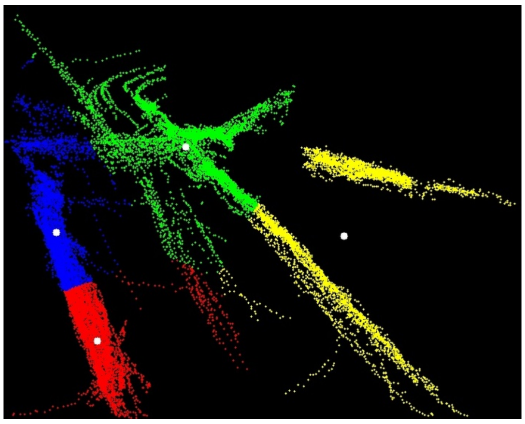


Figure 2.36: Four clusters

Chapter 3

Anomaly Events Detection

The concept how anomaly events is video sequence can be found is described in this chapter. The goal is to find people whose motion is exceptional in comparison with all other people motions in the scene, and to detect motions which are unusually fast.

First, it is necessary to create people motion trajectories. This process is composed of two parts. People detection, described in section 3.1, and the consequent creation of motion trajectories, described in section 3.2. Finally, the analysis of trajectories where the anomaly events are detected is made and described in section 3.3.

3.1 Moving People Detection in Static Camera Viewpoint

People detection is one of the typical problems solved by computer vision. There exist numerous different approaches to people detection. Typically, it is necessary to estimate features in an image and to use learned classificator for search the features which correspond with the learned people features. Sliding window concept, as is described in section 2.1.6, is typically used for seeking through image.

Doppia library by Rodrigo Benenson is employed for people detection in this work. It uses integral channel features mentioned in section 2.1.4, model scaling method `VeryFast` described in section 2.1.7 and AdaBoost algorithm for classification, described in section 2.1.5.

3.1.1 Input Pre-processing

In Doppia library people detection is made with the usage of sliding window concept, as described in section 2.1.6. Initial sliding window size can be set in graphical user interface, as described in section 4.1.2. Sliding window stride can be set in both dimensions to arbitrary values, window scale values so well as the number of scales can be set. All these parameters participate in detection process time. Although these parameters are carefully set (3 pixel stride, 50 % maximal scale up and 50 % maximal scale down with 8 total scales), the detection process on image containing 640×400 pixels takes about two seconds. This is unacceptable slow and some input pre-processing must be done to reduce detection time.

The main input modification is consisted of detecting regions in an image where motion occurs and executing people detection process only in these regions. Regions without the motion are skipped, as they probably do not contain nothing interesting for this work focus. Another reduce of detection time consists in frame skipping. Only every nth frame is processed where n can be set in graphical user interface. Typically convenient value of n

is dependent on fps of input video. If value is too small no detection process acceleration is achieved and if value is too large important information can be skipped.

For detection regions with motion the information about static camera usage is convenient. Frames difference can simply be done to detect motion as described in section 2.1.1. Actual frame and previous frame difference is computed, threshold is made and closing operator is applied as described in section 2.1.2. The example output is in figure 2.6. For the location moving objects in a binary image contours are used as described in section 2.1.3. Bounding rectangles of contours mark the regions with motion. Detected bounding rectangles are in figure 2.8.

3.1.2 Suitable Regions Creation

After locating the motion regions in an image which are marked for example in figure 3.1, a situation where some rectangles are inside each other can occur. People detection in these rectangles does not make sense, thus rectangles placed inside each other are removed.

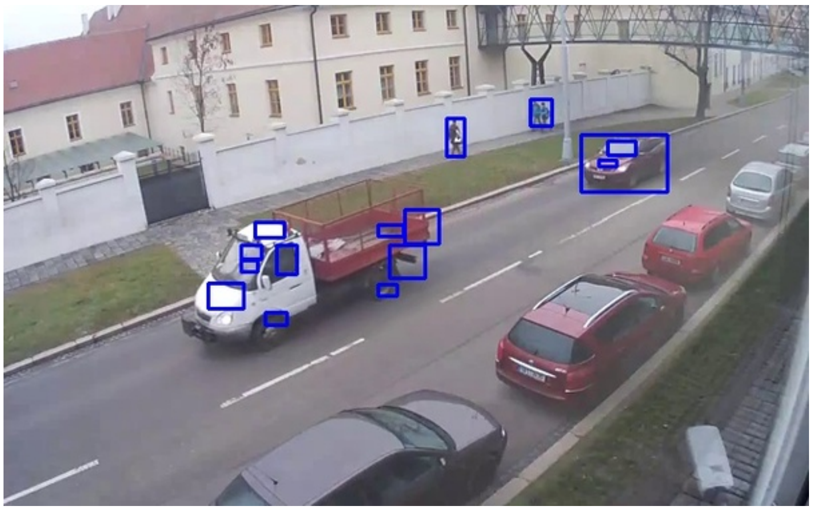


Figure 3.1: Motion bounding rectangles

Also, some contours in motion search process do not have to be continuous and some inaccuracy can occur. This problem is solved by joining close rectangles into a large one. If rectangles centers are closer than the predefined value, they are joined into one rectangle, with the usage of the most top-left corner and the most bottom-down corner of these rectangles. This join is made for all found motion regions in the whole image. The remove of inner rectangles and joining the close rectangles from figure 3.1 can be seen in figure 3.2. The example videos with frame differences and detected motion regions can be found on the included DVD in directory `video/preprocessing`.

3.1.3 Detection Process

As was mentioned, for people detection is used Doppia library. In figure 3.2 are marked regions containing motion. Only these regions are sent for people detection process. Sometimes motion detection do not find whole moving object, thus for better detection are added borders to all rectangles. In figure 3.3 are regions from figure 3.2 with enlarged borders. From the figure 3.3 is evident decrease of searched area for people detection with corresponding detection time decrease in comparison with the whole frame. The advantage is

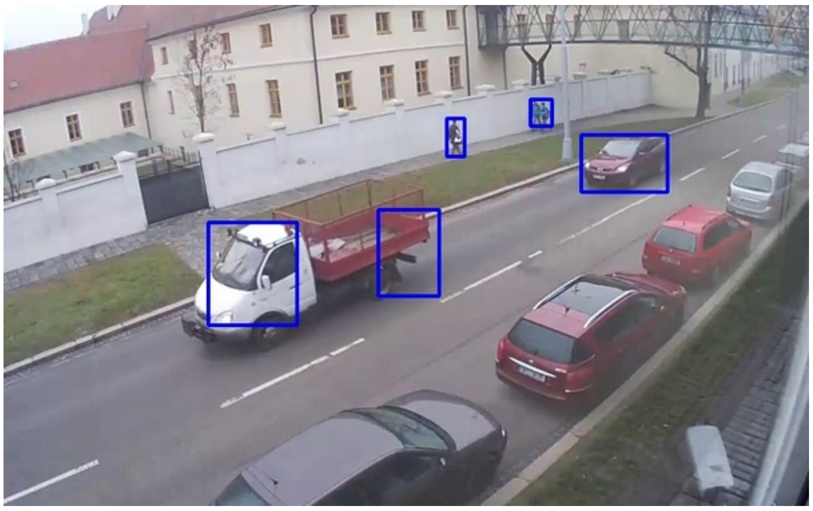


Figure 3.2: Motion bounding rectangles modification

also in fast skipping of frames without motion because no detection process is made in these frames.

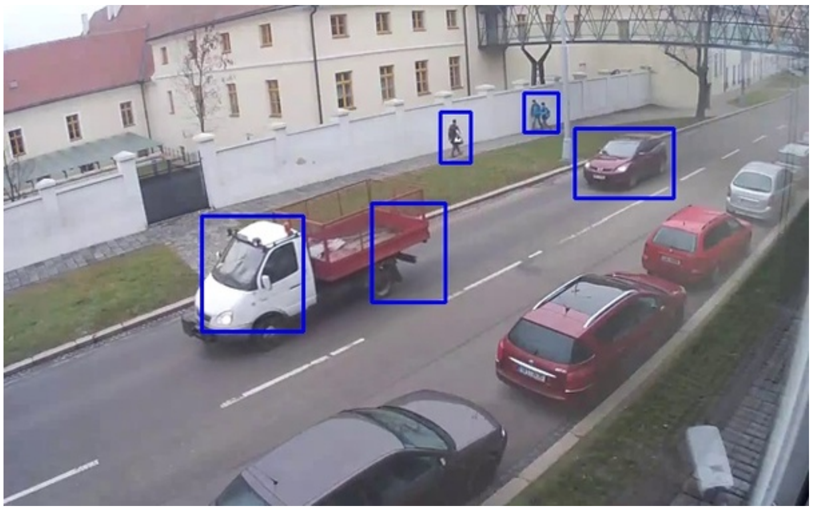


Figure 3.3: Motion regions with enlarged borders

Another improvement before people detection process made is selecting regions of interest (ROI) and the expected corresponding detection height. For every region of interest an initial expected detection height is set, which is used in detection process as initial sliding window height as mentioned in section 3.1.1. It allows to detect people with different heights in whole image. If only one sliding window height should be set, much more sliding window scales had to be done, to detect people in whole image, due to image perspective projection.

Regions of interest also provides technique to choose only those areas where is some motion expected. For example, it is sure there will be no detections in treetops or house wall, etc. There is necessary user interaction who has some scene information. The user can set regions of interest and expected detection height in graphical user interface, as described in section 4.1.2. The example of setting regions of interest and expected detection heights is shown in figure 3.4. Green rectangles mark regions of interest and yellow rectangles mark expected detection height for each region of interest.

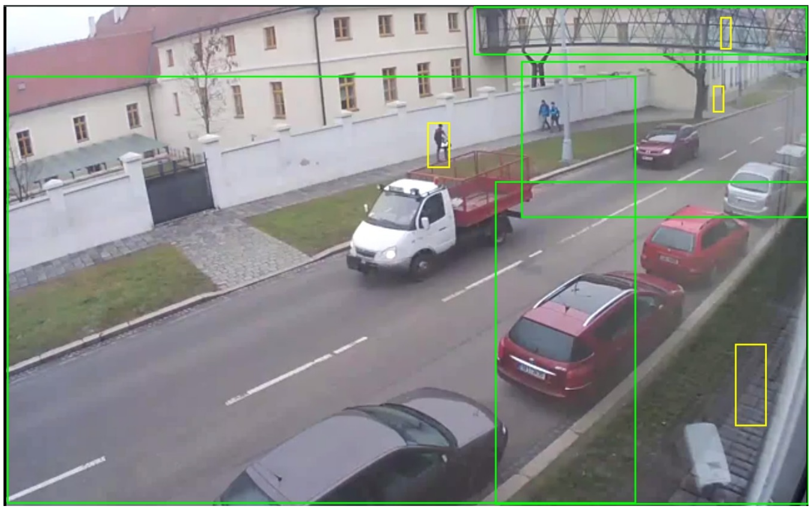


Figure 3.4: Regions of interest and expected detection height mark

Expected detection height is set through rectangles but only rectangle height is taken into account. Rectangle width is ignored, thus can be set arbitrary. Regions of interest can overlap each other. If the same detection in more regions is found, only one is saved and the others are skipped.

3.1.4 People Detection Output

Regions of interest are processed sequentially. Each region of interest is cropped from frame and is processed separately. One region of interest for the whole video sequence is processed and, consequently, another one starts its processing. After all regions of interest are processed, output detections can be saved into a file. This option is useful for better testing. As described in section 4.3, people detection process is the most time consuming part of the whole application. It would be disadvantageous to wait all the detection time every time new analysis is required. It is more comfortable to save detections into the file and when a new analysis should be done, read the saved detections from the file. Detections from all regions of interest are sorted due to frame number and are saved into the file. See detection file structure below:

```
DETECTIONS_FILE
filepath
4
678   123   23   45    -0.104659    16
674   127   23   45   -0.0601096    20
689   132   13   28   -0.0880687    20
662   127   30   61   -0.0311963    24
670   135   30   61    -0.083175    28
626   169   26   53    -0.106938    40
601   191   26   53    -0.106137    48
543   240   20   41    -0.108602    56
586   243   15   30   -0.0936339    56
587   222   23   45   -0.0532941    56
                          ⋮
```

The first line serves for correct file recognition. The second line contains an absolute file path of video file which detections belong to. On the third line, there is number n which sets how many frames are processed (every nth frame is processed, the rest is skipped). Then on each new line is saved one detection. Rectangle top-left corner of detection is saved on first two positions on the line. Next two positions describe detection width and height. The fifth position on the line is detection score, which expresses the measure of detection confidence. At the last position on the line is a number of frame where the detection occured.

The example output videos of the people detection process can be found on the included DVD in directory `video/detection`.

3.2 Tracking People Detections

People detection process gives information about people occurence in image. However, no information about motion are known. Tracking serves to follow an object in video sequence and provides information about the start of motions, their end and progression. The goal is to follow the object as long as it is possible, and to collect as many information about motion as possible. Typically, some features to object description are set and then followed by method which serves to motion computation. Two method are discussed here - Optical flow and CAMSHIFT.

3.2.1 Tracking Features Extraction

To realize object tracking it is necessary to extract some features from image, on which can be tracking process executed. Significant points in an image are suitable for optical flow algorithm described in section 2.2.2. Harris corners, described in 2.2.1, are convenient and can be used for tracking by optical flow. In OpenCV library optical flow implementation is available, where input points and two images are given as input parameters and new points positions are set at output. The example of optical flow computation done by Harris corners is in figure 2.21.

CAMSHIFT algorithm, described in section 2.2.4, is based on color tracking. CAMSHIFT is also implemented in OpenCV library. Histogram back projection image, initial search window and terminating criteria are the inputs. Rectangle, where object with corresponding histogram was found, is the output. The example of CAMSHIFT output is in figure 2.32.

3.2.2 Tracking Technique

As mentioned in section 3.2, two tracking methods are used (Optical flow and CAMSHIFT). Computing Harris corners for optical flow tracking consist in convert input image to grayscale image and set regions where Harris corners detection should be executed (people detections). Harris corner detection is implemented in OpenCV library.

CAMSHIFT features computing needs pre-processing. If histogram back projection of whole image should be done, a lot of noise can appear. Only the regions with motion are separated from the whole image. Thanks to the pre-processing, histogram back projection is more accurate. Regions with motion are separated in the same way as described in section 3.1. The only difference is that regions are not cropped from image, but they are marked into a mask. The example of a motion marked mask in an image is given in figure 3.7.

