Analysing pedestrian movement with the Flowity software

Johan Karlelid | Division of Fire Safety Engineering | LTH | LUND UNIVERSITY

Analysing pedestrian movement with the Flowity software

Johan Karlelid

Lund 2020

Title: Analysing pedestrian movement with the Flowity software

Author: Johan Karlelid

Report: 5605
ISRN: LUTVDG/TVBB--5605--SE

Number of pages: 71
Illustrations: Images in the report, not produced by the author are referenced to and were either found online as open access or adapted from the original. The author claims fair use.

Keywords
Detection, Pedestrian detection, Egress, Computer vision, Video analysis, Speed, Flow, Density, Deep learning, Neural network, Fire safety, Evacuation, Engineering, Lund university, LTH

Abstract
This thesis surveyed existing technologies and research on pedestrian detection systems using video as input. The purpose of the study was to investigate what such a software would need to produce to be relevant for the fire safety engineer in outdoor circulation- or egress applications. The thesis tested a newly developed detection software called Flowity. The software, developed by ÅF Digital Solutions AB (a subsidiary to AFRY AB) utilizes machine learning and artificial intelligence algorithms to identify and detect objects and pedestrians. The objective was to establish a list of factors that the Flowity software should be able to extract and how, if to be useful for the fire safety engineer. The objective was also to conduct a case-study test of the software on a video of pedestrian movement and to evaluate/compare the capabilities to accurately identify and quantify pedestrian movement with the Flowity software and do a comparison to manually collected data. Results from the study show that the Flowity software could identify people and automatically presented them as detected pedestrians with a detection accuracy of 70 %. This applies for this case study, which was an outdoor high-resolution video recording containing 576 pedestrians, with the camera placed 4 meters above the walking area. The software managed to provide data on movement patterns, route choices of detected pedestrians as well as measuring movement speeds, flows and densities at different sections. The maximum global people density measured with the software was 0,13 persons/m^2 and the maximum local density was 3 persons/m^2. When comparing the manually and software measured flows and densities, there was no statistically significant difference between the measurement methods. However, a comparison between manually and software measured movement speeds showed a statistically significant difference between the measurement methods. A 14,2 % higher average flow was measured with the manual counting and a 15,1 % higher average global denisity. The software measured a 32 % higher average speed than what was manually measured. Uncertainties connected to the manual measurements and unknown influence of factors on detection performance might have impacted the results.

Brandteknik
Lunds tekniska högskola
Lunds universitet
Box 118
221 00 Lund

www.brand.lth.se
Telefon: 046 - 222 73 60

Division of Fire Safety Engineering
Faculty of Engineering
Lund University
P.O. Box 118
SE-221 00 Lund
Sweden

www.brand.lth.se
Telephone: +46 46 222 73 60

Acknowledgements

This rapport is produced as part of the course *Degree Project in Fire Safety Engineering VBRM01 (22,5 credits) during one semester at Lund University. This thesis concludes the Fire Protection Engineering Program.*

I would like to thank all the people who have patiently listened to me ramble on about the thesis and to all the persons listed below for their guidance, feedback and assistance.

Supervisor
Enrico Ronchi, Associate Professor at Division of Fire Safety Engineering, Lund University.

External supervisor
Steve Gwynne, Research Lead at Movement Strategies.

Flowity development team
Fredrik Hofflander, Section Manager at Future Technologies, AFRY AB.

Per Brendelökken, Consultant at Future Technologies, AFRY AB.

Others

Mohammad Mashhadawi, Fire Protection Engineer at AFRY AB.

Viktor Arozenius, Bi16

The author is responsible for the content in this report

Johan Karlelid
Lund, 2020

Summary

One way of better understanding crowd dynamics and how people behave in an evacuation situation is to accurately identify and quantify the pedestrian movement. In a time of increased building complexity, the need to deviate from the prescriptive design increases. The BBRAD, (The Swedish National Board of Housing, Building and Planning's general recommendations on the analytical design of a building's fire protection) offers a performance based design approach, with the possibility to numerically evaluate that, for instance, the ability of the population to escape safely in case of fire is upheld. To do so, the fire safety engineer would need accurate data on pedestrian movement as an input for this approach and to further improve the design of the evacuation modelling.

The purpose of this thesis was to survey existing technologies and research on pedestrian detection systems using video as input. The purpose was also to investigate what such a software would need to produce to be relevant for the fire safety engineer in outdoor circulation- or egress applications. The thesis tested a newly developed detection software called Flowity. The software, developed by ÅF Digital Solutions AB (a subsidiary to AFRY AB) utilizes machine learning and artificial intelligence algorithms to identify and detect objects and pedestrians. The objective was to establish a list of factors that the newly developed Flowity software should be able to extract and how, if to be useful for the fire safety engineer. The objective was also to conduct a case-study test of the software on a video of pedestrian movement and to evaluate/compare the capabilities to accurately identify and quantify pedestrian movement with the Flowity software and do a comparison to manually collected data.

The case-study was an outdoor video containing pedestrians entering a large Lund University building. The video was 760 seconds long and contained 576 unique individuals where the software should identify and quantify each pedestrian and their movement. The video did not include a high people density scenario, indoor spaces, evacuating pedestrians, presence of smoke, fire or limited visibility. However, the concept of automatically detecting pedestrians and extracting movement data are still relevant for circulation and egress applications, which are related to evacuation and fire safety engineering.

Results from the study show that the factors the Flowity software managed to extract were data on movement patterns and route choices of detected pedestrians as well as measuring movement speeds, flows and densities at different sections. Factors that were not possible to extract, given the current state of the software, were the age, gender, body size, and any movement impairments of the pedestrians. The Flowity software could identify individual people and automatically presented them as detected pedestrians with a detection accuracy of 70 %, meaning, that 30 % of the visible pedestrians were not detected by the software. This applies for this case study, filmed at a 1080p resolution, with a camera placed 4 meters above the walking area and at an angle of 21 degrees. The maximum global people density measured with the software was 0,13 persons/m^2 and the maximum local density was 3 persons/m^2. When comparing the manually and software measured flows and densities, there was no statistically significant difference between the measurement method. However, a comparison between manually and software measured movement speeds showed a statistically significant difference between the measurement methods. A 14,2 % higher average flow was measured with the manual counting and a 15,1 % higher average global denisity. The software measured a 32 % higher average speed than what was manually measured. This could be due to the high outliers that the software produced or the uncertainties in the manual measurements. Further investigation in required.

Flowity is a new software, never tested before for circulation/egress applications and there is need for further testing and research. The development of a general protocol to be used when systematically testing pedestrian detection softwares needs to be researched in the future along with research into factors and how they influence detection performance. If pedestrian detection softwares are proven to work successfully in getting accurate and relevant pedestrian movement data, the applications could be useful to the fire safety engineer but also broad in nature.

Table of content

Abbreviations

AI – Artificial Intelligence
LUP – Lund University Publications
TP – True Positive
FP – False Positive
TN – True Negative
FN – False Negative
MR – Miss rate
FPPI – False Positive Per Image
DNN – Deep Neural Network
CNN – Convolutional Neural Network
ToF – Time of Flight
LiDAR – Light Detection and Ranging
HOG – Histogram of Oriented Gradients
LBP – Local Binary Pattern
SVM – Support Vector Machine
AdaBoost – Adaptive Boosting
PCA – Principal Component Analysis
GDPR – General Data Protection Regulation
API – Application Programming Interface
LoS – Level of Service
KS – Kolmogorov-Smirnov

List of figures

Images in the report, not produced by the author are referenced and were either found online as open access or adapted from the original. The author claims fair use.

List of tables

1 Introduction

As modern buildings get increasingly more complex the importance of knowing where people are and how people move in an emergency situation cannot be overstated. Accurate information about movement speeds, route-choices, flows and people densities are crucial in the understanding of crowd dynamics, pedestrian egress movement and the kind of emergency system that need to be in place. Knowledge and skill in these areas are required by the fire safety engineer in order to facilitate safe buildings in the future.

The use of sensors or cameras (either mono or in stereo), combined with algorithms to detect and track pedestrians have been actively utilized and researched the last decade. Computer vision involves having computers conduct high-level distinctions, detections or other tasks similar to what the human visual system can achieve (Huang, 1992). The use of computer vision has in recent years become much more popular. However, a survey of the top 16 pedestrian detectors utilizing algorithms on mono camera footage (Dollár, Wojek, Schiele, & Perona, 2011) concludes that the detection performance at that time was not adequate enough for partially occluded pedestrians or for low resolution footage, i.e. the persons within the image was not represented with enough pixels to be detected. In (Boltes & Seyfried, 2013), an extensive review of research on pedestrian detection using different camera techniques was completed. They state that most of the detection using mono camera setups have a detection accuracy of about 90% and the stereo setups will often fall short in distinguishing between people in larger crowds due to the difficulty in segmenting fore- and background objects.

The most recent pedestrian detectors use a hybrid between feature algorithms, AI- (artificial intelligence) technology and deep learning, showing an excellent performance in detection (Baabou, Ben, Ben Farah, Abubakr, & Kachouri, 2019). Deep learning models apply features from deep neural networks (DNN) or convolutional neural networks (CNN), which is considered a subset of artificial intelligence. Simplified, they are a mathematical way to calculate a probability that an input (visual imagery) is equal to a selected output (e.g. pedestrians) by enabling features to be extracted on multiple layers (Schmidhuber, 2015).

The Flowity software developed by ÅF Digital Solutions AB (a subsidiary to AFRY AB) uses the latest machine learning and AI (deep neural networks) technologies in order to detect and analyse any object or situation from a video dataset. If this software is proven to work successfully in getting accurate and relevant pedestrian movement data, the applications could be substantial and broad in nature.

1.1. Purpose and objectives

The purpose of this project was to survey existing technologies and research on pedestrian detection systems using video as input. The purpose was also to investigate what such a software would need to produce to be relevant for the fire safety engineer in outdoor circulation- or egress applications.

The objective was to establish a list of factors that the Flowity software should be able to extract and how, if to be useful for the fire safety engineer.

The objective was also to conduct a case-study test of the software on a video of pedestrian movement and to evaluate/compare the capabilities to accurately identify and quantify pedestrian movement with the Flowity software and do a comparison to manually collected data.

1.2. Limitations and delimitations

The review of existing detection technologies was limited to systems using video as input and applications utilizing sensors, cameras, algorithms or neural networks, as the Flowity software is able to utilize these technologies. No investigation into detection through laser, wifi-, Bluetooth- or GPS technologies were made.

The testing of the Flowity software was limited to testing only one outdoor case and the analysis of one video. This was done to focus on trying out a new software and to check if the detection of pedestrians was possible for a simple scenario. This was performed to assess if relevant pedestrian movement data could be extracted. Therefore, no testing of different video resolutions or camera angles were done. The testing did also not include indoor videos with limited visibility (e.g. where darkness or smoke were limiting the visibility) or footage taken from thermal imaging cameras.

The case-study was an outdoor video and it did not include some aspects relevant for the fire safety engineering applications such as a high people density scenario, indoor spaces, evacuating pedestrians, presence of smoke, fire or limited visibility. However, the concept of automatically detecting pedestrians and extracting movement data are still relevant for circulation and egress applications, which are related to evacuation and fire safety engineering.

Measurements of the area where the video was recorded were taken on-site and will be generating some margin of error and uncertainties. Manual collection of pedestrian data was made, and the entire video was watched two times, to count all the pedestrians at different sections. This was only made by the author. The time for pedestrians to pass certain sections (transit time) was collected, counting only seconds with no decimals. The distance the pedestrians covered between two sections was manually measured but was considered to be a simplification of the exact travel path and will generate uncertainties. Taking the manual collection of measurements and pedestrian data into account, the final results could have been affected.

In reality, the accuracy of the Flowity software will be dependent on a range of different factors. An investigation into factors and how they impact performance was not done in this thesis given the time constraints. The video was applied in two separate instances of the software, one to achieve a video render showing the detection of pedestrians and one where raw pedestrian data were output. This approach was required given a current limitation of the software. The video was used to analyse detection accuracy and the manually measured pedestrian data could be used as a benchmark to which the Flowity system results was then compared. In addition, certain details about the Flowity software are not addressed, at the request of the developers, given commercial sensitivities.

The list of factors that the Flowity software should be able to extract and how, did not include factors such as detection of flames, smoke, debris or lowered visibility. Those were not the focus in this thesis.

The high technical level, with regards to the section on existing methods for pedestrian detection (especially algorithms and neural networks) led to descriptions being simplified to fit an audience not familiar with the computer vision research field.

2 Methodology

The workflow for the project can be seen in Figure 1. The initial phase of the project comprised a literature review of previous research on existing methods for pedestrian detection and performance evaluation. In the next phase, a list of factors (deemed relevant to the field of circulation and egress) was established and was introduced to the Flowity development team in an effort to develop a suitable version of the software, to later be tested on a video of pedestrian egress movement. The next phase included, sourcing a video suitable to fit the purpose and objective of the thesis and the implementation of the Flowity software, which was completed at the AFRY-offices in Gothenburg by the development team at Flowity. Lastly an analysis of the software capabilities and a discussion about the project, limitations and future research was elaborated on.

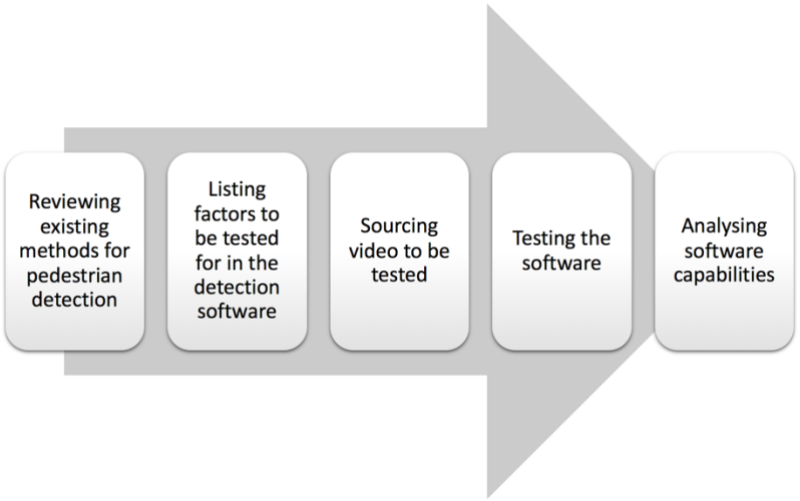


Figure 1 Thesis workflow.

Sourcing information and research on the subject was done mainly on the following platforms: Elsevier/ScienceDirect, ResearchGate, LUP (Lund University Publications) and Google Scholar. Keywords included pedestrian detection, egress, computer vision, video analysis, deep learning, neural network, convolutional networks, fire safety, evacuation modelling to mention some. An estimated 50-60 research articles were read, and the available literature was comprehensive, e.g. the number of research articles on “pedestrian detection” on ScienceDirect was 1128 as of this year. The interest and amount of research done has gone up immensely the last two decades, see Figure 2 below.

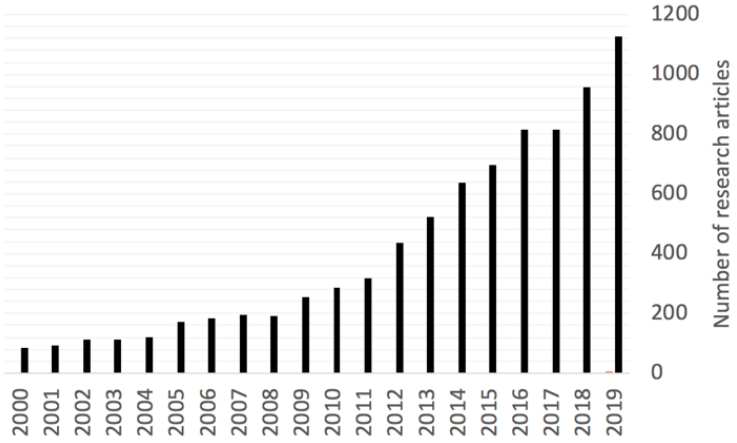


Figure 2 Number of published research articles on ScienceDirect with keywords "pedestrian detection".

2.1. Flowity software

Flowity is an artificial intelligence software developed in the beginning of 2019 by the future technologies' division at ÅF Digital Solutions AB (a subsidiary to AFRY AB) in Gothenburg. The development team specialises in AI, machine learning, computer vision, programming, statistical methods, algorithms and communication. The software utilizes AI, DNN, machine learning and a range of algorithms to detect objects within a video. The information regarding details about the software will be relatively limited and some crucial parts will be left out as they are not to be made public by company request.

The system is machine learned by different image datasets containing pedestrians, cars, trucks, trains, traffic lights or other objects that suit the specific need for a developed system. The detection is done by the feature-layers in the neural network and the data can subsequently be integrated to an automatic report generation with the desired statistics. This is achieved through the use of an open API (application programming interface) and the current output data format is either a video render of the detections or a JSON (JavaScript Object Notation) file containing raw data, for instance all the detected pedestrians GPS-coordinates and timestamps. The application of Flowity as of today is mostly used within traffic analysis (Tedblad, 2019).

The software developers claim that the software features are GDPR (General Data Protection Regulation) compatible as well as having the ability to be run on any present video setup (CCTV or otherwise). Analysis can be made either in real time or by postprocessing, depending on the applications. These facts were neither attested nor ruled out in this thesis.

The initiation of a constructive dialog with the team behind the Flowity software was done during the early phases of the thesis. This gave both parties an understanding of the purpose and objective of the project as well as enabling an in depth understanding of the Flowity software and its limitations.

3 Factors relevant to consider in a pedestrian detection system

The following section describes which factors and what kind of outputs would be relevant for a pedestrian detection system to extract and how, when doing video analysis of pedestrian movement. The factors all have a relevance to the field of circulation and egress. The factors were introduced to the Flowity software developing team in an effort to establish a desired approach for analysing a video of pedestrian movement. The result of this approach is later described in the results.

As stated by (Bukowski & Tubbs, 2016) in the SFPE Handbook of Fire Protection Engineering, *"Among the most important concepts in fire safety in buildings is to manage those potentially exposed to the fire and its effects, either by protecting them in place or by moving them to a place of safety"*. The concept of egress is explained and is generally defined as a safe path of travel that leads all the way to the exit. When doing analysis of pedestrian movement, knowing which exits and route choices people make as their means of safe egress as well as the flow through these openings will be important factors to consider. A pedestrian detection system could facilitate the protection of those potentially exposed to harm by quickly detecting people and knowing where they are located.

The Swedish building code regulations ('Boverket´s building regulations – mandatory provisions and general recommendations, BBR') states in the latest edition (BFS 2011:6 with amendments up to BFS 2019:2) that: *"Buildings shall be designed to ensure that there is an adequate time for evacuation during a fire"* and follows up with general recommendations about maximum walking distance to the nearest escape route as well as recommendation for number of exits and their respective widths and heights. Knowing the distance people cover during an egress situation and the time it takes are important factors to monitor when doing analysis or assessing evacuation safety.

With increased building complexity, the need to deviate from the prescriptive design increases. The BBRAD ('The Swedish National Board of Housing, Building and Planning's general recommendations on the analytical design of a building's fire protection') offers a performance based design approach, with the possibility to numerically verify that, for instance, the ability of the population to escape safely in case of fire is upheld. Verification of deviations from the prescriptive design can be made either through qualitative assessment, scenario analysis or quantitative risk analysis.

One option in order to evaluate the ability of safe escape in case of fire is to use an evacuation modelling tool. Work by (Kuligowski, Peacock, & Hoskins, 2010) reviews over 25 different computer evacuation models. The continuous egress model, Pathfinder is one of the reviewed models and is currently frequently used by Swedish fire protection engineers. A recent paper by (Lovreglio, Ronchi, & Kinsey, 2019) identified 72 different models currently used, Pathfinder being the most well-known and well-used. Surveying the technical reference from Pathfinder, (Thunderhead Engineering, 2019) the user defined parameters with regards to pedestrian movement can be examined. Factors such as comfort distance, body sizes, movement speeds, impairments etc. can be defined by the user either by a constant value, a uniform distribution between two values, a normal- or a log-normal distribution. Acquiring data about these factors from analysis of pedestrian movement will be important to the present work within evacuation modelling.

