FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Crossing pedestrian flows – analysis of video records from experiments
Student:	Anna Sajdoková
Supervisor:	Ing. Pavel Hrabák, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Goal of the thesis is to analyse video records from crossing-pedestrian-flows experiments conducted at FNSPE in 2014. The focus is on extraction of pedestrian trajectories and participants identification (each participant was equipped by contrast hat with unique binary code).
1. Perform research of appropriate tools for trajectory extraction (multiple object tracking, binary codes detection).
2. Explore the applicability of found tools for analysis of provided video records, using appropriate tools extract trajectories and participants' IDs.
3. Suggest and implement algorithm for detection and completion of incomplete trajectories.
4. Compare obtained data with results from Bamberger et al. (2015) using appropriate pedestrian flow characteristics.

J. Bamberger et al. (2015) Crowd Research at School: Crossing Flows. Traffic and Granular Flow '13. Springer, Cham.

Electronically approved by Ing. Karel Klouda, Ph.D. on 14 December 2021 in Prague.

Bachelor's thesis

CROSSING PEDESTRIAN FLOWS – ANALYSIS OF VIDEO RECORDS FROM EXPERIMENTS

Anna Sajdoková

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Ing. Pavel Hrabák, Ph.D.
May 12, 2022

Contents

List of Figures

List of Tables

I would like to thank my supervisor Ing. Pavel Hrabák Ph.D. for his time, patience, and invaluable advice in the field of pedestrian dynamics. I also express my appreciation to Ing. Jakub Novák for his valuable remarks. Finally, I wish to thank my family and friends for all their support. I would like to name David Horský and Lukáš Rynt.

Declaration

In Prague on May 12, 2022 . . . .

Abstract

The thesis focuses on processing and analysis of a video record from crossing pedestrian flows experiment conducted at CTU FNSPE in 2014. The result is an algorithm for automatic extraction of trajectories from this video. Pedestrians in the video had special hats for recognition. The tracking of people is based on hats detection from video frames. Track identity association is done using the shortest distance. When tracking, it can happen that part of the trajectory is missing. The missing parts are approximated by a line segment. Next aim is to recognize binary code from hats. With usage of a convolutional neural network 45% accuracy was achieved on 20 randomly picked hat samples. The outcome of the thesis is a dataset of trajectories and its analysis using pedestrian flow characteristics (average speed, velocity, density, and fundamental diagram).

Keywords pedestrian dynamics, crowd dynamics, pedestrian crossing flows, pedestrian flow, fundamental diagram, multiple object tracking, image processing

Abstrakt

Práce se soustředí na analýzu a zpracování videozáznamu experimentu křížení chodců pořízeného na ČVUT FJFI v roce 2014. Výsledkem je algoritmus na automatickou extrakci trajektorií chodců z tohoto videa. Chodci na sobě měli speciální čepičky pro rozpoznání. Sledování osob je založeno na detekci čepiček z jednotlivých snímků videa. Asociace detekcí identitám je provedena pomocí nejmenších vzdáleností. U sledovacích algoritmů se může stát, že se trajektorie ztratí. Tyto lokace jsou aproximovány úsečkou. Další částí je rozpoznání kódu z čepiček. Konvoluční neuronová síť detekovala správně 45% čepiček na náhodném vzorku 20 čepiček. Výstupem práce je dataset trajektorií a analýza pohybu chodců (průměrná rychlost, rychlost a fundamentální diagram).

Klíčová slova pohyb chodců, dynamika davu, křížení chodců, proud chodců, fundamentální diagram, sledování více objektů, zpracování obrazu

Abbreviations list

BFS	Breadth-first search
BI-VZD	Data mining course at CTU FIT
CNN	Convolutional neural network
CTU	Czech Technical University in Prague
DNN	Deep neural network
FIT	Czech Technical University in Prague, Faculty of Information Technology
FNSPE	Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering
MOT	Multiple object tracking
NN	Neural network

Introduction

The thesis presents a computer vision model for extracting pedestrian trajectories from the crossing pedestrian flow experiment that was carried out in CTU FNSPE in 2014. The aim is to automatically obtain the pedestrian trajectories from this experiment. The aim is not to build robust software that would handle all experiments of this type.

The crossing flows experiment is inspired by Bamberger et al. [1]. They organized an experiment in a German school. In their work, they analyze the crossing area with pedestrian flow characteristics and attempt to investigate whether the phenomenon of self-organization (e.g. line formation) appears. Future experiments could further support or diminish the ideas. Nevertheless, the aim of the thesis is not to perform a thorough analysis.

Motivation

It is important to research pedestrian traffic. The knowledge obtained can be used e.g. in pedestrian traffic simulators. Precise traffic simulation is crucial to make evacuations safer and faster. Today, traffic modeling tools are widely adopted in the building design process [2], where performance-based design is applied.

The video from pedestrian crossing flows experiment (described in Section 4.1) was not processed by anyone yet. Extensive analysis of the dataset can increase the knowledge of pedestrian crossing flows.

Chapter 1

Pedestrian Dynamics

Modeling traffic is a broad area of research. Biologists model the movement of cells. Physicists look at particles. In recent years, research on self-driving cars, such as modeling vehicular traffic and tracking vehicles, has gained a lot of attention. The movement of pedestrians is also a kind of transport in complex system.

The chapter introduces concepts in pedestrian dynamics with focus on explaining observables in pedestrian traffic, such as density, velocity, and flow. More detailed information on pedestrian dynamics can be found in a book called Stochastic Transport in Complex Systems [3]. Further reading about methods for measuring pedestrian observables can be found in an article by Steffen et al. [4]. Both sources are the foundation for the following information.

Traffic models differ greatly depending on the objects modeled. There are rules that the model must follow. They are strongly connected to the nature of the tracked object. For example, in vehicular traffic, vehicle movement is restricted by road, speed limits, light signs, and traffic rules. Interactions with the closest cars are the most important. The vehicular traffic is modeled in one-dimensional or quasi-one-dimensional space.

Pedestrian traffic is specific by its complexity [3]. The speed of walking differs depending on motivation (rushing to work versus shopping for groceries). It is not clear which interactions are the most important. Cars only travel on roads; pedestrians are much less space constrained and their movement needs to be modeled in two dimensions.

1.1 Observables

Observables in pedestrian dynamics are divided into *microscopic* and *macroscopic*. Microscopic variables describe an individual pedestrian (e.g. pedestrian position, individual velocity, and acceleration). Macroscopic variables describe the movement of groups of pedestrians. The most used are velocity, density, and flow.

Flow

Flow is a physical quantity. In general, it measures how many units of mass flow through the investigated area per unit of time.

Flow J, in the context of pedestrian dynamics, is the number of pedestrians N crossing a specified location per time interval T. That is expressed in the following equation:

$$J = \frac{N}{T} \tag{1.1}$$

How to calculate the flow? Let us record all the times a pedestrian crosses the location. The flow can then be calculated from the time gaps $\triangle t_i = t_{i+1} - t_i$ of two consecutive pedestrians i and $i+1$:

$$J = \frac{1}{\langle \triangle t_i \rangle}\ , \langle \triangle t_i \rangle = \frac{1}{N} \sum_{i=1}^{N} (t_{i+1} - t_i) \tag{1.2}$$

where $\langle \triangle t_i \rangle$ is understood as mean value of time gaps between pedestrians.

Density

Density ρ is estimated by the number of pedestrians N per some specified area A:

$$\rho = \frac{N}{A} \tag{1.3}$$

The uncertainty is in determining the area A. There are two approaches for measuring the observable [4]:

- *On fixed location* during a given period of time $\langle \tau_n, \tau_{n+1} \rangle$, then average values over the period of time. Repeat for $\langle \tau_{n+1}, \tau_{n+2} \rangle \dots$.
- *In fixed time* τ_t on a given section $\langle x_A, x_B \rangle$, then average values in this section. Repeat for $\tau_{t+1}\ \dots$.

Although the two options appear similar, the number of pedestrians included in an area may differ.

Velocity

Individual velocity $\vec{v}_i(t)$ of a pedestrian i at a time t is the time derivative of their position.

$$\vec{v}_i(t) = \frac{d}{dt}\vec{x}(t) \tag{1.4}$$

It can be estimated as the rate of change in position on a small interval $\triangle t$ around t.

$$\vec{v}_{\triangle t, i}(t) = \frac{\vec{x}(t + \triangle t/2) - \vec{x}(t - \triangle t/2)}{\triangle t} \tag{1.5}$$

Other possible calculation of individual velocity:

$$\vec{v}_{\triangle t, i}(t) = \frac{\vec{x}(t) - \vec{x}(t - \triangle t/2)}{\triangle t/2} \tag{1.6}$$

Note that velocity is a vector and **speed** s is the norm of velocity $||\vec{v}|| = s$.

When dealing with a group of pedestrians, individual velocity is averaged either over time or space. The space mean velocity is defined as it is further used.

Space mean velocity v_A of pedestrians at a fixed time t in a specified area A is the aritmethic mean of the individual velocities $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$ at t:

$$v_A = \frac{1}{N} \sum_{i=1}^{N} v_i(t) \tag{1.7}$$

Trajectory

As obtaining trajectories is the main focus of this thesis, what trajectories are is also explained. Intuitively, the trajectory is the path that an object follows through space as a function of time.

Trajectory $x_\alpha(t)$ of a pedestrian α are all locations of the pedestrian during a given period of time.

$$x_\alpha(t) = [x_\alpha^{(1)}, x_\alpha^{(2)}](t) \tag{1.8}$$

Time is understood to be discrete $t \in \{t_0 + n\triangle t\}$. In this thesis $\triangle t = 5/47\ s$ as the frame rate of the video is 47 frames per second and every fifth frame is processed.

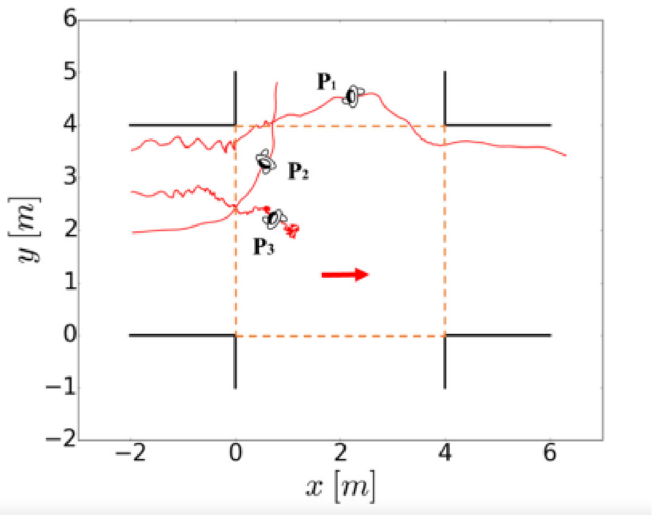


Figure 1.1 Exemplary trajectories observed in the PeTrack experiments in Jülich [5]. Further information about PeTrack can be found in Section 4.2.1

1.2 Fundamental Diagram

Fundamental diagram connects density, velocity, and flow. It is clear that these variables are connected. For example, people on a crowded street move much slower than those on empty streets because the movement is slowed down by the crowd. There are multiple equivalent forms of the fundamental diagram [6]:

- velocity-density $v(\rho)$
- flow-density $J(\rho)$
- flow-velocity $v(J)$

These variables are connencted via *fundamental relation of pedestrian flow* [7]:

$$J = \rho v \tag{1.9}$$

This relation allows to choose any representation mentioned. The fundamental diagram quantifies the capacity of pedestrian facilities and thus can be used e.g. for the rating of escape routes.

In this thesis, a fundamental diagram will be modeled by the relation *velocity as a function of density*. As it is relatively easy to construct from camera recordings.