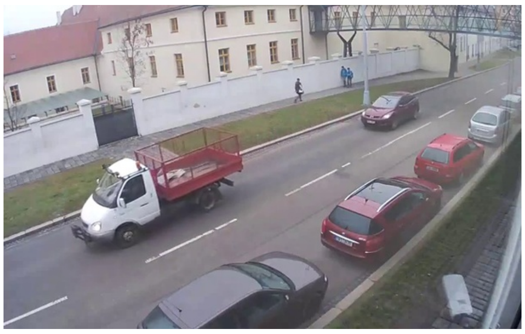


Figure 3.5: Input image

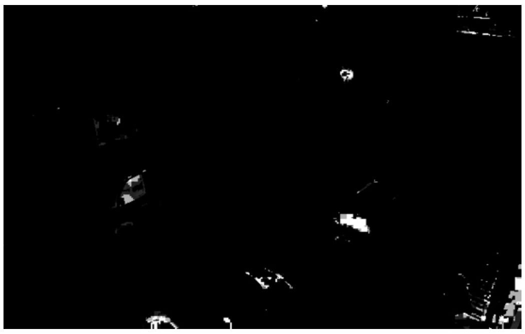


Figure 3.6: Histogram back projection

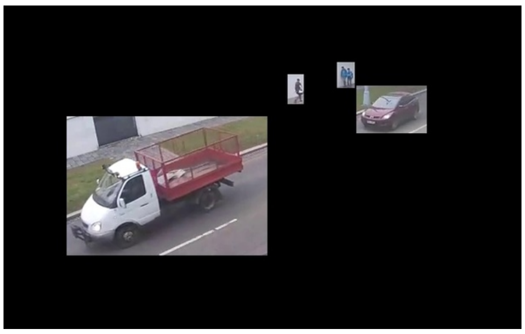


Figure 3.7: Image with motion only

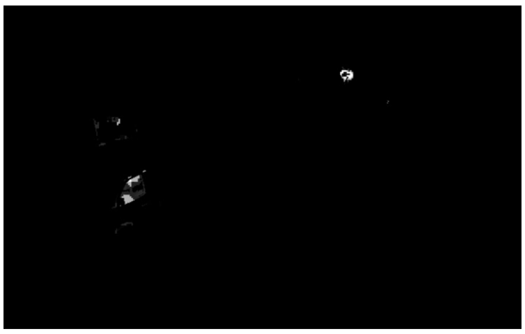


Figure 3.8: Histogram back projection

Figure 3.6 shows a computed histogram back projection of figure 3.5. The histogram for back projection is shown in figure 2.23 and is corresponding to figure 2.22. Figure 3.8 shows a computed histogram back projection of figure 3.7. The histogram is as in previous case. Lower noise for easy CAMSHIFT computation is apparent. CAMSHIFT computation is more accurate in image with removed non motion regions, due to lower noise.

The next steps are similar for both mentioned methods (Optical flow and CAMSHIFT). Tracking algorithm skeleton is in algorithm 6. New tracked object position (Harris corners by optical flow or window position by CAMSHIFT) is firstly computed for every processed

Algorithm 6 Tracking algorithm

```
for all Processing frames do
    Compute new tracked object position for all active trajectories
    if New computed position is not valid then
        Stop trajectory
    end if
    for All actual frame detections do
        if Detection confirms any active trajectory then
            if Detection score is high then
                Recompute confirmed trajectory features
            end if
        else New detection occurs - create new trajectory
        end if
    end for
end for
```

frame. Computed position is not valid in case it is too big or too small in comparison with initial size (first detection size or size set by features recomputation), or if the features do not change the position. In this case the trajectory is stopped. If the position is valid, the center of Harris corners or the center of search window is saved for trajectory description. All detections for actual frame are checked for trajectory confirmation (confirmation technique is described in section 3.2.3). If detection confirms any trajectory and has a high score (positive), the features of corresponding trajectory are recomputed with confirming detection, thus new Harris corners or new color histogram are computed, and the initial size for possible trajectory stop is set (detection height). If detection did not confirm any trajectory, a new trajectory is created on place of detection.

3.2.3 Track Confirmation

As described in algorithm 6, trajectories and detections confirm check is done with all detections in every processed frame. If there are no detections for actual frame, only new tracked object position is computed. Trajectory confirmation is based on the intersection of previous object position, and actually computed object position. In case of optical flow, computed Harris corners bounding rectangle is used, in case of CAMSHIFT, detected object window is used.

If an intersection area of previous object rectangle and actually computed object rectangle is at least 20 % of smaller rectangle area, the trajectory is considered to be confirmed. With a higher intersection area there often come trajectory break. If more detections can confirm the trajectory, the detection with the best score is selected and the other ones are skipped. The example of confirmation process with optical flow method is shown in figure 3.9. The blue rectangle marks tracked bounding Harris corners rectangle, the green rectangle marks detection, which confirms the trajectory. The red rectangle marks detection which did not confirm any trajectory and the yellow rectangle marks bounding rectangle for newly created Harris corners. The same situation is seen in figure 3.10, with usage of CAMSHIFT method. The blue rectangle marks CAMSHIFT tracking window, the green rectangle marks confirming detection and the yellow rectangle marks newly added trajectory (place where new histogram is computed), because of new detection without trajectory confirmation.

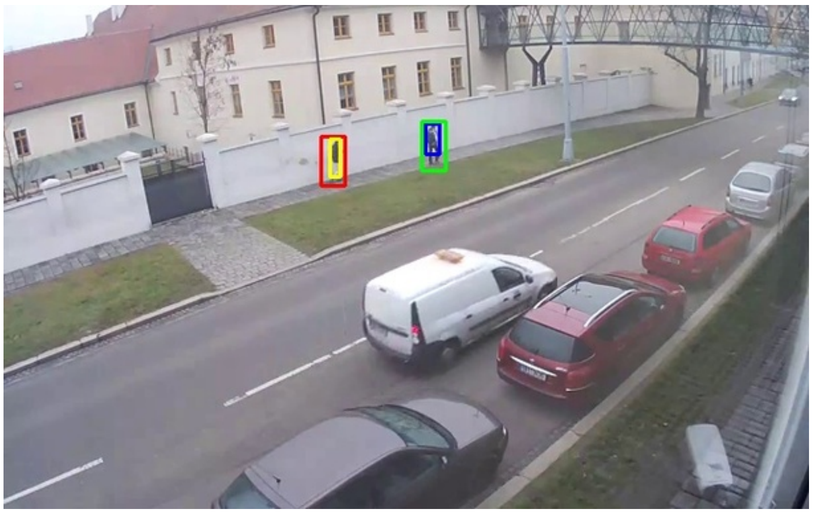


Figure 3.9: Trajectories confirmation process - Optical flow

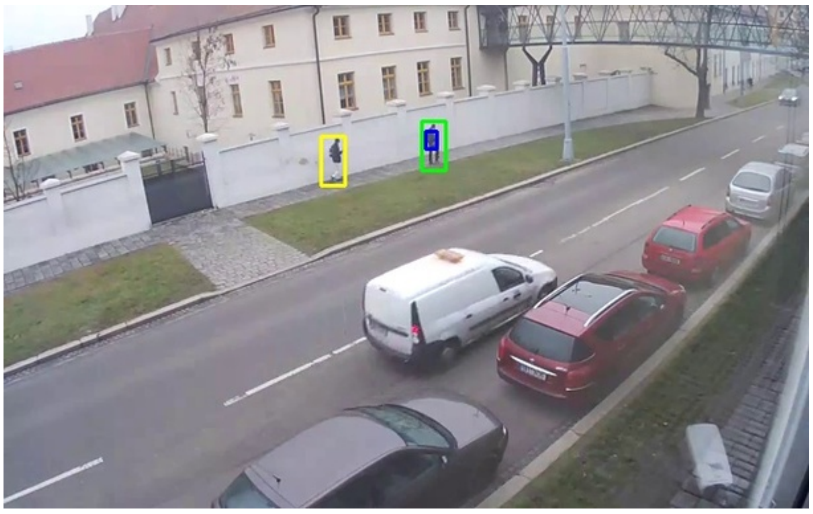


Figure 3.10: Trajectories confirmation process - CAMSHIFT

Every time a trajectory is confirmed by detection, counter of confirmations is increased and the score of detection is added to trajectory variable. When confirmation detection has a high score (positive), it expresses higher people detection confidence and special positive confirmation counter is increased. All these values are used in analysis part, as described in section 3.3.

3.2.4 Tracking Process Output

After tracking process is done, the information about people motions in video sequence are available. For every trajectory sequence of points describing the motion, count of confirmations by people detection, count of positive conformations and sum of confirmations scores is saved. The example of detected trajectory is shown in figure 3.11. The example outputs of the tracking process can be found on the included DVD in directory `video/tracking`. Both methods used (Optical flow and CAMSHIFT) are presented side by side.

Similar to the detection output, described in section 3.1.4, trajectories can be saved into

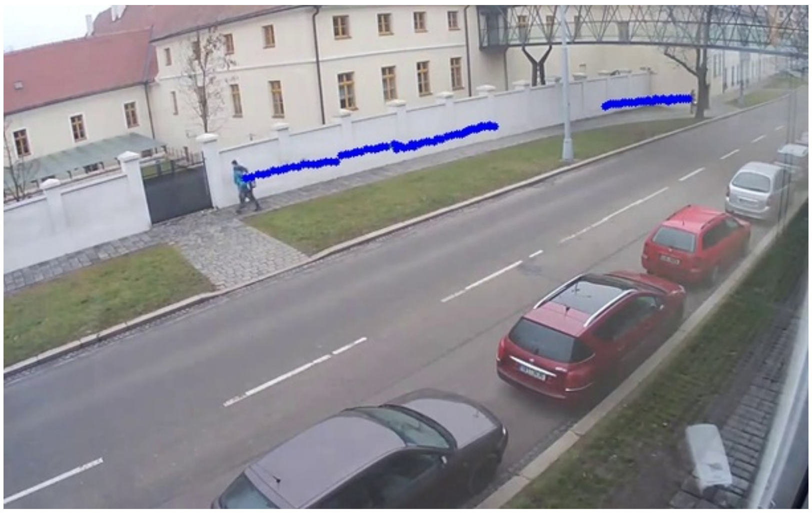


Figure 3.11: Detected trajectories

a file. Later, the trajectories can be load from the file, and the whole process of creating trajectories can be skipped. Start analysis with another parameters is much faster. All trajectories are saved, including weak trajectories which are, for example consisted of only one point. Good trajectories filtration is made in the analysis part, as described in section 3.3.1. Output image with trajectories has following structure:

```
TRACKS_FILE
filepath
4
TC 9442
TRACK    3       -0.0884743   1
179.5    57.5    8            40
178      57      12           40
177      56.5    16           47
176      57.5    20           47
175      57      24           47
174.5    56.5    28           47
TRACK    1       -0.0354231   0
347.5    255.5   56           42
348      255     60           42
349      266.5   64           42
349.5    266     68           42
TRACK    4       -0.302945    0
148.5    37.5    72           40
145.5    37.5    76           40
                 ⋮
```

The first line serves for correct file recognition. The second line contains absolute file path of video file, which trajectories belong to. On the third line there is number n, which sets how many frames are processed (every nth frame is processed, the rest is skipped). On the fourth line is the count of all trajectories saved in the file, for easy file loading. Then each trajectory begin with TRACK text and three values. The first value is the total number of track confirmations, the second value is the sum of all confirmations scores and the last value is the number of positive confirmations. Next, trajectory points until the next TRACK line are saved. Each trajectory point has saved position on the first two positions, on the third position is point frame number, and on the last position is the height of the first detection or last positive confirmative detection.

3.3 People Motion Mapping by Trajectory Analysis

Trajectory analysis is made for scene understanding and for getting a semantic information about the scene. There exists many different approaches to trajectory analysis. Some works are focused on this task, like [22] and [25], but they are often too complex and works with many factors (for example objects interaction), which are not meaningful for the goal set by this work. Sequential trajectory scanning approach is used in this work.

3.3.1 Trajectories Filtration

As mentioned in section 3.2.4, all detected trajectories are saved into a file or directly processed. Trajectories that are not correct are also loaded, thus some trajectories filtration must be done. Hence before the analysis, the trajectories which do not meet the set conditions are removed from the processing.

Several conditions can be set in the graphical user interface as described in section 4.1.2. The conditions are minimal length of trajectory, minimal count of confirmations, minimal count of positive confirmations and average confirmation score.

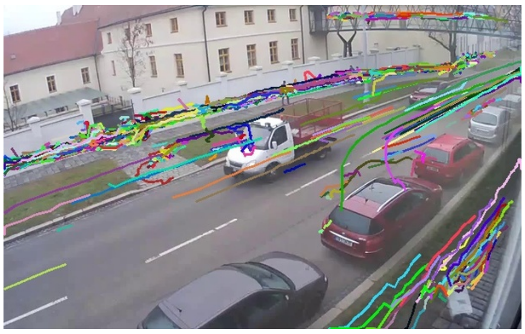


Figure 3.12: All trajectories

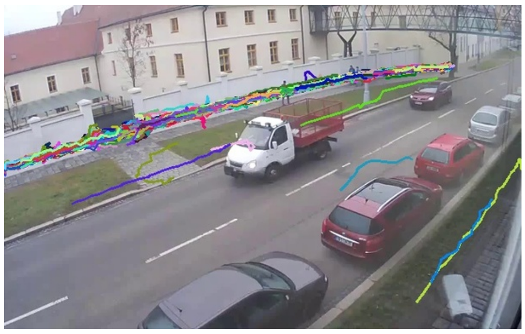


Figure 3.13: Confirmations constraint

In figure 3.12 almost all detected trajectories are marked. The only condition is to have at least the length of four points in the trajectory. Some false trajectories are present in this case because of very low detection score where false positive detections occurs. In figure 3.13 the same scene with the minimal count of confirmations set to 15 can be seen. The trajectories with a lower count of confirmations are skipped. In this case is not taken into account detection score. That designates the detection confidence, so the false trajectories can still be marked in some types of scenes.

In figure 3.14 the trajectories with at least 6 positive confirmations are marked. In this case, mainly trajectories with the high measure of confidence to be correct people trajectories are marked. In figure 3.15 the most limitative condition is shown. The average score of confirmation must be at least 0.0 and the count of confirmations must be at least 4. This condition removes the majority of trajectories because even good trajectories

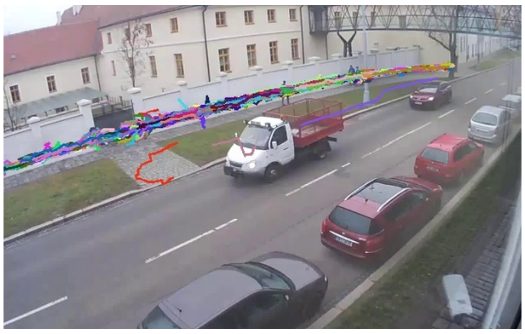


Figure 3.14: Positive confirmations constraint

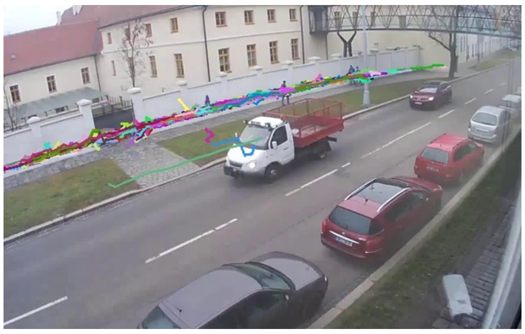


Figure 3.15: Average track score constraint

have some detections with a lower score and this condition requires, on average, a positive confirmation score.

