Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

A new algorithm for estimating pedestrian flows during massive touristic events, optimized for an existing camera setup

Relatore:
Chiar.mo Prof.
Sandro Rambaldi

Correlatore:
Chiar.mo Prof.
Stefano Sinigardi

Presentata da:
Gianluca Guidi

Sessione II
Anno Accademico 2016/2017

It is the time you have wasted
for your rose that makes your
rose so important.

Antoine de Saint-Exupéry, The Little Prince

Abstract

In questa tesi presento un nuovo algoritmo per l'analisi di filmati che permette di calcolare il flusso di persone che attraversano un passaggio anche in presenza di condizioni sfavorevoli della telecamera. Il lavoro di tesi si è concentrato sull'analisi di una serie di sequenze video precedentemente estratte da una telecamera di sicurezza rivolta verso il Ponte della Costituzione a Venezia, con lo scopo di stimare il flusso pedonale sul ponte. La scarsa qualità dei video dovuta alla bassa risoluzione ed il posizionamento non ottimale della telecamera, che provoca numerose sovrapposizioni, causano il fallimento di molte tecniche di computer vision esistenti, perciò è stato necessario creare una nuova soluzione. È stata inoltre effettuata una verifica dell'algoritmo attraverso un programma che lo implementa, analizzando sia dati artificiali che reali.

Introduzione

Venezia è una città turistica d'eccellenza, famosa in tutto il mondo. Alcuni eventi in particolare richiamano verso la città massicci flussi turistici, con rischi di sovraffollamento e congestione di alcune aree. In letteratura è già ben noto che le folle non controllate possono comportarsi in maniera pericolosa e provocare disastri [1]. Quando un luogo è già sovraffollato, è troppo tardi per applicare misure di sicurezza per evitare incidenti. È perciò importante essere in grado di predire in anticipo quante persone stanno raggiungendo un certo posto, in modo che in caso di sovraffollamento sia possibile fermarle in aree sicure ed impedire ad altri di raggiungere la città stessa. A tal proposito, Venezia si presenta come un caso particolare: a parte le imbarcazioni che vengono sporadicamente dalle spiagge e dalle località vicine, esiste solo un numero limitato di passaggi che permettono l'accesso alla città. Si può entrare a Venezia in treno o in automobile/autobus: tutti i treni arrivano in una specifica stazione, e tutte le automobili e gli autobus arrivano in Piazzale Roma. Monitorare i passaggi attorno a Piazzale Roma è un primo importante passo nell'essere in grado di contare le persone che entrano ed escono dalla città.

Per molti anni, il gruppo di Fisica dei Sistemi Complessi, nel Dipartimento di Fisica e Astronomia dell'Università di Bologna, ha avuto un forte interesse nello sviluppo di modelli di mobilità umana, particolarmente in relazione alla città di Venezia. I lavori precedenti sono stati devoti al problema delle interazioni e dinamiche umane in luoghi molto affollati [2], analizzando le decisioni delle persone in queste peculiari condizioni. In quel contesto è

stato necessario pianificare una specifica organizzazione e sistemazione delle telecamere, perché studiare il modello richiedeva di essere in grado di tracciare le persone. In questa tesi, l'interesse è ora posto sull'ottenimento di dati sulle persone che entrano nella città attraverso i passaggi principali, senza avere la possibilità di installare nuove telecamere.

Questa tesi si concentra sull'analisi di una serie di sequenze video estratte da una telecamera di sicurezza rivolta verso il Ponte della Costituzione, e si propone di fornire un modo automatico di stimare il flusso di persone che attraversano questo ponte. La situazione della telecamera è sfavorevole, ma non poteva essere modificata, perciò si è reso necessario sviluppare una soluzione che fosse in grado di rilevare flussi di alto volume anche in casi di risoluzione estremamente bassa. Data questa situazione, inizialmente c'era un grado di incertezza sulla possibilità di una soluzione sufficientemente precisa. Il lavoro descritto in questo elaborato ha implicato l'esplorazione di procedure di computer vision esistenti, ed il test di alcune di queste sui dati video disponibili, per scoprire se e come fosse possibile analizzare la difficile scena del Ponte della Costituzione.