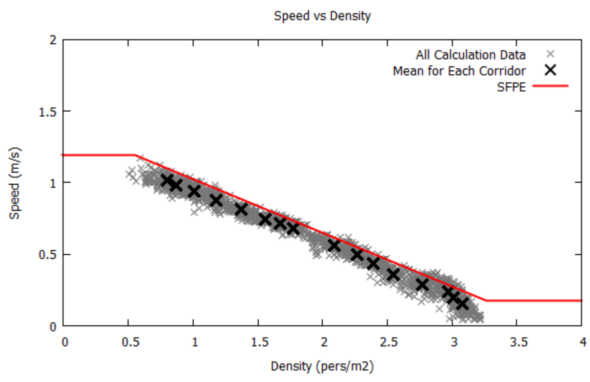


Figure 1.2 The relation of velocity and density is sometimes standardized in programs modeling pedestrian traffic. This standardization is called SFPE curve. The figure shows such standardization in Pathfinder [8], an agent-based pedestrian flow simulation software. Pathfinder has two modes [9] – Steering and SFPE. In the SFPE mode Pathfinder looks at the SFPE curve and based on the density finds appropriate velocity. It moves pedestrian with this velocity and resolves collision later.

1.3 Self-organization Phenomena

Self-organization, also called spontaneous order, is a process where some form of order emerges after interactions between the examined objects in a initially disordered system. An example of self-organization phenomena that might emerge in the pedestrian crossing experiment is *formation of lines*. Another might be the observation of *diades* (a pair of persons walking in close distance) and *triades* (three persons walking near each other).

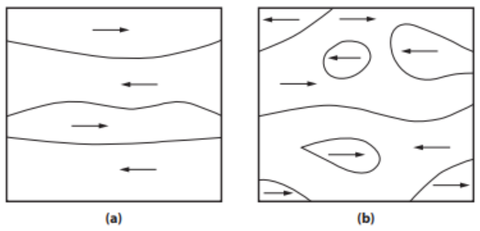


Figure 11.7 Typically in bidirectional pedestrian motion, lanes (a) and clusters (b) are formed. These can be distinguished by the the band index [1491], which is basically the ratio of pedestrians in lanes to their total number. This index is close to 1 in (a) (from [768]).

Figure 1.3 Line formation as shown in an article by Bamberger et al. [1].

Chapter 2

Multiple Object Tracking

Multiple object tracking (MOT) is an essential building block of a wide range of applications. It is used in fields such as human tracking [10, 11, 12], autonomous driving [13, 14], and biology (tracking ants [15], cells [16, 17], or fish [18]).

The chapter provides an introduction to MOT. The main resource is a summary article called Multiple object tracking: A literature review [19]. Pedestrian trajectories are obtained with MOT algorithm called centroid tracking in Subsection 5.4. The centroid tracking algorithm is located in this chapter in Subsection 2.3.

2.1 Task Description

MOT is a task in which the objective is to estimate the positions of multiple variables. Typically, it is an estimation of trajectories for objects of interest in video sequences.

There is a MOT challenge website[1] [20]. It concentrates benchmark datasets in one place. Ground truth bounding boxes are not rotated in these datasets and this thesis task differs a lot, due to the usage of hats. Therefore, the models trained on these datasets cannot be compared with the model in this thesis.

2.2 Principle

Detection-based tracking is today a preferred paradigm for solving the MOT task [21].

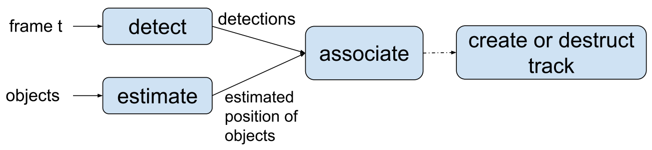


Figure 2.1 Tracking by detection for a frame at time t. A detector detects objects from the frame. Further such objects will be denoted as *detections*. Positions of before detected objects (from now on just *objects*) is estimated. Objects and detections are associated, so the algorithm knows which detection is part of which track. If the detection does not associate with any object a unique ID is created for it (new *track* is created). Similarly if an object is not associated with any detection, the track is destructed. Same principle can be applied to time $t + 1$.

[1]https://motchallenge.net/

Detection

The performance of the MOT algorithm is highly dependent on a reliable and efficient object detection. In most state of the art for pedestrian tracking the detection part is done by a neural network [21]. Nevertheless, any traditional methods, such as finding contours by segmentation or Hough's transformation, can be used, too.

What detection algorithm to use depends on the amount of data, conditions in the tracked area and the nature of the tracked object (shape, color, etc.). In places like laboratories, warehouses, or conveyor belts, the conditions (lighting, area of interest, weather, etc.) should be constant. In this thesis, it can be chosen whether to use classical methods, neural networks, or both. Under unstable conditions, neural networks have outperformed classical methods [22].

The output of detection part are the object bounding boxes in the current frame.

Estimation

The estimation of the object in the current frame is calculated from the previous locations of the object. The simplest estimation is to keep the object in its place. Nevertheless, it is also possible to look at past several frames and estimate some position using some more sophisticated methods e.g. Kalman filters, central differences.

Association

Association step needs a metric to be able to associate objects with detections. There is a simple metric, like the Euclidean distance, which assigns a minimal value to the pair of detection and object that appears closest on the video. Nowadays, deep association metrics such as detection feature are used. The final metric is often a combination of space and deep feature metrics.

Track identity creation and destruction

MOT is trying to be robust to oclusions. Because of that in some algorithms the object track still exists even when no detection is associated with it in current frame. How long should be the track remembered is stored in a variable `DISSAPEARED_MAX`. When detection is not assigned to track for `DISSAPEARED_MAX` frames the track is destructed. For object that did not associate with any track a track is created. Sometimes there is a trial period for track, where the track is only counted in when the number of object in the track exceed certain limit. Trial period is not used in this thesis as there is high reliability for detections of hats.

2.3 Centroid Tracking

A centroid tracking is a fundamental algorithm in MOT. The main idea is that it uses the Euclidean distance between the centroids of existing objects and the centroids of objects currently detected to associate between frames.

The algorithm relies on object detection in every frame. The tracker remembers object it saw, where it was and how long ago. When the object disappears for more than `MAX_DISSAPEARED` tracks, it is deleted.

Diagram 2.2 shows the update function that is called for every frame. In each call, if the detector did not detect any objects, just increase the disappeared counter for all objects and return. If there are no objects remembered from the past frames, the algorithm needs to register detections as new objects and return. If none of the above is true, then the distance matrix D is computed between the detections and the objects. Then association is just picking up the minimal values as shown in Algorithm 1. If the objects and detections do not have the same length, there will be some leftovers. If those are detections, register them. If not (that means there are objects that were not associated) increase disappeared counter for the objects unused in association step.

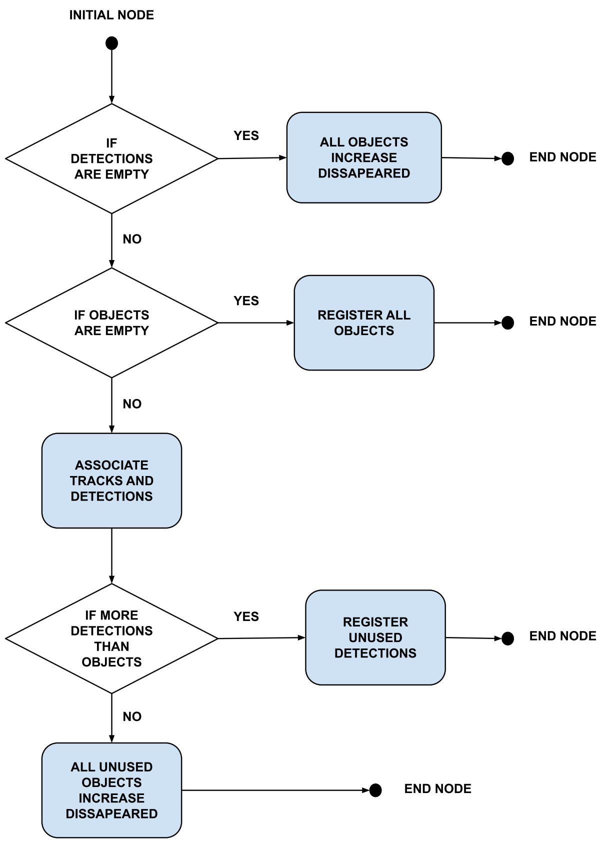

INITIAL NODE
IF DETECTIONS ARE EMPTY
YES
ALL OBJECTS INCREASE DISSAPEARED
END NODE
NO
IF OBJECTS ARE EMPTY
YES
REGISTER ALL OBJECTS
END NODE
NO
ASSOCIATE TRACKS AND DETECTIONS
IF MORE DETECTIONS THAN OBJECTS
YES
REGISTER UNUSED DETECTIONS
END NODE
NO
ALL UNUSED OBJECTS INCREASE DISSAPEARED
END NODE

Figure 2.2 Diagram of update function in centroid tracking algorithm. The function takes as input *detections* made by a detector and *objects*. It updates the set of objects to correspond to the state on the frame.

Algorithm 1 Centroid tracking association

```
1: procedure ASSOCIATE
2:     $D \leftarrow compute\_centroid\_distances(objects, detections)$
3:     $rows \leftarrow D.min(axis = 1).argsort()$        ▷ sort indexes based on smallest value in row
4:     $cols \leftarrow D.argmin(axis = 1)[rows]$ ▷ find index of the smallest element in each col, sort using rows index list
5:     for row,col in rows,cols do
6:         if row in used_rows or col in used_cols then
7:             continue
8:         end if
9:         $ID \leftarrow IDS[row]$
10:        $object[ID] \leftarrow detections[col]$
11:        used_rows append row
12:        used_cols append col
13:    end for
14: end procedure
```

Chapter 3

Image processing

Computer vision is used for hats detection and the extraction of binary codes. First part of this chapter introduces concepts used for traditional image processing. The main resource for this chapter are The Scientist and Engineer's Guide to Digital Signal Processing [23]. The second part is about processing images by neural networks.

3.1 Image Processing Methods

Since the quality of information that comes out cannot be better than the quality of information that comes in, image processing is vital. Images contain noise and unwanted distortions. This can be suppressed by the right processing techniques. Image processing also helps enhance desired features and suppress unwanted ones. The information below is used for preprocessing (Section 5.2), detection in centroid tracking (Section 5.3), and classification of the binary code on image (Section 5.5).

3.1.1 Camera Calibration

Camera calibration is a technique for determining camera parameters. It is used to eliminate image defects caused by the combination of camera and lens [24]. Camera calibration is most frequently used to remove distortions. Distorsions are briefly introduced as they are evident in the video footage and camera calibration coefficients were not preserved.

Distorsion

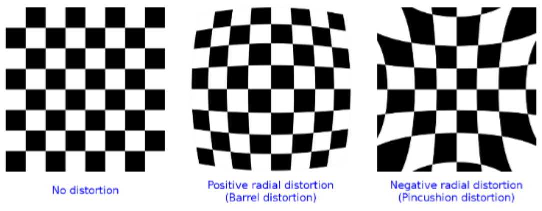


Figure 3.1 Radial distortions on a chessboard [25].

There are two types of distorsions [25]:

radial: It causes straight lines to appear curved. The two types Barrel and Picushion are shown in the Figure 3.1.

tangencial: It occurs due to imperfectly aligned lenses and imaging plane (they should be parallel). So, some areas in the image may look nearer than expected.

3.1.2 Filters

A filter is a set of local transformations. It aims to suppress low or high frequencies in an image. There are two types of filtration: in *space* and in *frequency* domain. This thesis uses only filtration in space domain.

Median Filter

Median filter is a non-linear filtration in space, that replaces the value of given pixel with the median of the neighbor pixels. The disadvantage of this filter is that it is time consuming because it needs to sort the values to obtain the median. Advantages are that it is robust to outlier values and it preserves edges.

3.1.3 Morphological Operations

Morphological operations are a way of obtaining knowledge from an image with usage of a structural element. The input is a binary or grayscale image and a structural element. During the transformation the structural kernel is applied on every pixel.