In an egress situation, the relationship between people density and flow is of interest. This relationship has been widely researched and the most frequently used example, (Fruin, 1987)

describes a model called LoS (Level of Service). Basically, with increased people density (people per square meter), the flow will fall. Fruin divides the model into six different design levels (A-F) as described by Fruin´s LoS, which all represent different densities and resulting flows, LoS "A" represents the threshold of unimpeded free flow. Fruin's concepts can be related to the so-called "fundamental diagrams", depicting relationships between speed, density and flow. These have long been researched and (Vanumu, Ramachandra Rao, & Tiwari, 2017) reviewed different pedestrian flow characteristics that were developed for a variety of situations. Measuring the people density, speeds and flows during analysis of pedestrian movement will be essential in the understanding of the relationships and to aid the continuous improvement of the evacuation modelling. The Flowity software should be able to extract data on these factors in order to be useful for fire safety engineering applications.

3.1. Factors

Based on information from the previous section and knowledge acquired from the fire engineering program a list of factors relating to pedestrian (1) detection and (2) counting/tracking. This was done to differentiate between factors and because the software addresses these approaches differently.

Within detection, the most obvious capability of a pedestrian detection software will be to distinguish people from other objects and to exclude non-desirable detections, for instance mannequins, reflections or certain staff members (to ensure high detection accuracy). It will also be important to detect certain pedestrian attributes, e.g. age, body size, gender, impairments etc. An adaptation to the specific purpose and scene will always have to be done. Once the detection has been achieved the software should be able to track and count pedestrians as well as track them in relation to user defined object, i.e. a corridor wall, a door opening, a line etc. This requires the software to enable user inputs and memorize characteristics of the pedestrians. Tracking the pedestrian should be able to produce representative walking speeds, distance walked, the distance kept to other pedestrians or objects, the time to safety and route choices made by each pedestrian. Monitoring user defined areas, factors such as flow, people densities and space occupation can be extracted. Table 1 below presents some detection factors that the Flowity pedestrian detection software ought to extract from the case video.

Table 1 Detection factors.

	Factor	***Description***
Detection	Pedestrian	Detect humans and classify them as pedestrians., not including mannequins, reflections, certain staff, people on bikes, inside cars or carrying heavy objects etc.
	Age	Make a distinction between different age groups, children, elderly or adults.
	Gender	Make a distinction between male and female pedestrians.
	Body size	Make a distinction between different body sizes, measuring shoulder width and height in meters.
	Movement impairment	Detect wheelchair users or people with movement impairing disabilities.
	Incapacitation	Detect pedestrians who are immobile (e.g. passed out due to smoke)

Table 2 below present some counting/tracking factors that the Flowity pedestrian detection software ought to extract from the case video.

Table 2 Counting/tracking factors.

	Factor	***Description***
Counting/Tracking	Pedestrian speed	Track individual walking/running speed at any given time and position in meters per second.
	Pedestrian density	Count pedestrian density at any given time and area in persons per square meter.
	Flow rate	Count the number of pedestrians passing by a section per second (door, stairs, corridor or other user defined sections).
	Distance walked	Track individual distance covered in meters.
	Distance between pedestrians	Track the distance (in meters) maintained between pedestrians, indicating level of comfort distance.
	Distance to objects	Track the distance (in meters) that pedestrians maintain with obstructions. Could be walls/corridors etc. (user defined).
	Dwell time	Count time (in seconds) that an individual has been standing still, indicating congestion or indecisiveness
	Time to safety	Count the time (in seconds) it takes for the pedestrians to get to a safe place (user defined)
	Route/exit usage	Track the route/exit (user defined) usage that pedestrians take
	Space occupation	Count the number of pedestrians passing through an area (user defined)

3.2. Outputs

This section will describe the key desired data output from a pedestrian detection software. Graphical aspects or aesthetics will be characterised by the authors subjective opinion and are obviously subject to change. This section is only meant to serve as an example.

The highest possible accuracy in detecting pedestrians is obviously desired. Performance and evaluation are described in Section 4.2. The software should, for visual and presentational purposes, output a video-render with detected pedestrians and also make distinctions between age groups and gender, e.g. by having colour coded bounding boxes and symbols above the boxes (there are probably other means of achieving the same effect, this is just one example). Pedestrians having movement impairments or who are incapacitated should be visually distinct e.g. by having a separate geometry of the bounding boxes or separately marked. Figure 3 below illustrates how some of these implementations might look like. This is only the authors subjective opinion.

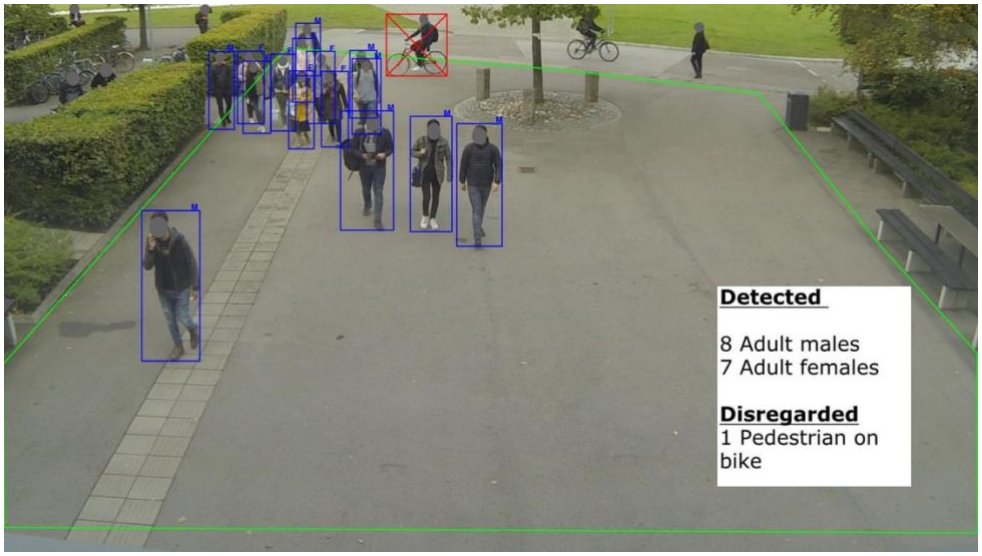


Figure 3 Hypothetical implementations of detection factor outputs

The software should apart from a video-render also generate data statistics, perhaps in the form of spreadsheets containing each pedestrian present within the footage and all the measured factors. Through the use of parametric equations, each pedestrian would have the coordinates and time recorded which in turn could led to the extraction of other data, e.g. speed, acceleration, density, distance to other pedestrians, distance covered, etc. Figure 4 illustrates an example of how the situation and parametric data might look like.

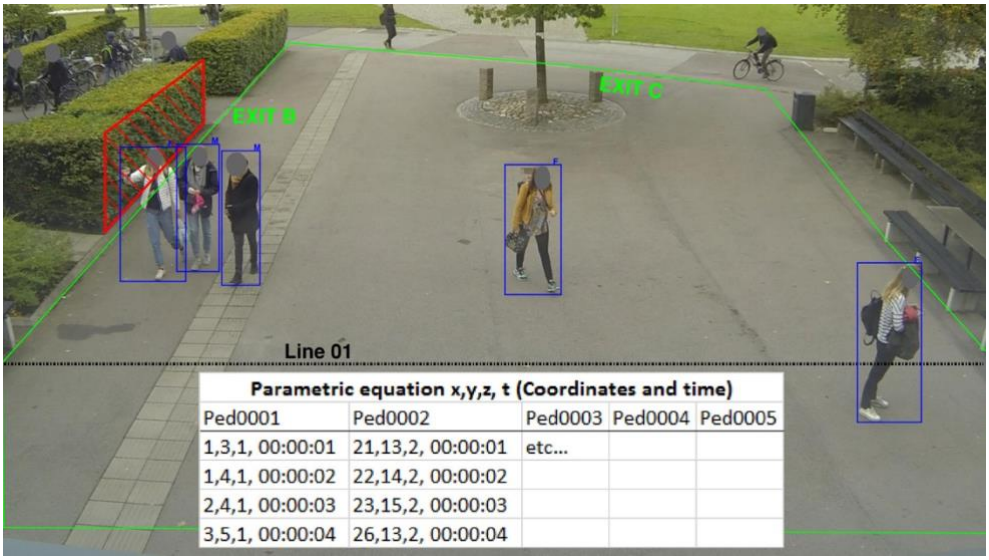


Figure 4 Hypothetical implementation of parametric outputs.

Some of the detection factors combined with the counting and tracking ones could be extracted and logged for each pedestrian separately. Table 3 below illustrates how this might look. In addition, the software could display the routes pedestrians take for egress, preferably on a 2D or 3D drawing of the area either as static trajectory map or a dynamic video.

Table 3 Hypothetical detection-, counting- and tracking factors logged for each pedestrian

	Pedestrians				
	Ped0001	Ped0002	Ped0003	Ped0004	Ped0005
Gender (male / female)	m	m	f	f	f
Age group (child / adult / elderly)	a	a	a	a	a
Shoulder width (m)	0,43	0,4	0,36	0,41	0,38
Height (m)	1,78	1,83	1,69	1,74	1,77
Wheelchair or other impairment					
Distance covered (m)	23	24	26	20	19
Dwell time (s)	1	1	8		
Time to safety (s)	29	23	26	27	28
Incapacitated					

Some of the user dependent factors, such as, which sections to measure flow at and which obstructions to measure distance to are preferably presented as individual data sheets. Table 4 and Table 5 below illustrates how this may look, in different tables. The hypothetical obstruction in this case, is the hedge section, marked by a red box and the investigated flow sections corresponds to Line01 present within Figure 4.

Table 4 Example outputs of user some user defined factors.

	Area usage	***Flow rate (p/section*s)***
Time (1 sec interval)	No. Of Pedestrians	Line01
00:00:01	2	2,1
00:00:02	3	1,9
00:00:03	4	1,8
00:00:04	5	2,4
00:00:05	5	2,3
00:00:06	5	1,9

Table 5 Example outputs of user some user defined factors.

	Distance kept to obstruction (m)		
Time (1 sec interval)	Ped0001	Ped0002	Ped0003
00:00:01	0,15	0,13	etc…
00:00:02	0,15	0,13	
00:00:03	0,2	0,1	
00:00:04	0,35	0,15	
00:00:05	0,4	0,22	
00:00:06	0,4	0,21	

Further analysis of the statistics is not crucial as a standalone output, e.g. presentation of the lowest, highest, average and standard deviation value of the different factors measured can easily be extracted from the data sheets.

4 Existing methods for pedestrian detection

This section will introduce the reader to the subject of pedestrian detection and give examples of some of the ways of measuring detection performance. It will also review some existing technologies that claim success in pedestrian detection, tracking, or counting. The survey will give a short introduction on the technology and exemplify research efforts performed with them. A large portion is predominantly allocated to deep neural networks and machine learning as they are used in the Flowity software.

4.1. Pedestrian detection

Pedestrian detection is the computational process of localising people within a scene, an image or a video frame and then represent the detected human with points, boxes, lines or otherwise. The choice of ways to represent a detected human will have implications on further aspects and should be chosen with that in mind. For example, representing a detected human with a rectangular box might not be suitable when investigating pedestrian leg movement but might be suitable for some other objective. Figure 5, from Object Tracking: A Survey by (Yilmaz, Javed, & Shah, 2006) presents different ways to represent detected pedestrians.

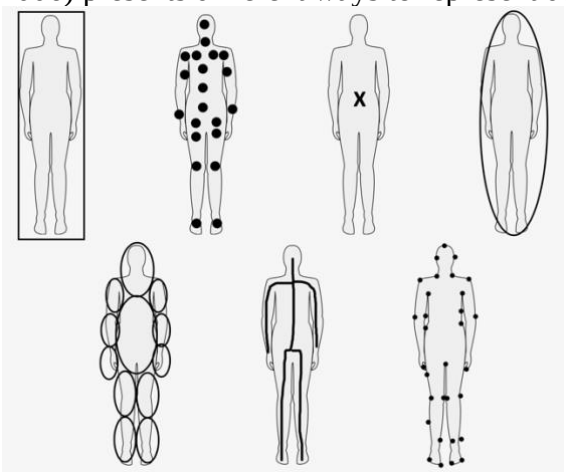


Figure 5 Different pedestrian representations. Figure adapted from (Yilmaz et al., 2006)

4.2. Performance and evaluation

There will always be different detection errors, either due to software- or hardware limitations or due to the scene's characteristics. How well a software will detect, track and count pedestrians will be dependent on a range of different factors. The software's implemented features and algorithms will predominantly affect performance, however environmental, video and pedestrian factors will also have a large influence (Yilmaz et al., 2006). Figure 6 seen below shows some of these factors and at the same time highlights the flow of performance dependence i.e. the software's counting performance will be dependent on the tracking performance which in turn is dependent on the detection performance and so on. Notably, this is probably not all of the factors that might influence this process and the influence they have on the final performance is not known by the author (i.e. the number and connectivity of the branches shown in Figure 6 might be more complex than that shown). This is only meant as a quick overview and to give the reader some insight.

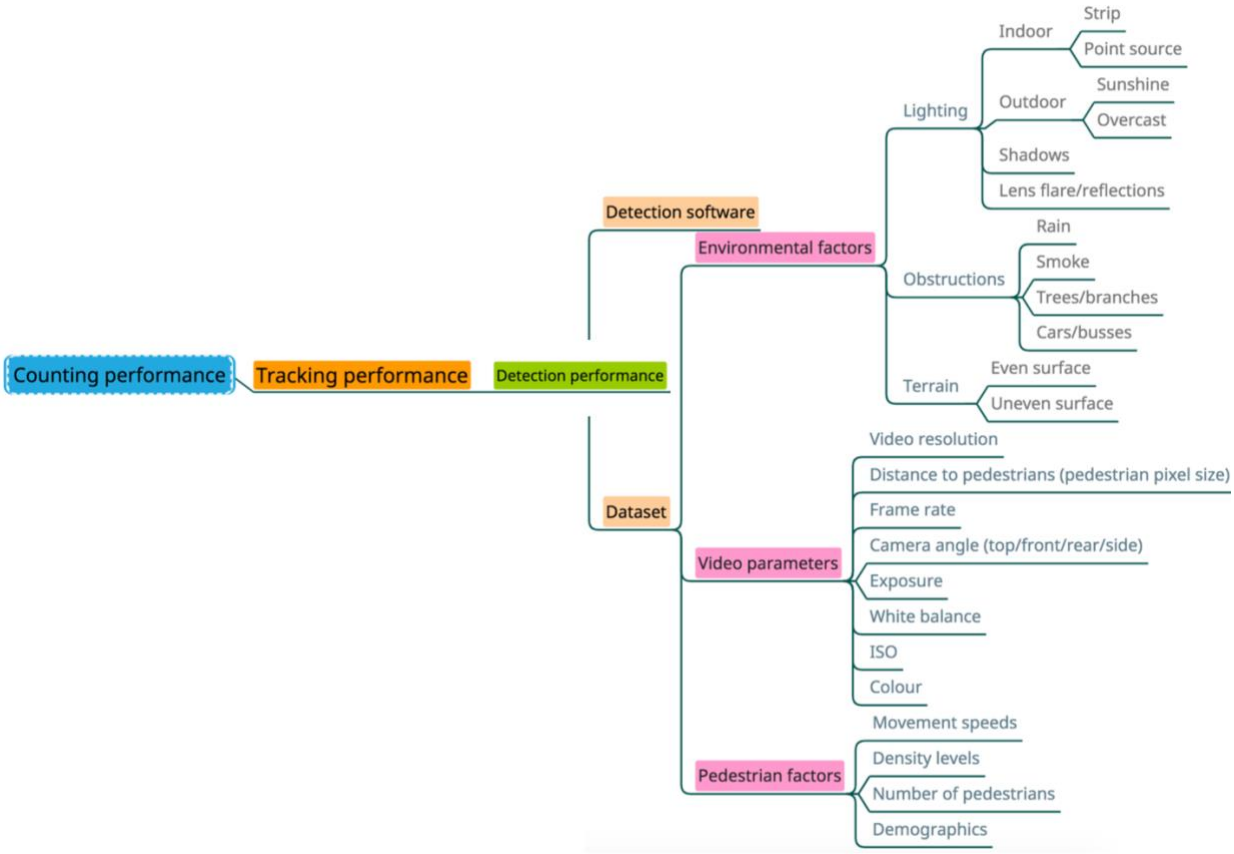


Figure 6 Factors thought to influence the performance in counting, tracking and detecting pedestrians.

Detection errors can be categorized into false positives or false negatives. False positive being the error of displaying a detected pedestrian, where there really is none. False negative is the absence of detection where there really is a pedestrian. Table 6 adapted from (Ujwal & Brémond, 2016) illustrates the concept of false and true detection. If the actual scene contains a pedestrian or not is referred to as "Ground Truth".

Table 6 True/False Positives/Negatives. Table adapted from (Ujwal & Brémond, 2016)

		Pedestrian Detector	
		Pedestrian	**Other**
Ground truth	**Pedestrian**	True Positive (TP)	False Negative (FN)
	Other	False Positive (FP)	True Negative (TN)

Furthermore, in their report the concept of MR (Miss Rate) is explained, which is considered one method of rating a detection systems performance. Dividing the number of false negatives by the sum of false negatives and true positives will yield the MR, as seen in equation (1)

$$Miss\ rate = \frac{False\ Negatives}{True\ Positives + False\ Negatives} \quad (1)$$

One other criterion for evaluating a system is the FPPI (False Positive Per Image). Dividing the total number of false positives by the total number of images analysed will yield the FPPI, as seen in equation (2).

$$FPPI = \frac{\sum False\ positives}{Number\ of\ images} \quad (2)$$

Accuracy is defined as 100 minus the MR and the FPPI, as seen in equation (3). In the literature review, a 100% detection accuracy for any pedestrian detection system has not been found.

$$Accuracy\ (\%) = 100 - MR(\%) - FPPI\ (\%) \quad (3)$$

4.3. Sensors

The use of sensors as a means of pedestrian detection is widely used, relatively inexpensive as well as offering a more anonymous alternative than for instance standard monocular video capture (Kurkcu & Ozbay, 2017). Sensors range from those utilizing infrared beams (electromagnetic radiation, non-visible to the human eye) to ToF (time of flight) systems with LiDAR (light detection and ranging). The sensor detection heavily relies on the use of algorithms to interpret the output data and to classify it as pedestrians. Algorithms are described in section 4.6 Algorithms.

A student thesis from Lund University, (Lundh & Pettersson, 2017) used Microsoft Kinect (infrared beams and depth sensor used for controlling an XBOX) in order to detect people throughout a building. Their objective was to be able to obtain information about people's movements and whereabouts during an egress situation using the Kinect cameras in a frontal view orientation. The information could then be used by rescue services as a way to locate people or fed into simulation programs enabling egress alternatives to be assessed, potentially with the results informing a dynamic way-finding system, all in real-time. The results showed a lack of range in the detection as well as a limited capacity to track more than six people at a time, which would pose a problem when trying to have full coverage of the building or tracking pedestrians in high density situations. However, work by (Seer, Brändle, & Ratti, 2014) displayed a significantly high detection accuracy using Kinect sensors. They managed to achieve a 94% accuracy and a 4 cm tracking precision on their dataset containing 2674 pedestrians by using overhead mounted sensors. Furthermore, (Corbetta, Meeusen, Lee, & Toschi, 2016) displayed good results when using Kinect sensors facing down in the Eindhoven train station. Using the 3D-depth sensor they were able to detect and track several millions of pedestrian trajectories. The range limitation is still a relevant issue for the infrared sensors, whereby a lot of sensors would be needed to cover an entire building. Figure 7 below shows the output signatures captured by the depth sensor as well as the setup used in the train station.