All conditions can be combined together to reach the best possible output. Conditions setting depends on the type of scene, as described in section 4.2.

3.3.2 Typical Motion Regions Location

One of the basic information that is possible to extract from loaded trajectories is information about motion placement. The goal is to locate the places in the scene where the motion is the most common and where the motion did not appear.

For this purpose, a heat map which is described in section 2.3.1, can be used. It provides the information where the motion in a scene occurs. In figure 3.16 there is a scene with marked trajectories and in figure 3.17 there is a corresponding heat map. Places where the motion is more common are marked by color with a higher saturation value. All trajectories which are analysed are marked in heat map, including separate trajectories.

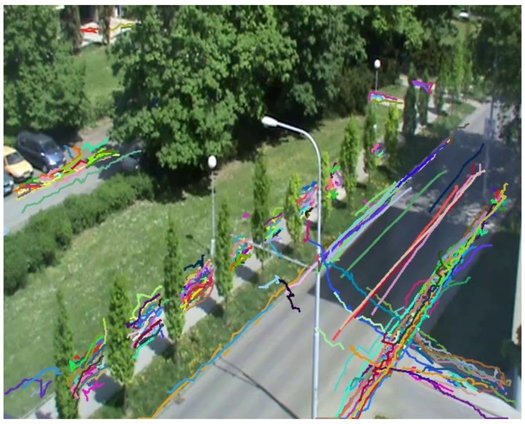


Figure 3.16: Marked trajectories

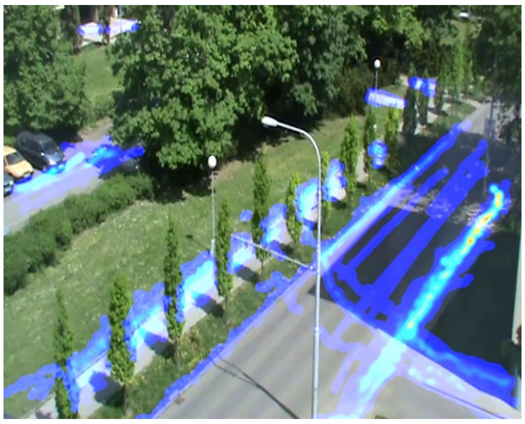


Figure 3.17: Corresponding heat map

The goal is to detect only the areas with a typical motion, so separate trajectories should not be included. For this purpose can be heat map values thresholded and borders of areas with high thresholded values can be found. The example output of this purpose can be seen in figure 3.18. The areas where is the highest density of motion occurence are marked.

Another proposed technique is based on clustering. It comes out from work [21], where trajectories similarity is computed and according to similarity are trajectories assigned to

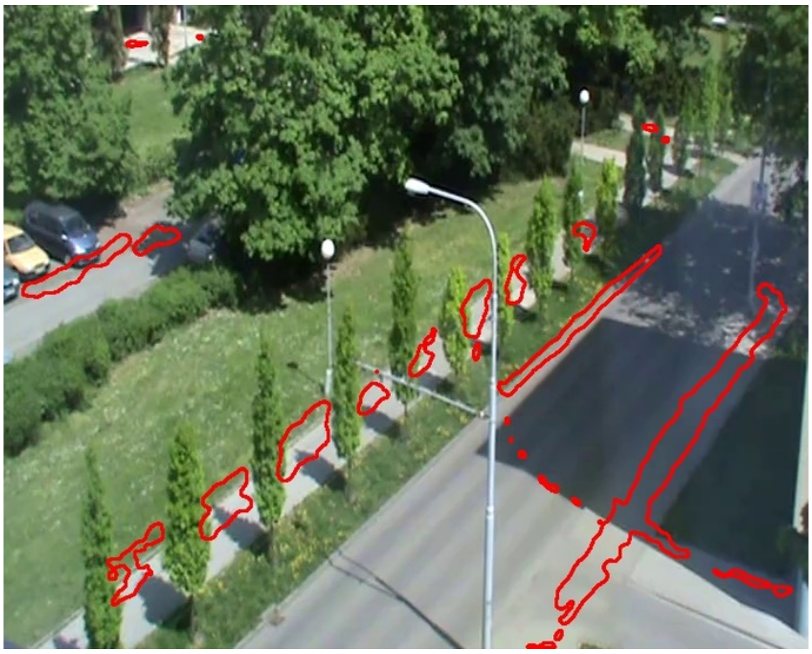


Figure 3.18: Areas with high density of motion occurence

clusters. In this work is approach a bit different. All trajectories points are assigned into large amount of clusters and these clusters are consequently merged on account of their distance. For clustering is used OpenCV library implementation of K-means++ algorithm, as described in section 2.3.2.

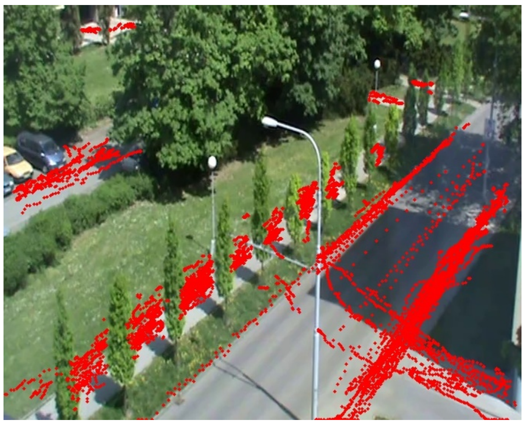


Figure 3.19: Trajectories points

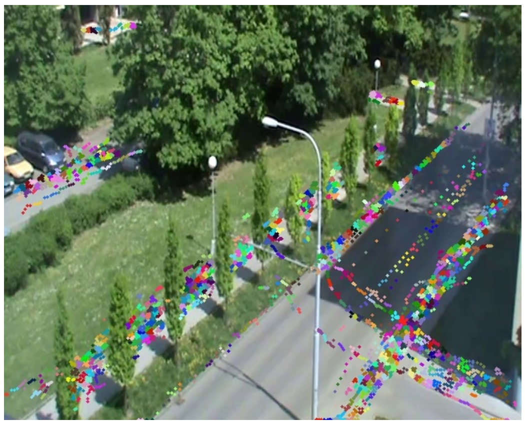


Figure 3.20: Points assigned to 750 clusters

Input trajectories points are in figure 3.19. Points assigned to 750 clusters are in figure 3.20. Created clusters are merged into groups with respect to initial average clusters centers distance. New group is marked as one of existing cluster centers and another centers are recursively inserted into same group, when they are close enough. Clusters merging approach is similar to flood fill algorithm. New groups creating continue, until all cluster centers are not assigned to some created group. Output are clusters centers assigned to groups, where each assigned center in group is closer to some other center in the same group than to any center in another group.

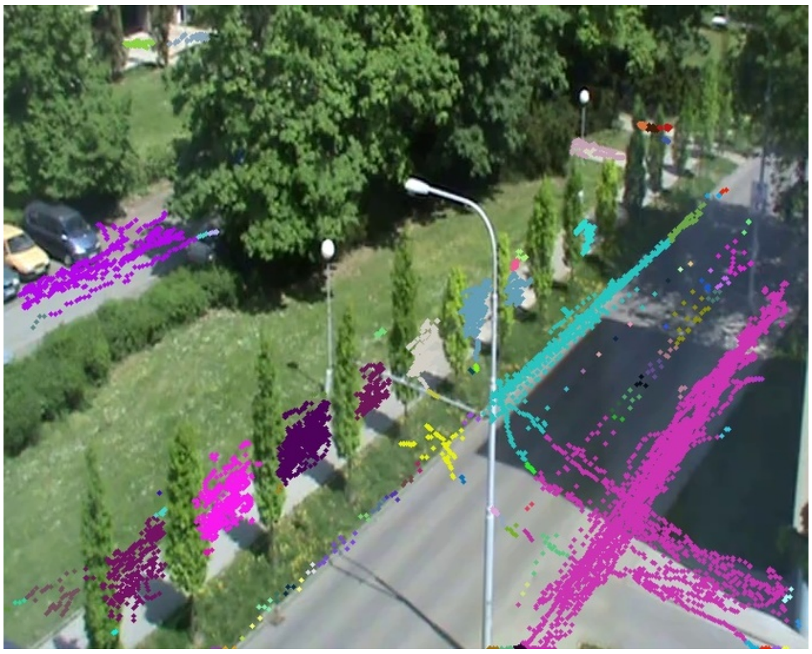


Figure 3.21: Merged clusters

In figure 3.21 clusters centers are merged to groups. The points which are close and almost continuous create always one group. The points which do not have other neighbour points than in the same trajectory are assigned to separate groups. The borders of groups, which meet condition of minimal containing points, are found and marked.

In figure 3.22 areas where typically motion occurs are marked. In comparison with heat map in figure 3.17, separate motions are not marked, because the groups with small amount of points are skipped. Threshold value is set in graphical user interface, as described in section 4.1.2. In comparison with the previous method based on heat map thresholding, there are not marked only areas with high motions density. Trajectories which are close to the places with high motion density are also marked.

3.3.3 Motion Direction Analysis

The next interesting information which can be extracted from trajectory analysis is typical direction of motion. Areas where people walk mainly in only one direction can be located.

The approach is based on the computation of direction vector for each two points of each trajectory with respect to trajectory creating sequence. According to direction vector a line between two trajectory points is marked to relevant mask. Masks for four directions are saved - left, right, up and down.

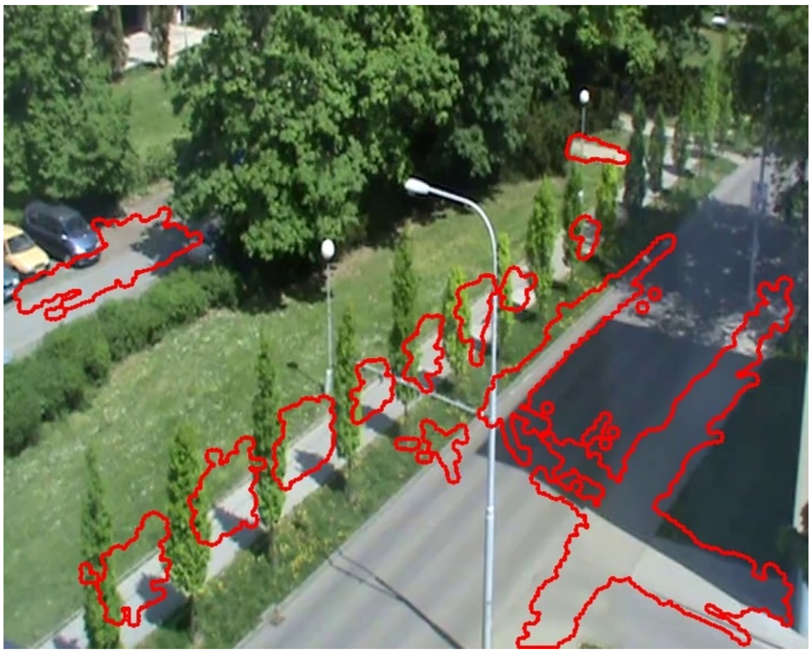


Figure 3.22: Merged clusters

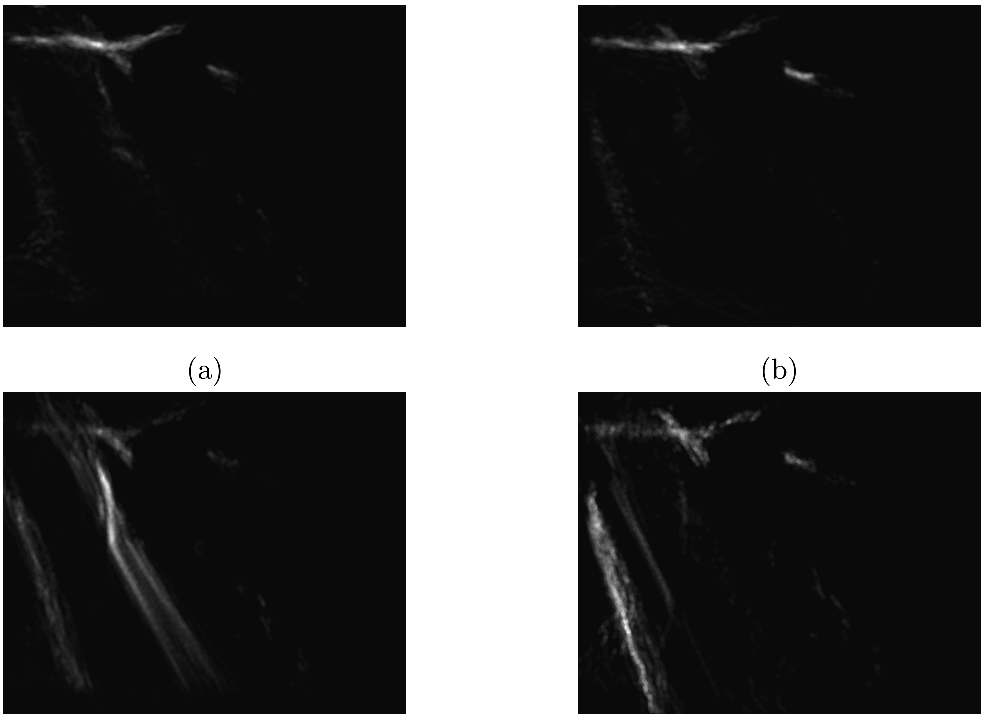

(a) (b) (c) (d)

Figure 3.23: Direction maps

Each direction mask represents 90° of possible direction, according to sectors in figure 3.24. If a direction vector points exactly in the middle of sectors, it is added to both relevant directions. In figure 3.23 there are examples of direction masks. In (a) there is left direction

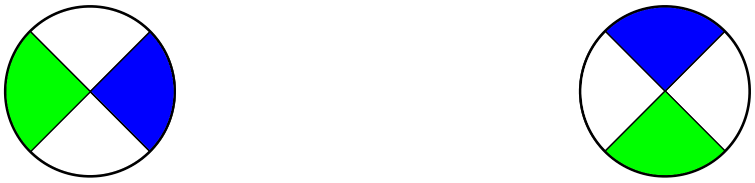


Figure 3.24: Horizontal and vertical sectors

mask, in (b) there is right direction mask, in (c) there is up direction mask, and in (d) there is down direction mask.