There are four main morphological operations:

Dilation ($\oplus$) assigns 1 to the pixel in the binary image, if at least one pixel marked with structural element is equal to 1. In grayscale images, the pixel is assigned the maximal value from the pixels value covered with structural element. In practice, that means it adds pixels to object boundaries. Dilatation makes objects bigger.

Erosion ($\ominus$) assigns 1 to the pixel in the binary image if all values of the pixels marked with the structural element are equal to 1. In grayscale images, the pixel is assigned the minimal value from the pixels values covered with structural element. In practice, it removes pixels from the boundaries. After erosion objects are a bit thinner and smaller. It can help to remove small lines and noise.

Opening ($\circ$) first erodes the picture, then dilates. It helps to eliminate noise and still keep the same size for the objects.

Closing ($\bullet$) first dilates the picture, then erodes. This morphological operation helps, for instance, to close gaps in objects. The holes are eliminated by dilation, erosion reconstructs the objects to be the same shape and size.

3.1.4 Background Subtraction

When using a static camera to make a video stream, a foreground (moving objects) and background (static scene) can be determined. The approach is to see the difference between current frame and a reference background frame, often called background model. This technique detects any movement, so for instance weather conditions like rain can easily influence the output. Example for an image and its background model is shown in Figure 3.2.

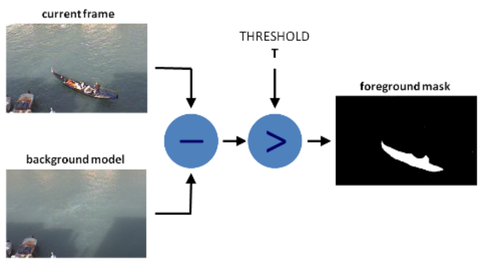


Figure 3.2 Background substraction principle.

3.1.5 Segmentation

Segmentation divides the image into segments with the same properties. Usage would be e.g. to split desired objects and background. The most used segmentation in this thesis is segmentation with two tresholds.

Segmentation with Two Tresholds

This segmentation creates a binary mask. It takes an image and two tresholds as input. If the pixel of input image lies in between of these given thresholds, then the pixel in mask is assigned 1 else 0.

HSV Segmentation

HSV segmentation is used to obtain areas of the image that have similar color.

HSV is a color model. To provide some examples of color models:

RGB: Stands for red, green, blue. It is the most used and well known.

HSV: Stands for hue, saturation and value (brightness). Most common current application is in color selection tools like this color picker on image 3.3. It is also used for color segmentation, as it is easier to write the thresholds than it would be for example in RGB.

Lab: Stands for lightness, a* is the red-green axis, b* is the blue-yellow axis. It covers the entire range of human color perception.

CMYK: Stands for cyan, magenta, yellow, and key (black). It is used in color printing.

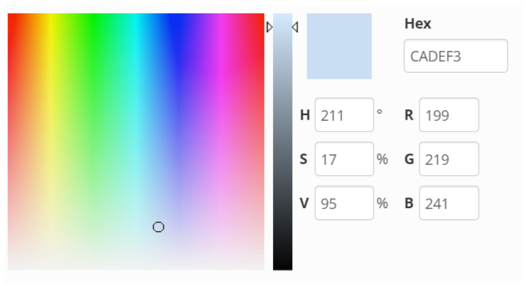


Figure 3.3 HSV color picker CTU blue is picked.

Wathershed Segmentation

The idea behind the algorithm is that for images high intensity pixels can be viewed as peaks, low intensity pixels as valleys. Start filling each valley with different colored water. When water with different color meets a barrier is created between them. The water level rises until it floods the entire area. Each water body is then a segment. The barriers determine the segmentation result. Implementation of this algorithm and some more information can be found for instance in Python image processing library called OpenCV [26].

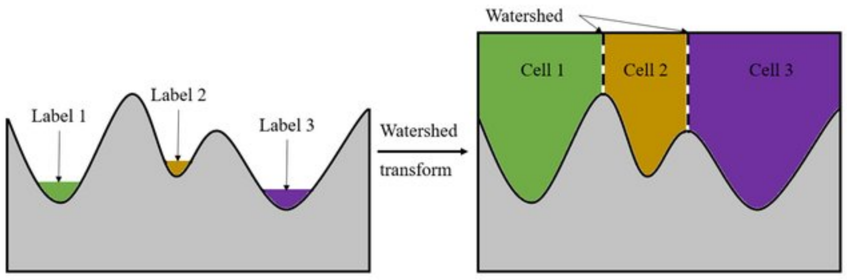


Figure 3.4 Watershed segmentation process [27].

3.2 Neural Networks

The section introduces neural networks. In this thesis, convolutional neural networks are used for binary codes classification (Section 5.5) that were on the pedestrian hats. Main resource for this section is a Deep learning book [28] from Ian Goodfellow et al. This thesis presumes knowledge in range of CTU FIT subject called Data mining (BI-VZD) [29]. Nevertheless the main concepts are briefly introduced. Then training and convolutional neural networks are described in more detail.

3.2.1 Brief Introduction

Neuron

The basic unit in neural network (NN) is called a neuron. A neuron has n inputs edges with weights $w_1, \ldots, w_n$, a bias w_0 and an activation function f. The neuron gets an input $x_1, \ldots, x_n$. The output y is calculated as the activation function applied to the weighted sum of inputs plus bias:

$$y = f(\sum_{i=1}^{n} w_i x_i + w_0) \tag{3.1}$$

A single neuron as a network (usually called perceptron) can represent only linear functions. It cannot learn to predict a simple function like $xor(x_1, x_2)$. So, to predict more complex data, neurons are usually grouped into layers.

Feed Forward Network

A feed forward network has neurons organized into layers. It consist of an input layer, some number of hidden layers and an output layer. A network with one hidden layer can approximate any continuous function for inputs within a specific range (that is called an universal aproximation theorem).

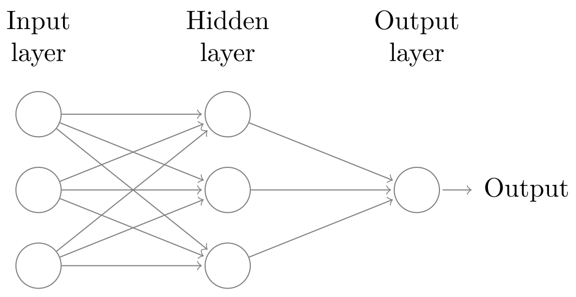


Figure 3.5 Image [30] of neural network with an input layer consisting of 3 neurons, a single hidden layer with 3 neurons and an output layer which has one neuron.

To continue, multi-layer feed forward networks are universal approximators. However, finding such parameters is rather difficult. NN used in practice have dozens of layers and millions of parameters. It is a matter of days or months to train them.

A network with large number of hidden layers is called a deep neural network. They have the ability to learn and model even some complex non-linear relationships.

3.2.2 Training

NN have the ability to learn. It means somehow adjusting the trainable parameters (w – weights and bias) so the network output $\hat{Y}$ is close to the desired output Y. How do the neural networks do that?

A loss function measures how much the predicted label is wrong. The NN learns by trying to minimize the prediction error measured on the average value of the loss function L on the training set. Common loss function also used in this thesis is called Mean Squared Error (MSE). It is defined as:

$$L(\theta) = \frac{1}{N}||Y - \hat{Y}|| \tag{3.2}$$

where θ are the learned parameters ($\theta = w$), N is the number of input vectors, $\hat{Y}$ are the predictions of NN and Y are the target labels.

A stochastic gradient descent (SGD) algorithm is used for the iterative updates of the trainable parameters. There are extensions like minibatch SGD, Adam, SGD with momentum and others.

What is the error for each trainable parameter in current step of the SGD can be found by backpropagation. The only conditions are that we need a differentiable loss function and all activation functions need to be differentiable too. Then the NN error in w can be calculated with backpropagation.

The backpropagation is an algorithm that computes the gradient of the loss function with respect to w. The gradients are needed for the update of the trainable parameters. The negative value of gradients tell us the direction where to move in order to minimize the loss function. Gradients are computed iterating backwards from the last layer using the chain rule. More information can be found in the Deep learning book [28].

3.2.3 Convolutional Neural Networks

Convolutional neural network (CNN) is a type of neural network that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. CNNs process inputs with a grid-like structure. They have proven to be particularly useful in image and video recognition.

There are three main types of layers to build CNN: convolutional layer, pooling layer, and fully connected layer. Sometimes there is also a batch normalization layer. A brief explanation of these layers is given below.

Convolutional Layer

Convolution is a mathematical operation. The first argument x is called the input, and the second argument w is called the kernel. Convolution operation on function x with kernel w is defined as:

$$(w * x)(t) = \int_{-\infty}^{\infty} x(t-a)w(a)da \tag{3.3}$$

For vectors (discrete case):

$$(w * x)(t) = \sum_{i} x_{t-i}w_i \tag{3.4}$$

Convolution can be generalized to multiple dimensions.

For two dimensions were I is the input matrix and K is the kernel of shape m, n:

$$(K * I)_{i,j} = \sum_{m,n} I_{i-m,j-n}K_{m,n} \tag{3.5}$$

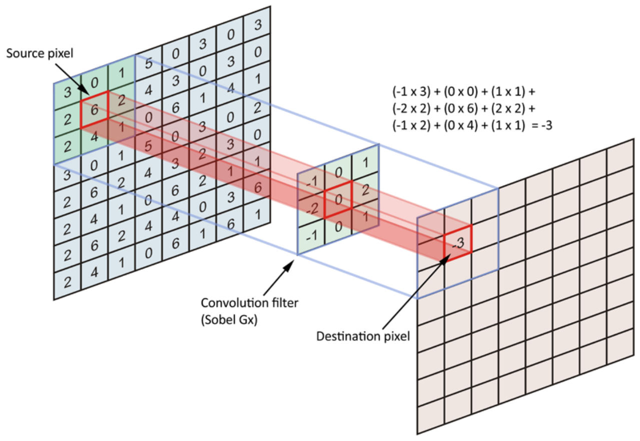

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 3.6 Convolution [31].

Usually in image processing, there is a whole vector of values for a single pixel, the so called channels. Lets have c channels. The channel values for single pixel do not have any spatial structure. So every channel needs its own set of weights for every input dimension. Also specifying the number of output channels o for every pixel is convenient.

When processing images, we need to specify:

- the width W and height H of the kernel
- number of input channels F
- the stride S denotes that the output pixel is computed for every S-th input pixel

The convolution operation that is computed in the neuron of a convolutional layer then looks like this:

$$(K * I)_{i,j,o} = \sum_{m,n,c} I_{i \cdot S+m, j \cdot S+n, c} K_{m,n,c,o} \tag{3.6}$$

Because of shared weights architecture of the convolution kernels CNNs are shift invariant (CNN outputs equal response for same structure in different location).

Pooling Layer

A pooling layer applies some summary statistics on outputs of the previous layers. For instance, max pooling operation reports only the maximum output within a rectangular neighborhood. Other types are average pooling and min pooling. The pooling layer reduces the number of parameters, that decreases the computation time and helps to control overfitting.

Batch Normalization Layer

Normalization is a technique to standardize data (rescales data to have a mean of zero and a standard deviation of one). Batch normalization layer recenters and rescales layer inputs. The batch normalization layer is useful, as it usually reduces the computation time. The authors of ResNet one of the well known CNN architectures wrote: "By only using Batch Normalization [...], we match the accuracy of Inception in less than half the number of training steps." [32] Inception is another type of CNN. Both state of the art at their times.

Chapter 4

Analysis

The chapter presents the crossing flows experiment conducted at CTU FNSPE. Results of other known crossing flows experiments are introduced.

4.1 CTU FNSPE Experiment Setup

Around 80 volunteers (second year students of CTU FNSPE) participated in this experiment. It was conducted by Bukáček, Hrabák et al. on the same day as the bottleneck experiment [33] in 2014. The setup is shown in the Figure 4.1. At the beginning of the experiment, students were divided into two groups. Each group started at the narrow entrance (approximately 80 cm wide). Students were instructed to walk in the direction of the arrows. The idea was to observe the two streams of pedestrians that cross in the crossing area.