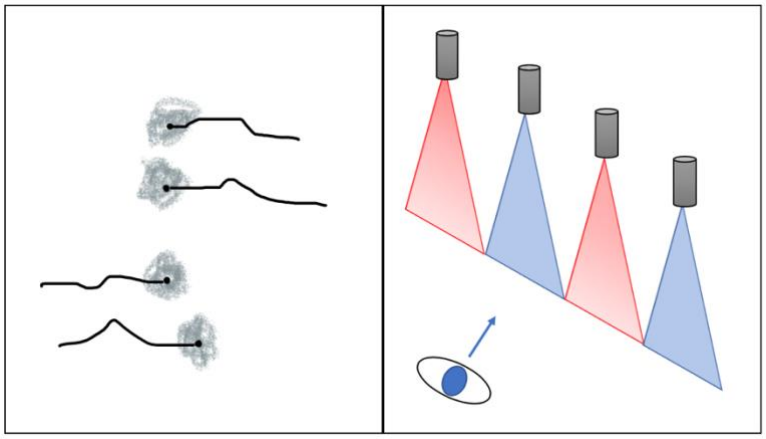


Figure 7 (Left) Output from depth sensor with projected trajectories. Right displays the setup used at Eindhoven train station. All images are adapted from (Corbetta et al., 2016)

ToF (time of flight) uses the time it takes to reflect back light from an object to the emitter. Combining LiDAR sensors and a ToF-system, (Ogawa, Sakai, Suzuki, Takagi, & Morikawa, 2011) examined the pedestrian detection possibilities from an in-vehicle LiDAR sensor. Their work showed great potential in using the sensor in combination with classification algorithms. The accuracy was around 80% even at long distances up to 70 meters. Furthermore, rain or heavy particles in the air impacted the emitted light resulting in false detections.

4.4. Deep neural networks

The concept of neural networks first emerged with (Hopfield, 1982) and has in the last decade ended up being used in the state-of-the-art computer vision and image recognition systems, albeit in other various forms of the original structure. It is considered to be a branch of artificial intelligence. The word “neural” stems from the structure being similar to the brain’s neurons and its interconnected structure. To say that a network is deep is simply stating it has more layers and, in a sense, more connected “neurons”. To explain the use of deep neural networks in computer vision, the concept of having a computer detect a handwritten number and classify it a number is used. Figure 8 below shows a picture of a handwritten “9” (left). The picture in this case is 28 by 28 pixels (in total 784 pixels) and each pixel has a value indicating the brightness. To the right each pixel is assigned to a specific neuron that hold this brightness value. The next layer (column of neurons) is meant to detect certain features or patterns that occur within the image, e.g. the “9” is composed of different sub-parts illustrated by the coloured lines. The next layer combines certain elements that make up the “9”, such as a circle on top of a straight line. In the end the pictures distinct features will lead the programs algorithms to identify (probabilistically) that this picture displays the number 9.

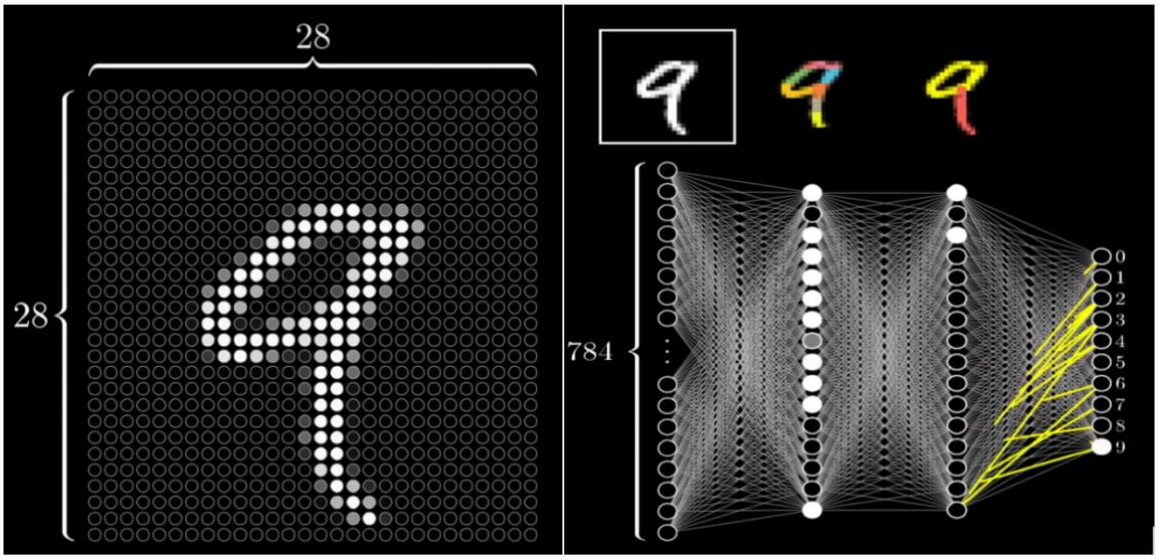


Figure 8 (Left) Picture of a handwritten "9" displayed as 784 individual pixels. (Right): The layout of the neural network used to classify the image as the number 9. Images taken from (But what is a Neural Network?)

One key point to be made is that the system needs to know that these features correspond to a number or object etc. That is where “training” of the systems come in place. This will be covered in the section 4.5 Machine learning.

The distinct features of other more complex items, such as humans, cars, animals etc. would in turn need far more layers and filters in order to be fully classified. The term convolutional neural networks (CNNs) described in (Schmidhuber, 2015), simply is a deep neural network but with many more layers. In the report “Visualizing and Understanding Convolutional Networks” by (Zeiler & Fergus, 2014), their system has detected distinct features from different pictures. Figure 9 illustrates how some of these convolutional layers and filters detect and display important features.

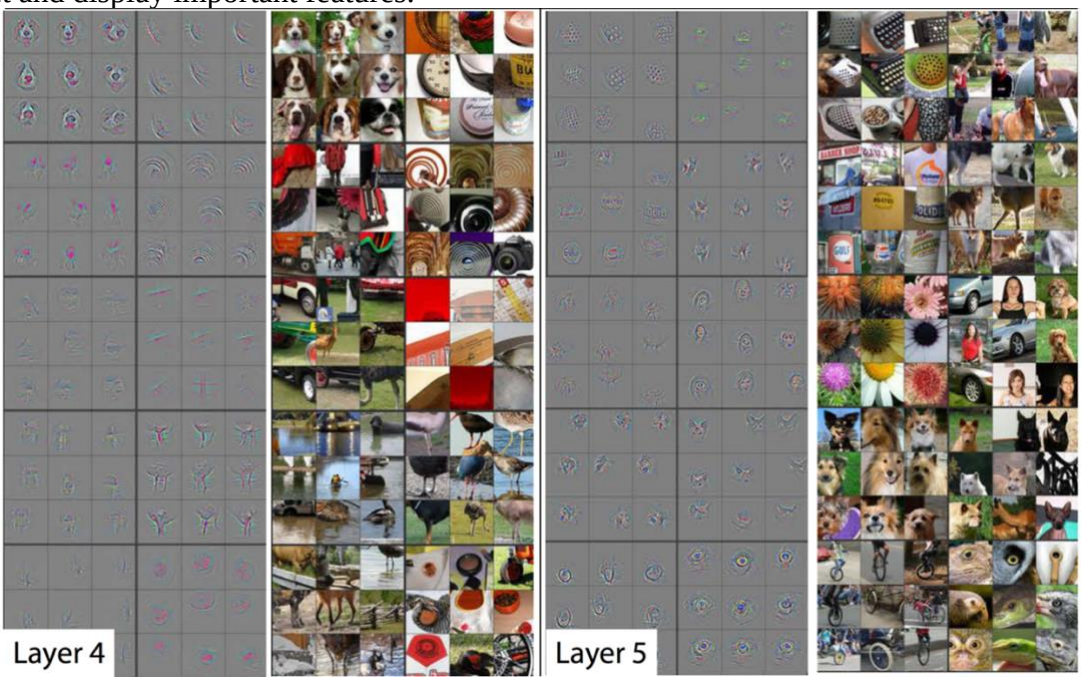


Figure 9 Features detected in different layers in a CNN. Image taken from (Zeiler & Fergus, 2014)

In recent years, several research efforts have been conducted on deep learning utilized for object- and pedestrian detection. This includes the use of thermal, multispectral imagery combined with regular imagery by (Guan, Cao, Yang, Cao, & Yang, 2018) managed to boost the overall performance in their pedestrian detection model (based on CNN). Figure 10 below illustrates detection on both the image inputs.

Figure 10 Regular monocular imagery and multispectral images shows detected pedestrians. Image from (Guan et al., 2018)

Research done by (Tomè et al., 2016) proposed CNN-solutions for pedestrian detection. Their system was later tested on a computer thought to be the forerunner of self-driving cars in the future. The figure below is displaying the segmentation of an input image and the different features extracted at the layers. The feature detection ranges from edge detection to more abstract and complex patterns.

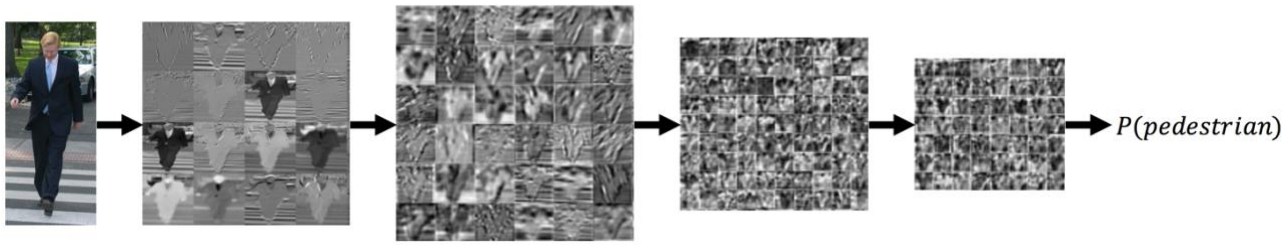


Figure 11 Pedestrian detection. Feature extraction from the layers in the CNN. Image from (Tomè et al., 2016).

More research on the subject of deep neural networks and pedestrian detection include (Bojarski et al., 2017; Brunetti, Buongiorno, Trotta, & Bevilacqua, 2018; Gouda & Nayak, 2015; Lavi, Serj, & Ullah, 2018) to mention some.

4.5. Machine learning

The introduction of the term can be tracked back to the late 1950s (Samuel, 1959) , and later on established by (Koza, Bennett, Andre, & Keane, 1996) with the working question “How can computers learn to solve problems without being explicitly programmed?”.

The question is nowadays answered by means of algorithms, by constructing models using mathematics. The learning part comes from inputting data into the model and having the

algorithms distinguish what is important in the data given the information that has been previously provided. Almost as a synonym, the phrase “pattern recognition” can be used when describing what machine learning is, (Bishop, 2006).

Learning can broadly be separated into a supervised- and an unsupervised form. Supervised meaning that an input image has the desired output already labelled. Unsupervised would be inputting the same image but with no annotation of the desired output, the algorithm itself will look for patterns or features. A good visualisation adapted from (Seebo, 2019) shown in Figure 12 below categorizes the learning process and specifies what sort of algorithm is used to achieve the desired learning (there are more than those listed however). Classification, regression and other algorithms are described more in detail in Section 4.6 Algorithms.

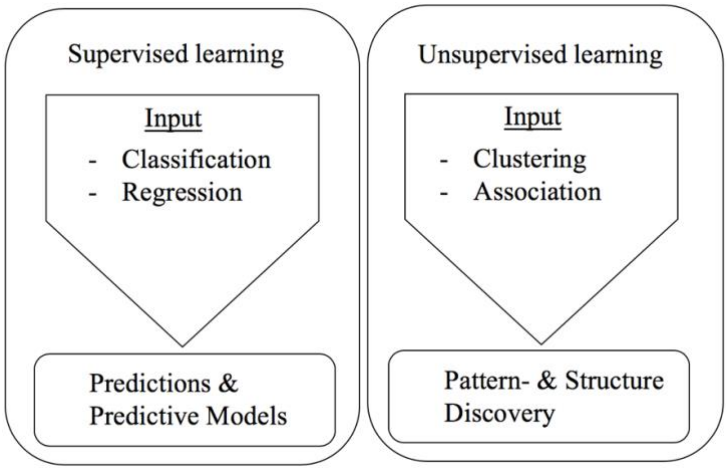


Figure 12 Supervised- and unsupervised machine learning. Image adapted from (Seebo, 2019)

To demonstrate further, Figure 13 below illustrates an example input (in this case a pedestrian). The picture on the left is the input without anything indicating what the image is supposed to represent. If this is fed into an algorithm, perhaps one that specifically tries to find patterns or structures (unsupervised) by utilizing histogram oriented gradient analysis (explained more in Section 4.6 Algorithms), it may output an image similar to the picture in the middle. The picture on the right has already been classified by a human (supervised) to contain these specific structures representing a pedestrian and indicating the location by using a bounding
box.

Figure 13 (Left): Input image showing a pedestrian. (Middle): Pattern recognition. (Right): Pattern and pedestrian labelled as desirable output from input image.

If a system is given enough input data (supervised or unsupervised), it will with time be trained and will theoretically detect a previously unseen image as an image of a pedestrian, given the learned patterns or structures are present within this new image. There are several image datasets made public that have a large subset of images displaying pedestrians. They are manually labelled and annotated and can be used for training a detection system or for benchmarking one. Some of the more prevalent pedestrian datasets are:

- INRIA ('INRIA Person dataset')
- Caltech ('Caltech Pedestrian Detection Benchmark')
- Pascal ('The PASCAL Visual Object Classes')
- Daimler ('Daimler Pedestrian Benchmark Data Set')

4.6. Algorithms

The ways to detect and track pedestrians using algorithms are numerous. The spectrum ranges wide and in (Pulford, 2005) a taxonomy of up to 35 algorithms used for single- and multiple target-tracking is outlined. Recent research (Pathak & Sivraj, 2018) concludes that the most commonly used within pedestrian detection are those utilising feature extractors, classifiers or cluster algorithms.

Two of the most commonly used feature extractors, HOG (Histogram of Oriented Gradients) and LBP (Local Binary Pattern) are explained first, as their output lay the foundation for the computation in the classifiers or cluster algorithms. HOG, first introduced to the field of computer vision by (Dalal & Triggs, 2005) is a method to represent an image as gradients of vectors indicating light intensity difference in cells (sections of pixels). Figure 14 below illustrates an input image and albeit the image is very blurry, a machine can make out the features of the human silhouette from the oriented gradients on the images to the right.

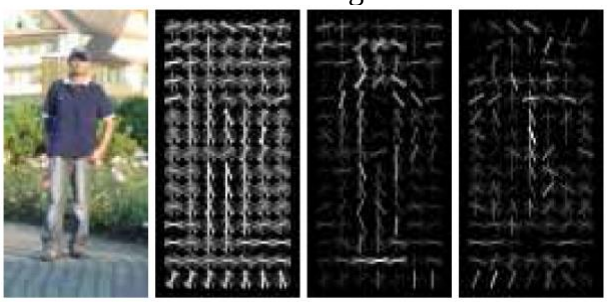


Figure 14 Histogram of oriented gradients. Image taken from (Dalal & Triggs, 2005).

Texture based detection, also known as LBP, introduced by (Ojala, Pietikäinen, & Harwood, 1994), utilizes pixel brightness (values 0-255) to convert a 3 by 3 matrix into a binary subset through a series of thresholding operations. Figure 15 below shows the procedure and a final result of the histogram generated by the LBP extractor.

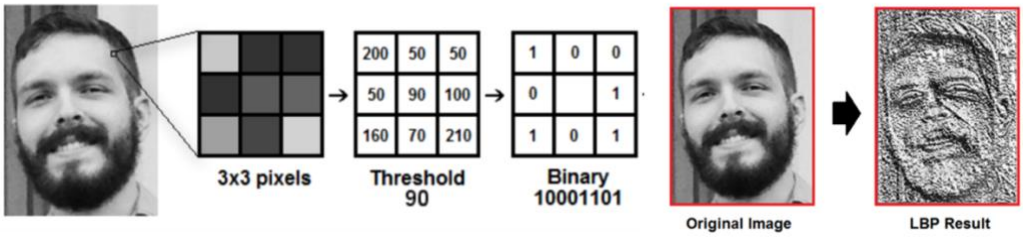


Figure 15 LBP procedure on an input image. Image taken from (Prado, 2018).

Taking the output histograms from either feature extractor and applying classifiers or cluster algorithms, will allow the detection system to recognise certain features corresponding to a pedestrian. This is the essence of machine learning.

Classification algorithms, e.g. the SVM (Support Vector Machine) are a method to separate data into two distinct subsets divided by a hyperplane. (Pathak et al., 2018) state that the success and wide spread use of SVMs has to do with its good performance and quick computational speed. The concept of image classification within pedestrian detection is in a sense a way of saying either: “Yes, the image (extracted features) corresponds to a pedestrian” or “No, the image does not correspond to a pedestrian”. Furthermore (Pathak et al., 2018) highlights AdaBoost (Adaptive boosting) as an algorithm highly used for large scale pedestrian detectors. It combines several weak classifiers into a strong one making estimates and weighing in the confidence of each classifier. It is more suitable for training with large amounts of data. The classification schematics seen in Figure 16 below can be used to, in a simple manner describe the distinguishing process of supervised machine learning. If the data is above the red line, the machine classifies the data as a triangle, if below, it is a circle.

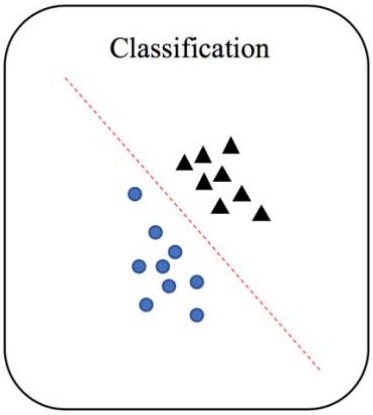


Figure 16 Classification concept. Image adapted from (Seebo, 2019).

PCA (Principal Component Analysis), an algorithm used for unsupervised machine learning stems from the concept of clustering seen in Figure 17. PCA was used by (Mehralian & Palhang, 2013) as a way to take advantage of a clustered output, in their case HOG histograms from different cells throughout an image and then only apply classifiers on a section of the cells containing gradients of interest.

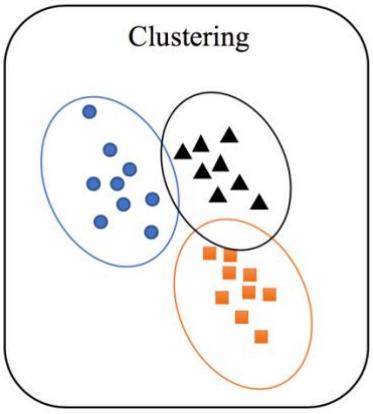


Figure 17 Clustering concept. Image adapted from (Seebo, 2019).

5 Case Study

The testing of the Flowity software was completed on a video of pedestrians entering a large Lund University building (entrance of *V-huset*). The video was chosen as it included suitable pedestrian movement that fitted the thesis objective and was available to the author. The video contained several hundred people, different levels of occlusion and people on bicycles passing through the frame. Manual analysis was conducted to have data a benchmark comparison with the software output. The analysis focused on the number of unique pedestrians present, walking speeds, flows at certain sections, people density and route choices. Figure 18 below illustrates a top-down view (left) of the entrance area and a screenshot of the video capture (right).