Il capitolo 1 presenta gli obiettivi del progetto e le caratteristiche dello specifico scenario che doveva essere analizzato, insieme alle sfide tecniche ed ai problemi correlati; viene poi discusso lo stato dell'arte del conteggio di persone. Il capitolo 2 descrive la ricerca fatta per trovare una tecnica di sottrazione del background adatta, che è parte della soluzione che è stata sviluppata. Il capitolo 3 introduce il nuovo algoritmo che è stato creato per contare i pedoni che attraversano il ponte. Sono inclusi una descrizione dettagliata del funzionamento della procedura di conteggio, una discussione della complessità, e una panoramica delle tecnologie usate per l'implementazione. Il capitolo 4 spiega come sono stati raccolti manualmente i dati reali e mostra i risultati prodotti eseguendo l'algoritmo su una sequenza di test.

Il codice del programma che implementa l'algoritmo sviluppato è disponibile su GitHub al seguente URL: `https://github.com/physycom/peopleflow`

Introduction

Venice is a popular touristic city of world-wide fame. Some events in particular attract massive touristic flows towards the city, with risks of overcrowding and congestion of certain areas. In literature it is already well known that unmanaged crowds can behave dangerously and provoke disasters [1]. When a location is already overcrowded, it is too late to apply security measures to prevent incidents. It is therefore important to be able to predict in advance how many people are reaching a certain place, so that in case of overcrowding it is possible to keep them in safe areas and prevent others to reach the city itself. Venice has a peculiar feature in this regard: apart from boats sporadically coming from nearby beaches and resorts, there is only a limited number of passages that allow access to the city. Venice can be entered by train or by car/bus: all trains arrive at one specific train station, and all cars and buses arrive at *Piazzale Roma*. Monitoring the passages around *Piazzale Roma* is a first important step in being able to count the people who enter and exit the city.

For many years, the Physics of Complex Systems group, in the Physics and Astronomy Department of the University of Bologna, has had a strong interest in developing human mobility models, particularly in relation to the city of Venice. Previous works have been devoted to the problem of human interactions and dynamics in very crowded locations [2], analyzing people's decisions in these peculiar conditions. In that context it was necessary to plan a specific camera setup, because studying the model required being able to track people. In this thesis, the interest is now placed on obtaining data

about people entering the city through the main passages, without being able to install new cameras.

This thesis focuses on the analysis of a series of video sequences extracted from a security camera facing *Ponte della Costituzione*, which connects the train station and *Piazzale Roma*, and aims to provide an automatic way to estimate the flows of people crossing the bridge. The camera setup is unfavorable, but it could not be modified, so the solution was required to be able to detect flows of high volumes even in cases of extremely low resolution. Given this situation, at first there was a degree of uncertainty about the possibility of a sufficiently accurate solution. The work described in this document entailed exploring existing computer vision procedures, as well as testing some of them on the available video data in order to find out whether and how it is possible to analyze the challenging scene of *Ponte della Costituzione*.

Chapter 1 presents the project goals and the characteristics of the specific scenario that had to be analyzed, along with the problems and technical challenges connected to it; the state of the art of people counting is then discussed. Chapter 2 describes the research done for finding a suitable background subtraction technique, which is part of the solution that was developed. Chapter 3 introduces the new algorithm that was created for counting the pedestrians who cross the bridge. It includes a detailed description of how the counting procedure works, a discussion of the complexity and an overview of the framework used for implementation. Chapter 4 explains how the ground truth data was collected and shows the results produced by running the algorithm on a test sequence.

The code of the program that implements the developed algorithm is available on GitHub at the following URL: `https://github.com/physycom/peopleflow`

Contents

List of Figures

List of Tables

Chapter 1

Problem

This chapter presents the goal of the thesis, along with an analysis of the challenges that had to be overcome. The state of the art is also explored, and the applicability of existing techniques is discussed.