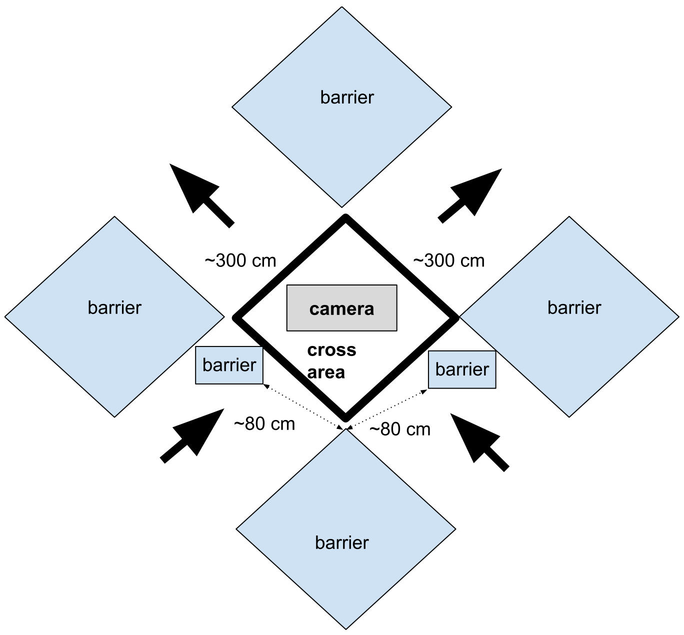


Figure 4.1 Experiment setup. It shows the approximate distance of the entrance point and the cross area. Camera was placed in the center of the room.

Camera Setup

In the experiment setup, the camera is overhead in the middle of the room to know the detailed position of every person at any time also in crowded scenes. This knowledge could not be obtained from the side view as the pedestrians would be sometimes ocluding each other. The camera was most probably Prestigio Roadrunner 700x, however it remains uncertain. The estimated room height is 2 meters. The video frame rate was 47 frames per second. The focal lenght is not available.

Hats

In application dealing with the safety of people, reliable empirical data are needed [34]. Therefore, the experiment used hats as markers. The marker has a red dot in the center. Together with a green dot, the dots can be used to see the rotation angle of the person. The hat is also equipped with a 2x4 binary code that holds an identification of the person. The down left corner of the binary code is a white control bit.

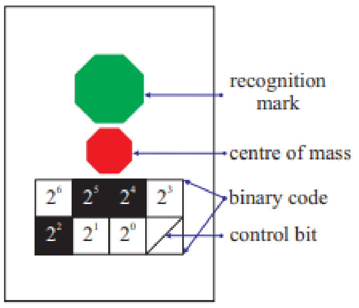


Figure 4.2 Hat used in experiment. The radius of the green dot is 3.75 cm. The radius of red dot is 2.5 cm. The square size is 3.5 cm.

4.2 Pedestrian Tracking in context of Pedestrian Dynamics

In Bamberger et al. [1] the experiment was evaluated manually. That means locating each person throughout the video and denoting their positions. Computer vision for evaluation of crossing flow experiment was used in article [5]. They used software, called PeTrack, that was developed by them.

4.2.1 PeTrack

PeTrack (**Pe**destrian **Track**ing) [35] is a software for automatic obtaining of trajectories. The source code is available here[1]. It uses **Lucas–Kanade method**, differential method for optical flow estimation, to obtain the trajectories. More detailed explanation of Lucas-Kanade method is well described here [36]. PeTrack software could not be used for processing this experiment due to different nature of hats (shown in Section 6.2).

In article [5] outcomes of experiments with multiple crossing flows and unidirectional flow are described. This set of experiments (called BaSiGo experiments) researched crossing pedestrian

[1]https://jugit.fz-juelich.de/ped-dyn-emp/petrack/-/wikis/home

Table 4.1 PeTrack experiments with crossing flows [5].

Run	Name	$b_{in}(m)$	$b_{cor}(m)$	N
01	CROSS_90_D_1	0.6	4	603
02	CROSS_90_D_2	0.9	4	604
03	CROSS_90_D_3	1.2	4	606
05	CROSS_90_D_5	1.8	4	600
06	CROSS_90_D_6	2.4	4	597
07	CROSS_90_D_7	3.0	4	604
08	CROSS_90_D_8	4.0	4	592

flows in a similar setup like in this experiment. The mean age of the participants was 25 years and they were mostly university students.

In total they did 8 runs of the crossing flow experiment (in paper called BaSiGo_Cross_D). What happened in experiment number 04 is not written in the paper. The width of entrance b_{in} varied for each run. The exit size stayed the same b_{cor}. The number of pedestrians N that participated also varied as describe in Table 4.1

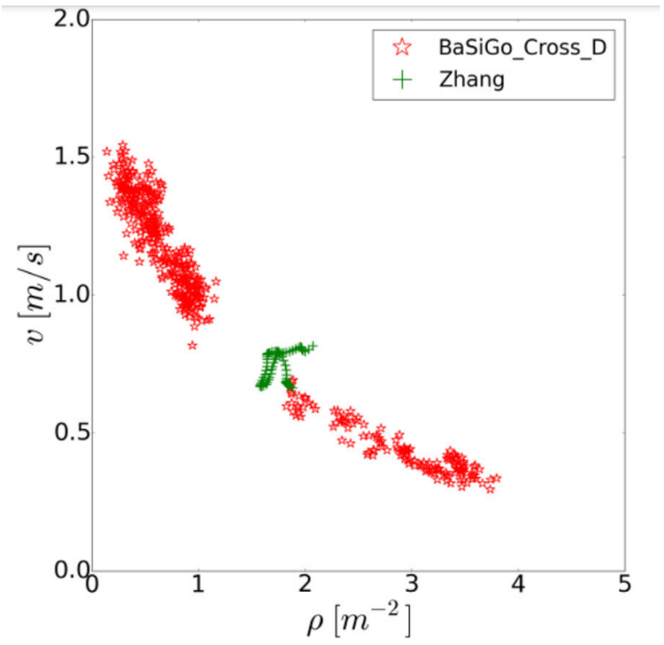


Figure 4.3 Fundamental diagrams for crossing pedestrian flows experiment BaSiGo_D [5].

4.2.2 Bamberger et al. Experiment

The experiment [1] was conducted in a German school with students in age of 16 or 17. The entrance and exits were 3 meters wide for the first run. Then they added barriers. For the second run it was 2, for third 1.5 m and the last run the width was 1 m.

They also found out that the minimal speed of person was 0.72 m/s and maximal speed was 1.61 m/s.

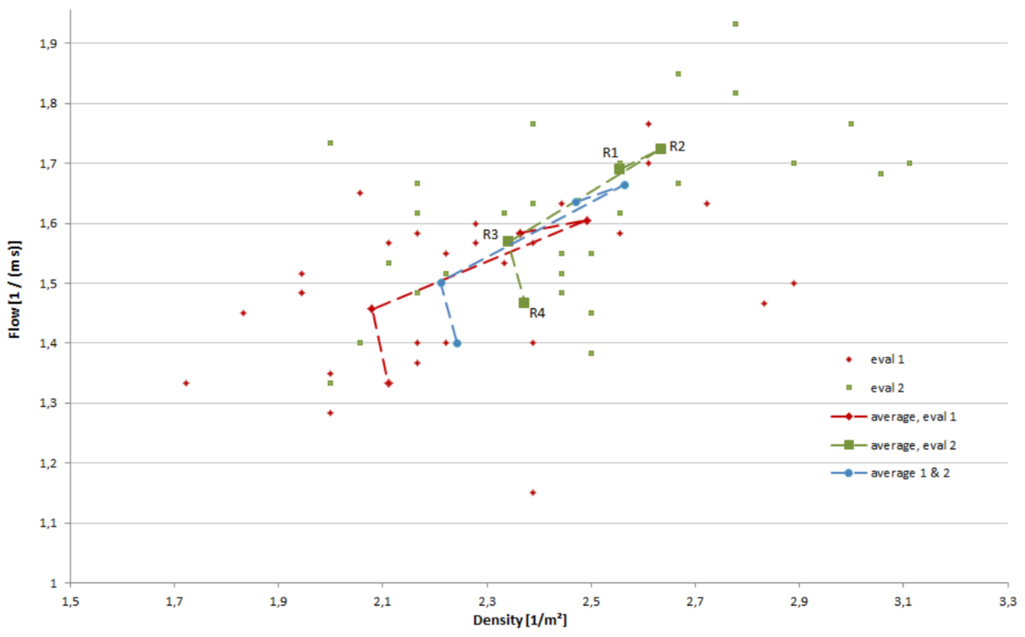


Figure 4.4 Bamberger et al. fundamental diagram [1]. in each run (R1 - R4) the results were manualy measured for two persons (in picture red diamond and green square). The blue circles mark the average of both counts. It is remarkable how stable the bias is between the two persons results.

4.2.3 Hatless Pedestrian MOT

Experiments without hats would facilitate easier realisation of moderated experiments in real environments, which would increase the amount of trajectory data.

In [34] they use stereo cameras to get the location of the pedestrian. However, it showed up that there needs to be lots of stereo cameras on a small place and these cameras are expensive. So the experiments continue to use some kind of marker to better obtain the pedestrians locations. To my knowledge, there was no pedestrian crossing flows experiment without hats.

Chapter 5

Model Design

The model is implemented in Python using the OpenCV[1] image processing library and a package called Improutils[2] [37] produced by ImproLab CTU FIT group. Tensorflow[3] [38] is used for the implementation of CNN that classifies binary codes. The chapter explains the model design.

5.1 Model Overview

The overview of model is presented in Figure 5.1. The model consists of 4 parts. It takes a video on input, outputs a dataset of trajectories.

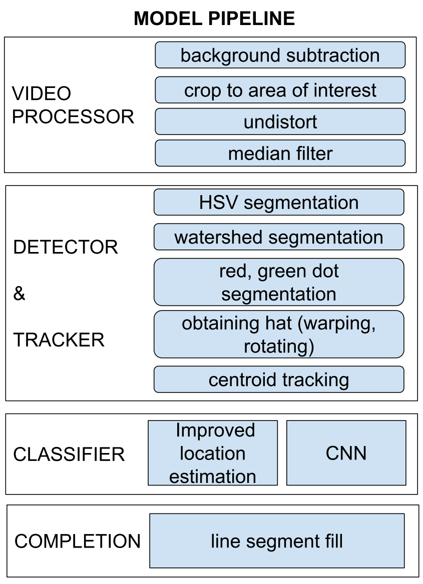


Figure 5.1 The figure shows model overview. The centroid tracking was programmed without using any MOT library. The CNN was trained from scrach.

[1]https://docs.opencv.org/4.5.5/
[2]https://gitlab.fit.cvut.cz/bi-svz/improutils_package/tree/master
[3]https://www.tensorflow.org/

5.2 Preprocessing

The video capture was loaded by the OpenCV VideoCapture object and the preprocessed frames were saved. The preprocessing included cropping to area of interest, undistorting with transformation, median filtering, and background subtraction.

5.2.1 Area of Interest

Frames were cropped only to the area of interest, as shown in Figure 5.2.