Figure 18 (Left) Google Maps top-down view of the main entrance and screenshot from video capture (right).

The video was 12 minutes and 44 seconds long and filmed with a resolution of 1920 x 1080 pixels at 24 frames per second. The time of the recording (8 am) was an overcast day with only natural light present. The camera, in this case a GoPro Hero 3 was situated 4 meters above the walking area and angled roughly 21º degrees down onto the walking area. Further dataset details can be seen in Table 7 below.

Table 7 Video dataset details.

Filename	***Description***	***Length***	***File size***	***Video resolution***	***Bitrate and frame rate***	***AOV/FOV (angle of view/field of view)***	***Lighting conditions***
GOPR1949.mp4	Pedestrians walking in and out of the main entrance in the morning starting at 08:00. The camera was located **4m** above the walking area	12 min, 44 seconds	2,88 GB	1080p	30mbit/s 24fps	H.FOV 64,4 º V.FOV 37,2 º Diag.FOV 73,6 º	Natural light during an overcast day

5.1. Manually collected pedestrian information

The video was manually analysed to obtain data that later could be compared to data that the software managed to produce. The video in its entirety was watched and analysed two times and the number of unique persons present within the video was counted. In total 576 unique persons were identified over the course of the 12 minutes and 44 seconds. A smaller area was more closely studied and can be seen in the Figure 19 below. This area contained no people on bicycles (except one person on an electric scooter) and there was a total of 316 unique pedestrians counted. Additional information on the measurements taken and the collection of pedestrian information in this section can be found in Appendix A.

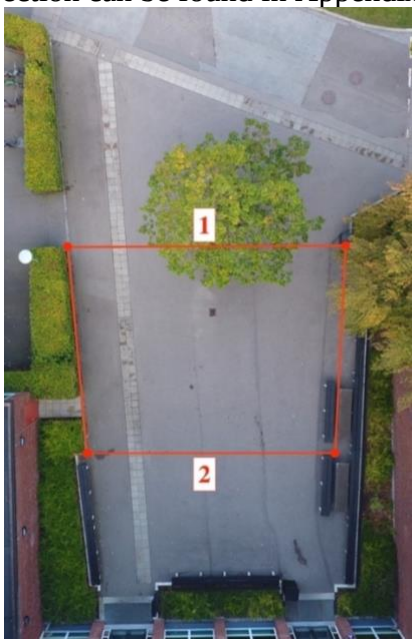


Figure 19 The red box represents a smaller area studied.

Manually counting the number of pedestrians using the two different entrances, A or B (seen in Figure 20) was conducted: total of 224 pedestrians used entrance A and 92 used entrance B during the course of the video.

Figure 20 V-huset, Entrance A and B

To obtain the walking speed of all the pedestrians within the area, all the pedestrians were visually tracked (individually) and their transit time (time between crossing line 1 and 2 in Figure 19) was noted. Measuring the distance walked by each pedestrian was not possible due to the immense task of manually extracting them. Therefore, watching 20 different pedestrians walk across the small area and then manually drawing their movement pattern onto a map over the area was instead completed. This was accomplished by using reference points in the video and transferring the travel path for each pedestrian onto the separate top-down view. Moreover, the paths were segmented and measured. For instance, the screenshot frame below (Figure 21), containing six pedestrians was represented by the drawing in Figure 22.

Figure 21 Screenshot of six pedestrians walking across the small area.

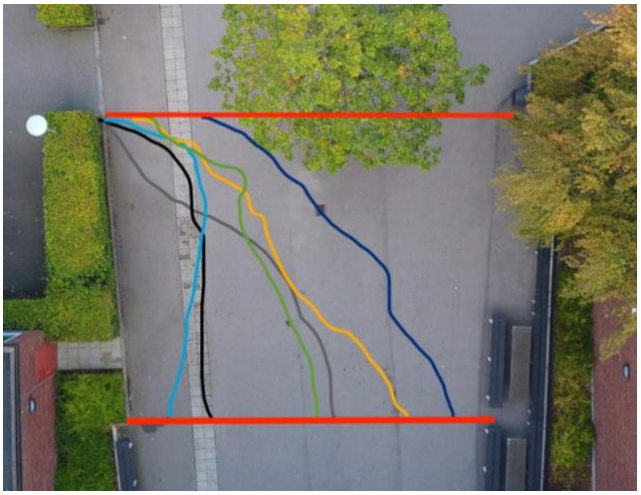


Figure 22 Six pedestrians travel paths manually drawn onto a map of the small area.

The cameras AOV (angle of view) and placement distorts the image which will have an impact on the accuracy of the manual measurements. This was addressed by using known reference points on the ground and using the on-site measurements of the area. The 20 pedestrians were individually analysed and, combined with the reference points, on-site measurements and the manually drawn travel paths led to an average distance walked between line 1 and 2 to be calculated. This distance was 10,25 meters. The uncertainty of the manual measurements was reduced by measuring everything twice. The two different measurements resulted in the uncertainty estimated to be in the range of 40-60 centimeters.

The average of 10,25 meter was then used for the all the 316 individually counted pedestrians. Table 8 summarizes these 20 pedestrians and the walking distance.

Table 8 Min, max, median, average and standard deviation for 20 manually measured travel distances.

	Min	*Max*	*Median*	*Average*	*Standard deviation*
Manual	7,44	14,13	10,15	10,25	1,61

The transit time for each pedestrian was divided by the average distance walked (10,25m) and in Figure 23, all the 316 individually counted pedestrians walking speeds can be seen. The average walking speed was 1,67 meters per second.

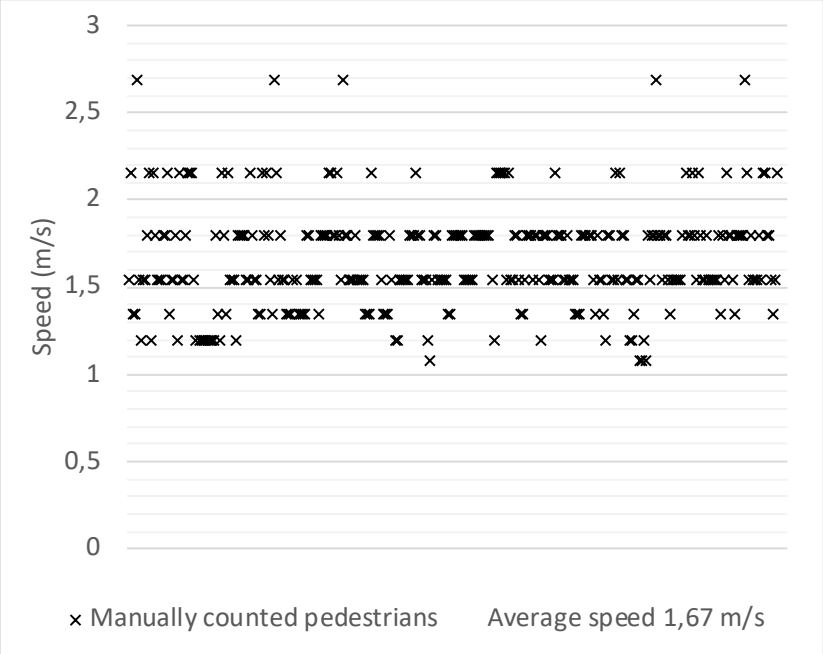


Figure 23 Manually counted pedestrian walking speeds (m/s) within the small area.

In Figure 24 below the speeds of all the pedestrians passing through the area at a 10 second interval is presented. The choice of a ten seconds interval had to do with the fact that most pedestrians passed through the area during the 10 seconds and that the total number of time intervals would be limited to 76, which was considered manageable for the counting of flows and densities. An average of all the pedestrians during the 10 second interval is presented.

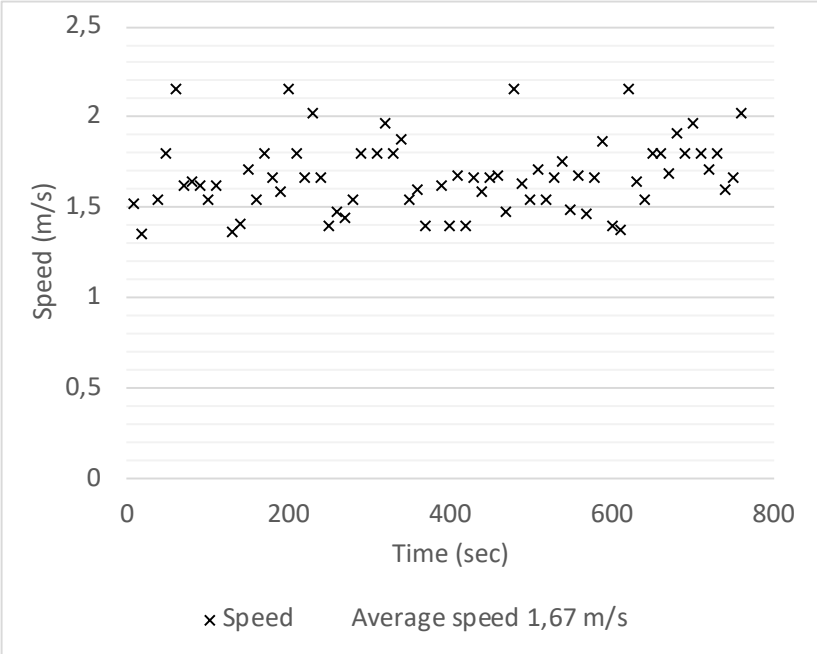


Figure 24 Manually counted pedestrian walking speeds within the small area (m/s) considering a 10 second interval.

The number of pedestrians crossing line 2 during a 10 second interval was counted and noted. That number was divided by the length of line 2 (12,13 meters) and the time it took (in minutes). Figure 25 below presents the flow at line 2. The average flow was 2,09 persons per minute and meter.

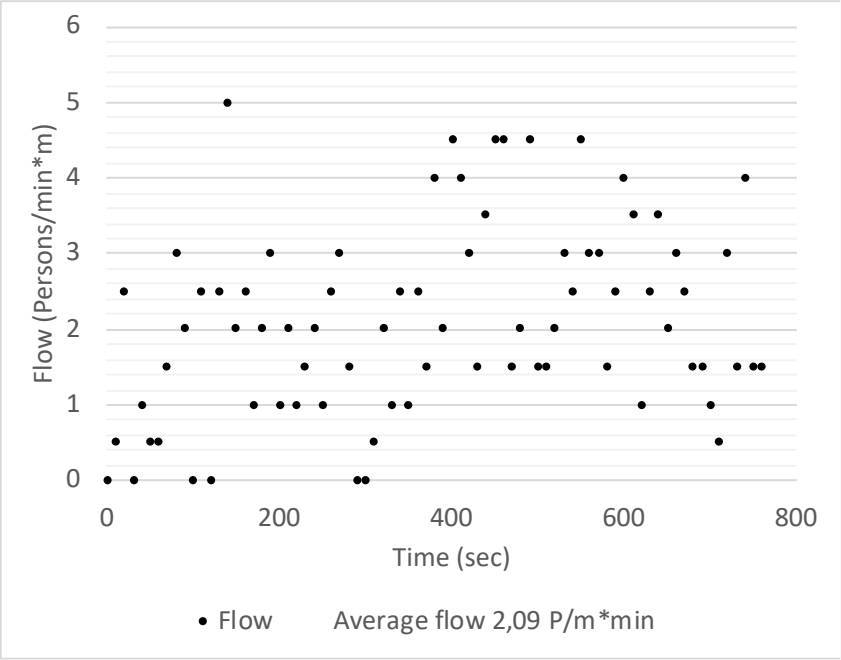


*Figure 25 Manually counted flow at line 2 (persons/min*m) within the small area considering a 10 second interval.*

The global people density, i.e. number of persons per square meter for an area was calculated by counting the number of pedestrians present within the small area at 10 second intervals. The number was divided by the area examined (100,3 m^2). Figure 26 below presents the global people density within the area. The average density was 0,038 persons per square meter.

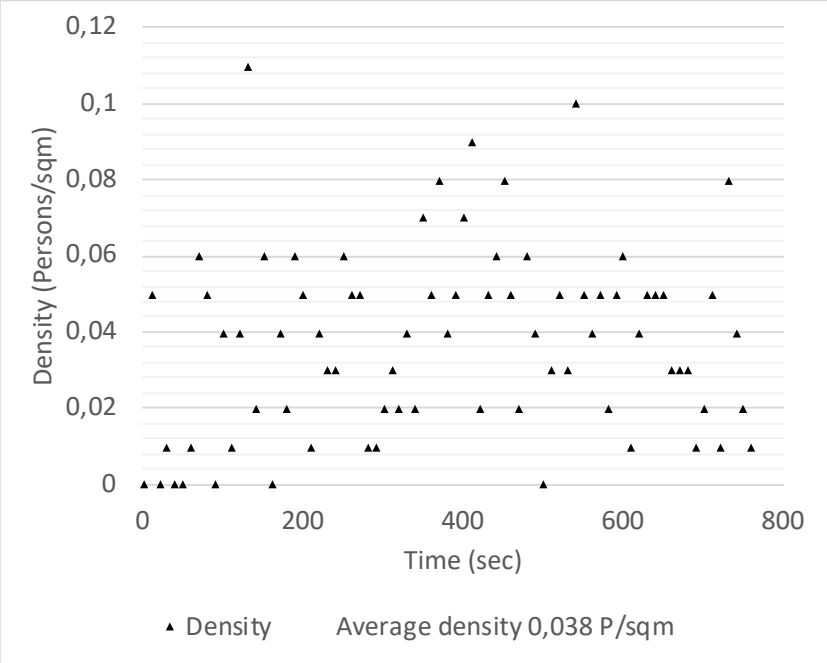


Figure 26 Manually counted global people density (persons/m2) within the small area considering a 10 second interval.

Local people density, which relates to having a direct impact on people's movement, was not part of this particular analysis, due to the difficulty of manually measuring local density on a larger scale and to include all the pedestrians at all times. However, an estimate of the local density at on particular point in the video was made. This was considered the visually most dense area and doing a manual measurement of the area shown in Figure 27 below on site resulted in a local density of 2 persons per square meter (4 persons in an area of 2x1 square meters). The video had played for 124 seconds when this occurred.

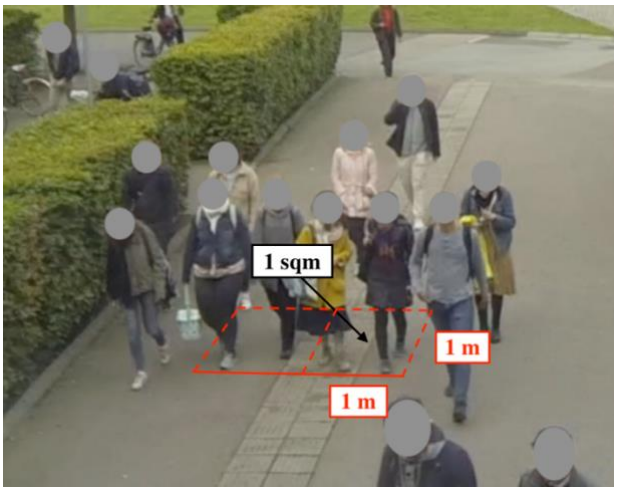


Figure 27 Estimate of maximum local density.

6 Results

This section will describe the outputs that the software provided as well as the results generated from the output files, detection accuracy and measured pedestrian factors.

The dialog and creative process with the Flowity development team resulted in the following factors, not being possible to incorporate in the system, given the current state of the software: age, gender, body size, movement impairment or people being incapacitated.

6.1. Video render

The software automatically generated a rendered video with annotated detections using different coloured bounding boxes. The Flowity software automatically identified pedestrians and the detected pedestrians were assigned a unique ID and colour. Figure 28 below is a screenshot from the render. The lines (dark blue, light blue, orange and purple) seen in the lower end of the image are manually placed by the developer at Flowity during the setup of the software. They are programmed into the system using a local coordinate grid. The purpose was to demonstrate a feature which created a digital footprint when a detected pedestrian (their bounding box) passes them from either direction. This is referred to as a "footfall" and has two ways of presenting data, either "A", a pedestrian is passing the line from above/left (in the video) or "B" from below/right. The text in the upper left corner of the image updates the "footfall" and displays the total number of pedestrians who have passed in either direction. In this particular render, the lines were only placed at random by the developer with no specific intent behind it. Therefore, no further analysis of the feature was completed. People have been anonymised using grey circles (added manually by the author of the report).

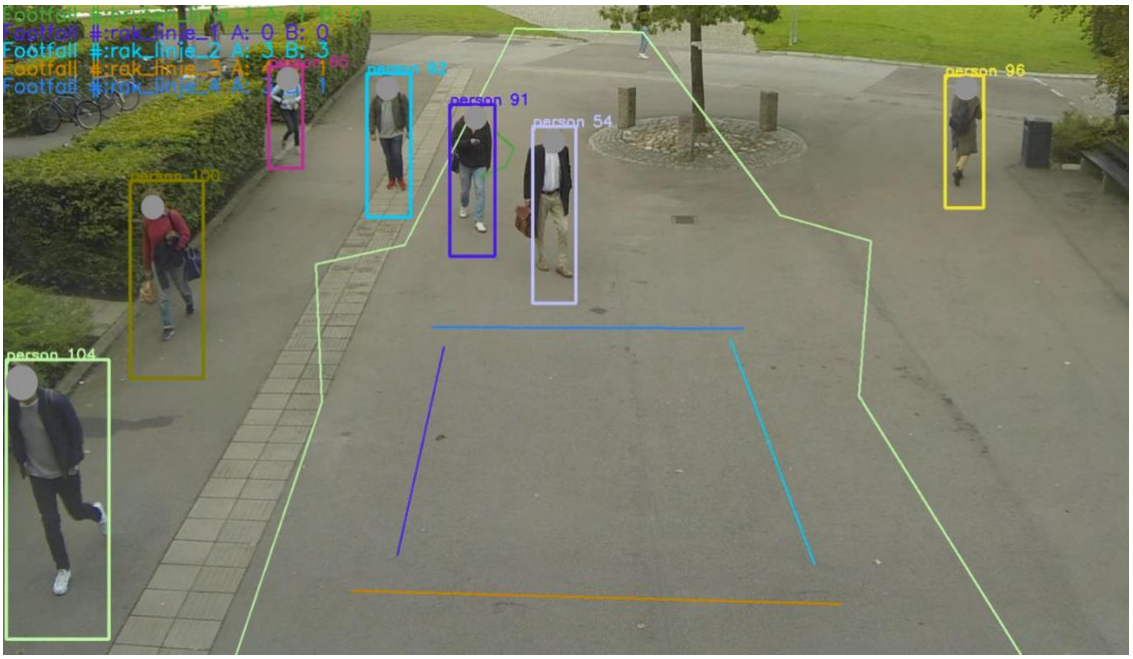


Figure 28 Screenshot from the Flowity video render with detected pedestrians.

6.2. Data statistics

The other kind of output that the Flowity software was capable of generating was a .json file containing a *"timestamp*" generated every 42ms, followed by the number of milliseconds and seconds that has passed since the start of the video. The interval of 42ms corresponds to the frame rate of the video, i.e. 24 frames per second. A short section can be seen in Figure 29 below. *"Gps_detections"* depicts the ID of the detected pedestrian and its latitude and longitude in the format *["ID", "latitude", "longitude"*] and then continues to write lines if more pedestrians are present during that timestamp. "*Detections*" indicate if a pedestrian has been identified in the given frame and outputs int the format *["x-position in video", "y-position in video", "pixel-width of bounding box", "pixel-height of bounding box", "ID"*] and then continues to write lines if there are more pedestrians present during that timestamp.