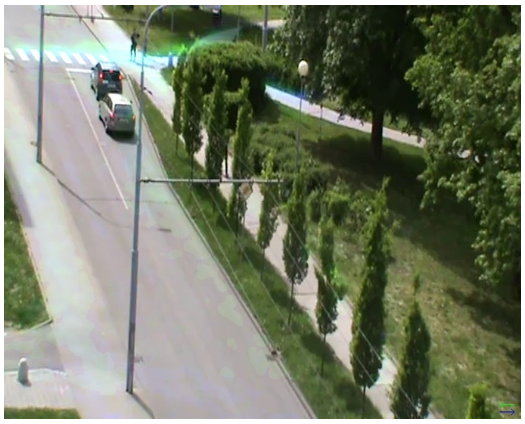


Figure 3.25: Horizontal direction

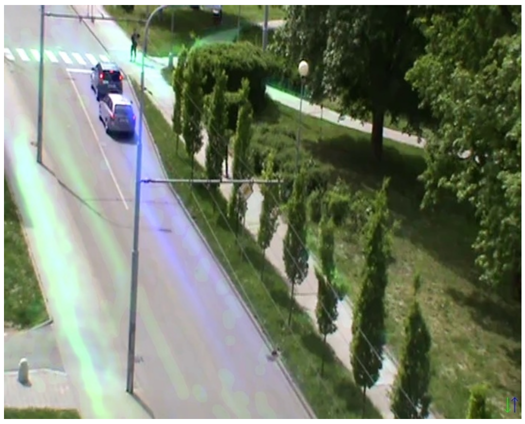


Figure 3.26: Vertical direction

Left and right mask are merged into one horizontal output image which can be seen in figure 3.25, with colors set according to figure 3.24. Similarly, up and down masks are merged into one vertical output image which can be seen in figure 3.26. Purposefully, the scene with kept car trajectories was processed to demonstrate direction analysis. According to figure 3.26, cars go on the right side of the road.

3.3.4 Exceptional Trajectories Detection

One of the goals of this work is to detect exceptional trajectories in scene. As an exceptional trajectory is regarded a trajectory, whose points have minimal intersection with points of the other trajectories. Exceptional trajectory has a unique way in comparison to another trajectories. The approach consists of two parts. Firstly, all trajectories are marked into image and then each trajectory is compared to be unique. To mark all trajectories into image, the empty image is created, and every trajectory is saved one by one.

The example of marked trajectories, corresponding to trajectories in figure 3.27, can be seen in figure 3.28. Separate trajectories are not visible, due to very low values. This is similar to heat map, with high values at places with high trajectories occurence, and low values at places with low trajectories occurence. The low values in map already mark exceptional trajectories. Trajectories in mask are wider than just one pixel. If a trajectory would be too narrow, numerous trajectories would be marked as the exceptional because many trajectories do not have exactly the same progression, but are close to each other, and cannot be marked as the exceptional ones. Locating the exceptional trajectories in mask is processed sequentially.

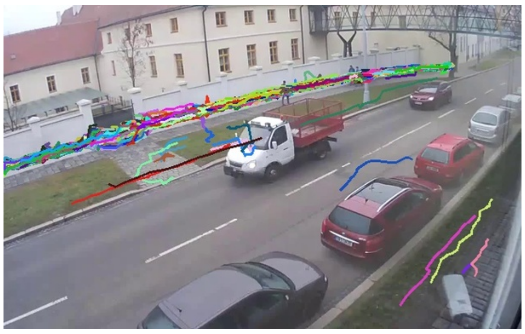


Figure 3.27: Input trajectories

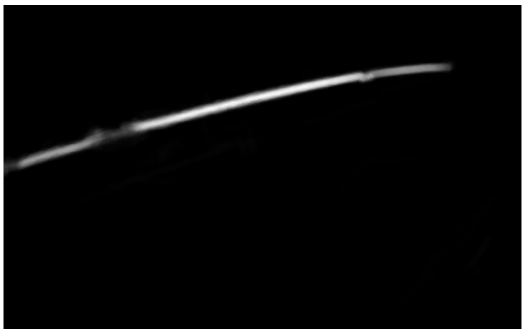


Figure 3.28: All trajectories map

Algorithm 7 Exceptional trajectories detection

```
for all Trajectories do
    Initialize  $start\_point$ = first trajectory point
    for all Trajectory points do
        Compute average trajectory value from $start\_point$
        if Average trajectory values is small enough then
            Set point distance from $start\_point$
        else
            if Point distance is large enough then
                Mark exceptional trajectory from $start\_point$ to actual point
            end if
            $start\_point$ = actual point
        end if
    end for
end for
```

Algorithm 7 describes the process of exceptional trajectories detection. Each trajectory is processed sequentially point by point. For every trajectory is searched as long as possible part, whose average value loaded from map is small enough (set suitably with respect to saved map values, so that trajectory can cross some trajectories and still can be marked as exceptional). If average value of processed track part is too large, the distance of processed part is checked. If the distance value is at least as large as value of parts first point detection height multiplied by entered parameter (parameter is set in graphical user interface as described in section 4.1.2), processed trajectory part is set as the exceptional one. Variable *start_point* holds beginning point of actually processed trajectory part. More parts of one trajectory can be marked as exceptional ones but they must have the distance value between start and end point at least as large as predefined value.

Figure 3.29 shows the detected exceptional trajectories from an input figure 3.27. In figure with exceptional trajectories the number for every trajectory is also saved. It is saved for distinguishing single trajectories because trajectories position and time are saved into a file. See the file structure below:

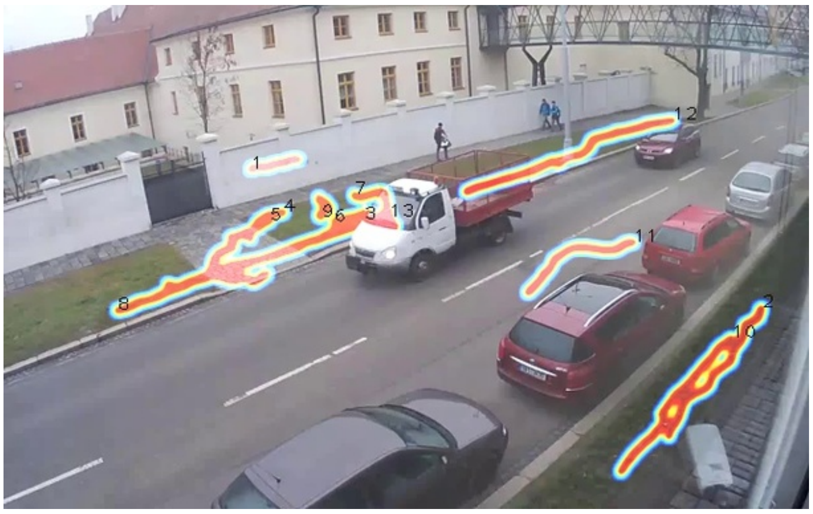


Figure 3.29: Detected exceptional trajectories

1	->	Pos:	196	130	Time:	00:00:30
2	->	Pos:	601	240	Time:	00:29:32
3	->	Pos:	286	170	Time:	00:42:38
4	->	Pos:	221	163	Time:	00:42:43
5	->	Pos:	211	171	Time:	00:42:51
6	->	Pos:	261	171	Time:	00:43:37

$\vdots$

Each line starts with a number which indicates the trajectory in the exceptional image as in figure 3.29. The next features are trajectory start position and time in a video sequence. Motions can be additionally searched in a video file with usage of these information.

3.3.5 Fast Moving Trajectories Detection

Another type of anomaly events are fast trajectories. These trajectories have a fast progression and typically runners or bikers stand for these events. This type of trajectories is detect by long distance between trajectory points. For a pedestrian trajectory is typical that trajectory points are close. When points are far, it signifies a fast motion.

Detection of fast trajectories algorithm 8 is similar to detection of exceptional trajectories algorithm 7. Again, every track is searched, and the part as long as possible, which meets condition, is searched. Every two following trajectory points are checked for condition

$$points\_distance/n > multiply * detection\_height \tag{3.1}$$

where *points_distance* is a distance between two processed points, n is a number which marks that every nth frame is processed, *multiply* is a predefined constant set in graphical user interface, as described in section 4.1.2 and *detection_height* is saved detection height of actually processed trajectory part first point. If two points are far enough, the part distance is accumulated until the points are still far or trajectory ends. If the distance value is at least as large as value of parts first point detection height multiplied by entered parameter (parameter is set in graphical user interface as described in section 4.1.2), pro-

Algorithm 8 Fast trajectories detection

```
for all Trajectories do
    Initialize  $start\_point$ = first trajectory point
    for all Trajectory following two points do
        if $points\_distance/n > multiply * detection\_height$ then
            $part\_distance += points\_distance$
        else
            if $part\_distance$ is large enough then
                Mark exceptional trajectory from $start\_point$ to actual point
            end if
            $start\_point$ = actual point
            $part\_distance = 0$
        end if
    end for
end for
```

cessed trajectory part is set as the fast one. More parts of one trajectory can be marked as fast ones.

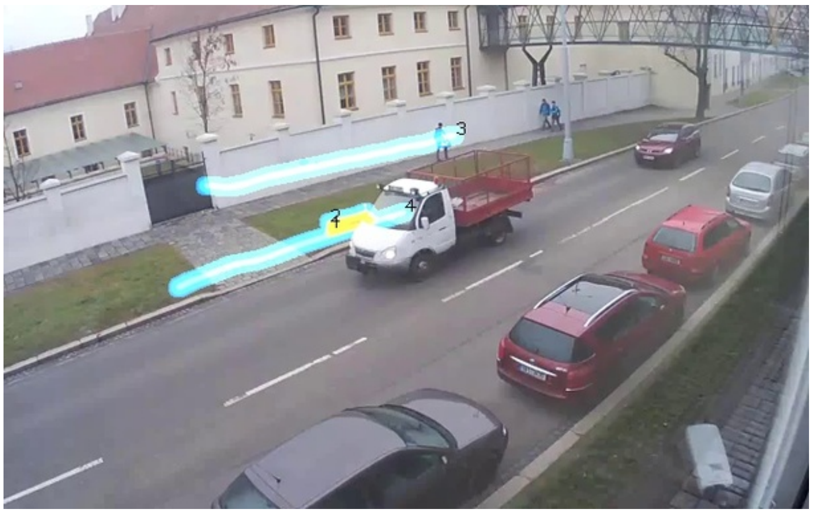


Figure 3.30: Fast trajectories detection

In figure 3.30 there are detected fast trajectories from an input figure 3.27. The number for every trajectory is again saved in the figure. Similar to exceptional trajectories detection, these numbers are saved for distinguishing single trajectories saved in a file. File is the same as the one described in the section 3.3.4. Again, positions and times of anomaly events are saved for additional event search in a video file.

Chapter 4

Surveillance Video Application and Results

In this chapter implementation of proposed approaches is described and the final application testing results are shown. In section 4.1, there is described how the final application is designed, how it is implemented, and what are different possibilities to execute application. Different scenes were recorded and the application was tested on these scenes. In section 4.2, the results of recorded scenes are discussed as well as the suitability of different scene types. In section 4.3 processing time durations over different scene types are discussed, and in section 4.4 the results of anomaly events detection in different scenes are shown.

Output video summarizing the whole application run can be found on included DVD in directory `video`.

4.1 Final Application

All techniques are implemented as described in chapter 3. The final application is implemented in C++ programming language. Doppia library is used for people detection, as mentioned in section 3.1. Because Doppia library is usable only in operating system Linux, the application can run only the same operating system. OpenCV library, which provides many implemented computer vision and image processing algorithms, is also used. For implementation of graphical user interface (GUI) is used Qt framework.

Application execution can be done through integrated development environment Qt Creator. The application was developed with the usage of Qt framework in version *5.4.1* with `gcc` compiler in version *4.6.1 64bit*. OpenCV library must be installed. Doppia library is included.

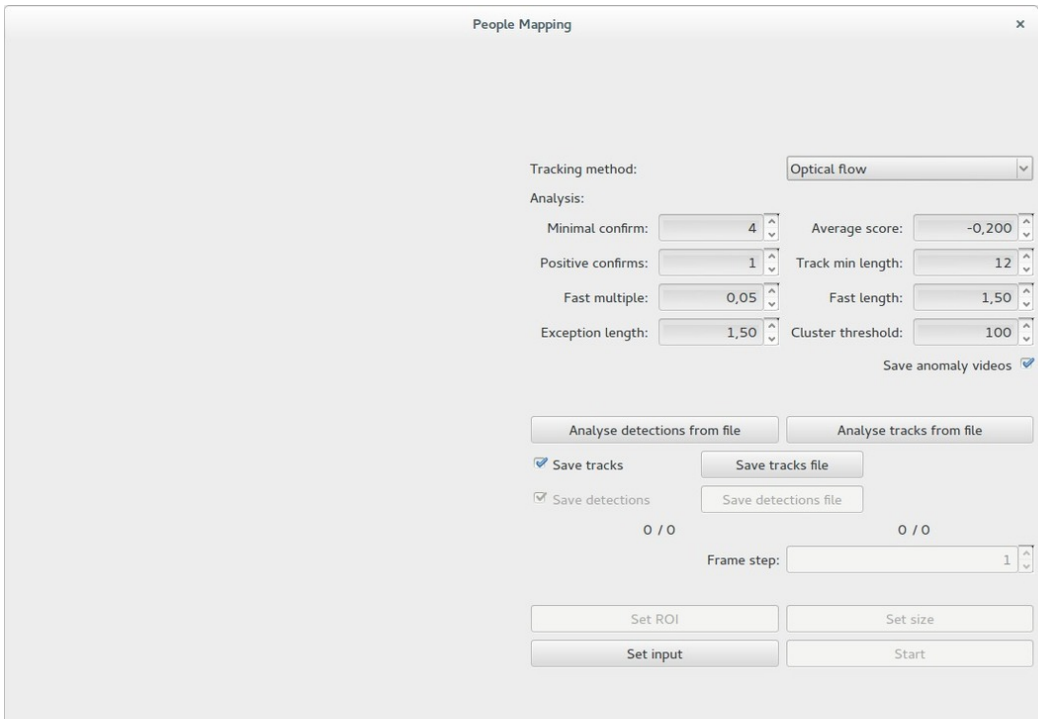
4.1.1 Application Modules

The whole application is structured into several modules. Each module has its own functionality. Application starts in *main.cpp* file, where `mainwindow` module is started. `Mainwindow` module provides the whole application control and displays graphical user interface. From `mainwindow` module another modules as needed are than started. The other modules are `pedestriandetection`, `pedestriantracking`, `pedestriananalysis` and `trackio`.

`Pedestriandetection` module serves for processing the pedestrian detection. It does all steps described in section 3.1. `Pedestriantracking` module execute pedestrian tracking, as described in section 3.2. Module `pedestriananalisis` execute trajectory analysis and saves detected anomaly events. All steps of analysis process are described in section 3.3. The last module `trackio` serves for detections and trajectories file operations. Detections and trajectories can be saved into a file and loaded from a file, as mentioned in sections 3.1.4 and 3.2.4. Separation application into modules provides future change of any module so, for example, another people detector can be used in future.