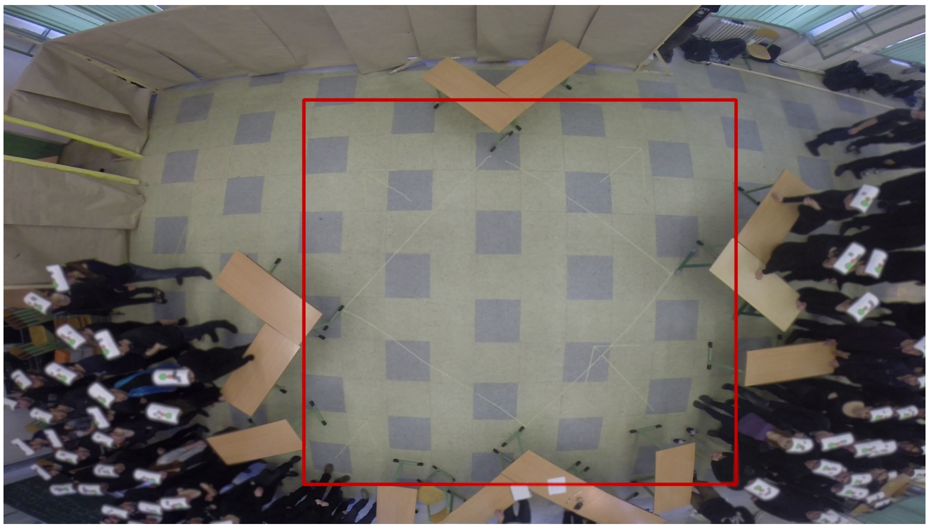


Figure 5.2 A frame of the experiment footage. The observed area is marked with a red rectangle. In this area, pedestrians are tracked. The location of the left corner is $(625, 197)$ pixels and has a size of $(900, 800)$ pixels. The pedestrians are blurred to comply with GDPR.

5.2.2 Undistort Image

To calibrate it is necessary to take enough pictures of known pattern (e.g. chessboard) from different angles. The camera matrix and the distortion coefficients can be computed from that.

Unfortunately, no calibration was done, so to remove distorsions some estimations described below were conducted. Tangential distortion is not simply observable in this case, so only radial distorsion is computed. Radial barrel distorsion can be comfortably seen by eye. This can be suppressed by estimating the radial coefficient. It is important because we want the locations of pedestrians to be as accurate as possible.

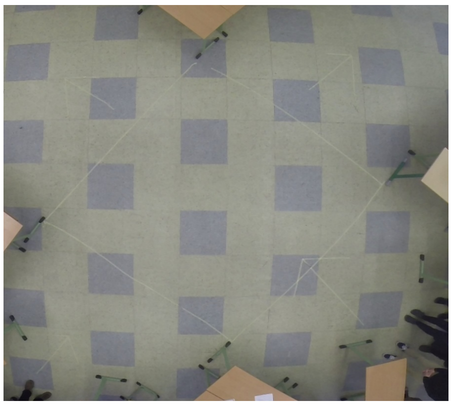
Distorsion coefficients are described in OpenCV [25]. The thesis works with k_1 coefficient. Since it is a positive radial distortion, the coefficients of undistortion must be negative. Found coefficients are in the Table 5.1. An OpenCV function called `undistort()` when given distorsion coefficient matrix (in this case only one coefficient was changed) performs the undistortion (in this case this is just a simple sinus transformation).

5.2.3 Background Subtraction

KNN background subtraction algorithm implemented in OpenCV was used. As shown in Figure 5.4, it creates a grayscale image where the foreground is marked white, the shades gray, and the

Table 5.1 Best found coefficients that fix radial barrel distorsion. The results were determined by looking at pictures by eye. Area of interest has a smaller coefficient (fixes larger distorsion). Since, the model works only with area of interest, whole image is not shown.

	area of interest	image
found coefficients	$-2.5e^{-5}$	$-4e^{-5}$


(a) The barrel distorted area of interest image. The corners of the image seem much further than they should.

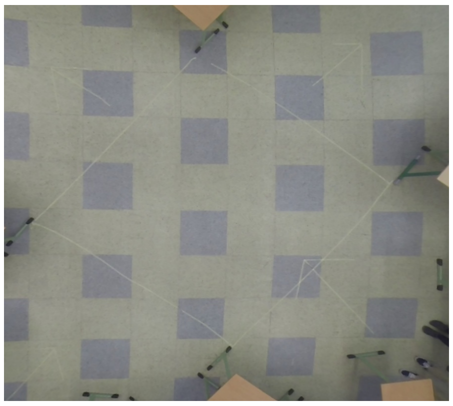

(b) Undistorted area of interest. After applying undistorion with coefficient -4^{-5}, barrel distorsion is less visible.

Figure 5.3 Fixing distorsion in the area of interest. Comparison of distorted and fixed area of interest.

background black. Since the algorithm sometimes produced small holes in the foreground mask, dilation with kernel (8,8) was used.

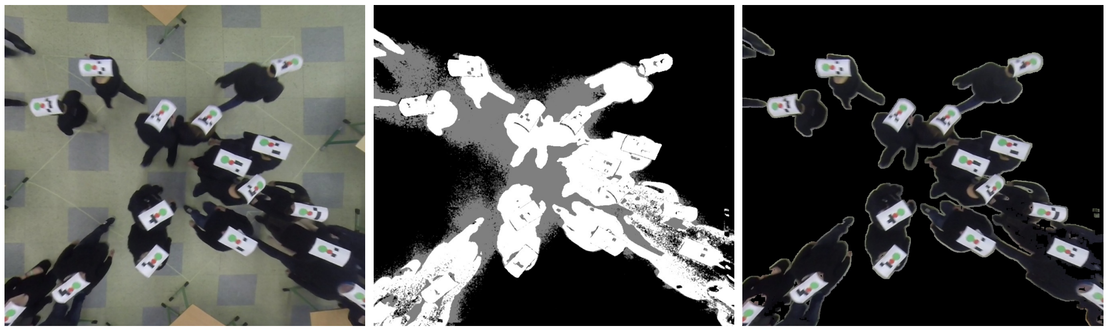


Figure 5.4 Background subtraction. On the left side, the original image. In the middle, a background subtraction model is placed. On the right side, output image, that was produced by application of dilated foreground mask. The foreground mask is obtained by only taking white pixels from the background model.

5.3 Object Detection

HSV segmentation with two tresholds computes a mask for white color. Since hats are the only objects of white color, this segmentation finds them. Then, convex contours are found.

Sometimes, it happens that the hats overlap. This is detected by finding more green points in one segmented area or by exceeding the threshold for hat size. In that case, the watershed

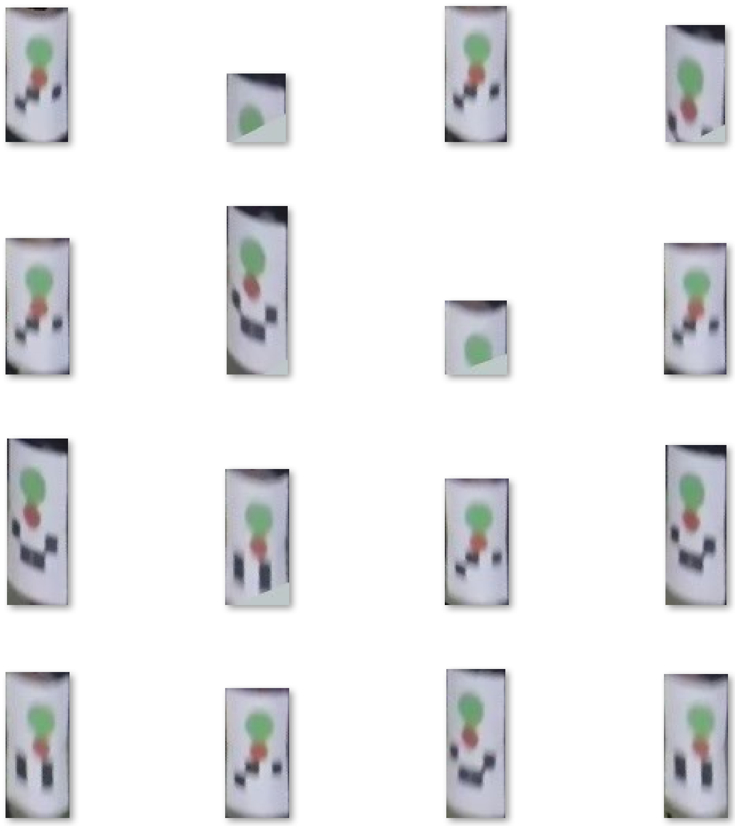


Figure 5.5 Example of segmented hats. There are two segmentations where the pedestrian is entering the area of interest so the hat is not whole.

segmentation is used to approximately divide the hats (shown in Figure 5.6).

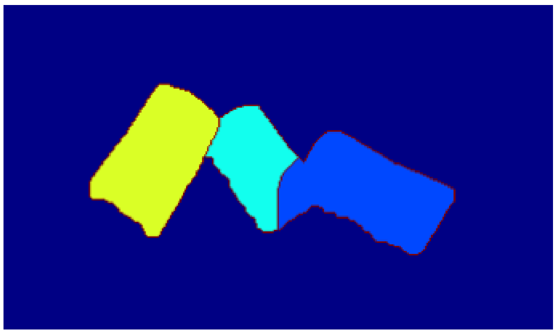


Figure 5.6 Division of the hats by watershed segmentation.

The green dots act as indicators of a segment for the watershed algorithm. Each pixel is assigned to the segment of nearest green dot. As shown in Figure 5.6 it is just an approximation of the hats; nevertheless, the bounding box will likely contain the binary code and red dot of the hat (these being the only features we are interested in). A contour is found for each watershed segment. Watershed algorithm in this case resembles Breadth-first search algorithm that would be run from each green dot. Breadth-first search algorithm was the first tried approach however the implementation was tremendously slow.

Now, the algorithm has all the contours of hats. It finds the minimal enclosing rectangle for each contour. With HSV segmentation detects green and red dots. Finds centers of dots and store the information about rotation angle.

The resulting images of hats are shown in Figure 5.5.

5.4 Multiple Object Tracking

For the tracking of multiple objects, a centroid tracking algorithm is used. The algorithm is described in Section 2.3. The model detects hats as described in Section 5.3. Euclidean distance is used as a metric for association of tracks and detection. The `MAX_DISSAPEARED` constant is set to 1, because the detection of hats is quite accurate and there are no oclusions of the objects.

The output of this algorithm is a dataset of tracks. The location of green dot, red dot, frame ID, and track ID is stored for each hat that was assigned to a track.

Centroid tracking is a simple algorithm, but it works well in this thesis. Kalman filter or a deep association metric was not used. Mainly because it was not necessary. Today centroid tracking is still used, e.g. in atmospheric research, when tracking thunderstorms [39]. It is very useful for tracking objects that are mostly still as the Euclidean distance is small. Another usage involves a video in a laboratory, where the environment is stable and the objects are not ocluded. That is the case of the crossing flows experiment.

5.5 Binary Codes Classification

The section presents different approaches for binary codes classification.

5.5.1 Naive approach - Estimation by Location

The naive approach was to locate the middle of each square, look at surrounding pixels and the majority color would determine the color of the square.

The color of the square was determined by looking at neighborhood of 4 pixels and taking the majority color. However, this approach had very little accuracy.

5.5.2 Improved Approach - Estimation with Contours of Binary Code

The improvement of the above approach was to make a mask and find all contours of black squares (sometimes connected into other shapes such as rectangles). Some areas of the picture

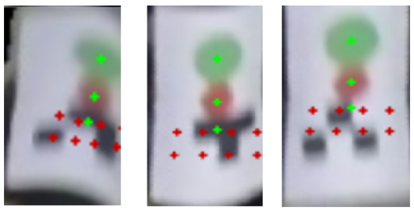


Figure 5.7 Approximate location of centers. The line determined by red and green dot was used to find perpendicular lines on which the centers of squares lie. With usage of the line segment between green center and red center, the ratio for sizes was measured.

were filtered out. Namely, everything before the ending of red dot contour was marked 0 in the mask. The lower crop line was determined by the absence of contours of the black squares. The number of squares $\hat{q}$ was estimated knowing that one square has area of aproximately 40 pixels. Then the $\hat{q}$ estimated location nearest to contours of black pixels were marked as 1 rest as 0.

5.5.3 CNN Approach

Another approach was to let a neural network decide for us what the binary code is. The task is determinig the color of 8 squares in a low quality picture. The neural network will have to somehow learn the positions where to look for the squares. CNNs are networks that work well when identifying objects on images and their locations.