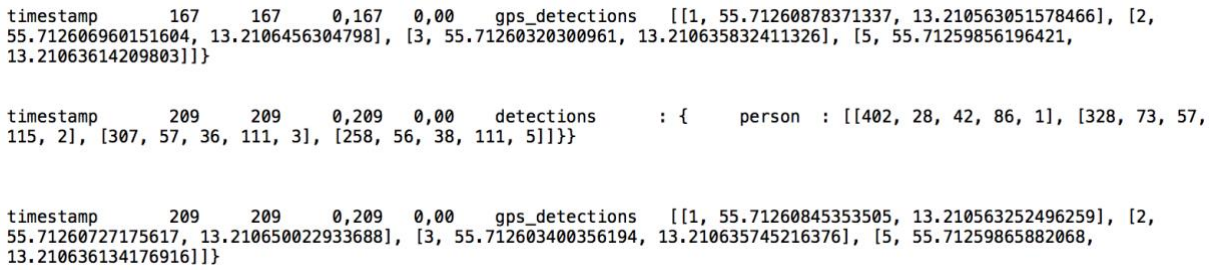


Figure 29 Example section from Flowity output data in .json file format

A conversion to Microsoft Excels .xlsx-format was completed which required substantial amounts of manual clean-up of the rows and columns, e.g. converting text to data and removing unwanted symbols. The end result was a dataset of 284986 rows with seven columns: [indication that a timestamp was done], [milliseconds since start of video], [same time but in seconds], [indication that there was a GPS detection], [detected pedestrians unique ID], [latitude of detected pedestrian], [longitude of detected pedestrian]. A short section can be seen in Table 9.

Table 9 Example section of the data set containing all the detected pedestrians.

Timestamp	***Time (ms)***	***Time (sec)***	***gps_detections***	***ID***	***Latitude***	***Longitude***
timestamp	125	0,125	gps_detections	2	55,7126066	13,2106426
timestamp	125	0,125	gps_detections	3	55,7126026	13,2106378
timestamp	125	0,125	gps_detections	1	55,7126091	13,2105629
timestamp	167	0,167	gps_detections	2	55,712607	13,2106456
timestamp	167	0,167	gps_detections	5	55,7125986	13,2106361
timestamp	167	0,167	gps_detections	3	55,7126032	13,2106358
timestamp	167	0,167	gps_detections	1	55,7126088	13,2105631

6.3. Detection performance

The video analysed was in total 760 seconds long which amounts to a total of 18240 individual frames (24 frames per second) where the software should detect pedestrians. To manually verify detection performance, a limit of 500 randomly selected frames was set in order to keep the workload to a reasonable. Those 500 frames contained 3010 possible pedestrian detections, which were all manually noted. The number of true positives, i.e. where the software correctly identified an actual pedestrian was 2236. The number where the software failed to detect present pedestrians (the false negatives) was 774. The software also detected 21 miscellaneous items as pedestrians, so called false positives. Figure 30 below illustrates the process for a single frame. The top image displays 10 different pedestrians (various degrees of occlusion) and the bottom one shows the software output for the same frame. In the output image, 7 true positives (indicated by the coloured bounding boxes), 3 false negatives (red annotated arrows) and one false positive (yellow arrow) can be seen.

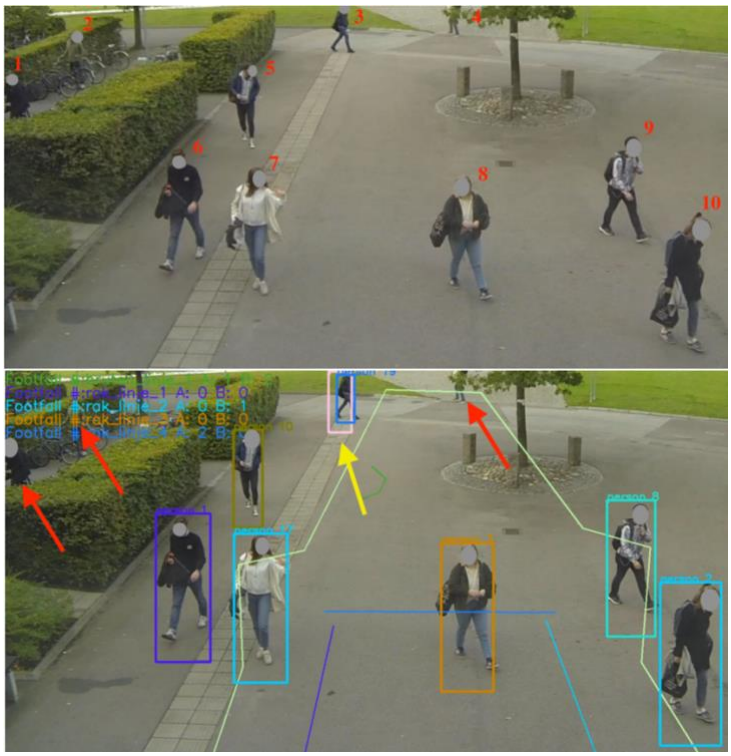


Figure 30 Detection evaluation. (Top image): Original video frame. (Bottom): Flowity-frame with arrow-annotations.

Table 10 below presents a summary of the 500 frames along with miss rates, false positives per image and detection accuracy for the system.

Table 10 Flowity detection performance

Flowity performance	
Total number of TP	2236
Total number of FN	774
Total number of FP	21
FPPI in %	4,2
MR in %	25,71
Accuracy in %	70,09

6.4. Software collected pedestrian information

The pedestrian GPS-data collected was analysed and could reveal that, during the course of the 760 second video a total of 513 unique pedestrians were detected. When only looking at a sub section of the video, in this case the same area as described in the manual counting, Figure 19, the number of detected pedestrians were 285.

The GPS-coordinates were plotted on a longitude, latitude axis, see Figure 31 below. The plot contains all the detected pedestrians at all the timestamps. The image to the right presents a heat-map of the most frequently occupied square meters. Red indicates the most occupied areas, i.e. more pedestrians passed through this area than those with a yellow or green colour. The heat-map was accomplished by dividing the coordinate-grid into smaller sections using functions in Microsoft Excel, further explained in Appendix C. Having each cell be 1x1 meter in size (with corresponding longitude and latitude) and then counting the number of pedestrians passing through each cell.

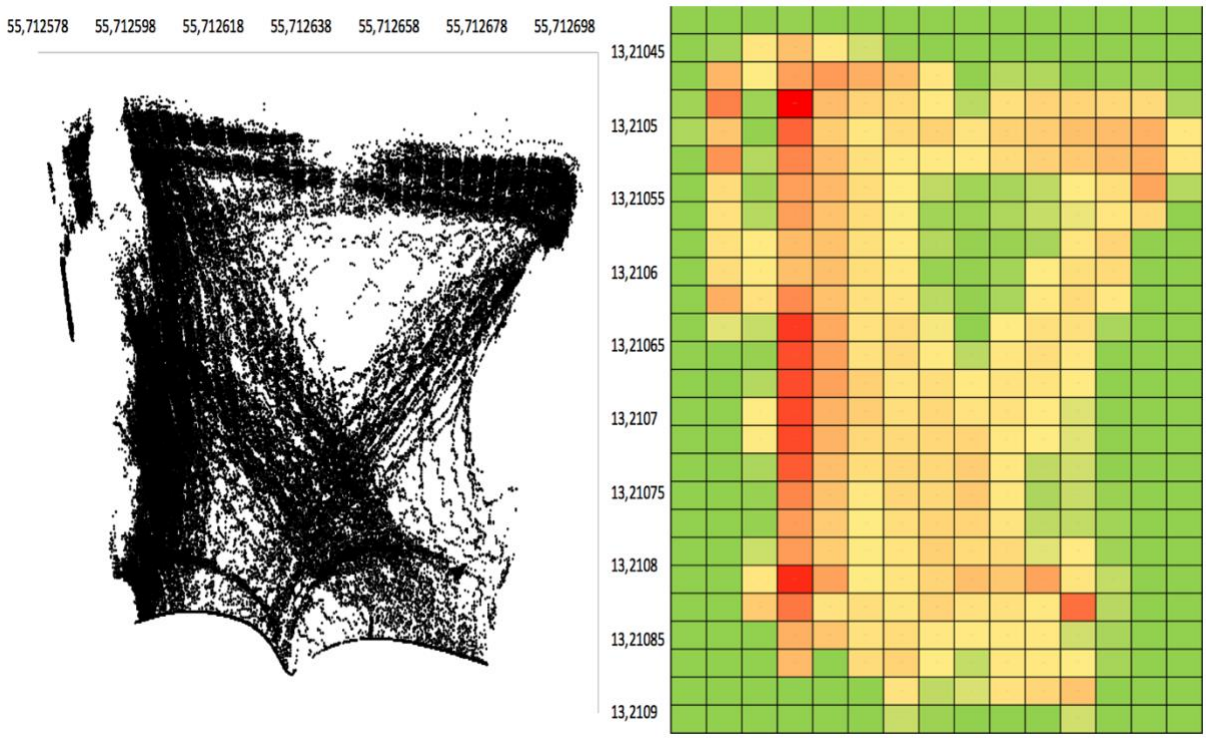


Figure 31 All pedestrian detections plotted on a latitude and longitude axis (Left). Heat-map displaying the most occupied square meters (Right).

Overlaying the plot from Figure 31 onto a map over the area, the result can be seen in Figure 32. The image and plot were not in the same scale and the figure is only meant to highlight the movement patterns features and how they might correlate to the background.

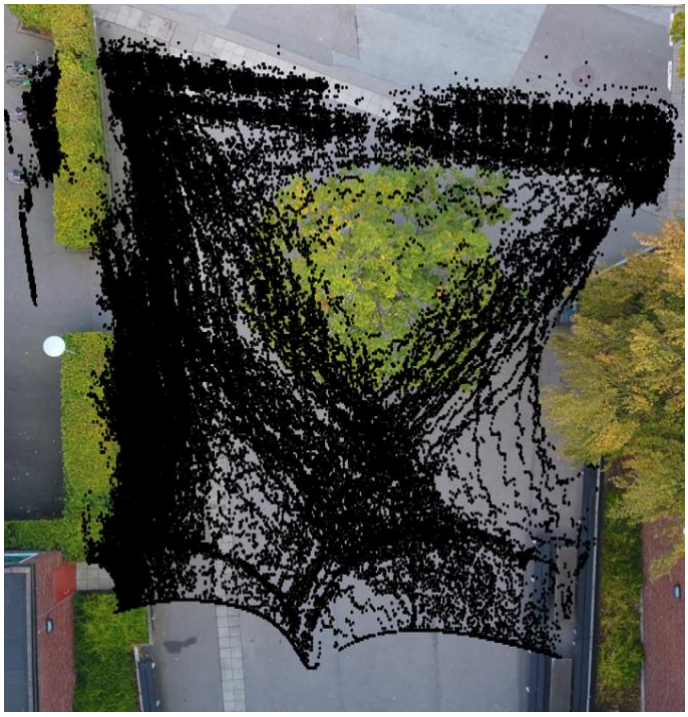


Figure 32 Movement patterns displayed on a background map.

Dividing the entire area into the same as presented in 6.2 manually collected pedestrian information (Figure 19) was done by setting longitude and latitude limits corresponding to line 1 and 2. That resulted in Figure 33 below.

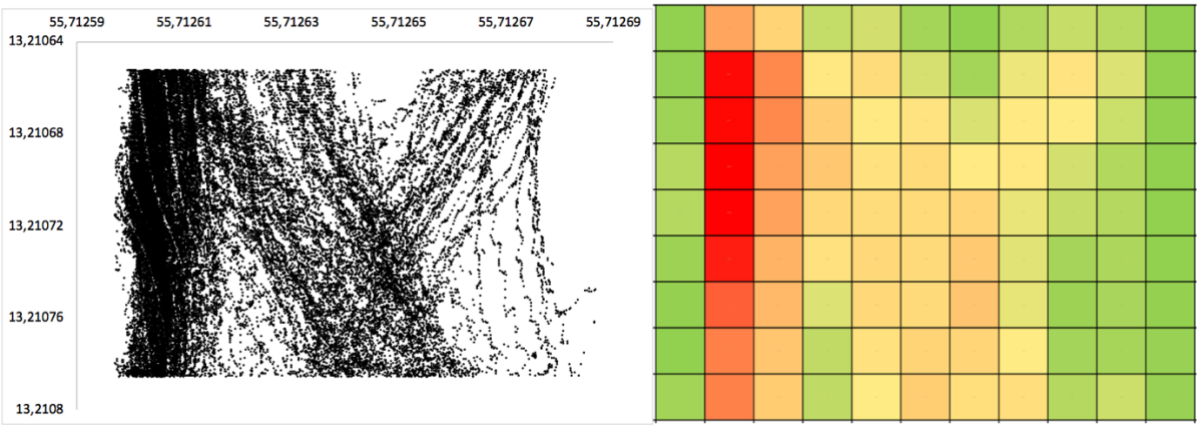


Figure 33 Small Area: All pedestrian detections plotted on a latitude and longitude axis (Left). Heat-map displaying the most occupied square meters (Right).

To obtain the pedestrian walking speeds within the small area, a similar approach to the manual counting was done, distance walked was divided by the transit time. Isolating unique pedestrians and determining the distance covered between line 1 and 2 was done by utilizing functions from Microsoft Excel and a coordinate distance formula called the Haversine formula. Further details on isolating unique pedestrians and the Haversine formula can be found in Appendix B. Dividing the distance covered by each pedestrian by their transit time

gave the walking speed in m/s and the 285 detected pedestrians can be seen in Figure 34. The average walking speed was 2,22 meters per second.

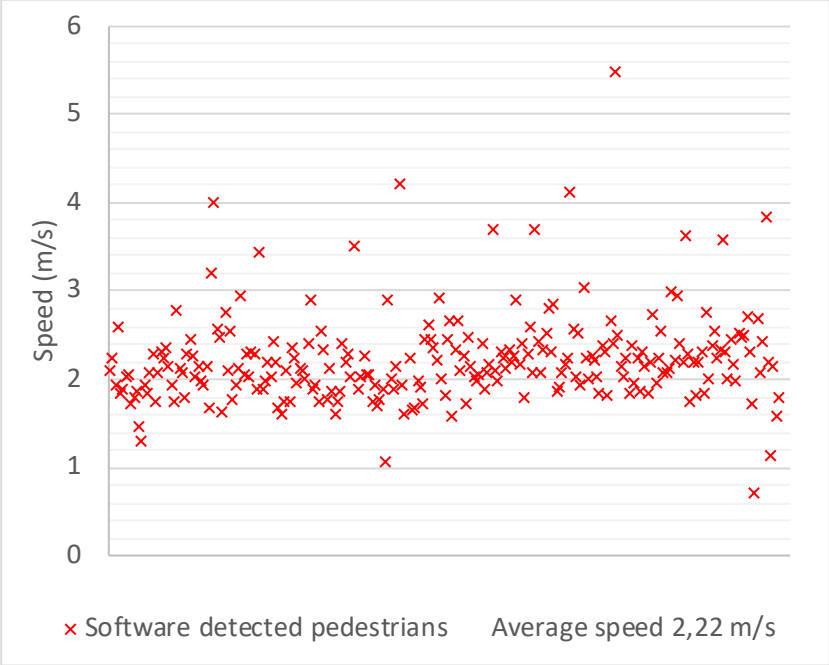


Figure 34 Software counted pedestrian walking speeds (m/s) within the small area.

An average of all the pedestrians during a 10 second interval is presented in the Figure 35 below.

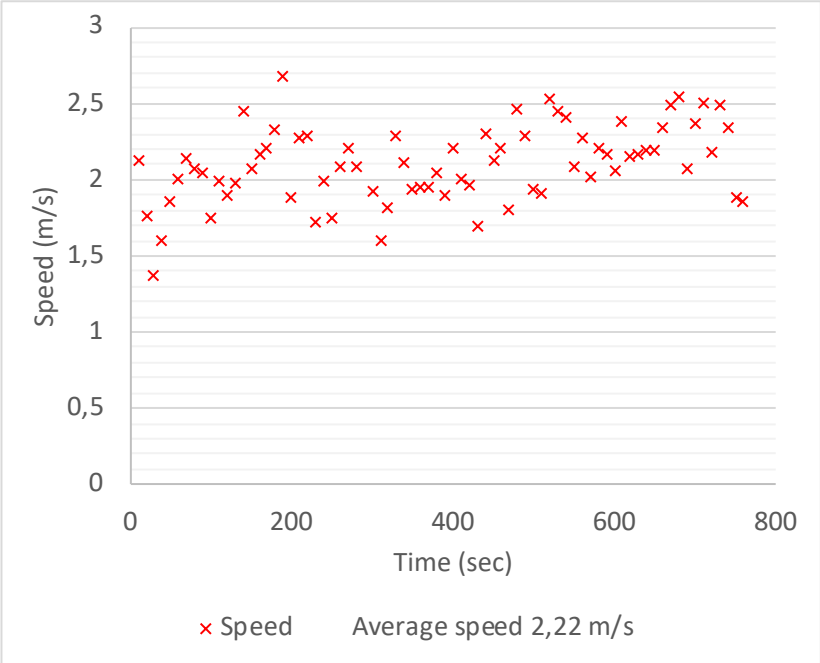


Figure 35 Software counted pedestrian walking speeds (m/s) within the small area considering a 10 second interval.

Similarly, to the manual count, the number of unique pedestrians crossing line 2 during a 10 second interval was sought after. Utilizing Microsoft Excel to isolate the number of pedestrians passing through a timestamp interval as well as a specific coordinate section (corresponding to that of line 2). The number of pedestrians was divided by the length of line

(12,13 meters) and the time it took (in minutes). Figure 36 below presents the flow at line 2. The average flow was 1,83 persons per minute and meter.

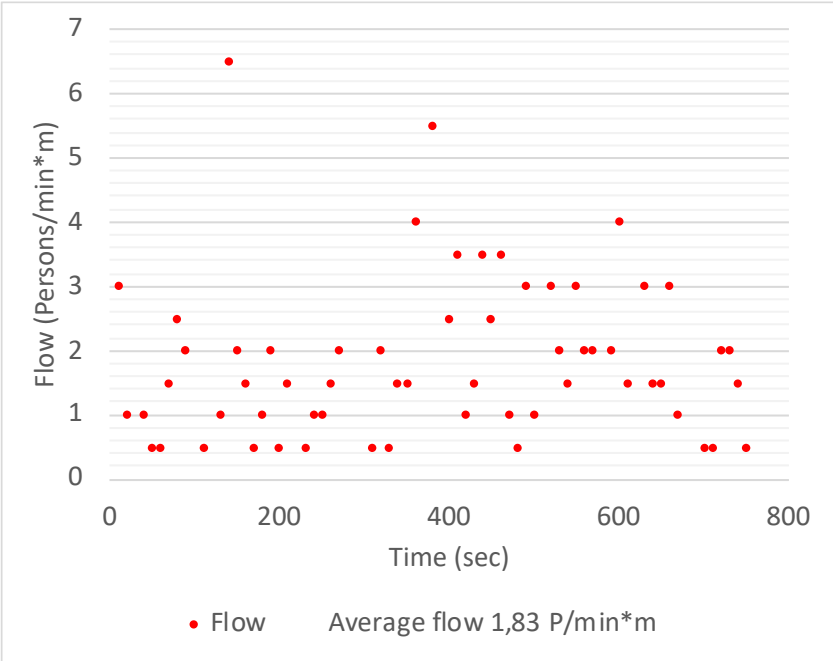


*Figure 36 Software counted flow at line 2 (persons/min*m) within the small area considering a 10 second interval.*

Similarly, to the manual count, the global people density was calculated by counting the number of pedestrians present within the area at 10 second intervals. By counting the number of unique pedestrians present within the area at different time intervals and divided by the red box area (100,3 m2) the people density could be counted. Figure 37 below presents the global people density within the area. The average density was 0,038 persons per square meter.