1.1 Venice Project

Venice attracts a large number of visitors each year, especially during famous public gatherings and events, such as the world-famous Venetian Carnival. Raising concerns about overcrowding have sparkled interest in the study of the unique congestion conditions of this location. The work described in this document is part of a larger project, carried out by the Physics of Complex Systems group inside the Department of Physics and Astronomy of the University of Bologna, that aims to analyze pedestrian traffic in this popular city. The studies are based on four types of data:

- cellular geolocation data
- captured wireless data
- photographs of crowds
- security camera footage

The cellular data, provided by a telecommunications provider, includes geo-location information, thus it can be used to trace paths and identify highly trafficked routes. This data can be used to create a rough estimate of how people move through the city; however, producing an accurate estimate is problematic because not all of the pedestrians are subscribers of the mobile operator, and because the tracking data is limited to the periods of active usage. In particular, a scale factor is required, and the result of this thesis can provide that factor.

Wireless data, which includes both Wi-Fi and Bluetooth, was acquired with a distributed network of sensors that can provide information about nearby active devices. Since most of the population is assumed to carry a smartphone, this kind of data can help in analyzing the pedestrian traffic.

Photographs of crowds in key locations, such as *Piazza San Marco*, can be systematically analyzed to create an empirical estimate of the size of crowds. The figures provided by both official and unofficial sources are often inaccurate and/or biased, thus a systematical analysis can provide valuable information to compare those figures.

Security camera footage can be used to extract information about the pedestrian flow. This can also provide the scale factor required for the analysis of cellular data. The thesis work consists in the development and implementation of a method for automated analysis of these video sequences. The work focused on a scene which includes *Ponte della Costituzione*: studying the pedestrian flow on this bridge is of particular interest because it connects the bus station and the train station.

1.2 Requirements

The main goal of this work consists in developing a vision system to create automated or semi-automated estimates of the pedestrian flow on *Ponte della*

Costituzione. The information that can be extracted in this way is useful for two main reasons:

1. An accurate estimate of the people flow on the bridge is important by itself, because it connects the bus station and the train station, which are two key locations for the analysis of the traffic in the city.

2. If the precision is sufficiently high, the output figures can be compared with those generated in the same location by cellular data. The latter covers the whole city, whereas security camera data is more limited in scope. By comparing the two results, a scale factor can be computed and applied to cellular data in order to obtain useful information on other parts of the city.

Bi-directional count The bridge is a constrained passage in which people usually (almost always) just walk from one end to the other, that is they do not turn back and reverse their path. This effectively means that pedestrian flow on a bridge is similar to car traffic on a straight road: at any time there are exactly two opposite flows of pedestrians. Thus, estimating two flows translates to counting how many people crossed the bridge in each of the two directions over a period of time.

Precision There is not a hard limit on the error that can be generated, since the obtainable precision can largely depend on the input conditions. An accuracy of 80% would be a very good one. Limiting the error (and increasing accuracy) of the estimate is the key property of a good solution.

Efficiency Efficiency is not a primary concern, but an obviously welcome property. Data that has to be analyzed was already collected and a large distributed computing framework is available for running the software, so optimization is not as important as other properties. However, the solution might be ported to an embedded system in the future, so that it is possible to

run it on-line on a local platform in the proximity of the camera. Therefore it should at least be possible to reduce the amount of required resources.

No camera setup control Since the traffic relative to a past event has to be estimated, the solution must work with existing sources. This means that there is no control over camera quality and placement. In particular, due the conditions described in the next section, the solution must work even in ultra-low resolution, high pedestrian density scenarios.

Reusability Focus should be placed on finding a working solution for analyzing a specific case, that is estimating pedestrian flow on *Ponte della Costituzione* in the scene described in the following section. Despite this, reusability and adaptabilty were also identified as important secondary requirements: the solution should also be easy to be adapted for working in a different but similar scenario.

1.3 Scenario

All available footage that includes a view of *Ponte della Costituzione* was extracted from a single security camera, located about 30 m away from the bridge's entrance and overlooking a large portion of the bus station. The camera is positioned on *Autorimessa Comunale*, a parking lot, which is the closest public building that was able to support the camera. Although all of the bridge is theoretically visible, people are only distinguishable to a human viewer in about half of it, so only a small fraction of the image frame is of significance. Due to the positioning of the camera in relation to the bridge, occlusions are a rather frequent event. An example of the described scenario can be seen in figure 1.1.