4.1.2 Graphical User Interface

Graphical user interface serves for execution of specific part of application and for setting parameters, which affects application processing. Three parts of application can be executed. Whole process including people detection, trajectories creation and trajectories analysis is first option. The second option is to load people detections from a file and execute only trajectories creation and trajectories analysis. And the last option is to load prepared trajectories from a file and execute only trajectories analysis. As described in section 4.3, every time is more convenient to load data from a file if it is possible. First, it is necessary to execute the whole process with all parts and save detections and trajectories into a file, and then these data can be loaded for following processing with other possible parameters.



Tracking method: Optical flow

Analysis:

Minimal confirm:	4	Average score:	-0,200
Positive confirms:	1	Track min length:	12
Fast multiple:	0,05	Fast length:	1,50
Exception length:	1,50	Cluster threshold:	100

Save anomaly videos ✓

✓ Save tracks

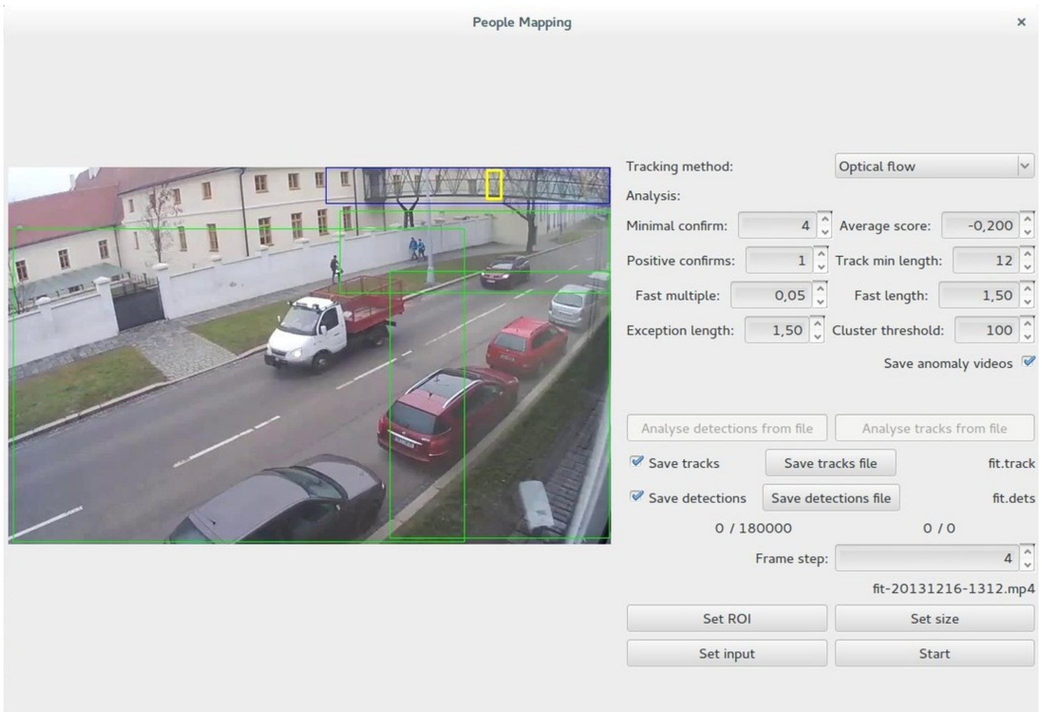
✓ Save detections

0 / 0 0 / 0

Frame step: 1

Figure 4.1: Graphical user interface

In figure 4.1 there is graphical user interface of the application. When application starts, the user can choose which part of application will be executed. If the button `Analyse detections from file` is pressed, the user will choose the file, where the detections are saved then trajectories creation and analysis will be done. If the option `Save tracks` is checked, created trajectories will be save into a selected file. If the user choose option `Analyse tracks form file`, trajectories are loaded from the file, and trajectories analysis will be done. The whole process including people detection is executed with option `Set` `input`. The video file is opened and the further setting is allowed.



Tracking method: Optical flow
Analysis:
Minimal confirm: 4 Average score: -0,200
Positive confirms: 1 Track min length: 12
Fast multiple: 0,05 Fast length: 1,50
Exception length: 1,50 Cluster threshold: 100
Save anomaly videos ✓
✓ Save tracks fit.track
✓ Save detections fit.dets
0 / 180000 0 / 0
Frame step: 4
fit-20131216-1312.mp4

Figure 4.2: Graphical user interface - `Set input` option

In figure 4.2 there is graphical user interface with an input setting. The user can set regions of interest and the expected detections height, as described in section 3.1, with buttons `Set ROI` and `Set size`. At least one region of interest must be set. If the option `Save tracks`, respectively `Save detections` is set, the output file for save data must be set with `Save tracks file`, respectively `Save detections file` button. With the help of spin box `Frame step` a value which determines step between processed frames (every nth frame is processed) can be set. Check box `Tracking method` can be used to set which tracking method will be used (Optical flow or CAMSHIFT). By pressing the `Start` button the whole process starts.

Analysis Parameters

In figure 4.2 a lot of possible analysis parameters, which can be set, are visible. All these parameters are used in trajectory analysis part.

- `Minimal confirms`

  Value defines how many confirmations must belong to a trajectory to be processed by trajectory analysis. Trajectories with less confirmations number are ignored. The option is also described in section 3.3.1.

- `Average score`

  Value defines minimal average score of trajectory confirmations. Trajectories with lower average confirmation score are ignored. The option is also described in section 3.3.1.

- `Positive confirms`

  Value defines how many positive confirmations must belong to a trajectory to be processed by trajectory analysis. Trajectories with less positive confirmations number are ignored. The option is also described in section 3.3.1.

- `Track min length`

  Value defines a minimal trajectory length (number of points describing trajectory) which a trajectory must contain to be processed. Otherwise, the trajectory is ignored. The option is also described in section 3.3.1.

- `Fast multiple`

  Value defines *multiply* constant for mark trajectory as fast event. As described in section 3.3.5, trajectory points must meet the condition in equation 3.1. If the value is for example 0.05, than the distance between points must be at least 5 % of first point detection height for trajectory can be marked as fast.

- `Fast length`

  Value defines a constant which is used for marking a trajectory as the fast one. Algorithm 8 says that the trajectory part distance must be large enough. For this purpose this value is given. If the value is for example 1.5 than the distance must be at least 1.5 multiple of detection height of the first point of trajectory part.

- `Exception length`

  Definition is similar to `Fast length` value. It is used for exception trajectories detection in algorithm 7.

- `Cluster threshold`

  Value defines threshold for selecting clusters to be shown in output image marking typical motion areas. As described in 3.3.2, clusters which contain at least specified amount of points are marked as a part of typical motion area.

- `Save anomaly videos`

  If the option is checked, videos with anomaly events are saved, as described in section 4.4.

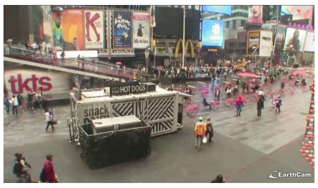
4.2 Different Scenes Suitability

For testing the application different scenes were used. Several videos were recorded. The list of videos is in table 4.1. These videos were used for application testing. Short previews of testing videos are on the included DVD in directory `video/input` (because of space limits, only 5 minute long previews are included) and application output results are in directory `output`.

Video name	Scene type	Video length	Preview in figure 4.3
timesSquare.mp4	Crowded scene	00 : 42 : 48	(1)
templeBar.mp4	Crowded scene	01 : 03 : 08	(2)
crossing.mp4	Traffic containing scene	03 : 13 : 57	(3)
fit.mp4	Traffic containing scene	02 : 00 : 00	(4)
road.mp4	Traffic containing scene	03 : 13 : 51	(5)
aghHall.mp4	Scene without traffic	01 : 14 : 17	(6)
playground.mp4	Scene without traffic	02 : 22 : 01	(7)

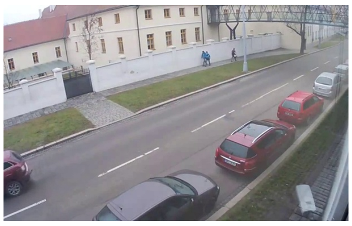
Table 4.1: List of testing videos

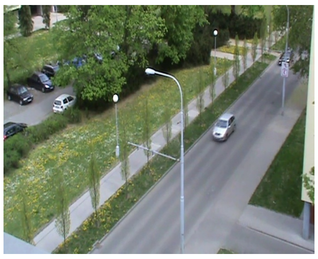
The previews of testing scenes are in figure 4.3. Different types of scenes were tested and the application behavior is evaluated.

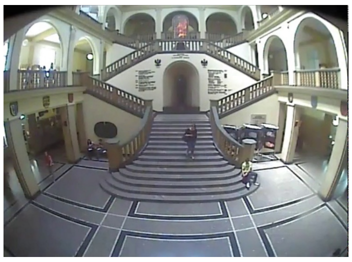

(1)

(2)

(3)


(4)


(5)


(6)

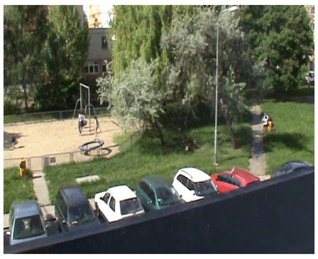

(7)

Figure 4.3: Testing scenes preview

For application results evaluation the following values are used:

(D-1) **Total people detections count**

Count of all people detections in the scene.

(D-2) **Positive people detections count**

Count of positive people detections in the scene.

(D-3) **Average detections count per video frame**

The average count of people detections per one frame of input video.

(D-4) **Average detections score**

The average score of all people detections in the scene.

(T-1) **Total trajectories count**

Count of all trajectories in the scene.

(T-2) **Total length of trajectories**

Total length (number of points describing trajectory) of all trajectories in the scene.

(T-3) **Average trajectory length**

The average length (number of points) of trajectory in the scene.

(T-4) **Average confirmations per trajectory**

The average count of confirmations per trajectory in the scene.

(T-5) **Average positive confirmations per trajectory**

The average count of positive confirmations per trajectory in scene.

4.2.1 Crowded Scenes

Crowded scenes are such scenes, where a high occurence of people is typical. These scenes are typically squares or streets where a high people frequency is typical. Scenes (1) and (2) from table 4.1 can be considered as crowded scenes. In table 4.2 the results of these scenes processing are contained. All trajectories without any filtration were processed.

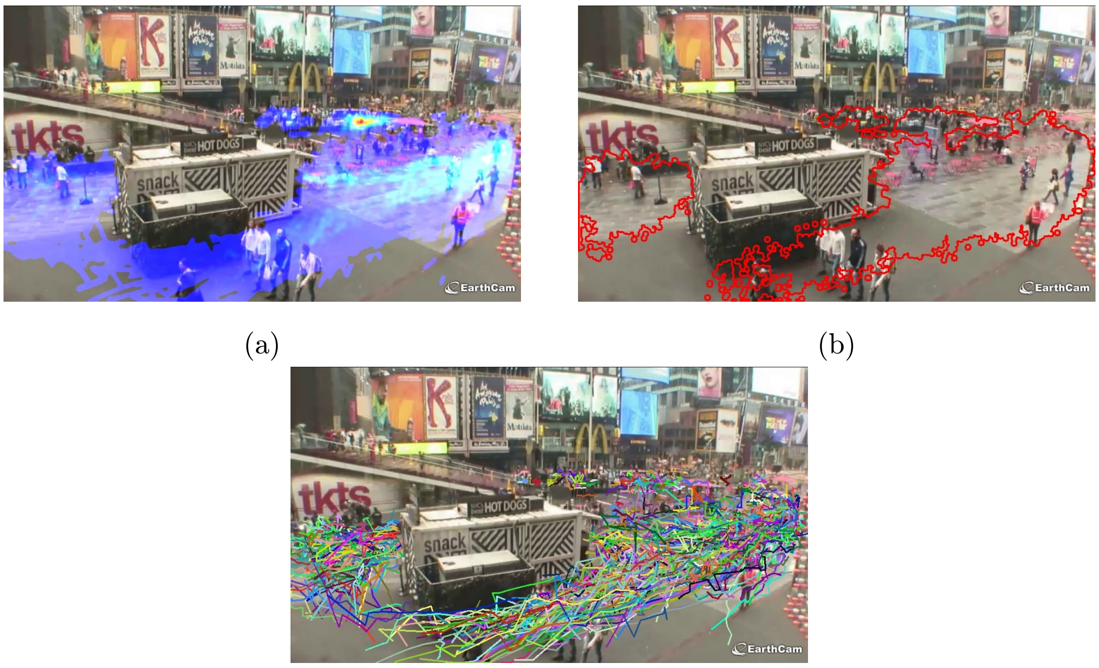
		Scene (1)	Scene (2)
Detections	(D-1)	134 994	160 726
	(D-2)	9 791	11 849
	(D-3)	6.306	1.697
	(D-4)	-0.067	-0.065
Tracking Optical flow	(T-1)	18 425	18 919
	(T-2)	110 966	199 963
	(T-3)	6.023	10.569
	(T-4)	2.690	4.665
	(T-5)	0.636	1.067
Tracking CAMSHIFT	(T-1)	6 429	14 516
	(T-2)	15 041	150 468
	(T-3)	2.340	10.366
	(T-4)	10.909	5.751
	(T-5)	2.025	1.213

Table 4.2: Crowded scenes results

From table 4.2 it is apparent, that crowded scenes contain a high number of detections and trajectories. Many of them can be skipped during trajectories analysis process, as shown in table 4.6. In crowded scenes a high number of average detections per frame is typical. In scene (1) it is 6.306 and in scene (2) it is 1.967. In other scenes, as apparent from tables 4.3 and 4.4, the value of average detections per frame is 0.580 at maximum and it is often even less.

Scene (1) is not suitable for both tracking methods, because of high color diversity in the background. In this case CAMSHIFT method reaches a bad result, although the average confirmations per trajectory value is high (10.909). It is caused by high number of detections per frame and the trajectories are often confirmed with an incompetent detection, so then the trajectories are not correct and it can jump to another person. The results are suitable mainly for detecting areas with a typical motion.