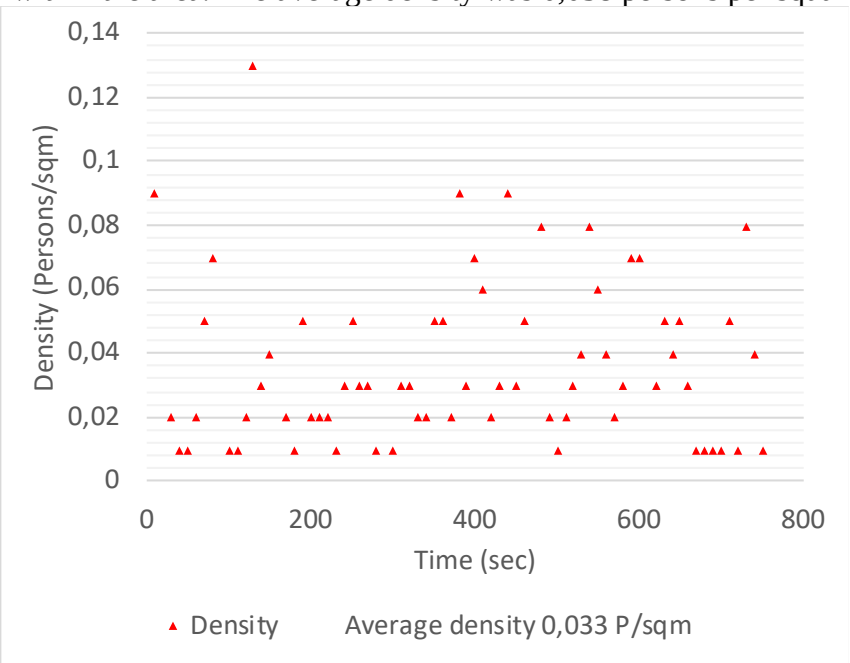


Figure 37 Software counted global people density (persons/m2) within the small area considering a 10 second interval.

Since the manual analysis only looked at the largest local people density that could be found (2 persons per square meter), the software would have to look for the largest local density present, to be able to compare to. Looking at the heat-map in Figure 33 and then count the number of unique pedestrians present within the 1 square meter cells at the same time, the maximum occurring local density could be produced. Figure 38 show the cell where the highest local density, 3 persons per square meter was achieved. The timestamp showed that this was at 125 seconds since the beginning of the video.

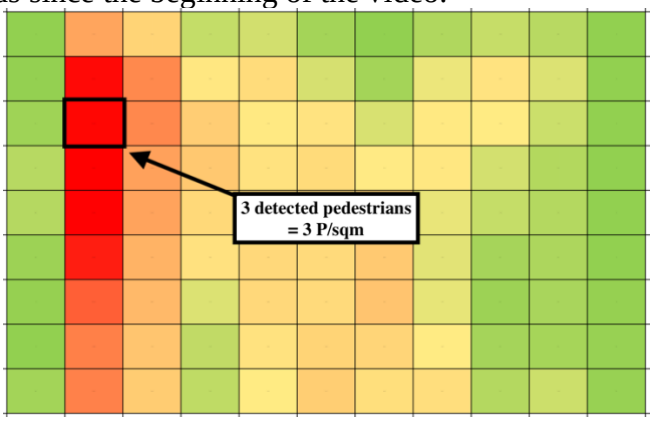


Figure 38 Cell containing the maximum local density measured with the software.

When determining the pedestrian usage of the entrances using software output an approach similar to the trajectory extraction (Figure 31) was used. Counting the number of unique pedestrians passing the area closest to either entrance resulted in a total of 173 pedestrians using entrance A and 112 using entrance B.

6.5. Comparing the manual and software collected data

The example made in Section 5.1 Manually collected pedestrian information, containing the manually collected walking distance for 20 pedestrians was compared to the 20 that the software collected. Analysing the difference in walking distance reveals that the average walkings distance collected by the Flowity software is 24,09 % higher than what was manually counted. A comparison between the walking distance of the 20 pedestrians can be seen in Table 11. It is worth noting that the uncertainty of the manual measurements was estimated to be in the range of 40-60 centimeters, as described in Section 5.1.

Table 11 Comparison between walking distance of the 20 pedestrians measured manually and with Flowity.

	Min	*Max*	*Average*	*Median*	*Standard deviation*
Manual	7,44	14,13	10,25	10,15	1,61
Flowity	9,32	18,03	12,72	12,82	2,13

Comparing the movement speed of the 20 pedestrians manually counted to the ones counted by Flowity, see Figure 39, reaveals a difference in walking speeds. Using a Kolmogorov-Smirnov test (KS-test) was done to see if the distribution of movement speeds is the same regardless of measurement method. The results show that the two data samples have statistically significant different distributions. The KS-test and all the detailed results can be found in Appendix D.

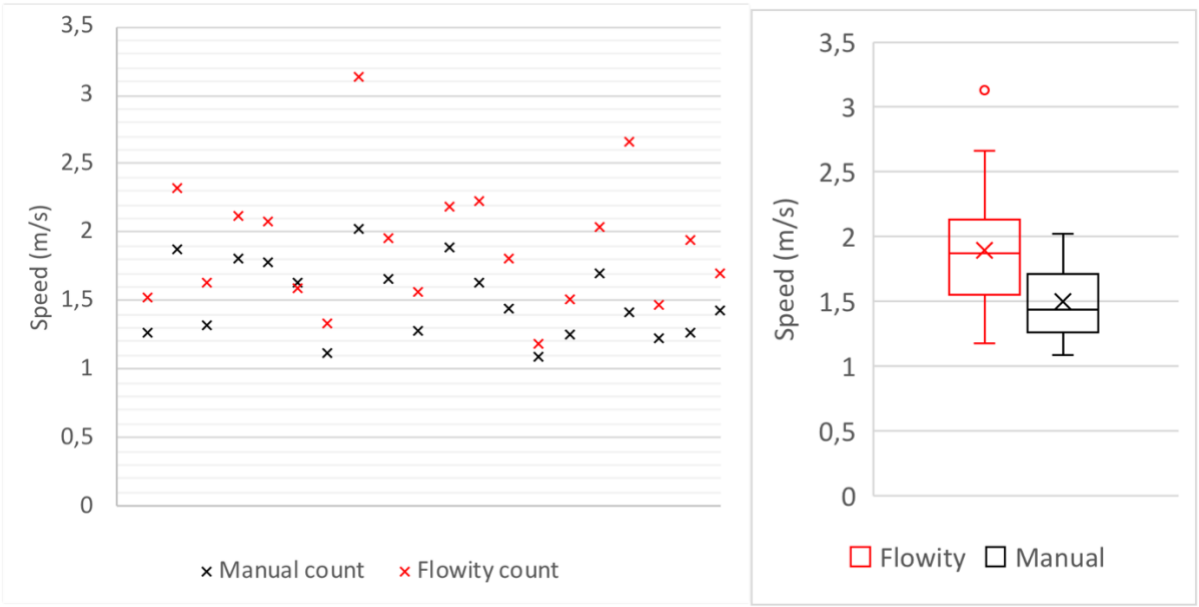


Figure 39 Comparing movement speeds of the 20 pedestrians measured manually and with Flowity.

Comparing the all the pedestrian walking speeds from the manual counting with the Flowity count, Table 12 and Figure 40, reveals a difference in the measured walking speeds. It is important to remember that the manually counted movement speeds are based on an average walking distance, with measurment uncertainties. The comparison show that software counted walking speeds are generally higher. The software counted average speed is 32,3 % higher than the manual count. A KS-test was done and it shows that the two data samples have statistically significant different distributions. Some high outliers can be seen in the Flowity counted movement speeds. The highest outliers were individually analysed further. They revealed that during some short sections the detected pedestrians had moved a long distance. Several sections where the pedestrians had moved over 40 cm in 0,042 seconds (9,5 m/s) were found, thus resulting in a high average walking speed. The pedestrian average distance covered during 0,042 seconds was 9,24 cm (using the average movement speed of 2,22 m/s).

Table 12 Comparison between pedestrian walking speed measured manually and with Flowity.

		Min	*Max*	*Average*	*Median*	*Standard deviation*
Manual count	Speed (m/s)	1,08	2,69	1,67	1,54	0,31
Flowity count	Speed (m/s)	0,69	5,48	2,22	2,14	0,51

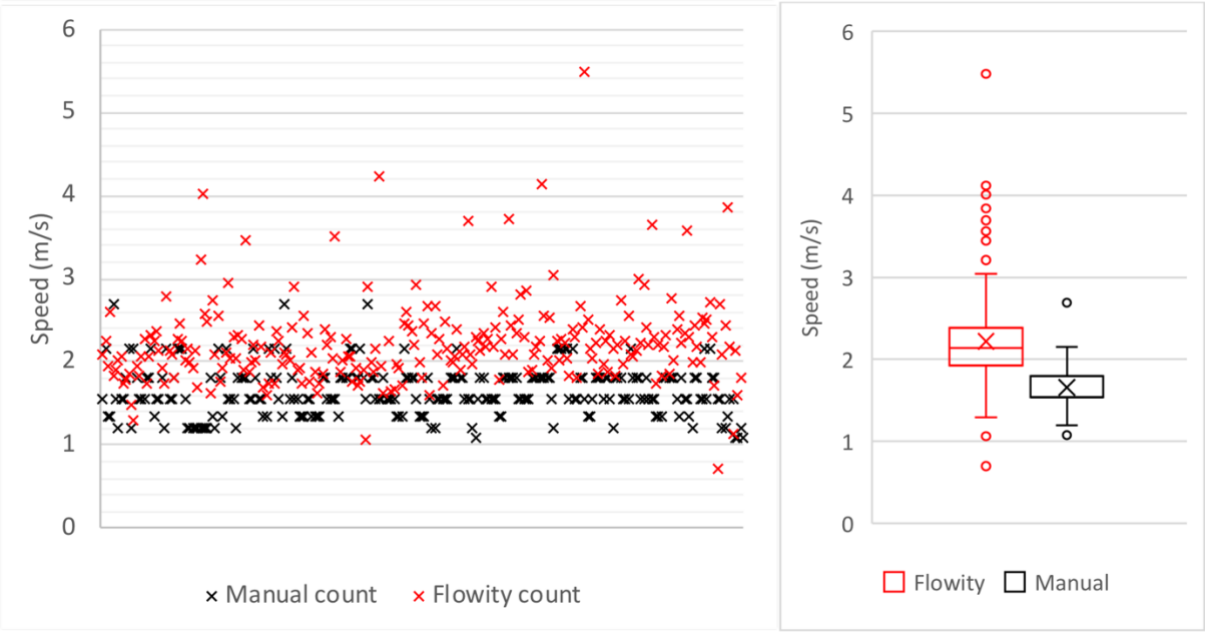


Figure 40 Comparison between pedestrian walking speed measured manually and with Flowity.

Comparing the flow across line 2 from the manual count with the Flowity software counting, Table 13 and Figure 41 seen below, reveals that the manually counted flow is slightly higher. For instance the manually measured average flow is 14,2 % higher than what the software produced. Results from the KS-test shows that the two data samples do not have statistically significant different distributions.

Table 13 Comparison between flow measured manually and with Flowity.

		Min	*Max*	*Average*	*Median*	*Standard deviation*
Manual count	Flow (persons/min*m)	0	5	2,09	1,99	1,27
Flowity count	Flow (persons/min*m)	0	6,49	1,83	1,51	1,25

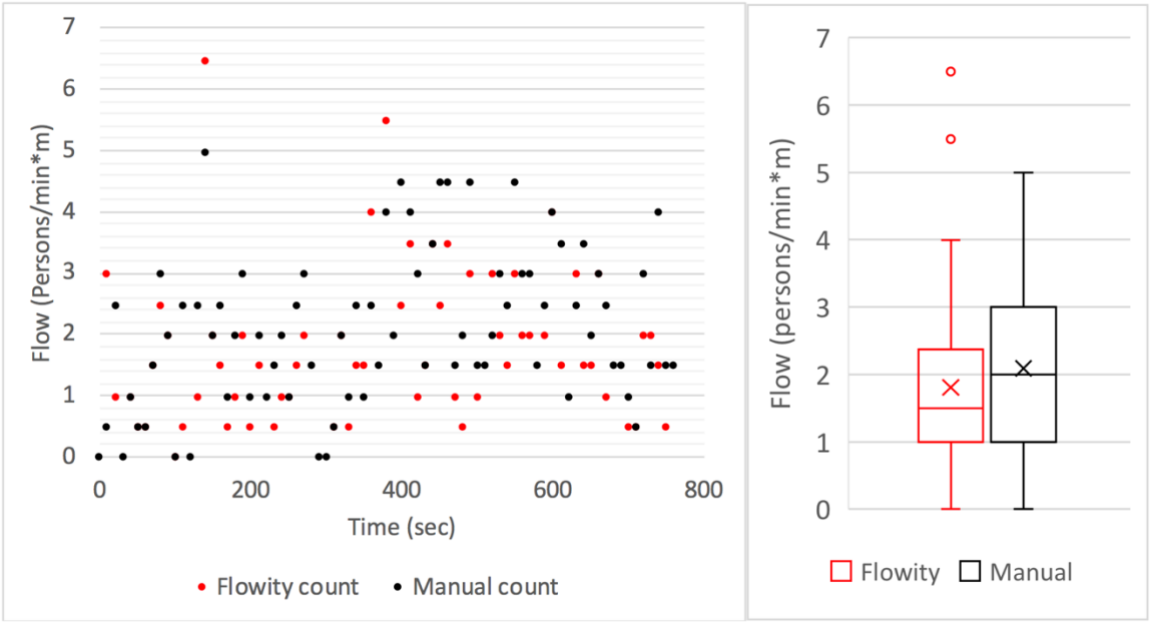


Figure 41 Comparison between flow measured manually and with Flowity.

Comparing the global density from the manual count with the Flowity software counting, Figure 42 and Table 14 seen below, reveals that the manually counted density is slightly higher than what the software produced. The manually counted average density is 15,1 % higher than what the software produced. Results from the KS-test shows that the two data samples do not have statistically significant different distributions.

Table 14 Comparison between global density measured manually and with Flowity.

		Min	*Max*	*Average*	*Median*	*Standard deviation*
Manual count	Global density (persons/m2)	0	0,11	0,038	0,039	0,025
Flowity count	Global density (persons/m2)	0,01	0,13	0,033	0,029	0,026

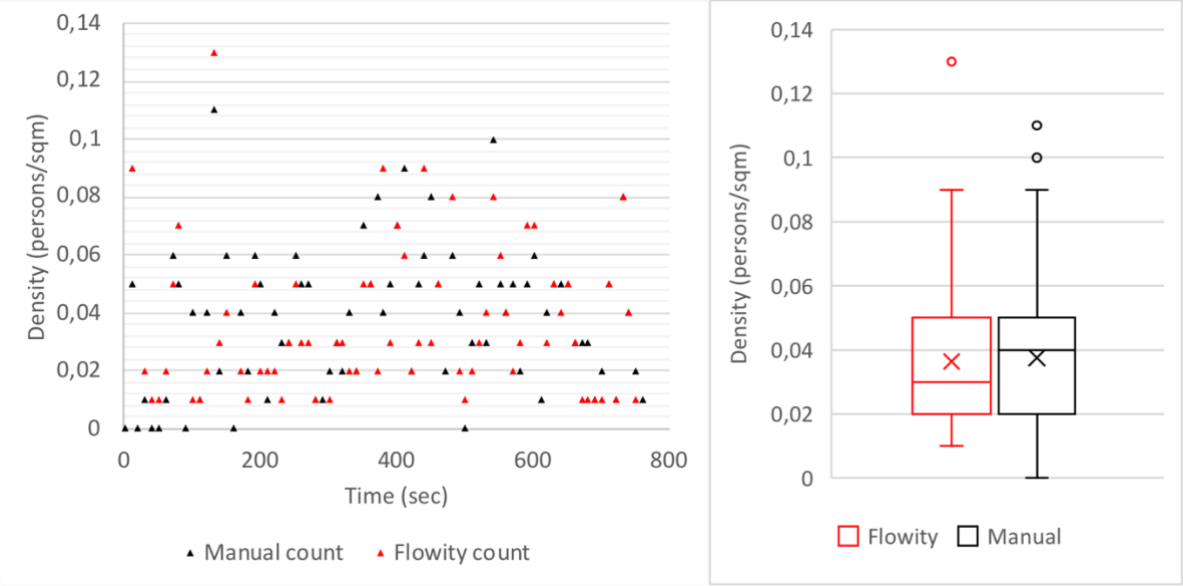


Figure 42 Comparison between global density measured manually and with Flowity.

Comparing the maximum local density measured manually with the software, a difference of 0,5 persons per square meters can be seen. The maximum local density that the software detected (3 persons per square meter) was found at 125 seconds into the video. This compares to the manually seen, 124 seconds and was at the same area as well.

In addition to the detection performance seen in Table 10, the number of unique pedestrians counted manually and the ones detected by the software is compared in Table 15.

Table 15 Comparison between manual- and software pedestrian count.

	Manual	*Software*	*Difference (%)*
Number of unique pedestrians detected	**576**	513	63 (11)
Number of unique pedestrians detected (small area,Figure 19)	**316**	285	31 (11)
Number of pedestrians using Exit A	**224**	173	51 (29)
Number of pedestrians using Exit B	92	**112**	20 (22)

7 Discussion

The results from testing the Flowity software in this case study showed that the software, utilizing deep neural networks, algorithms and machine learning managed to identify and detect pedestrians within the case video. Flowity also provided data on movement patterns, movement speeds, flows, densities and route choices of the detected pedestrians.

From the detected pedestrians plotted on a longitude and latitude, seen in Figure 31 and Figure 32, some clear movement patterns can be seen, e.g. the very prevalent tree that was located in the middle of the frame had been avoided by the pedestrians. Most of the pedestrians seem to have been walking at the left-hand side, which is consistent with what can be seen in the heat-map as well as observations made manually. The process of assigning which part of a detected pedestrians bounding box to represent the GPS-location is a company secret and thus no conclusive statements can be made. What can be seen however, is the fact that some detections seem to be located inside the bush on the left side, indicating that it was not the feet or the bottom of the bounding box that was the part representing the GPS-location. The rounded edges seen at bottom of Figure 31, is the edge of the video recording, stopping any further detection beyond that point.

There are uncertainties in the manual measurements taken at the location of the video along with the manual counting of the pedestrians, which would have an impact on all the collected factors, movement speeds, flows etc. As previously stated, the counting and measuring was only done by the author, however done twice and then averaged. All the manual counting and software accuracy determination was time consuming and having a second researcher, independently count or measure would have been more ideal and would likely have contributed to less uncertainties.

The manual measurement of distance covered by all pedestrians crossing the small area seen in Section 5.1 Manually collected pedestrian information was simplified to only be represented by the average travel distance of 20 different pedestrians. The manual analysis of these 20 pedestrians and their travel paths was difficult and time consuming and therefore only 20 were made. Using this average when calculating the remaining pedestrians movement speeds have likely contributed to the manually counted walking speeds being uncertain, for instance, by not taking all the pedestrians, direction changes and slight diagonal movements into account. The Flowity software produces walking distances that are 24,09 % higher than the manual count as seen in Section 6.5. taking into account that the uncertainties of the manual measurements were estimated to be in the range of 40-60 centimeters. Comparing those 24,09 % to the fact that Flowity also produces movement speeds that are 32,3 % higher the manual count can give an indication of an offset. It can also just be a problem with either the softwares distance measuring method or the manual measuring. The results from the KS-test also indicates that either the software or the manual counting is not representative of the other by showing that the two data samples have statistically significant different distributions. Some of the high outliers that can be seen in the Flowity counted movement speeds, Figure 40, could likely have contributed to the distributions being different. The high outliers occurred when the detected pedestrians had moved a long distance during short sections. Some of the pedestrians had several sections where the pedestrians had moved over 40 centimeters in 0,042 seconds (9,5 m/s) thus resulting in a high average walking speed. The problem seemed to arise as the detection of the pedestrian suddely seized and the uniquely assigned ID re-appeared on another pedestrian for a brief moment, then later coming back to the originally detected pedestrian. Further investigation and testing would have to be done in order to more precicely pin point the problem. Improvements to the manual counting method

in order to limit uncertainties, could be to use precision timing devices, pressure sensitive floor tiles, motion trackers on pedestrians, independent control-counts or the use of LiDar.