The video is 1280 pixels wide and 720 pixels high: at the time of writing this is already considered a fairly low resolution. However, since the region of interest (ROI) is limited to part of the bridge span, the actual area that

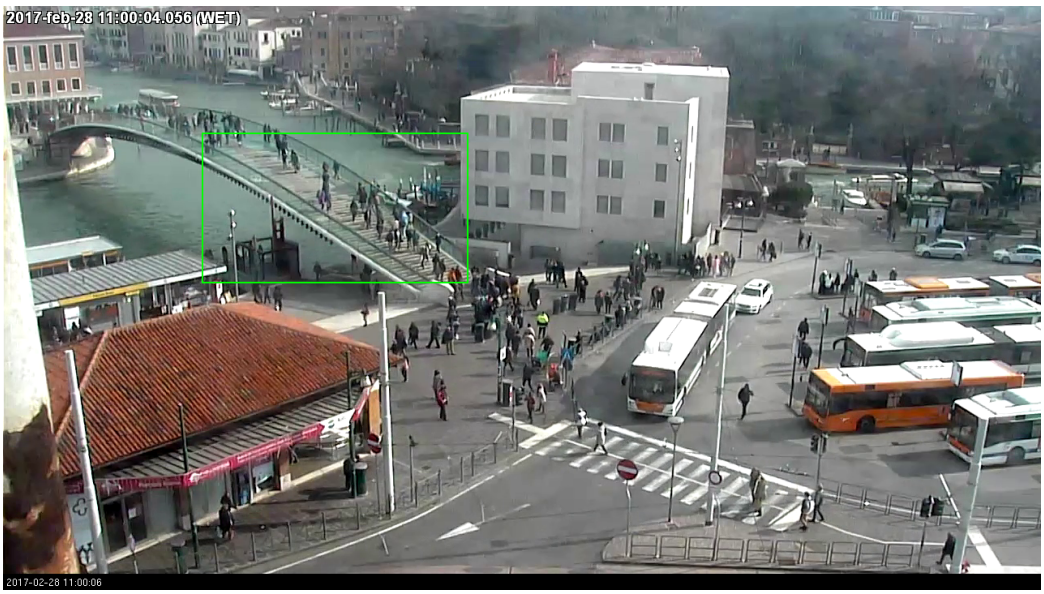


Figure 1.1: A whole image frame, with the useful part of the bridge highlighted in green.

can be worked on is only 327 x 184 pixels. This ultra-low resolution makes it hard to distinguish people as they usually just look like blurry dark shapes.

It should also be noted that in certain parts of the day, such as at night and in the early morning, the image quality is much worse. In the morning the camera is directly exposed to the sun, so light reflections on the protective glass in front of the camera sensor create visible artifacts, while at the same time a large portion of the bridge is shaded by a nearby building. These conditions were deemed to be too harsh for extracting useful information from the video, therefore, without additional work, the proposed solution does not apply to these particular cases.

The camera's low altitude, distant positioning and the low resolution make for a challenging setting. An obvious solution would have been to upgrade the camera, move it closer to the bridge and possibly have an aerial view of it. However, in order to be given the permission to do so by the authorities, one would first have to provide a proof of feasibility. Obtaining

Figure 1.2: A high density situation

this kind of proof would divert too many of the project resources into this particular task, causing other useful data not to be exploited thoroughly. Moreover, pedestrian traffic during a past event had to be analyzed, so acquiring higher quality data would not be possible. In conclusion, the camera setup and the resulting scene could not be chosen, and the solution must work regardless of this suboptimal situation.

1.3.1 Technical Challenges

To sum up the situation described above, there are several challenging aspects of the video sequence that have to be addressed.