The need of training data

Neural networks need enough training data. One possibility would be to annotate the dataset of hats and train a network with it. Since, the task is to annotate a single video automatically, anotating half of the video frames was not an obtion. Other option is to create an artifical dataset for the network training.

Artificial hats for all possible IDs were created. Possible hat IDs are even numbers since last bit of the code is a control zero (white) bit. Hats were created with OpenCV shapes. It is necessary to maintain proportions of the shapes on hats.

Network architecture

The hidden blocks architecture was inspired by Alexnet [40]. The building block of Alexnet consists of a convolutional layer, relu activation, and max pooling. This block is repeated several times, then there are some dense layers. Alexnet is trained for image classification of complex objects, like animals. It has 61M trainable parameters. The hats are way more simple. Therefore, less layers and computation power is needed.

Some improvements were made to increase the accuracy of the neural network. Namely a batch normalization layer was added after each convolution layer. The network also easily overfitted and probably only looked at few pixels and ignored the rest. To enforce the network to look at more pixels a dropout layer was added. Dropout layer trashes outputs of the previous layer with defined probability. That forces the network to learn from many more pixels. When adding the dropout layer, the needed epochs to train the networks doubled from 25 to 50.

A CNN with architecture shown below in Table 5.2 is used. The input is an image of shape (95, 63, 3), which was estimated that almost all images of hats would fit in. The output of the network were 8 neurons, each representing one digit in the binary code. The number of hidden blocks and dense layers were experimented with, so it would minimize error on 20 images (development data).

SGD optimizer performed better than Adam. MAE was used as a loss function. It looked at each predicted number from binary code and performed a mean squared error. The values closer together were not penalized that much as in mean absolute error. Those were the only losses considered for this task.

Data augmentation

The network without augmenting the training data was almost immediately overfitted on the train dataset. To prevent this behavior, artificial data are augmented for each epoch slightly differently.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 95, 63, 3)	0
conv2d (Conv2D)	(None, 48, 32, 8)	224
batch_normalization	(None, 48, 32, 8)	32
re_lu (ReLU)	(None, 48, 32, 8)	0
dropout (Dropout)	(None, 48, 32, 8)	0
max_pooling2d	(None, 24, 16, 8)	0
conv2d_1 (Conv2D)	(None, 24, 16, 16)	1168
batch_normalization_1	(None, 24, 16, 16)	64
re_lu_1 (ReLU)	(None, 24, 16, 16)	0
dropout_1 (Dropout)	(None, 24, 16, 16)	0
max_pooling2d_1	(None, 12, 8, 16)	0
conv2d_2 (Conv2D)	(None, 12, 8, 32)	4640
batch_normalization_2	(None, 12, 8, 32)	128
re_lu_2 (ReLU)	(None, 12, 8, 32)	0
dropout_2 (Dropout)	(None, 12, 8, 32)	0
max_pooling2d_2	(None, 6, 4, 32)	0
conv2d_3 (Conv2D)	(None, 6, 4, 32)	9248
batch_normalization_3	(None, 6, 4, 32)	128
re_lu_3 (ReLU)	(None, 6, 4, 32)	0
dropout_3 (Dropout)	(None, 6, 4, 32)	0
max_pooling2d_3	(None, 3, 2, 32)	0
flatten (Flatten)	(None, 192)	0
dense (Dense)	(None, 100)	19300
dense_1 (Dense)	(None, 50)	5050
dense_2 (Dense)	(None, 8)	408
Total params: 40,390		
Trainable params: 40,214		
Non-trainable params: 176		

Table 5.2 Summary of the CNN model.

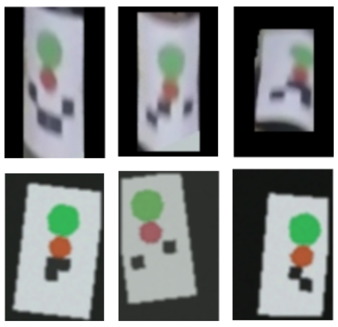


Figure 5.8 The first row are the hats from video. The second row are artificial augmented hats. They have different sizes, rotation, transformations and color alternations. The artificial hats are cropped and padded to have the same size.

The used data augmentation includes rotation with random angle up to 0.15 radians, shifts to left and right (maximum shift of 20 pixels), color changes (random hue and value), and adding Gaussian noise. They were implemented with the image module[4] in Tensorflow addons.

5.5.4 Binary Codes Classification Methods Results

As shown in Table 5.3 the results are not very satisfiyng. Both methods encountered problems with the bits near edges of the paper.

Traditional methods could perform better. This was just a first prototype. If we had found the centers of squares more precisely for connected squares and taken the center estimation shown in Figure 5.7, the score could have been much higher.

The hat paper edge tends to fall down, and the transformations are resembling a half of sinus vawe. When trying to add these transformations to the train dataset for CNN the network generalized very poorly on the hats even after long training.

The architecture of CNN was developed with trying a lot of possible architectures, all with AlexNet-like blocks. Most of the architectures overfitted quickly and did not generalize for hats data. The picked network 5.2 achived 93% accuracy on the train data and 45% accuracy on test data (20 randomly picked images).

Table 5.3 The models were evaluated on 20 randomly picked images, so the score is only partly indicative. The score was measured as the ratio of exact matches and all evaluation data. There were a lot of near mismatches (one bit was wrong) with both methods. Also this accuracy is later on increased as for each track there are more hats and the majority binary code is assign as the track ID.

model type	score
CNN	0.45
conventional methods	0.4

[4]https://www.tensorflow.org/addons/api_docs/python/tfa/image

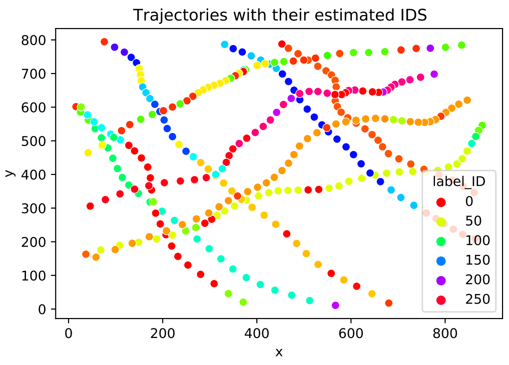


Figure 5.9 The estimated hat binary codes for trajectories in dataset. Majority hat identification was taken as the hat ID for the track.

5.5.5 Dead-ends

This section groups some experiments that were done but appeared to be dead end.

There were aproximately 47 000 images of hats. The image quality enhancement with OpenCV deep neural network (EDSR) for super-resolution would take approximately 16 days on a CPU with the available resources.

The images are shaped so the edges remind convex and concave curves. So the idea was to add some sinus transformed pictures to the dataset. The network that achived 0.45 on the hats sample had even more errors when using this data.

5.6 Completion of Incomplete Trajectories

At this point, a dataset of trajectories with hat identifications is available. The trajectories starting points, ending points, and the staring side were marked in the dataset. Also the trajectory hat ID was determined by the majority binary code present in the trajectory.

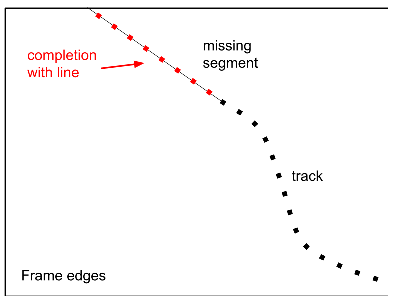


Figure 5.10 Track completion visualisation. The track is marked with black color. The missing segment completion is marked in red.

It can happen that the trajectory ends or starts before it reaches an edge of image. In that case the trajecotry is completed with a line connecting the last known location and an image edge. The direction is taken as a vector from last known two frames. An example of such trajectories can be found in Figure 5.11.

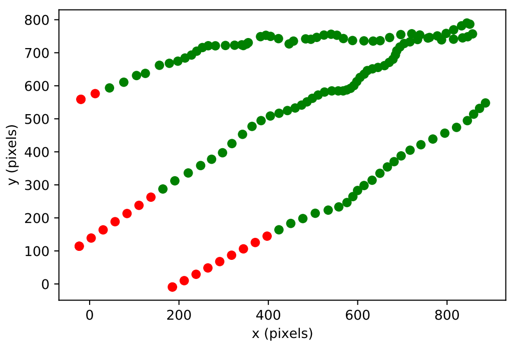


Figure 5.11 Example of completed trajectories. The red dots are the approximation for missing parts. Green dots represent the location of pedestrians on this frame.

Chapter 6

Results

The section analyzes the dataset of obtained trajectories, compares the results with results of other crossing flows experiments, and writes possible improvement instructions for the model.

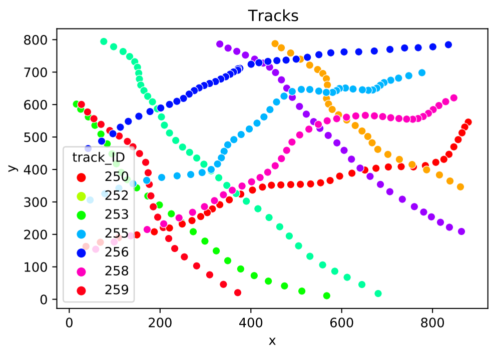


Figure 6.1 Example of extracted trajectories. Axis x and y are in pixels.

6.1 Data Analysis

After finding trajectories several interesting variables from pedestrian dynamics were found and observed.

Table 6.1 Overview of measured observables and their values for minimum, maximum, and mean.

observable	min	max	mean
speed	0 m/s	23.84 m/s	1.41 m/s
track speed	0.65 m/s	3.03 m/s	1.41 m/s
space mean speed	0.92 m/s	2.6 m/s	1.41 m/s
density	0.11 pedestrian/m^2	3.08 pedestrian/m^2	2.17 pedestrian/m^2

Velocity

Individual pedestrian velocity was measured by the relation specified in Equation 1.6. Since during preprocessing every 5th frame was taken the velocities are well detectable and measurable.

The basic review of velocity is provided with histogram in Figure 6.2.

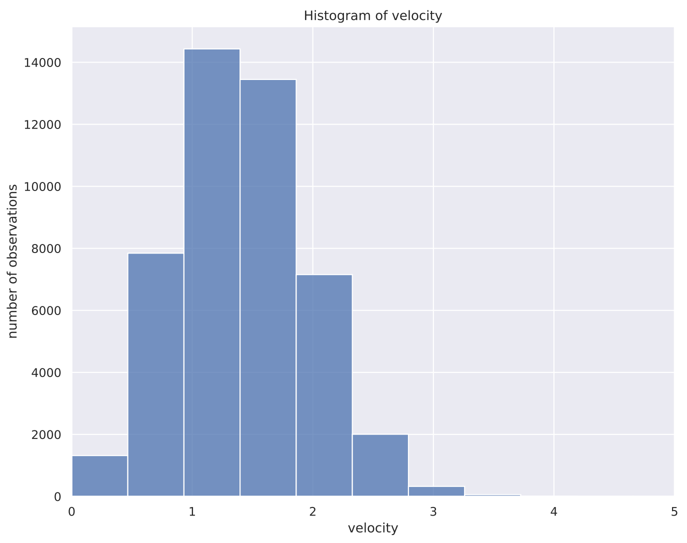


Figure 6.2 Histogram for velocity. The highest speed measured was 23.28 m/s, but it is evident from the histogram it was an outlier value. Most values were around 0 m/s to 4 m/s.

Track Speed

Track speed was measured as the mean value of all the track speeds. The average track speed was 1.41 m / s. Which corresponds to the average walking speed of a human: 1.4 m/s. The maximum track speed was 3.03 m/s, that is still less than the average speed for running being around 5 m/s. Histogram of track speed is shown in Figure 6.3.

Compared to Bamberger et al. [1] (described in Subsection 4.2.2) pedestrians in this experiment move much faster. In Bamberger et al. the minimal speed was 0.71 m/s and maximal speed 1.61 m/s. The average speed is also higher compared to BaSiGo_Cross_D experimet (described in Subsection 4.2.1). There were around 600 people in those experiments and the area of interest in the experiment was only a bit larger, so the lower speed is expected.