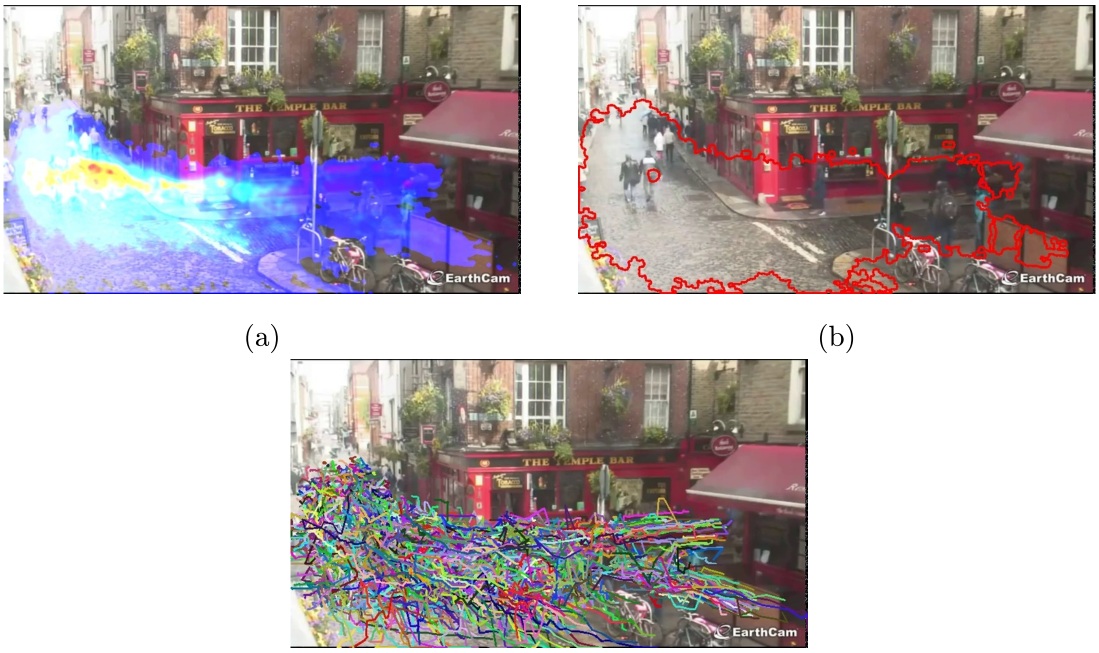
Example results from crowded scenes are present in figures 4.4 and 4.5. As described in section 4.4, crowded scenes are not suitable for anomaly events detection. In crowded scenes, a lot of false detections do not occur as in case of scenes containing traffic, so the analysis parameters setting does not have to be strict.



(a) (b)

(c)

Figure 4.4: Scene (1) results



(a) (b)

(c)

Figure 4.5: Scene (2) results

4.2.2 Traffic Containing Scenes

Scenes with traffic do not typically contain so many people as the crowded scenes. These are typically roads with pedestrian crossings or scenes where are walkways along roads. As traffic containing scenes can be marked scenes (3), (4) and (5) from table 4.1. In table 4.3 the results of processing these scenes are contained.

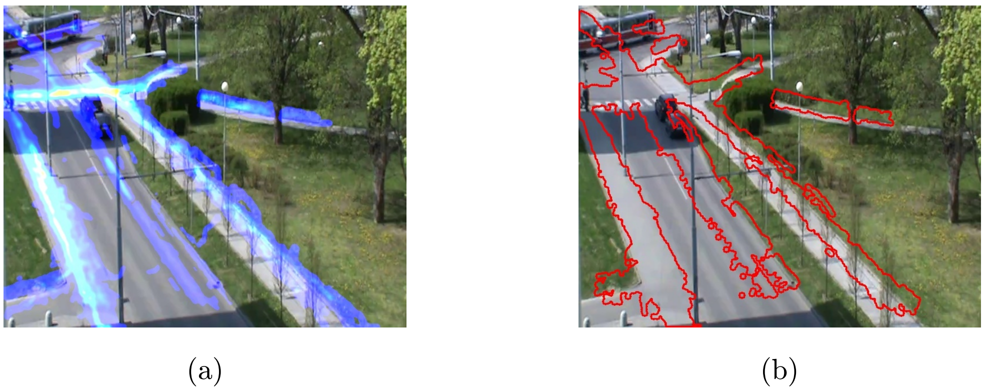
		Scene (3)	Scene (4)	Scene (5)
Detections	(D-1)	152 560	38 719	62 546
	(D-2)	9 185	6 387	5 693
	(D-3)	0.524	0.215	0.215
	(D-4)	-0.069	-0.048	-0.066
Tracking Optical flow	(T-1)	16 019	2 439	6 862
	(T-2)	196 266	39 573	70 012
	(T-3)	12.252	16.225	10.203
	(T-4)	4.250	10.135	4.147
	(T-5)	0.765	3.411	1.225
Tracking CAMSHIFT	(T-1)	14 218	2 255	6 206
	(T-2)	151 720	26 005	59 496
	(T-3)	10.671	11.532	9.587
	(T-4)	4.275	8.139	4.006
	(T-5)	0.752	2.836	1.015

Table 4.3: Traffic scenes results

In scene (3) there is a high number of detection because of a high motion occurence but many of them are false vehicles detections. These trajectories must be filtered in analysis process, as described in section 3.3.1. This represents a problem in all scenes containing traffic. Trajectories filtration is more complicated and must be set conveniently.

High average count of confirmations per trajectory value (10.135 by optical flow and 8.139 by CAMSHIFT) is exceptional in scene (4). In all other scenes this value is markedly lower. This is caused by convenient detection background in scene (4). Tracking features computation is made often in front of a white wall. There is not many disturbing elements in the background, and the features are computed better. This is valid for both tracking methods. For the same reason average positive confirmations per trajectory value (3.411) is in scene (4) markedly higher than in other scenes.

The example result are shown in figures 4.6, 4.7 and 4.8. In scene (5) the tendencies of people walking across road, although there is not pedestrian crossing are apparent. Many of these trajectories are detected as exceptional ones, as shown in section 4.4. This type of scenes is suitable for anomaly events detection, as mentioned in section 4.4.



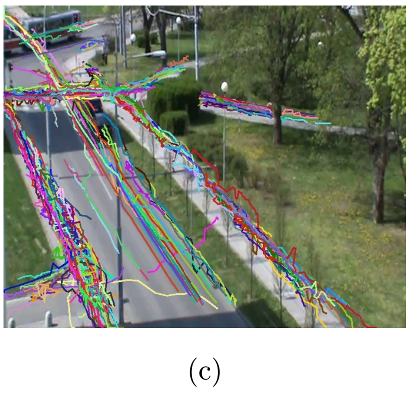


Figure 4.6: Scene (3) results

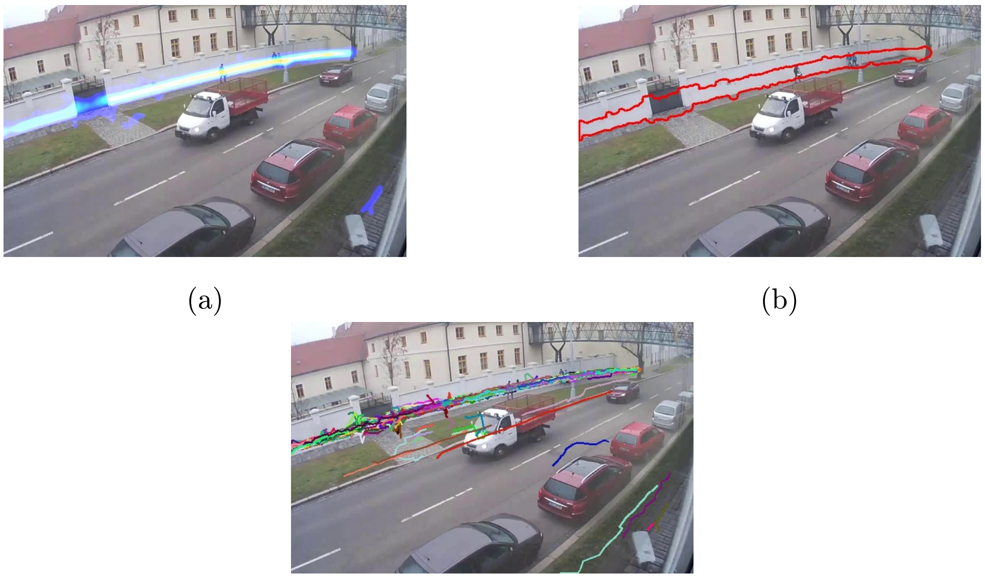

(a) (b) (c)

Figure 4.7: Scene (4) results

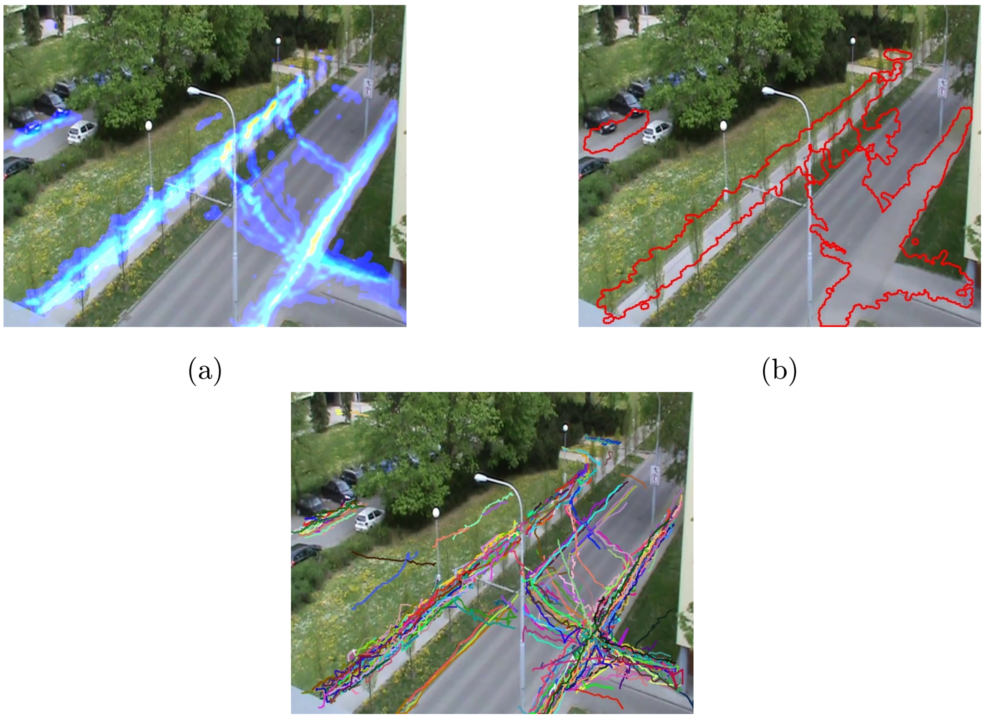

(a) (b) (c)

Figure 4.8: Scene (5) results

4.2.3 Scenes without Traffic

Another type of tested scenes are the scenes where no traffic occurs but they cannot be marked as crowded scenes, because there are not so many people. These scenes are typically parks or offices. From the tested scenes the scenes (6) and (7) form table 4.1 can belong to this type scenes. The results of these scenes processing are in table 4.4.

		Scene (6)	Scene (7)
Detections	(D-1)	33 591	39 383
	(D-2)	1 614	3 571
	(D-3)	0.580	0.185
	(D-4)	-0.071	-0.060
Tracking Optical flow	(T-1)	4 070	4 912
	(T-2)	36 644	42 514
	(T-3)	9.003	8.655
	(T-4)	3.147	3.881
	(T-5)	0.431	0.814
Tracking CAMSHIFT	(T-1)	4 054	4 497
	(T-2)	32 715	45 657
	(T-3)	8.070	10.153
	(T-4)	3.072	4.113
	(T-5)	0.441	0.801

Table 4.4: Scenes without traffic results

Scenes without traffic are suitable for processing. There are not so many detections as in crowded scenes and risk of trajectory jump is low. As described in section 4.3.3, these scenes are processed very fast. There also are not vehicles false positive detections as in traffic containing scenes, so trajectories filtration does not have to be so strict.

Scene (7) is the only scene, where tracking method CAMSHIFT reaches better results than optical flow method. This is caused by strong background colors in scene. Green grass and sand areas are unicolored, thus back histogram computation, as described in section 2.2.3, can be more accurate. Computation is not so influenced by background colors as for example in scene (1).

Example results are in figures 4.9 and 4.10.

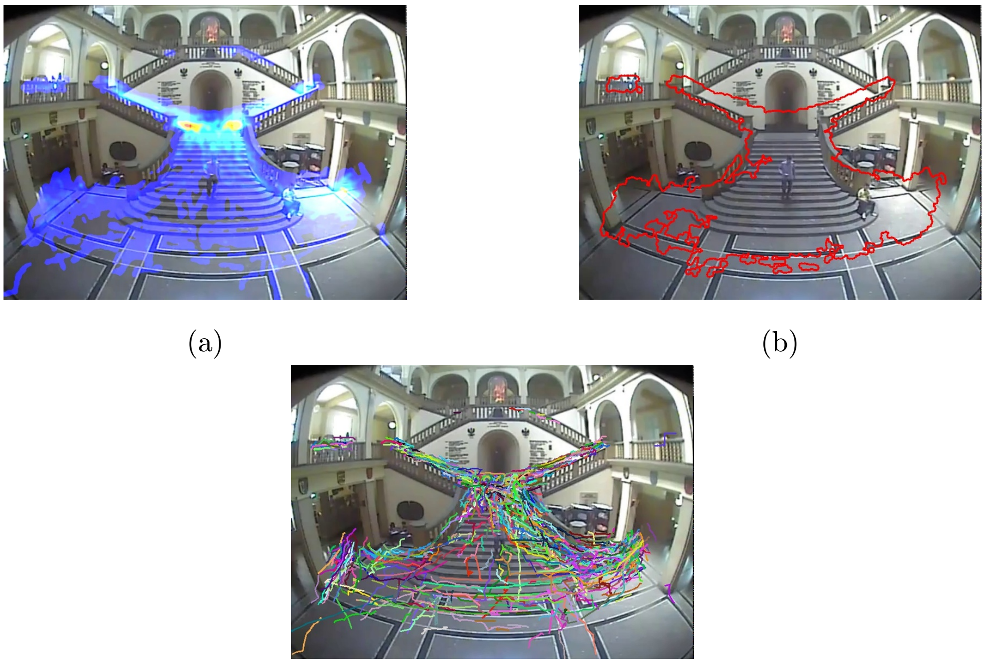


Figure 4.9: Scene (6) results

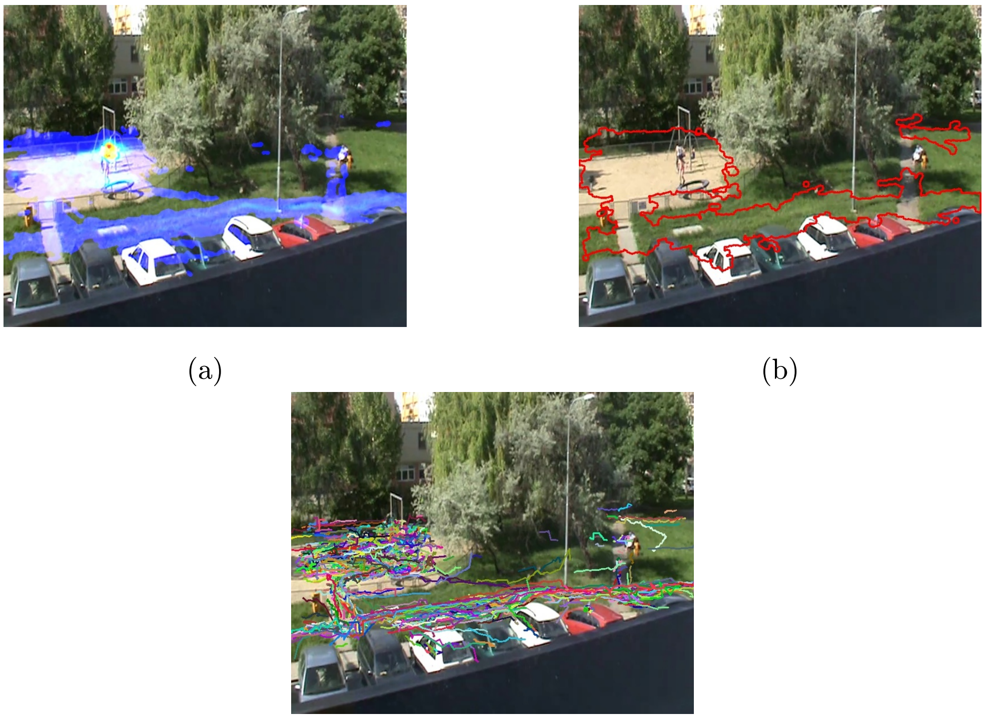


Figure 4.10: Scene (7) results

4.3 Processing Time Duration

Another application observed factor is processing time duration. The tendency is to create a real-time runnable application. Although improvements for acceleration have been done, a real-time processing has not been reached. For possible real-time processing should be convenient to use GPGPU, but the application is implemented only with the usage of CPU. The input video is processed in three steps. First, people detection is made. Second, trajectories creation is made, and lastly trajectory analysis is made. Each part of the whole process is measured separately.