- The video resolution is low at 1280 x 720 pixels; this is further worsened by the ROI being only a small part of the whole image.
- The camera is positioned at a side of the passageway, rather than above: this means that people will frequently overlap each other.
- The bridge is often heavily trafficked, so the density of people is high. Because of all of the above, it is very hard to tell people apart if they

are close to each other and wear clothing of similar color. An example of a high density situation can be observed in figure 1.2.

- The typical problems associated with outdoor scenarios, such as lighting changes and weather effects, are present.

1.4 State of the Art

Directionally counting people is a known topic in literature: there are working techniques that can be applied given the right conditions. However, because of the unique set of technical challenges outlined in section 1.3.1, none of the existing solutions was found to be adequate for this situation.

1.4.1 Tracking Blobs

Chen, Chen, and Chen proposed a method to bi-directionally count people passing through a door or gate[3]. They first extract the foreground from the image, then they count the people in each blob. The blob area is used to create a first estimate, which is then refined using color features. They also briefly track the blobs, taking the possibility of splitting and merging into account, in order to derive the direction of each counted individual.

While foreground extraction and area-based counting can be applied to the case at hand, color features and blob tracking would not work in this instance. In fact, the authors consider the behavior of blobs of more than 5 people to be unreliable, but in the scene analyzed in this thesis blobs of more than 15 people have been observed.

Blobs could be successfully analyzed and tracked in [3] mainly in virtue of the low density of people and the advantageous position of the camera, which was placed on top of the gate. This contrasts with the previously described setting.

While color features could potentially provide some useful information when people dress with uncommon colors, a visual inspection revealed that

most people dress with dark colors and that they look quite uniform. A color histogram-based approach was also attempted with the intent of re-identifying people in subsequent frames, but the results were not satisfying for this same reason.

A distant camera setup was considered in [4], where KaewTraKulPong and Bowden successfully tracked multiple low resolution targets. They maintain a background model, which allows them to segment frames by subtracting the background. Connected component analysis is performed on the foreground mask, discarding small areas and adding the rest to the list of objects to be tracked. The objects are then tracked based on motion information, shape and color features. Their tracking algorithm is able to handle partial occlusions, making it useful in real world outdoor scenarios, where security cameras cannot be placed exactly on top of the targets.

Although this tracking method is promising, it heavily relies on being able to initially identify the targets by analyzing the foreground mask generated by background subtraction. Unfortunately, this cannot always be done in the bridge scene. Since the pedestrian density is very high, one foreground blob often represents many people and it potentially includes a large number of partial occlusions, along with a few total ones. Such blobs might include people going both ways, so they cannot be tracked unambiguously. Although the tracking algorithm is able to handle partial occlusions, it cannot do so if the foreground blobs that provide the list of tracked objects are not separated: in the bridge scene at certain times a person would even be able to pass through the whole ROI by staying within one large blob, due to partial and total occlusions (see figure 1.2 for an example of a situation with very large blobs).

For these reasons, this method could not be used successfully for analyzing the pedestrian flow on *Ponte della Costituzione*. However, the idea of discarding small connected components and applying morphological operations to improve the result of background subtraction was eventually adopted for this task, as described in section 2.4.