Looking at the comparison between manually and software measured flows and global densities seen in Section 6.5 Comparing the manual and software collected data, the difference is not that big. A 14,2 % higher average flow when doing manual counting and a 15,1 % higher average global denisity when measured manually. Results from both of the respective KS-test indicates that the software- or the manual counting is representative of the other by showing that the two data samples do not have statistically significant different distributions. The uncertainties connected to the process of manually counting pedestrians along with the lack of knowledge of how diffrerent factors influence detection perfomance may have caused this difference in flows and densities. The maximum local density produced by the software did not compare to well with the manual approximation, 3 versus 2 persons per square meter. The fact that the software detected one more pedestrian, could just be a faulty detection or that pedestrians bounding box was inside of the cell, even though the person was outside of the cell, thus contributing to the count. More investigation would be needed and it would be interesting to do further testing and include more measurments of local densities, but it would require a better way of manually obtaining local densites.

The pedestrian detection accuracy was shown to be around 70 % for this case study, as seen in Table 10. This percentage is below the average detection accuracy of about 90%, stated by, (Boltes & Seyfried, 2013) in their study of mono camera based pedestrian detection systems. This is probably due to a lot of reasons. All of the information about the software and how it operates was not fully disclosed (due to secrecy) and, as stated by (Yilmaz et al., 2006), the software's implemented features and algorithms will predominantly affect the performance. An in-depth investigation into which factors, for instance those listed in Figure 6, and how they impacted the detection performance was not done but will be an important topic to research in the future. The basis for the detection accuracy in this study was made by counting the number of correct and incorrect pedestrian detections in 500 of the 18240 individual frames. A limit of 500 frames had to be set to keep a reasonable workload. Any indication of how representative these 500 frames were of the total population cannot be made. The frames were not specifically chosen and did not represent any particular composition of pedestrians, either high or low densities. Future detection accuracy measurements need to specify at which densities, levels of occlusion and conditions they are representative for.

Moreover, another aspect of detection and accuracy, which very well may impact the overall performance is high densities and the occlusion that may occur. If pedestrians walk in close proximity of each other and only the head or some limbs are visible, the detection system needs to still detect these pedestrians as separate and individual persons. Another aspect important for the detection accuracy is a phenomenon that will be referred to as a re-identification problem. The automatic pedestrian detection by the Flowity software assigns a unique ID to each pedestrian. If a pedestrian becomes occluded or re-appears after being out of frame, the system is meant to recognise the unique features of that particular pedestrian and re-assign the correct ID. This re-identification can be seen to work at times but there are some instances where issues appear, and the complete magnitude of this issue is not known. This was done as a case study and no investigation into factors, influencing detection accuracy was made, will add uncertainties. There is a need for research on what factors impacts detection performance. Furthermore, what would be considered an acceptable detection accuracy in order to use a pedestrian detection software for any real application?

The “footfall” feature explained in Section 6.1 Video render was not investigated further, due to the placements of the feature being without any real intent and the output data generated from it, inconsistent. This feature is however an interesting addition to the pedestrian detector and should be tested more. The output data was generated at a 42 ms interval. This interval corresponds to the input video being recorded at 24 frames per second and was at the time not possible to change due to software limitations.

The time it took to run the detection software is unknown and not explicitly stated by the company. If a pedestrian detection software is able to produce real-time pedestrian egress analysis a shift from the now vary static approach to evacuation systems into something that is dynamic might be the future. The concept of an intelligent evacuation guidance system is elaborated on by (Wu, Lv, & Yu, 2016). Maybe the future will have emergency signs that update instantly and always show the most optimal route to limit congestion or hinder people from evacuating in an undesired direction.

Factors that were initially thought to be extracted from the case video, e.g. pedestrian age, gender, body size, movement impairment or incapacitation was not done due to limitations in the software. The testing did not include other important aspects for fire safety engineering application such as indoor spaces and actual evacuating pedestrians. The project did not include any presence of smoke, fire, limited visibility, alarms or difference in video quality (such as use of low-resolution video or thermal imagery). This case study was done with a video, that was outdoors, high-resolution and not including any real evacuation egress. However, the concept of being able to detect pedestrians, track, and count them is the essence and core aspect of using a pedestrian detection system. If the system cannot provide any detection, tracking or counting it does not matter if the testing is done indoors, outdoors, in an evacuation situation or with the presence of smoke. The development of a general protocol to be used when systematically testing pedestrian detection systems for circulation/egress applications needs to be researched in the future. Future research also includes tests with different cameras, environments or events taking place. This research is essential in moving the technology in a direction where it would be more applicable for the fire safety engineer. Future research into pedestrian detection systems that might not just give accurate information about pedestrians would be also interesting. Maybe detection systems could detect, alert and highlight blocking debris within a corridor or doorway or detect and alert for lowered visibility, flames or other dangers.

8 Conclusion

Results from the study show that:

- The factors the Flowity software managed to extract were data on movement patterns and route choices of detected pedestrians as well as measuring movement speeds, flows and densities at different sections. Factors that were not possible to extract, given the current state of the software, were the age, gender, body size, and any movement impairments of the pedestrians.
- The Flowity software could identify people and automatically presented them as detected pedestrians with a detection accuracy of 70 %. This applies for this case-study, which was an outdoor video, filmed at a 1080p resolution, with a camera placed 4 meters above the walking area and at an angle of 21 degrees. The maximum global people density measured was 0,13 persons/m^2 and the maximum local density was 3 persons/m^2.
- Comparing manually and software measured factors, a statistically significant difference in measured movement speeds was observed. This could be due to the high outliers that the software produced or the uncertainties in the manual measurements, however this needs to be further investigated. The software measured a 32 % higher average speed than what was manually measured.
- Comparing manually and software measured flows and densities, there was no statistically significant difference between the measurement method. A 14,2 % higher average flow was measured with the manual counting and a 15,1 % higher average global denisity.

Flowity is a new software, never tested before for circulation/egress applications and there is need for further testing and research. The development of a general protocol to be used when systematically testing pedestrian detection softwares needs to be researched in the future along with research into factors and how they influence detection performance.

9 References

Baabou, S., Ben, A., Ben Farah, M., Abubakr, A., & Kachouri, A. (2019, March 1). *A Comparative Study and State-of-the-art Evaluation for Pedestrian Detection*. 485–490. https://doi.org/10.1109/STA.2019.8717226

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., & Muller, U. (2017). Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. *ArXiv:1704.07911 [Cs]*. Retrieved from http://arxiv.org/abs/1704.07911

Boltes, M., & Seyfried, A. (2013). Collecting pedestrian trajectories. *Neurocomputing*, *100*, 127–133. https://doi.org/10.1016/j.neucom.2012.01.036

Boverket´s building regulations – mandatory provisions and general recommendations, BBR. Retrieved 21 October 2019, from Boverket website: https://www.boverket.se/en/start/publications/publications/2019/boverkets-building-regulations--mandatory-provisions-and-general-recommendations-bbr/

Brummelen, G. V. (2013). *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press.

Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, *300*, 17–33. https://doi.org/10.1016/j.neucom.2018.01.092

Bukowski, R. W., & Tubbs, J. S. (2016). Egress Concepts and Design Approaches. In M. J. Hurley, D. Gottuk, J. R. Hall, K. Harada, E. Kuligowski, M. Puchovsky, … C. Wieczorek (Eds.), *SFPE Handbook of Fire Protection Engineering* (pp. 2012–2046). https://doi.org/10.1007/978-1-4939-2565-0_56

But what is a Neural Network? | Deep learning, chapter 1. (n.d.). Retrieved from https://www.youtube.com/watch?v=aircAruvnKk

Caltech Pedestrian Detection Benchmark. (n.d.). Retrieved 8 October 2019, from http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/

Corbetta, A., Meeusen, J., Lee, C., & Toschi, F. (2016, October 18). *Continuous measurements of real-life bidirectional pedestrian flows on a wide walkway*.

Daimler Pedestrian Benchmark Data Set. (n.d.). Retrieved 8 October 2019, from http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/daimler_pedestrian_benchmark_d.html

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 886–893. https://doi.org/10.1109/CVPR.2005.177

Dollár, P., Wojek, C., Schiele, B., & Perona, P. (2011). Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*,

743–761. https://doi.org/10.1109/TPAMI.2011.155

Fruin, J. J. (n.d.). *Pedestrian planning and design. Revised Edition*. *1987*.

Gouda, R., & Nayak, J. S. (2015). *Survey on Pedestrian Detection, Classification and Tracking*.

Guan, D., Cao, Y., Yang, J., Cao, Y., & Yang, M. Y. (2018). Fusion of Multispectral Data Through Illumination-aware Deep Neural Networks for Pedestrian Detection. *Information Fusion*, *50*. https://doi.org/10.1016/j.inffus.2018.11.017

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558.

Huang, T. S. (n.d.). Computer Vision: Evolution and Promise. *1992*, *1992*.

INRIA Person dataset. (n.d.). Retrieved 8 October 2019, from http://pascal.inrialpes.fr/data/human/

Koza, J. R., Bennett, F. H., Andre, D., & Keane, M. A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. In J. S. Gero & F. Sudweeks (Eds.), *Artificial Intelligence in Design '96* (pp. 151–170). https://doi.org/10.1007/978-94-009-0279-4_9

Kuligowski, E. D., Peacock, R. D., & Hoskins, B. L. (2010). A Review of Building Evacuation Models, 2nd Edition | NIST. *Technical Note (NIST TN) - 1680*. Retrieved from https://www.nist.gov/publications/review-building-evacuation-models-2nd-edition

Kurkcu, A., & Ozbay, K. (2017). Estimating Pedestrian Densities, Wait Times, and Flows with Wi-Fi and Bluetooth Sensors. *Transportation Research Record*, *2644*(1), 72–82. https://doi.org/10.3141/2644-09

Lavi, B., Serj, M. F., & Ullah, I. (2018). Survey on Deep Learning Techniques for Person Re-Identification Task. *ArXiv:1807.05284 [Cs]*. Retrieved from http://arxiv.org/abs/1807.05284

Lovreglio, R., Ronchi, E., & Kinsey, M. J. (2019). An Online Survey of Pedestrian Evacuation Model Usage and Users. *Fire Technology*. https://doi.org/10.1007/s10694-019-00923-8

Lundh, K., & Pettersson, F. (2017). *Microsoft Kinect för utrymning i smarta byggnader*. *2017*, 85.

Mehralian, S., & Palhang, M. (2013). Pedestrian detection using principal components analysis of gradient distribution. *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, 58–63. https://doi.org/10.1109/IranianMVIP.2013.6779950

Ogawa, T., Sakai, H., Suzuki, Y., Takagi, K., & Morikawa, K. (2011). Pedestrian detection and tracking using in-vehicle lidar for automotive application. *2011 IEEE Intelligent Vehicles Symposium (IV)*, 734–739. https://doi.org/10.1109/IVS.2011.5940555

Ojala, T., Pietikäinen, M., & Harwood, D. (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. *Proceedings of 12th International Conference on Pattern Recognition*, *1*, 582–585 vol.1. https://doi.org/10.1109/ICPR.1994.576366

Pathak, R., Sivraj, P., Sivraj, P., & Sivraj, P. (2018). Selection of algorithms for pedestrian detection during day and night. *Lecture Notes in Computational Vision and Biomechanics*, *28*, 120–133. https://doi.org/10.1007/978-3-319-71767-8_11

Pathfinder Resources | Thunderhead Engineering. (n.d.). Retrieved 22 October 2019, from https://www.thunderheadeng.com/pathfinder/resources/

Prado, K. S. do. (2018, February 3). Face Recognition: Understanding LBPH Algorithm. Retrieved 10 October 2019, from Medium website: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

Pulford, G. (2005). Taxonomy of multiple target tracking methods. *Radar, Sonar and Navigation, IEE Proceedings -*, *152*, 291–304. https://doi.org/10.1049/ip-rsn:20045064
Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Seebo. (2019, February 8). Supervised vs. Unsupervised Machine Learning. Retrieved 8 October 2019, from Medium website: https://medium.com/datadriveninvestor/supervised-vs-unsupervised-machine-learning-5200ffa7301a

Seer, S., Brändle, N., & Ratti, C. (2014). Kinects and human kinetics: A new approach for studying pedestrian behavior. *Transportation Research Part C: Emerging Technologies*, *48*, 212–228. https://doi.org/10.1016/j.trc.2014.08.012

Tedblad, R. (2019). *Flowity – Artificial Intelligence for traffic analysis*.

The PASCAL Visual Object Classes Homepage. (n.d.). Retrieved 8 October 2019, from http://host.robots.ox.ac.uk/pascal/VOC/

The Swedish National Board of Housing, Building and Planning's general recommendations on the analytical design of a building's fire protection, BBRAD. (n.d.). Retrieved 22 October 2019, from Boverket website: https://www.boverket.se/en/start/publications/publications/2013/the-swedish-national-board-of-housing-building-and-plannings-general-recommendations-on-the-analytical-design-of-a-buildings-fire-protection-bbrad/

Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., & Tubaro, S. (2016). Deep Convolutional Neural Networks for pedestrian detection. *Signal Processing: Image Communication*, *47*, 482–489. https://doi.org/10.1016/j.image.2016.05.007

Ujwal, U., & Brémond, F. (2016). *New Results—Pedestrian Detection on Crossroads*. Retrieved from INRIA website: https://raweb.inria.fr/rapportsactivite/RA2016/stars/uid129.html

Vanumu, L. D., Ramachandra Rao, K., & Tiwari, G. (2017). Fundamental diagrams of pedestrian flow characteristics: A review. *European Transport Research Review*, *9*(4), 49. https://doi.org/10.1007/s12544-017-0264-6

Wu, Z. Y., Lv, W., & Yu, K. (2016, October). *A Framework of Intelligent Evacuation Guidance System for Large Building*. Presented at the 2016 5th International Conference on Civil, Architectural and Hydraulic Engineering (ICCAHE 2016). https://doi.org/10.2991/iccahe-16.2016.111

Yilmaz, A., Javed, O., & Shah, M. (2006). Object Tracking: A Survey. *ACM Comput. Surv.*, *38*(4). https://doi.org/10.1145/1177352.1177355

Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (Vol. 8689, pp. 818–833). https://doi.org/10.1007/978-3-319-10590-1_53

Appendix A

The manual measurements taken on-site can be seen in Figure 43. These were complemented with further measurements when doing a manual determination of the pedestrian travel paths (Section 5.1). A tape measure was used to collect all the measurements.

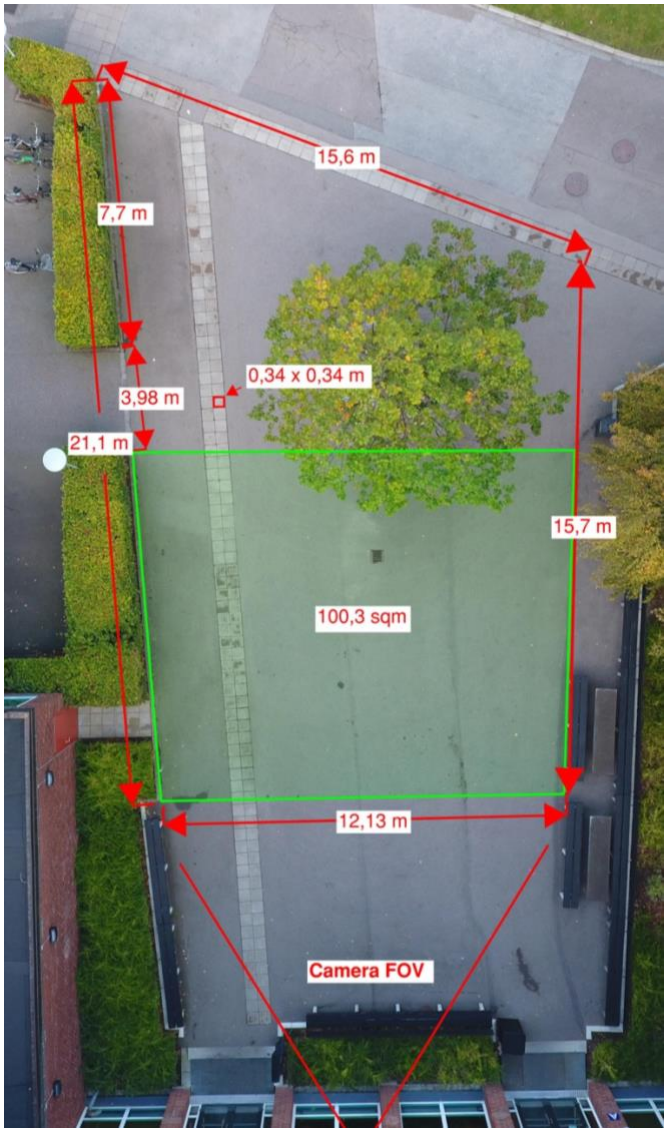


Figure 43 Manual measurements taken on-site.

Appendix B

The haversine formula is a way of calculating the distance between two points (longitude and latitude) on a sphere (Brummelen, 2013). Below in an example of a distance calculation done in Microsoft Excel based on the Haversine formula. The earths radius is equal to 6371km. This method was used to determine the distance between GPS-points from the software output data.

Table 16 Example, using the Haversine formula for distance calculation between two points on a sphere.

	A	*B*	*C*	*D*	*E*
1	**Latitude**	**Longitude**			**Distance (m)**
2	55,71261089	13,21065387			
3	55,71261502	13,21071197	**1.)** 8,28733E-14	**2.)** 5,75754E-07	3.) 3,668131439

1.) =SIN(ABS(A3-A2)*PI()/180/2)^2+COS(A2*PI()/180)*COS(A3*PI()/180)*SIN(ABS(B3-B2)*PI()/180/2)^2
2.) =2*ARCTAN2(ROT(1-C3);ROT(C3))
3.) =6371*D3*1000

Do retrieve the movement speed, the time between GPS-points is first produced. The distance is then divided by the delta t to produce the movement speed in meters per second, see Table 17.

Table 17 Example, using Microsoft Excel to produce movement speeds.

	A	*B*	*C*	*D*	*E*	*F*
1	**Timestamp (s)**	**Latitude**	**Longitude**	**Distance (m)**	**Delta t (s)**	**Speed (m/s)**
2	1,1337	55,71261089	13,21065387			
3	4,337	55,71261502	13,21071197	3.) 3,668131439	4.) 3	5.) 1,22266667

4.) =A3-A2
5.) =D3/E3

The output data is sorted by ID number and is placed in rising order, see example in Table 18 below. Using the Haversine formula and the functions in Microsoft Excel enables the calculation of the speed.