People detection time can be influence by setting regions of interest and stepping between processed frames, as described in section 4.1.2. Theoretically, the infinite settings of regions of interest are possible, so only the most suitable setting is tested. Lots of values of steps between frames are also possible, and the result is direct proportional to set value, thus only one convenient value is tested. Tracking can be influenced only by method selection (Optical flow and CAMSHIFT) thus both methods are tested. Most parameters can influence trajectories analysis, so more settings of parameters are tested, as they are described in table 4.5. All trajectory analyses are made with optical flow computed trajectories.

Firstly, detections into a file are saved and afterwards trajectories creation is processed. Created trajectories are again saved into a file and analysis is started with trajectories loaded from the file. Saving data into files in more comfortable because other parameters of action can be set and previous steps can be skipped, because the data from previous steps are loaded from the file. This technique provides time saving, when more parameters settings are tested. All measurements were made on computer with processor Intel® Pentium CPU 1.50GHz.

Setting-1	Value
Minimal confirm	0
Average score	-0.2
Positive confirm	0
Track min length	4
Fast multiple	0.05
Fast length	1.5
Exception length	1.5
Cluster threshold	100

Setting-2	Value
Minimal confirm	10
Average score	-0.2
Positive confirm	0
Track min length	6
Fast multiple	0.05
Fast length	1.5
Exception length	1.5
Cluster threshold	100

Setting-3	Value
Minimal confirm	0
Average score	-0.2
Positive confirm	4
Track min length	6
Fast multiple	0.05
Fast length	1.5
Exception length	1.5
Cluster threshold	100

Setting-4	Value
Minimal confirm	10
Average score	-0.05
Positive confirm	4
Track min length	10
Fast multiple	0.1
Fast length	1.2
Exception length	1.2
Cluster threshold	30

Table 4.5: Used parameters for trajectories analysis time measurement

4.3.1 Crowded Scenes

For crowded scenes, as described in section 4.2.1, long processing duration is typical. There are a lot of people in scene and people detection process is made very often. Detection time can be reduced with a higher frame step, when more frames are skipped, but worse results would be reached. In scene (1) the step 2 with fps 8 is used, and in scene (2) step 4 with fps 25 is used. As crowded scenes are again used the scenes (1) and (2) from table 4.1. The measured results are in table 4.6.

Time	Scene (1)	Scene (2)
Detection	04 : 11 : 39	06 : 16 : 02
Tracking Optical flow	00 : 04 : 02	00 : 09 : 05
Tracking CAMSHIFT	00 : 06 : 43	00 : 13 : 50
Setting-1 analysis	00 : 33 : 34	00 : 13 : 40
Setting-2 analysis	00 : 06 : 58	00 : 05 : 09
Setting-3 analysis	00 : 07 : 49	00 : 03 : 10
Setting-4 analysis	00 : 04 : 43	00 : 02 : 54

Tracks	Scene (1)	Scene (2)
Total	18 425	18 919
Setting-1	11 275	13 751
Setting-2	1 186	2 675
Setting-3	995	1 671
Setting-4	742	1 557

Table 4.6: Scenes (1) and (2) time measurement

The ratio between people detection time and video time is 5.880 for scene (1) and 5.956 for scene (2). In both tested scenes the people detection process is approximately six times slower than video real time. It is caused by many motions in crowded scenes and corresponding high number of detections per second as apparent from table 4.2.

The analysis is dependent on the amount of analysed trajectories and also on events in scene. Although for Setting-1 in scene (2) more trajectories are to process than in scene (1), the processing time is much lower. It is caused by high occurance of fast events in scene (1), because of wrong parameters setting. In scene (1) with usage of Setting-1 are over 800 fast events, so unsuitable parameters setting can increase the analysis time and result can be unusable.

4.3.2 Traffic Containing Scenes

For traffic containing scenes, as described in section 4.2.2, processing time is dependent on motion frequency in scene. If lots of vehicles, which are detected as moving objects are in scene, detection process is made and processing time is high. Also lots of false trajectories on vehicles are created, so with wrong set analysis parameters, can be analysis processing time slow. As traffic containing scenes are again used scenes (3), (4) and (5) form table 4.1. The measured results are in table 4.7.

Time	Scene (3)	Scene (4)	Scene (5)
Detection	09 : 52 : 08	03 : 24 : 47	04 : 48 : 33
Tracking Optical flow	00 : 23 : 38	00 : 08 : 41	00 : 23 : 30
Tracking CAMSHIFT	00 : 42 : 01	00 : 16 : 41	00 : 41 : 55
Setting-1 analysis	00 : 26 : 24	00 : 01 : 39	00 : 17 : 06
Setting-2 analysis	00 : 06 : 47	00 : 00 : 49	00 : 02 : 43
Setting-3 analysis	00 : 02 : 16	00 : 00 : 33	00 : 01 : 35
Setting-4 analysis	00 : 02 : 09	00 : 00 : 34	00 : 01 : 28

Tracks	Scene (3)	Scene (4)	Scene (5)
Total	16 019	2 439	6 862
Setting-1	12 845	1 495	4 797
Setting-2	1 855	596	657
Setting-3	777	375	416
Setting-4	712	366	394

Table 4.7: Scenes (3), (4) and (5) time measurement

The ratio between people detection time and video time is 3.053 for scene (3), 1.707 for scene (4) and 1.489 for scene (5). People detection time is dependent on frequency of moving objects in scene. In scene (3) there is a higher occurence of motion, because of road junction, so the detection time is higher than in other scenes. In other scenes traffic frequency is lower and detection time reduction is correspondent.

4.3.3 Scenes without Traffic

For scenes without traffic, as described in section 4.2.3, detection process is fast because not so many people as in crowded scenes are in these scenes and processing is not slowed down by vehicles as in traffic containing scenes. These scenes are scenes (6) and (7) from table 4.1. The measured results are in table 4.8.

Time	Scene (6)	Scene (7)
Detection	01 : 01 : 54	02 : 02 : 18
Tracking Optical flow	00 : 06 : 21	00 : 17 : 19
Tracking CAMSHIFT	00 : 14 : 44	00 : 30 : 39
Setting-1 analysis	00 : 02 : 37	00 : 02 : 26
Setting-2 analysis	00 : 00 : 50	00 : 00 : 56
Setting-3 analysis	00 : 00 : 26	00 : 00 : 34
Setting-4 analysis	00 : 00 : 24	00 : 00 : 36

Tracks	Scene (6)	Scene (7)
Total	4 070	4 912
Setting-1	3 150	3 445
Setting-2	352	506
Setting-3	161	277
Setting-4	130	242

Table 4.8: Scenes (6) and (7) time measurement

The ratio between people detection time and video time is 0.833 for scene (6) and 0.861 for scene (7). In both scenes is detection process faster than the real video time. It is caused by a high number of frames without motion which are processed very fast, because no people detection process is made.

4.3.4 Processing Time Duration Evaluation

From sections 4.3.1, 4.3.2 and 4.3.3 it is evident, that processing time depends on scene type. Detection process time is dependent on the occurence of events in a scene. If there are only a few people, the detection is fast. If the are also vehicles in the scene, the processing time increase. It also depends on the frequency of events. For example, in scene (3) and (4) the difference is evident. Although the scenes are of the same type, in scene (3) vehicle motion is much more frequent, so the whole detection process is much slower. People detection is the slowest part of the whole process.

As a faster tracking method the optical flow can be marked. The main difference is that for optical flow not so much input pre-processing is made as for CAMSHIFT method, as described in section 3.2.2. Tracking process time is dependent mainly on input video time and frequency of starting new trajectories, as it is evident on scenes (4) and (7). Both scenes have approximately same video length, but tracking is in scene (7) slower, because there are new trajectories started more often.

Trajectories analysis is dependent on count of anomaly event in scene. If there is a high count of anomaly events, the analysis is slow. Next element influencing the analysis time is the count of processed trajectories. Parameters which influence trajectories count must be set conveniently according to scene type and events occurence. Analysis, with suitably chosen parameters, is the fastest part of the whole process, so analysis with different parameters setting can be tried more times. The results of processed analysis tests are on the included DVD in directory `testing`.

4.4 Anomaly Events Detection in Scene

Anomaly events are detected as described in sections 3.3.4 and 3.3.5. If the option `Save anomaly videos` is checked in graphical user interface, as described in section 4.1.2, the videos with anomaly events are saved. Every time the whole anomaly trajectory is saved with five second before and five seconds after the trajectory occurence. If this option is not checked, only the images with anomaly events trajectory start are saved. Images and videos of detected anomaly events in scenes are saved on the included DVD in directory `output` (anomaly events are saved in subdirectory *scenename/events*).

4.4.1 Exceptional Events

Exceptional events detection is dependent on the count of processed trajectories. If there is a low number of trajectories in a scene, there is a high chance, that the trajectory will be marked as the exceptional one, because there are not enough trajectories in the scene to compare with. The result is also dependent on type of scene. In crowded scenes, the exceptional trajectories are not typical, because the motion is almost everywhere. In scenes with traffic occurence the exceptional events are typical for people crossing the road out of pedestrian crossing or for bikers. In a scene where only a walkway is, the exceptions are not typical because people motion is centered mainly on the walkway. Minimal exceptional trajectory length can be set in graphical user interface, as described in 4.1.2.

In figure (e) of the figures 4.11 and 4.12, all trajectories used when exceptions detection was made are present, and in figure (f) all found exceptions in the scene are marked. Figures (a)–(d) contain the examples of exceptional trajectories.

4.4.2 Fast Moving Events

Fast moving events detection is not dependent, in comparison with exception events detection, on the count of trajectories. Fast trajectories can also be detected in a scene with few trajectories because it is not dependent on the other trajectories. In crowded scenes and scenes without traffic, fast trajectories are not typical. In traffic scenes fast moving bikers are typical. False fast event detect can occur, when the person is not detected correctly and only a part of the person is detected. Because fast event detection is based on detection height, as described in equation 3.1, and the height is not complete, fast event can be detected although it is not correct. Approximate trajectory velocity as well as minimal trajectory length can be set in graphical user interface, as described in section 4.1.2. The examples of fast detections can be seen in figures 4.13 and 4.14.

In figure (d) of figures 4.13 and 4.14 all trajectories used when fast events detection was made are present, and in figure (c) all found fast events in the scene are marked. In figures (a) and (b) there are the examples of found fast trajectories.