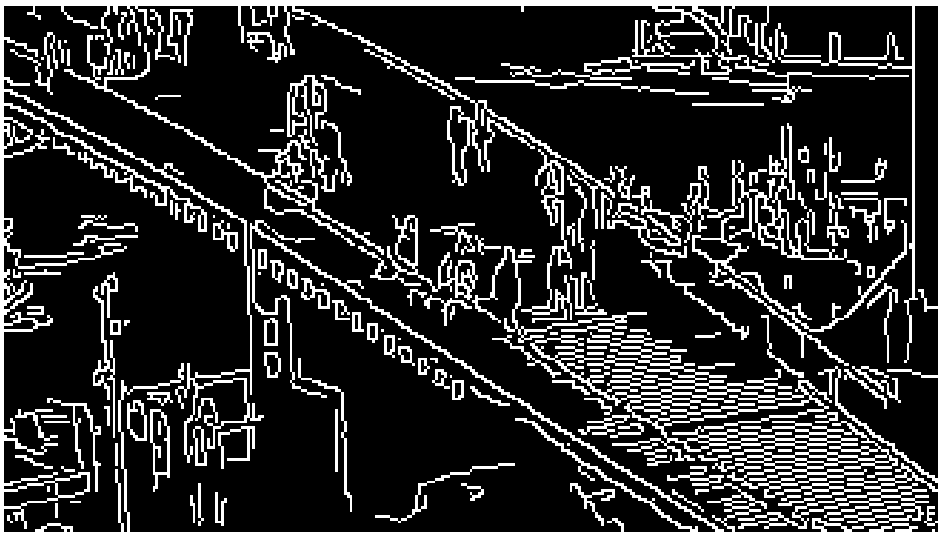


Figure 1.3: Edge map of the visible part of the bridge.

1.4.2 Computing Edge Motion

Bozzoli, Cinque, and Sangineto used a cheap camera to bi-directionally count people passing through bus or train doors[5]. Instead of tracking blobs obtained with standard background subtraction techniques, they chose to maintain a background model of the edges and used it to filter out static elements, allowing them to work on moving edge images. They obtain the necessary information about edge movement direction by computing the optical flow on the Kanade Lukas and Tomasi features[6, 7]. With this information, they were able to estimate the number of people that have passed through a gate by counting the number of movement vectors.

The choice of using gradient images instead of intensity images proved to be useful for the authors, because it works well despite sudden lighting changes. While this is a potentially good approach, application of the standard Canny algorithm[8] (also used in [5]) in the scene of Ponte della Costituzione reveals that a large part of the bridge is characterized by sharp edges even in the absence of people, as can be seen in figure 1.3. This is most likely caused by the bridge steps' sharp changes in color and shadows. Since that region is so rich of edges, using background subtraction only on edges would

not be very different than using it on the whole image. A visual inspection of the edge images sequence also suggests that groups of people would not be easily separated with this method. More importantly, since people in the scene are much smaller than the total field of view of the camera, the computation of the optical flows produces very coarse results, because it requires a tracked object to a have a large internal area, making this approach much less effective.

Overall, this system was deemed not to be applicable to the bridge scene. The key reason is that, even though this solution was built for cheap low quality cameras, in the paper the camera position is very close to the entry point, allowing for a good level of detail of the filmed people, which in turn makes it possible to analyze features and use an optical flow-based approach for estimating movement direction. As described in section 1.3, in the bridge scene the camera is distant and the subjects are not sufficiently detailed for most, if not all, feature-based procedures.

1.4.3 Tracking Heads

In [9], Xu, Lv, and Meng managed to count people by tracking head and shoulders. Their method consists of extracting the image foreground, detecting head and shoulders with a pre-trained Support Vector Machine (SVM) classifier and tracking the detected objects using Kalman filters as described in [10].

This method takes into account both potential high-density situations and non-orthogonal camera tilting. Tracking heads and shoulders, rather than whole bodies, works better in these cases because this part is usually visible even when partial occlusions occur. In very dense crowds this might not work though, because when humans are very concentrated even their shoulders tend to be hidden; in [11] it was shown that detecting only people heads can be effective even in dense crowds. In [12] García et al. proposed a method for directionally counting people by tracking only their heads.

An attempt at tracking people in the bridge scene has been made. Firstly,

the frame was segmented, extracting the foreground. Secondly, blobs of sufficiently small size were selected, in order to detect only one person at a time. These blobs were then tracked using a Kernelized Correlation Filters tracker[13] with fixed size bounding boxes, since the scale does not change very quickly in the selected region of the bridge. This procedure proved to be effective at identifying individuals and tracking them as long as they did not cross other people's paths. The results were visibly unreliable and erroneous in the presence of occlusions, often leading to tracking the wrong person.

In the experiment whole bodies were tracked, rather than only heads, so it would be conceivable that tracking heads might be much more accurate because it has been shown to work in crowded scenarios. A visual inspection of dense crowds crossing the bridge reveals though that it is often impossible to detect heads even for a human viewer, as it is mostly the product of guesswork. Again, this is mainly due to the low level of detail, caused by the region of interest being only a small part of a low resolution image.