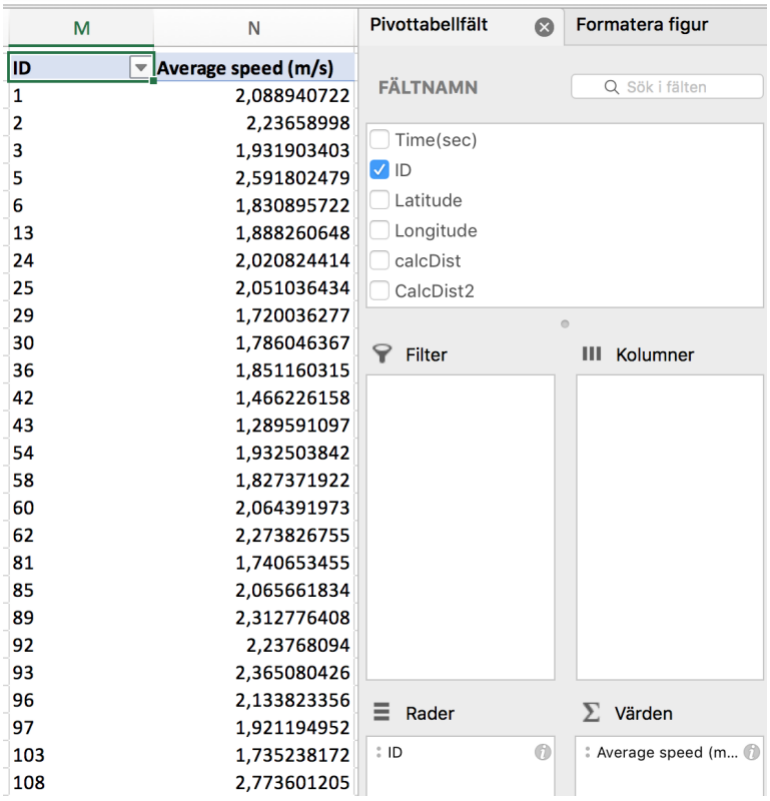
Table 18 Example of sorting IDs and calculating movement speed.

Timestamp (s)	*ID*	*Latitude*	*Longitude*	*Distance (m)*	*Delta t (s)*	*Speed (m/s)*
8,8	1	55,71261613	13,21075016	0,030044645	0,041	0,732796209
8,842	1	55,71261604	13,21074886	0,081700484	0,042	1,945249629
8,884	1	55,71261607	13,21074929	0,02714678	0,042	0,646351901
8,926	1	55,7126161	13,21074973	0,027234031	0,042	0,648429307
8,967	1	55,71261608	13,21075059	0,054310888	0,041	1,32465581
9,009	1	55,71261611	13,21075234	0,109614643	0,042	2,609872453
9,051	1	55,71261603	13,21075234	0,00921317	0,042	0,219361201
9,092	1	55,71261592	13,21075322	0,056463873	0,041	1,37716763
9,134	1	55,71261584	13,21075455	0,083793205	0,042	1,995076317
1,084	2	55,71261052	13,2106596	0,044994915	0,041	**0***
1,126	2	55,71261022	13,21065975	0,035381802	0,042	0,84242386
1,168	2	55,71260988	13,21065923	0,049232962	0,042	1,17221337
1,21	2	55,71260995	13,21065852	0,04528447	0,042	1,078201662
1,251	2	55,71261019	13,21065909	0,043850239	0,041	1,069518015
1,293	2	55,71261023	13,21065771	0,08659952	0,042	2,061893342
1,335	2	55,7126104	13,21065693	0,052223218	0,042	1,243409959
1,376	2	55,71261044	13,21065552	0,088417863	0,041	2,156533249

The formula down below prevents the system from calculating the distance between two different IDs and outputs a zero if the IDs are different.

0*.) =IF(B11=B10;6371*G79*1000;0)

Using Microsoft Excels Pivot Table function enables the presentation of all the unique Ids and the respective average, see Figure 44 for an example of the average movement speed.



ID	Average speed (m/s)
1	2,088940722
2	2,23658998
3	1,931903403
5	2,591802479
6	1,830895722
13	1,888260648
24	2,020824414
25	2,051036434
29	1,720036277
30	1,786046367
36	1,851160315
42	1,466226158
43	1,289591097
54	1,932503842
58	1,827371922
60	2,064391973
62	2,273826755
81	1,740653455
85	2,065661834
89	2,312776408
92	2,23768094
93	2,365080426
96	2,133823356
97	1,921194952
103	1,735238172
108	2,773601205

Figure 44 Using the Microsoft Excels Pivot table function.

Appendix C

In order to create a heat-map showing the frequency that a square meter was walked through, a function in Microsoft Excel and a conditional formatting table was used. The grid and function seen in Figure 45 below, describes the process of counting the number of times GPS-coordinates (longitude and latitude) passed through a cell of the grid. Rows A2-A141896 an B2-B141896 contained all the GPS-positions but cannot be seen here.

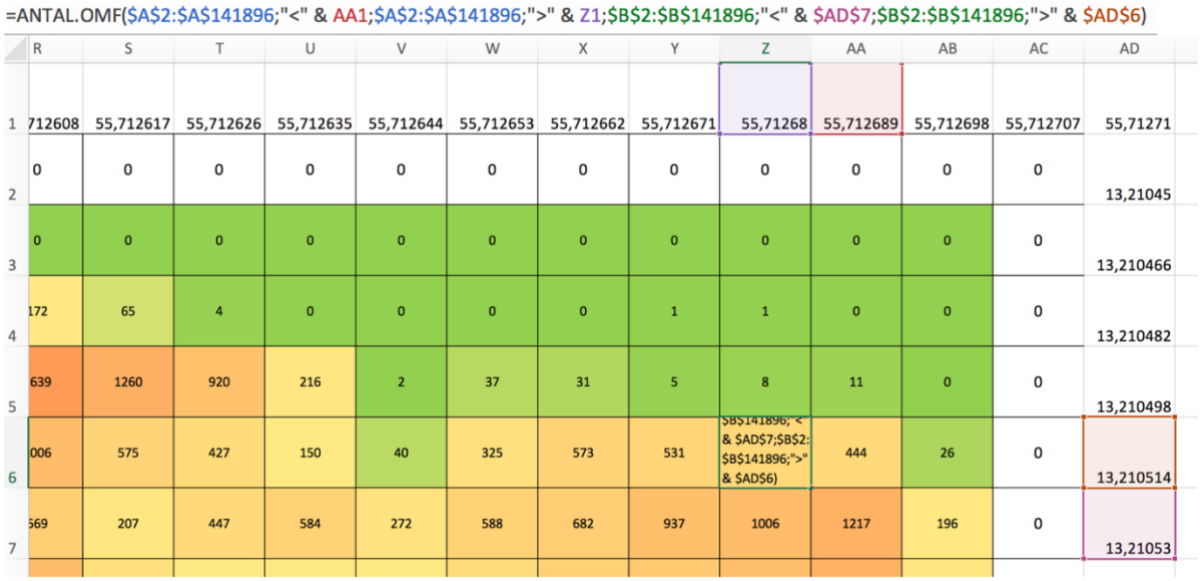


Figure 45 Creating a heat-map using Microsoft Excel functions.

Appendix D

The Kolmogorov-Smirnov (KS) test, with two samples, compares the cumulative distributions of the samples in a nonparametric test (Kirkman, T.W, 1996). This means that there is no notion of the kind of distribution the data is sampled from. The two independent samples were factors either measured manually or by the Flowity software. The KS-test was done using the IBM SPSS software (version 26), available as a student version and provided by Lund University.

Testing to either retain or reject a null hypothesis was done for all of the factors, movement speed, flow and density.

The null hypothesis for the manually- and software measured movement speeds of all pedestrians was formulated. “The distribution of movement speeds is the same regardless of measurement method”. Results from the SPSS software, seen in Figure 46 shows that the two data samples have statistically significant different distributions. This rejects the null hypothesis and indicates that either the software- or the manual counting is not representative of the other.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of Speed is the same across categories of Method.	Independent-Samples Kolmogorov-Smirnov Test	.000	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .050.

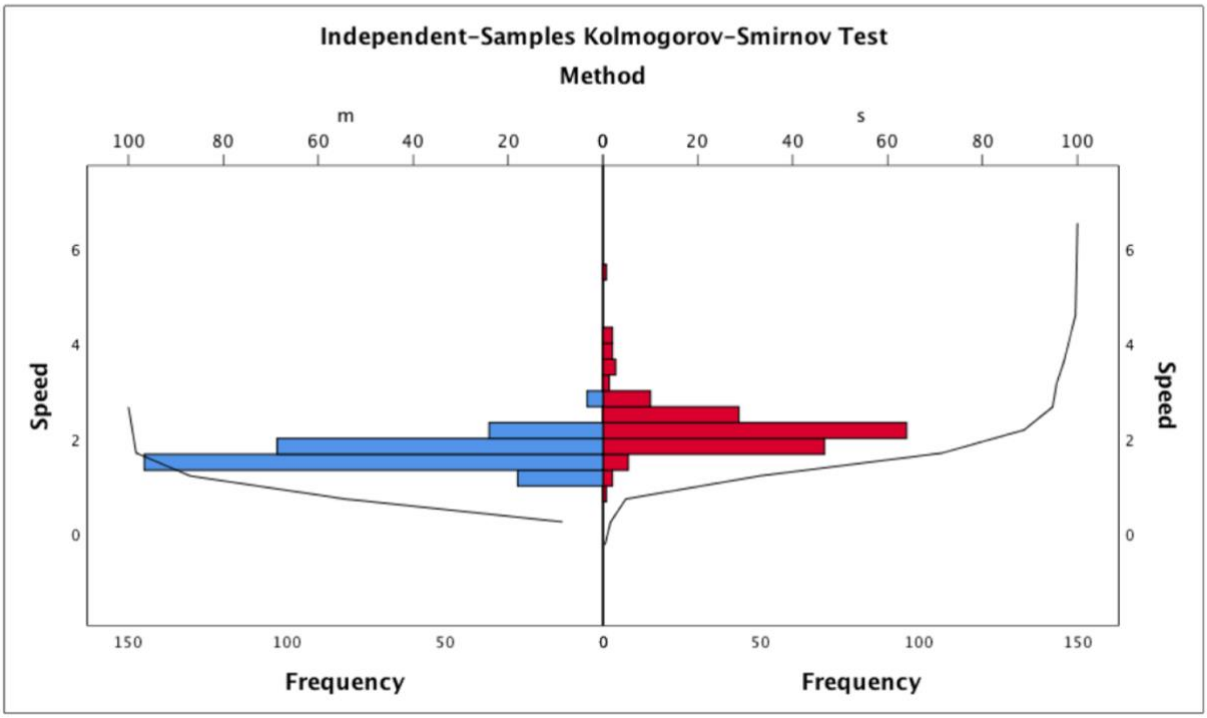


Figure 46 KS-test of all the measured pedestrian movement speeds. m = manual, s = software.

Following the same procedure but looking at the distributions of the 20 individually measured pedestrian movement speeds, results seen in Figure 47, also shows that the two data samples have statistically significant different distributions.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of Speed is the same across categories of Method.	Independent-Samples Kolmogorov-Smirnov Test	.035	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .050.

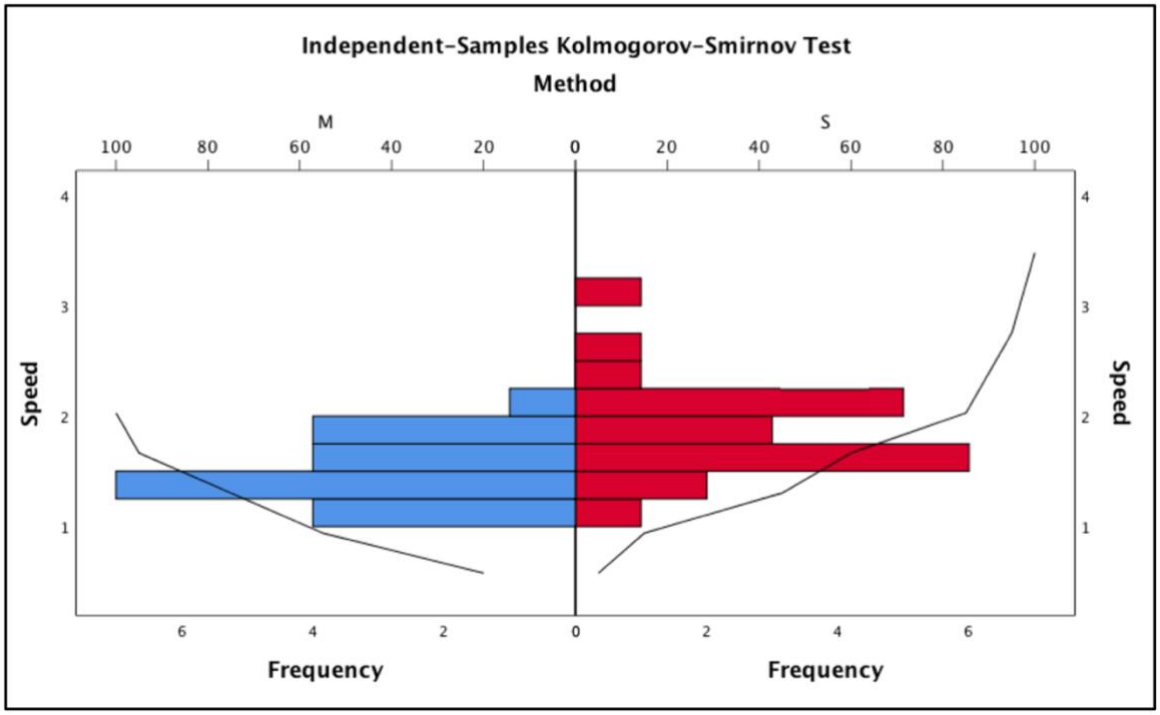


Figure 47 KS-test of the 20 individually measured pedestrian movement speeds. m = manual, s = software.

The null hypothesis for the manually- and software measured flows was formulated. “The distribution of flows is the same regardless of measurement method”. Results from the SPSS software, seen in Figure 48 shows that the two data samples do not have statistically significant different distributions. This retains the null hypothesis and indicates that the software- or the manual counting is representative of the other.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of Flow is the same across categories of Method.	Independent-Samples Kolmogorov-Smirnov Test	.271	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .050.

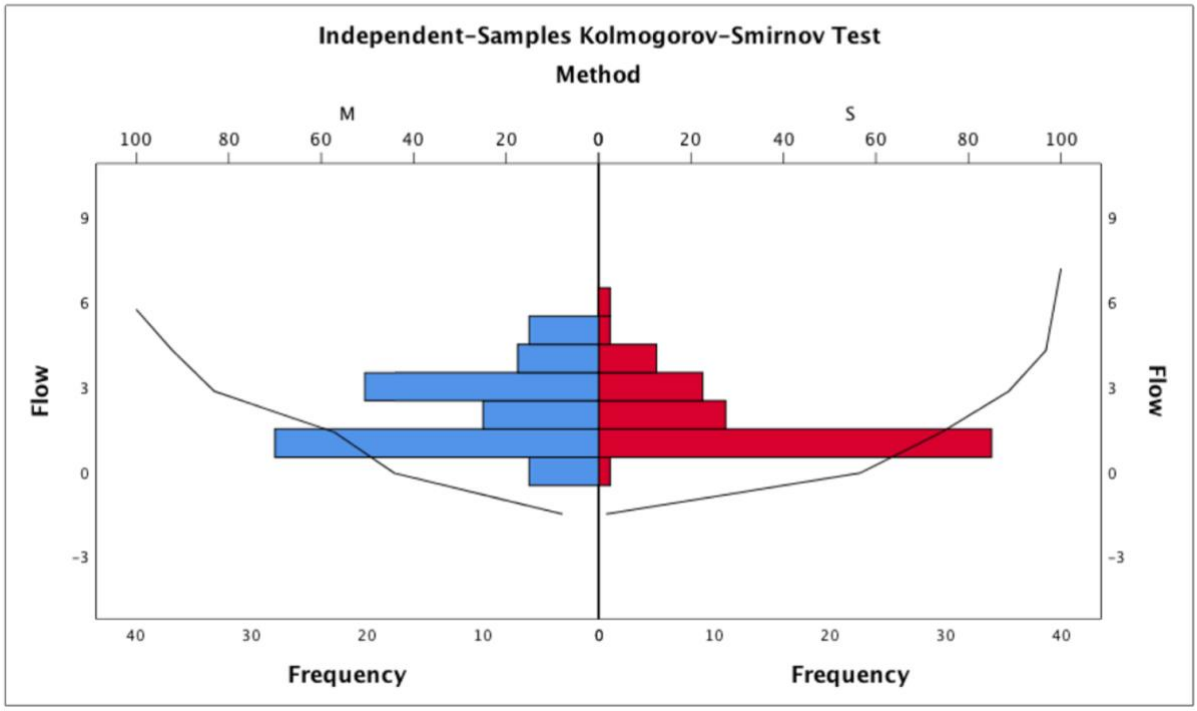


Figure 48 KS-test of the measured flows. m = manual, s = software.

The null hypothesis for the manually- and software measured densities was formulated. “The distribution of densities is the same regardless of measurement method”. Results from the SPSS software, seen in Figure 49 shows that the two data samples do not have statistically significant different distributions. This retains the null hypothesis and indicates that the software- or the manual counting is representative of the other.

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distribution of Density is the same across categories of Method.	Independent-Samples Kolmogorov-Smirnov Test	.353	Retain the null hypothesis.

Asymptotic significances are displayed. The significance level is .050.

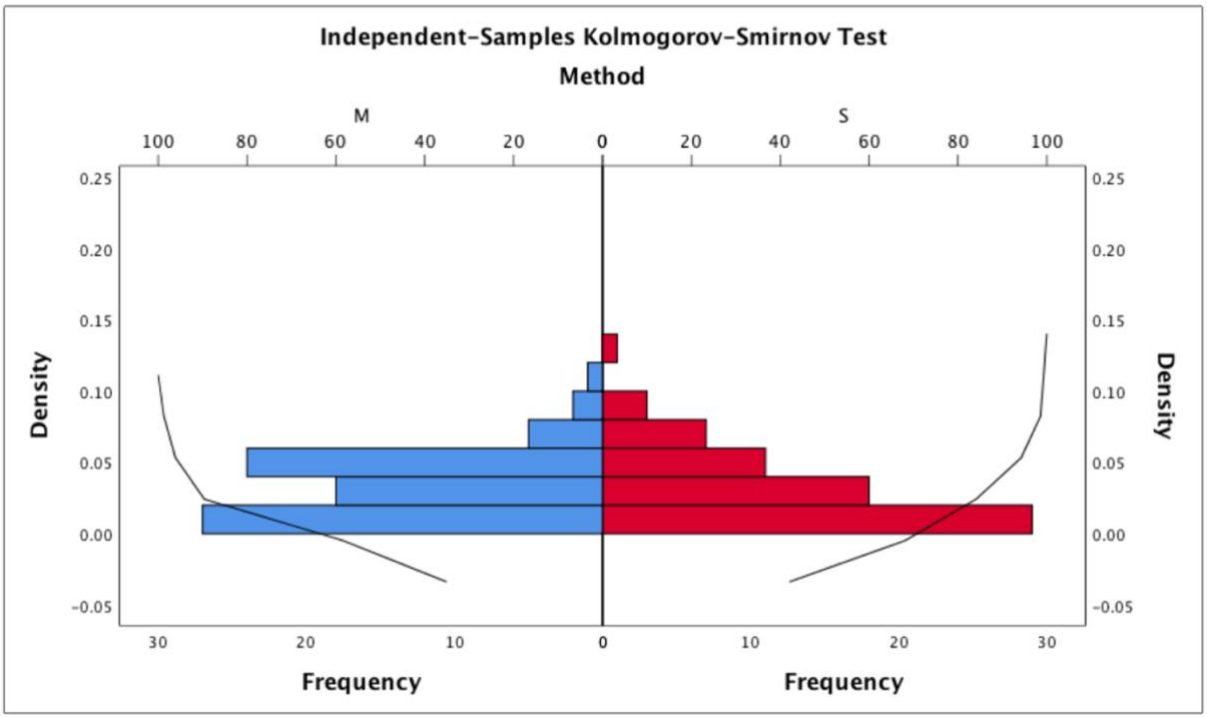


Figure 49 KS-test of the measured densities. m = manual, s = software.