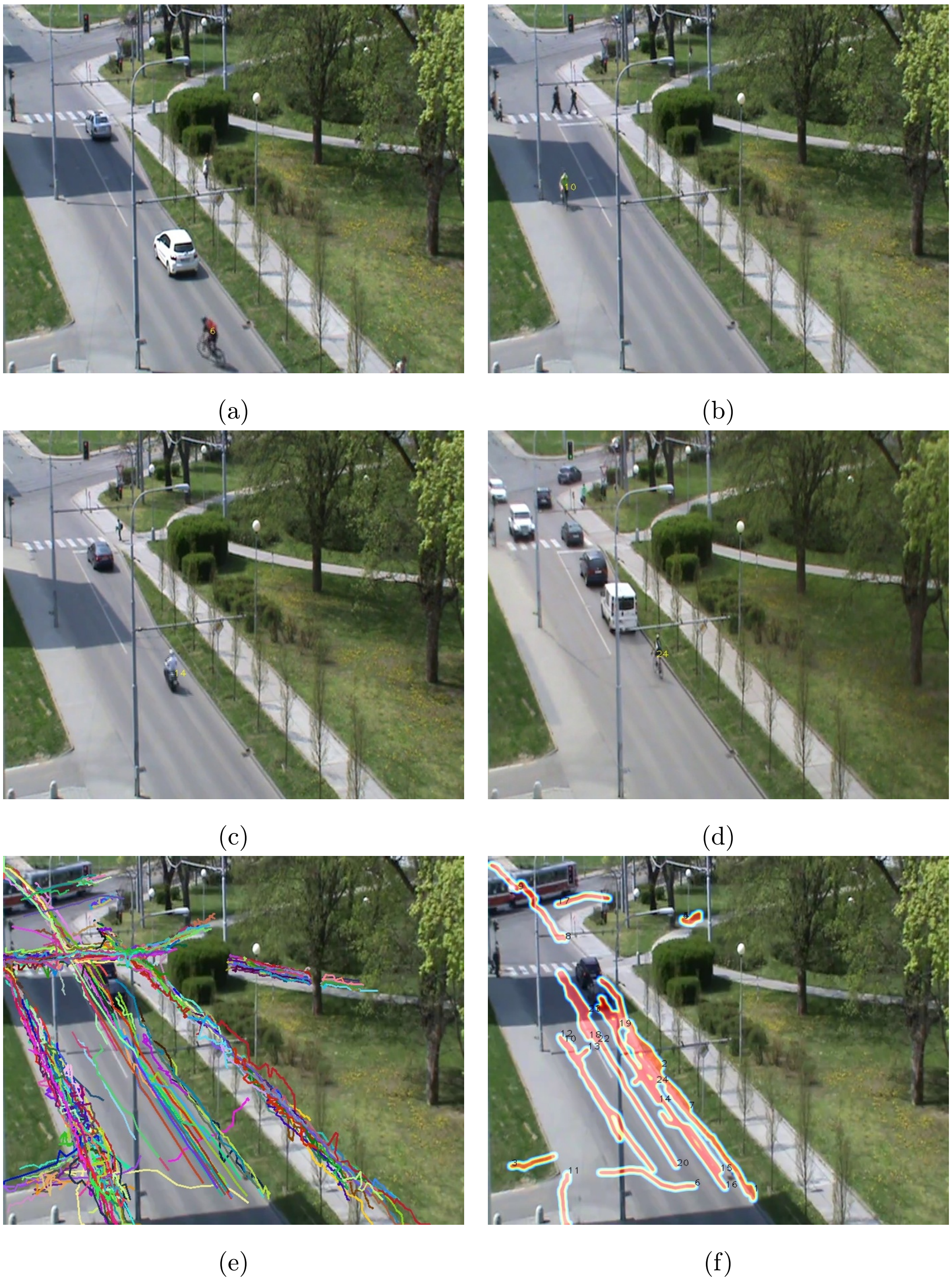


Figure 4.11: Scene (3) exceptional events example

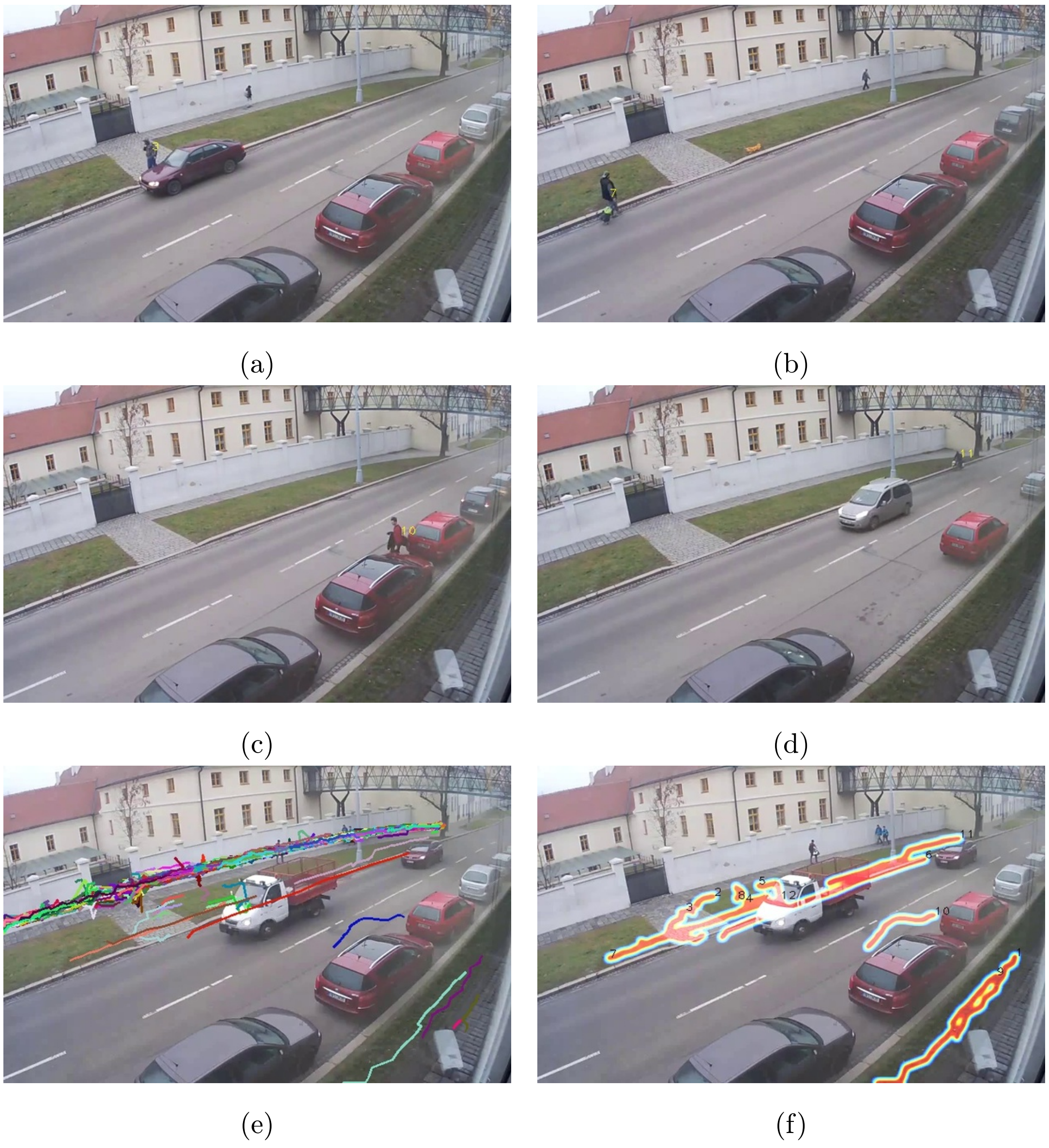


Figure 4.12: Scene (4) exceptional events example

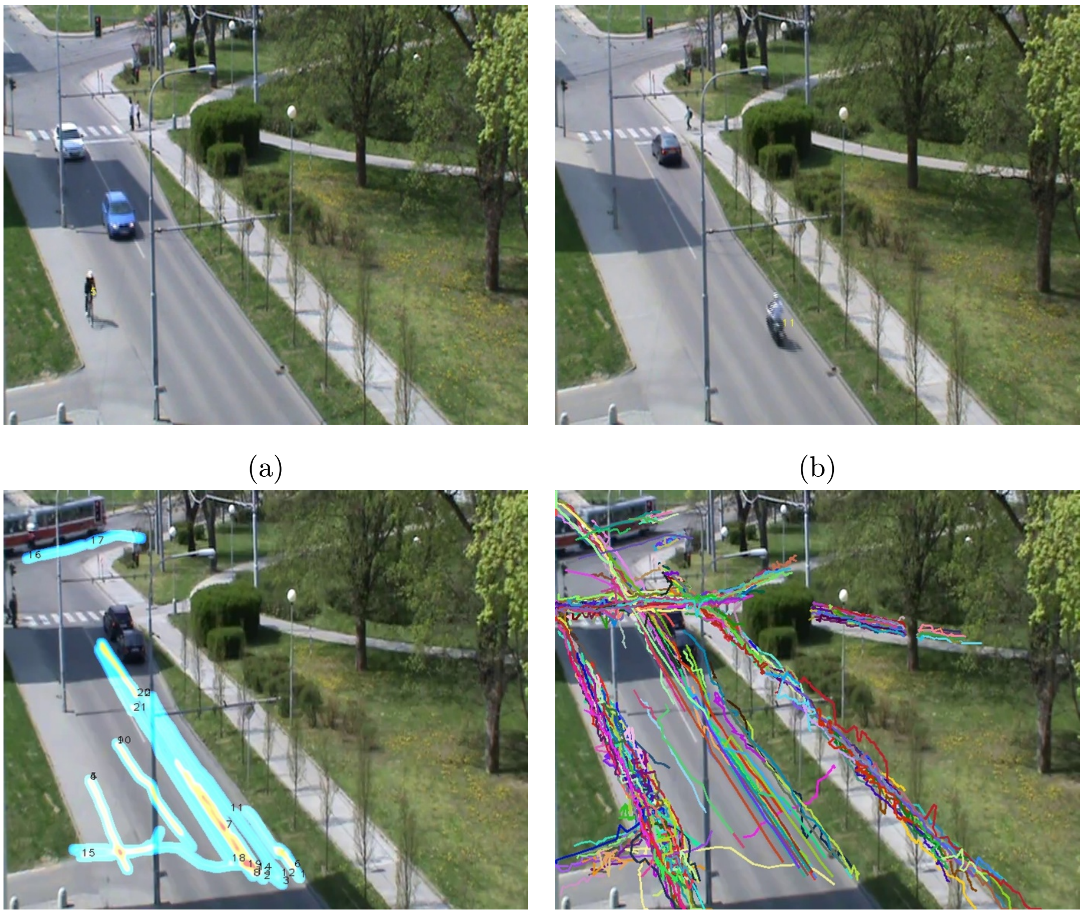
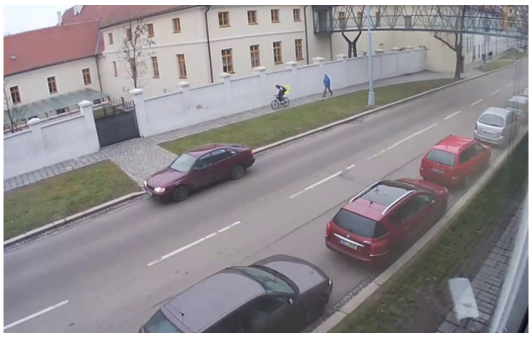
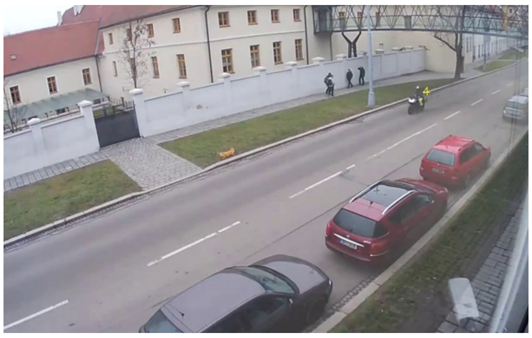

(a) (b) (c) (d)

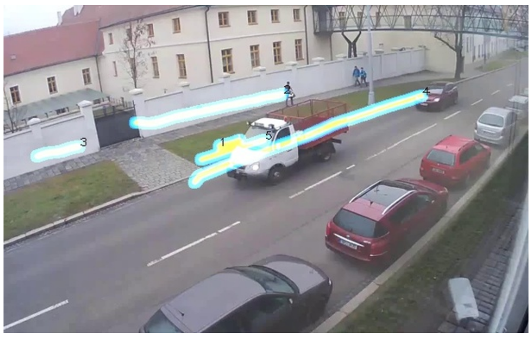
Figure 4.13: Scene (3) fast events example



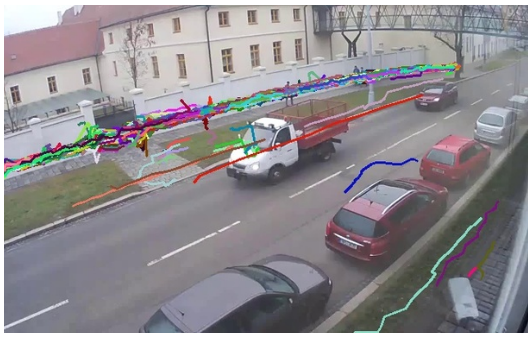
(a)



(b)



(c)



(d)

Figure 4.14: Scene (4) fast events example

Chapter 5

Conclusion

The aim of this diploma thesis was to propose and implement the application for mapping the motion of people. The application with the aim for anomaly events detection in a video sequence was implemented. Two methods for people tracking were implemented and tested. The proposed technique for anomaly events detection was tested on different scene types.

People tracking could be improved for example by Kalman filter. It should be used to cover cases when the view to a person is blocked by some other object. False positive detections on cars are quite common. Car detection could be used together with people detection to differentiate type of detection, and make people detection more accurate . Also some information from the previous trajectories could be used for better tracking process. A person is tracked without any knowledge of other trajectories. Trajectories are analysed sequential. There exists a lot of better approaches to the analysis, but for the purpose of anomaly events detection, the sequential approach provides sufficient results.

It has been managed to propose and implement the application which provides the information about people motion in a scene. It gives information about the occurance of the motion on a scene, about the regions where the motion in a scene is the most typical, and about a typical direction of the motion in scene. Additionally, exceptional trajectories and fast moving people are detected in the video sequence.

Bibliography

[1] Md. Atiqur Rahman Ahad, J.K. Tan, H. Kim, and S. Ishikawa. Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255 – 281, 2012.

[2] David Arthur and Sergei Vssilvitskii. k-means++: The advantages of careful seeding, 2007.

[3] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second, 2012.

[4] Rodrigo Benenson, Mohaned Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned?, 2014.

[5] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface, 1998.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273 – 297, 1995.

[7] Nevneet Dalal and Bill Triggs. Histogram of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, 1:886 – 893, 2005.

[8] Piotr Dollár, Serge Belongie, and Pietro Perona. The fastest pedestrian detector in the west, 2010.

[9] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features, 2009.

[10] Yoav Freund and Robert E. Shapire. *A decision-theoretic generalization of on-line learning and an application to boosting.* Kluwer Academic Publisher, 1995.

[11] K. Fukunga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory*, 21(1):32 – 40, 1975.

[12] Chris Harris and Mike Stephens. A combined corner and edge detector, 1988.

[13] Berthold K. Horn and Brian G. Schunck. Determining optical flow, 1980.

[14] Bahadir Karasulu and Serdar Korukoglu. Moving object detection and tracking in videos. In *Performance Evaluation Software*, pages 7 – 30. 2013.

[15] Brendan Tran Morris and Mohan Manubhai Trivedi. A survey of vision-based trajectory learning and analysis for surveillance. *Circuits and Systems for Video Technology*, 18(8):1114 – 1127, 2008.

[16] Dan Schonfeld, Caifeng Shan, Dacheng Tao, and liang Wang. Object trajectory analysis in video indexing and retrieval applications. In *Video Search and Mining*, pages 3 – 32. 2010.

[17] Pierre Soille. *Morphological Image Analysis.* Springler verlag, Berlin, 2003.

[18] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):23 – 46, 1985.

[19] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11 – 32, 1991.

[20] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137 – 154, 2004.

[21] Xiaogang Wang, kinh Tieu, and Eric Grimson. Learning semantic scene models by trajectory analysis. In *Computer Vision - ECCV 2006*, pages 110 – 123. 2006.

[22] Xiaogang Wang, Keng Teck Ma, Gee-Wah Ng, and W. Eric L. Grimson. Trajectory analysis and semantic region modeling using nonparametric hierarchical bayesian models. *International Journal of Computer Vision*, 95(3):287 – 312, 2011.

[23] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[24] Cheng Yizong. Mean shift, mode seeking and clustering. *Pattern Analysis and Machine Intelligence*, 17(8):790 – 799, 1995.

[25] Bolei Zhou, Xiaoou tang, and Xiaogang Wang. Learning collective crowd behaviors with dynamic pedestrian-agents. *International Journal of Computer Vision*, 111(1):50 – 68, 2015.

[26] Qiang Zhu, Shai Avidan, Mei-Cheng Yeh, and Kwang-Ting Cheng. Fast human detection using a cascade of histogram of oriented gradients. *Computer Vision and Pattern Recognition*, 2:1491 – 1498, 2006.

[27] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *International Conference of Pattern Recognition*, 2:28 – 31, 2004.

Appendix A

DVD Contents

`poster.pdf`	Poster presenting results of this thesis
`thesis.pdf`	This thesis in `pdf` format
`output/`	Application output example
`source/`	Application source code
`testing/`	Application output of testing data
`thesis/`	LaTeX source for this thesis
`video/`	Video files describing used techniques