Macroscopic view

From the macroscopic view observables, space mean velocity and space mean density was calculated. Density properties are described by a histogram in Figure 6.4.

Fundamental diagram

The relation between velocity and density is shown in the fundamental diagram in Figure 6.5. The velocity for this diagram is space mean velocity (averaged velocity for each frame). The density is averaged density for each frame. The highest density observed was 3 pedestrians / m^2.

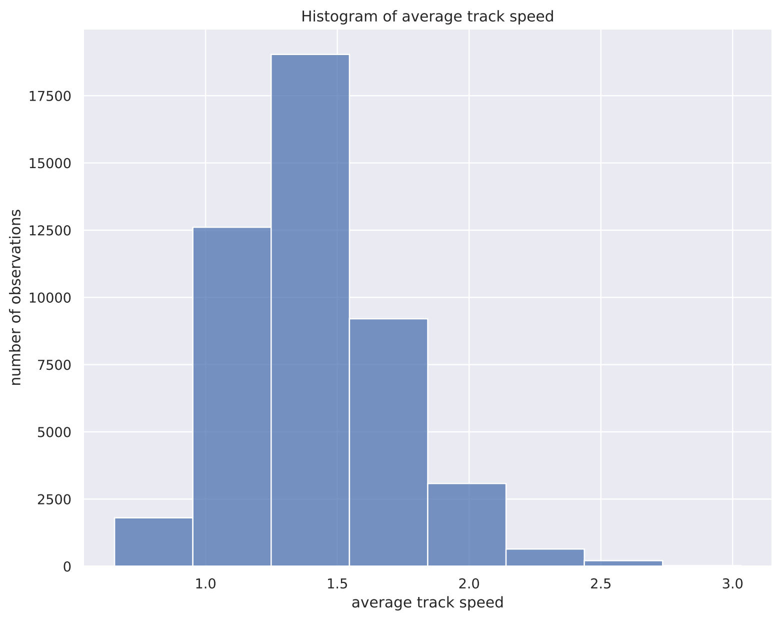


Figure 6.3 Histogram for average track speed.

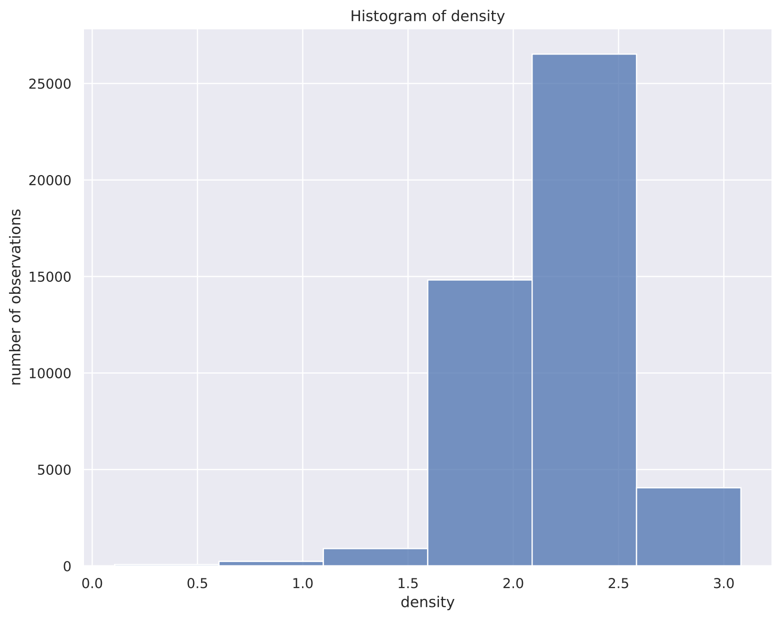


Figure 6.4 The Figure shows historgam for density.

6.2 Comparison with State of the Art

PeTrack was not able to detect pedestrians, due to different looking hats than what are used with PeTrack software. The inability to recognize hats is shown in Figure 6.7. PeTrack also expects

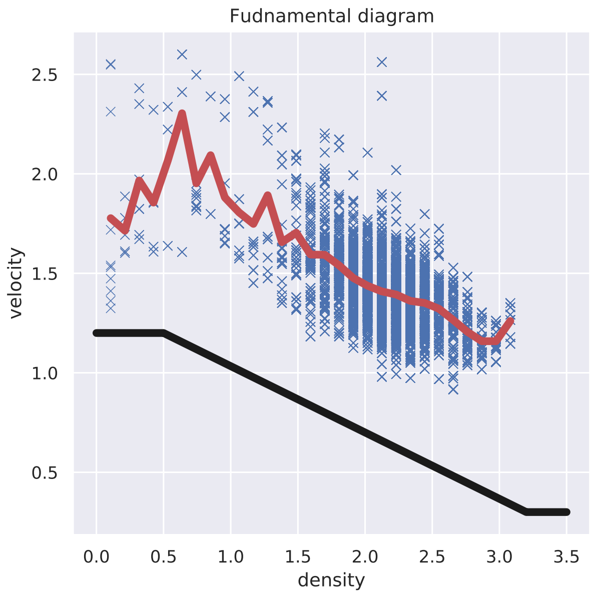


Figure 6.5 This figure presents fundamental diagram. The space mean velocity in m/s is on y axis. On the x axis is the density in pedestrian/m^{-2}. It can be seen that with higher density, pedestrians move with less speed. The red curve depicts the mean value of velocity for each unique density. This curve can be compared with the standard SFPE curve that is shown in black.

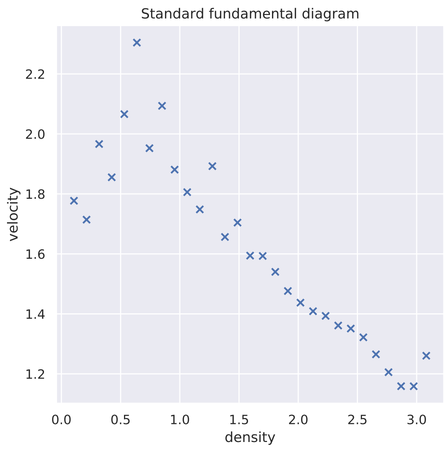
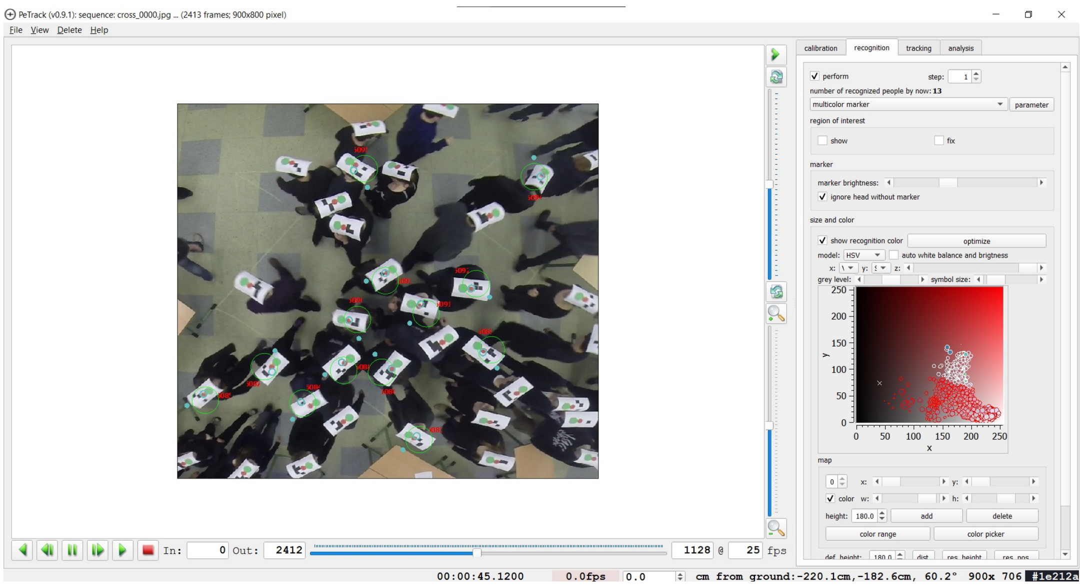


Figure 6.6 Standard fundamental diagram. The mean value of velocity was calculated for each interval of density.

a dark monochrome stable environment so it can use differential methods based on optical flow estimation. In this case the environment was a classroom with no such properties. That was solvable by background substraction. The problem with different looking hats remained. To process this experiment a custom model had to be made.



perform ✓ step: 1
number of recognized people by now: **13**
multicolor marker
region of interest: show ☐ fix ☐
marker: marker brightness:
ignore head without marker ✓
size and color: show recognition color ✓
model: HSV ☐ auto white balance and brigtness
height: 180.0
In: 0 Out: 2412 1128 @ 25 fps
00:00:45.1200 0.0fps 0.0 cm from ground:-220.1cm,-182.6cm, 60.2° 900x 706 #1e212a

Figure 6.7 Petrack software is unable to localize pedestrians due to different looking hats.

6.3 Improvements

Further improvements in MOT estimation step could be done. One of them is having a Kalman filter predict where the object should be next frame. That also with a more sophisticated association metric is done e.g. in a paper called DeepSORT [12].

There is a lot of room for improvement in the detection of binary codes. Some untried preprocessing methods could probably improve the detection. Or a network for scanning QR codes as a backbone for the model might achieve better performance.

Chapter 7

Disscusion

Frames from the video were obtained with OpenCV video capture object. A foreground mask was found with a background subtractor. The foreground mask was stripped of holes with dilatation. Then, it was used on the frame. The median filter was applied to reduce noise. The picture was undistorted with sinus transformation. The parameter for undistorion was approximated. For further experiments, it is recommended to calibrate the camera. Other approach for undistortion could have been detecting the corners of floor tiles and applying transformation so the floor tiles after transformation would be a square grid.

Second, the multiple object tracking was detection-based. Pedestrian hats were detected with HSV segmentation. The contours were made convex with the convex hull algorithm. Another approach could have been to detect the red dot that marked the center of mass. The approach with detecting hat contour had the advantage of having entire hat stored and not just the location of the center. The hats were rotated to become horizontal.

Multiple object tracking was performed using centroid tracking. It connects tracks in frames t and $t + 1$ by looking at the nearest centers of the bounding boxes. Centroid tracking is a simple algorithm. Several improvements can be made mainly during the association of tracks and detections. One of them is to incorporate the hat ID into the association part of the algorithm. Only Euclidean distance is used. Main drawback for centroid tracking compared to other algorithms is that object detection step needs to be run on every frame of the input video. If tracking were to be deployed somewhere, it would not be efficient, and there are much faster algorithms such as DeepSORT [12]. The efficiency in this thesis case did not matter, since it was for an evaluation of already taken video. The centroid tracking finishes in matter of minutes.

The naive approach for binary codes detection with traditonal methods looked at specific positions and its surrounding, then the majority color of pixels decided what the binary code is. The positions were estimated with usage of the red and green dot as described in Figure 5.7. However, this approach for reading binary codes was very inaccurate due to the deformations of the hats. This detection was improved by finding contours of black squares (each square representing a digit in the binary code). The q estimated locations closest to the black pixels were marked as 1 and the rest as 0. This algorithm achived 40% accuracy on a sample of 20 hats. The problem was to find the center of each square when the hat had connected squares. That can be further studied and when solved the accuracy will be much higher.

Other approach included using a CNN. The training dataset was made artificially with OpenCV. Then a CNN was trained to identify the codes on the artificial dataset. The crucial part was to do a lot of data augmentation for training examples (image rotation, shift left to right, flip vertically, and color variations). Then the detection of binary codes had better results. The CNN architecture was experimented with. The dropout layers were added and the number of hidden Alexnet-like blocks was determined to minimilize the loss on validation data. Further improvements include finetuning the CNN on a small sample of hats from the video. The hat ID of the entire trajectory was determined by the majority of binary codes present in the trajectory.