Chapter 2

Background Subtraction

The conditions described in the previous chapter are particularly difficult, so most techniques for analyzing the people flow fail. Despite this, it is still possible to process the video sequence in order to separate the foreground areas of the images from the background areas. This task is commonly referred to as background subtraction. The pixels representing a moving object will change values over time, so by comparing them with their previous values the changes can be detected and the area classified as foreground. Background subtraction is a popular topic in computer vision, because it is the basis for other advanced techniques. For example, extracting the foreground from an image can be used as a basis for finding targets that have to be detected or tracked.

For the analysis of the scene of *Ponte della Costituzione*, background subtraction is used to provide the indirected people counts, which are then separated into directed counts by solving a system of equations. It is assumed that people who are walking will be detected as foreground, thus the number of foreground pixels is used to estimate how many people are present. Since in the proposed solution people are counted based solely on the number of foreground pixels, a reliable and accurate background subtraction technique is paramount to the success of the analysis.

This chapter presents the research done for finding a good background

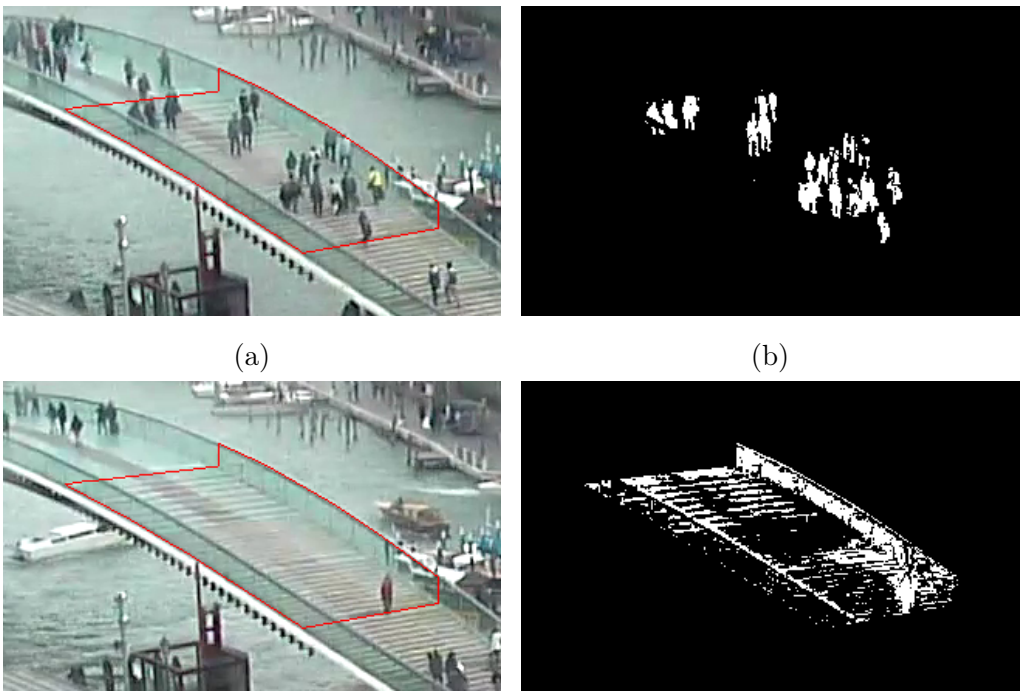
subtraction method, and describes how this task is performed in the final solution.

2.1 Static Background

First of all a model of the background must be obtained, in order to compare it with the target image and detect the variations. The simplest way of performing this task is to use a frame with no foreground objects as reference. The difference between the grayscale version of current frame and the background one is then computed and thresholded. Thresholding the difference image consists in classifying each pixel as having either a higher or lower intensity than a certain value, so the result is a binary image, also known as foreground mask, which represents foreground and background pixels with different values.

On one hand, while this approach could potentially work in a controlled indoor setting, it does not usually fare well in outdoor scenarios. Moving clouds can cause quick lighting changes, shadows can decrease the pixel intensities in specific areas and the weather can cause other undesirable effects such as reflections. On the other hand, a static background does not have the problems associated with a dynamic model, most importantly the potential inclusion of foreground objects in the background model due to slow or static foreground objects. For this reason, an attempt at using a static background model has been made.