The missing parts of trajectories were completed using a line connecting last known locations. One of the improvements could be to use the estimated hat IDs to connect disconnected trajectories.

In the analysis, the velocity was counted as the difference between the previous and current locations divided by the time as in Equation 1.6. Other approach could be to use central differences like in Equation 1.5. The approaches are very similar, and it is a matter of choice. From a macroscopic view, the density and space mean velocity was observed. The density was taken as an average density for each frame. The fundamental diagram showed the relationship between these observables.

The observed velocities were slightly higher than what Bamberger et al. [1] observed. The SFPE curve also has lower values.

Future work can attempt to detect self-organization phenomenom such as line formation.

7.1 Recommendations for Future Experiments

The usage of dark clothes and hats were very useful for the automatic extraction as the rest of background was filtered out by background subtraction. This setup can be kept for future experiments with some small alternations.

To recommend, it is important to calibrate the camera. Otherwise the cofficients for undistorting need to be guessed. Noting the camera and lens can be useful.

The white control bit is not ideal as it is impossible to detect. Another color entirely would be advised.

The binary codes would be easier detetectable if the papers were not falling down on edges. So doing the hats from plastic or other solid material would be helpful.

If the papers contained a QR code like encoded number, the detection of such numbers would be much easier. QR codes have three control bits at three corners. That ensures that the location and angle of rotation is found. Also the center of the mass could be a special different colored bit. By that the people would be very easy to detect by an altered QR code scanner and the information about center of the mass would be there too.

Conclusion

The aim of this thesis was to obtain trajectories from the crossing pedestrian flow experiment.

The result of the thesis is an algorithm implemented in Python. The centroid tracking algorithm was used to extract pedestrian trajectories. Both, traditional methods and CNN, were experimented with to obtain hat IDS. The algorithm for hat ID extraction achieved 45% accuracy on a random sample of 20 hats. The missing parts of the trajectories were approximated with a line segment.

The thesis instructions were fulfilled. The resulting trajectories dataset can be further analyzed to gain more knowledge about pedestrian crossing flows. Compared to Bamberger et al. [1] and the standard FSPE curve, the thesis results show higher velocities.

Bibliography

1. BAMBERGER, Johanna; GESSLER, Anna-Lena; HEITZELMANN, Peter; KORN, Sara; KAHLMEYER, Rene; LU, Xue Hao; SANG, Qi Hao; WANG, Zhi Jie; YUAN, Guan Zong; GAUSS, Michael, et al. Crowd research at school: crossing flows. In: *Traffic and Granular Flow'13*. Springer, 2015, pp. 137–144.
2. KULIGOWSKI, Erica D. Computer evacuation models for buildings. In: *SFPE handbook of fire protection engineering*. Springer, 2016, pp. 2152–2180.
3. SCHADSCHNEIDER, Andreas; CHOWDHURY, Debashish; NISHINARI, Katsuhiro. *Stochastic transport in complex systems: from molecules to vehicles*. Elsevier, 2010.
4. STEFFEN, Bernhard; SEYFRIED, Armin. Methods for measuring pedestrian density, flow, speed and direction with minimal scatter. *Physica A: Statistical mechanics and its applications*. 2010, vol. 389, no. 9, pp. 1902–1910.
5. CAO, Shuchao; SEYFRIED, Armin; ZHANG, Jun; HOLL, Stefan; SONG, Weiguo. Fundamental diagrams for multidirectional pedestrian flows. *Journal of Statistical Mechanics: Theory and Experiment*. 2017, vol. 2017, no. 3, p. 033404.
6. SEYFRIED, Armin; STEFFEN, Bernhard; KLINGSCH, Wolfram; BOLTES, Maik. The fundamental diagram of pedestrian movement revisited. *Journal of Statistical Mechanics: Theory and Experiment*. 2005, vol. 2005, no. 10, P10002.
7. MAERIVOET, Sven; DE MOOR, Bart. Traffic flow theory. *arXiv preprint physics/0507126*. 2005.
8. THUNDERHEAD ENGINEERING CONSULTANTS. *Pathfinder software* [online]. 2019 [visited on 2022-02-20]. Available from: `https://www.thunderheadeng.com/pathfinder/`.
9. THUNDERHEAD ENGINEERING CONSULTANTS. *Pathfinder Technical Reference Manual* [online]. 2021 [visited on 2022-04-08]. Available from: `https://files.thunderheadeng.com/support/documents/pathfinder-technical-reference-manual-2020-1.pdf#page=84&zoom=100,96,133`.
10. PELLEGRINI, Stefano; ESS, Andreas; SCHINDLER, Konrad; VAN GOOL, Luc. You'll never walk alone: Modeling social behavior for multi-target tracking. In: *2009 IEEE 12th international conference on computer vision*. 2009, pp. 261–268.
11. LU, Wei-Lwun; TING, Jo-Anne; LITTLE, James J; MURPHY, Kevin P. Learning to track and identify players from broadcast sports videos. *IEEE transactions on pattern analysis and machine intelligence*. 2013, vol. 35, no. 7, pp. 1704–1716.
12. WOJKE, Nicolai; BEWLEY, Alex; PAULUS, Dietrich. Simple online and realtime tracking with a deep association metric. In: *2017 IEEE international conference on image processing (ICIP)*. 2017, pp. 3645–3649.

13. RAVINDRAN, Ratheesh; SANTORA, Michael J; JAMALI, Mohsin M. Multi-object detection and tracking, based on DNN, for autonomous vehicles: A review. *IEEE Sensors Journal.* 2020, vol. 21, no. 5, pp. 5668–5677.
14. KOLLER, Dieter; WEBER, Joseph; MALIK, Jitendra. Robust multiple car tracking with occlusion reasoning. In: *European conference on computer vision.* 1994, pp. 189–196.
15. KHAN, Zia; BALCH, Tucker; DELLAERT, Frank. An MCMC-based particle filter for tracking multiple interacting targets. In: *European Conference on Computer Vision.* 2004, pp. 279–290.
16. LI, Kang; MILLER, Eric D; CHEN, Mei; KANADE, Takeo; WEISS, Lee E; CAMPBELL, Phil G. Cell population tracking and lineage construction with spatiotemporal context. *Medical image analysis.* 2008, vol. 12, no. 5, pp. 546–566.
17. SMAL, Ihor; DRAEGESTEIN, Katharina; GALJART, Niels; NIESSEN, Wiro; MEIJERING, Erik. Particle Filtering for Multiple Object Tracking in Dynamic Fluorescence Microscopy Images: Application to Microtubule Growth Analysis. *IEEE Transactions on Medical Imaging.* 2008, vol. 27, no. 6, pp. 789–804. Available from DOI: `10.1109/TMI.2008.916964`.
18. SPAMPINATO, Concetto; PALAZZO, Simone; GIORDANO, Daniela; KAVASIDIS, Isaak; LIN, Fang-Pang; LIN, Yun-Te. Covariance based Fish Tracking in Real-life Underwater Environment. In: *VISAPP (2).* 2012, pp. 409–414.
19. LUO, Wenhan; XING, Junliang; MILAN, Anton; ZHANG, Xiaoqin; LIU, Wei; KIM, Tae-Kyun. Multiple object tracking: A literature review. *Artificial Intelligence.* 2021, vol. 293, p. 103448.
20. PATRICK DENDORFER Aljosa Osep, Laura Leal-Taixé. *MOT challenge website* [online] [visited on 2022-02-20]. Available from: `https://motchallenge.net/`.
21. MILAN, Anton; LEAL-TAIXÉ, Laura; REID, Ian; ROTH, Stefan; SCHINDLER, Konrad. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831.* 2016.
22. REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems.* 2015, vol. 28.
23. SMITH, Steven W et al. The scientist and engineer's guide to digital signal processing. 1997.
24. FIT CTU. *SVZ camera calibration* [online]. 2022 [visited on 2022-04-08]. Available from: `https://courses.fit.cvut.cz/BI-SVZ/tutorials/files/3/lens-defects.html`.
25. DEVELOPER TEAM OPENCV. *Camera Calibration OpenCV Documentation* [online]. 2019 [visited on 2022-04-08]. Available from: `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`.
26. DEVELOPER TEAM OPENCV. *Watershed segmentation OpenCV Documentation* [online]. 2019 [visited on 2022-04-08]. Available from: `https://docs.opencv.org/4.x/d3/d47/group__imgproc__segmentation.html#ga3267243e4d3f95165d55a618c65ac6e1`.
27. ZHENG, Tao; DUAN, Zhizhao; WANG, Jun; LU, Guodong; LI, Shengjie; YU, Zhiyong. Research on Distance Transform and Neural Network Lidar Information Sampling Classification-Based Semantic Segmentation of 2D Indoor Room Maps. *Sensors.* 2021, vol. 21, p. 1365. Available from DOI: `10.3390/s21041365`.
28. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.
29. FIT CTU. *Knowledge mining from data* [online]. 2021 [visited on 2022-04-20]. Available from: `https://courses.fit.cvut.cz/BI-VZD/`.

30. DARREN, Sam. *Typesetting neural network diagrams with TeX* [online]. 2019 [visited on 2022-04-08]. Available from: `https://medium.com/momenton/typesetting-neural-network-diagrams-with-tex-4920b6b9fc19`.

31. FAVONI, Matteo. *Equivariance and generalization in neural networks* [online]. 2021 [visited on 2022-03-20]. Available from: `https://indico.uis.no/event/12/contributions/245/attachments/175/271/Equivariance_and_generalization_in_neural_networks_QCHS_2021.pdf`.

32. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. Available from DOI: `10.1109/CVPR.2016.90`.

33. BUKÁČEK, Marek; HRABÁK, Pavel; KRBÁLEK, Milan. Experimental study of phase transition in pedestrian flow. *Transportation Research Procedia*. 2014, vol. 2, pp. 105–113.

34. BOLTES, Maik; SEYFRIED, Armin. Collecting pedestrian trajectories. *Neurocomputing*. 2013, vol. 100, pp. 127–133.

35. BOLTES, Maik; BOOMERS, Ann Katrin; ADRIAN, Juliane; BRUALLA, Ricardo Martin; GRAF, Arne; HÄGER, Paul; HILLEBRAND, Daniel; KILIC, Deniz; LIEBERENZ, Paul; SALDEN, Daniel; SCHRÖDTER, Tobias. *PeTrack*. Zenodo, 2021. Version v0.9. Available from DOI: `10.5281/zenodo.5126562`.

36. BOUGUET, Jean-Yves et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel corporation*. 2001, vol. 5, no. 1-10, p. 4.

37. IMPROLAB FIT. *Improutils package* [online]. 2021 [visited on 2022-03-20]. Available from: `https://gitlab.fit.cvut.cz/bi-svz/improutils_package/tree/master`.

38. TENSORFLOW DEVELOPER TEAM. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Available also from: `https://www.tensorflow.org/`. Software available from tensorflow.org.

39. DEL MORAL, Anna; RIGO, Tomeu; LLASAT, Maria Carmen. A radar-based centroid tracking algorithm for severe weather surveillance: identifying split/merge processes in convective systems. *Atmospheric Research*. 2018, vol. 213, pp. 110–120. ISSN 0169-8095. Available from DOI: `https://doi.org/10.1016/j.atmosres.2018.05.030`.

40. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, vol. 25.

Content of Attached Medium

```
│ readme.txt ........................ brief description of medium content
└─ src
   │ └─ mot ........................ implementation of centroid tracking
   │ └─ classification ............. implementations for classification
   │ └─ data ....................... data obtained from centroid tracking
   │ └─ dataset .................... dataset of trajectories
   │ └─ results.ipynb .............. notebook with visualisation of the results
└─ latex ........................... source codes of the thesis in LaTeX
└─ thesis.pdf ...................... thesis text in PDF format
```

The medium does not include video footage from crossing pedestrian flows experiment conducted at FNSPE in 2014 to comply with GDPR guidelines.