Testing static background subtraction over a 10 minutes long sequence resulted in remarkably accurate foreground masks, as can be seen in figures 2.1(a) and 2.1(b). Slow changes in the shape of the shadow projected by a nearby building could be accounted for by using different background images at different hours, thus making this system potentially viable for the analysis of pedestrian flow during most of the day (excluding early morning and night, for the reasons explained in chapter 1). Despite this, it has been observed that a system based on a static background model does not fare well during



(a) (b)

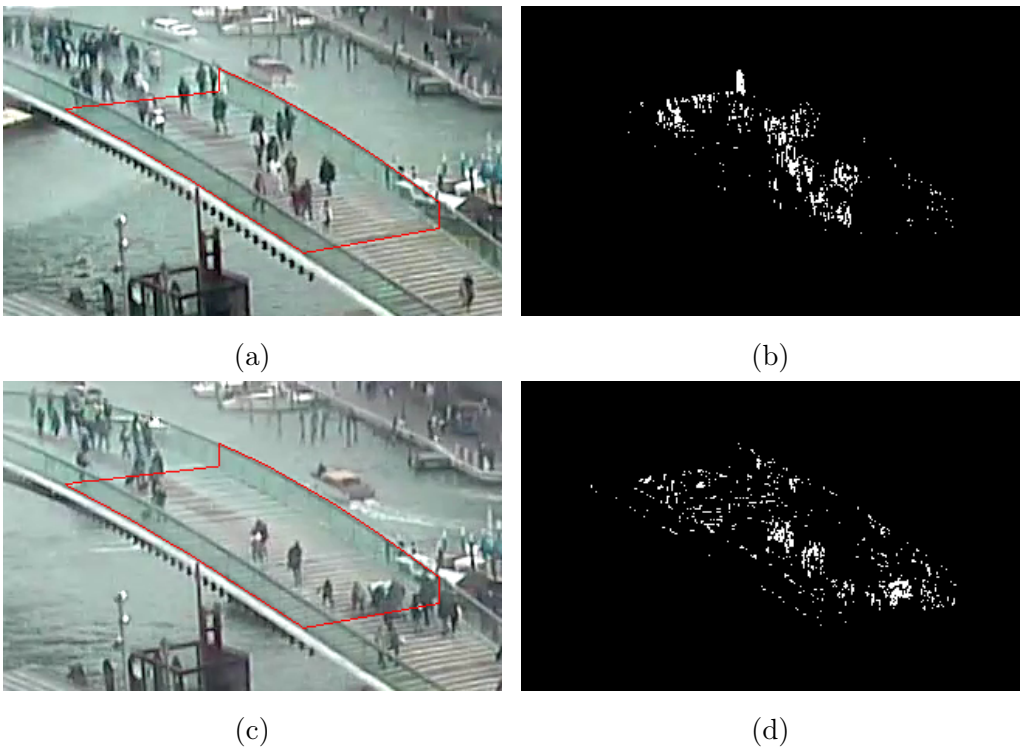
(c) (d)

Figure 2.1: Applying static background subtraction to (a) results in a good foreground mask (b), while applying it to (c) results in a foreground mask with large wrongly detected foreground areas (d). The ROI is highlighted with a red contour.

quick changes in illumination caused by moving clouds. Moreover, the scene naturally changes throughout the day, so using a static image is not a robust method and can lead to wildly inaccurate counts. Figures 2.1(c) 2.1(d) highlights this issue: a large part of background is misclassified as foreground even though at this time the bridge is almost empty.

2.2 Dynamic Background

Since using a static background was proven not to be adequate for the video, it was necessary to maintain an adaptive background model that is updated to reflect the current condition of the scene. A popular and readily



(a) (b)

(c) (d)

Figure 2.2: While applying MOG2 background subtraction to (a) results in a reasonable foreground mask (b), applying it to (c) results in (d), which includes a lot of noise in the foreground and ignores some pedestrians. The ROI is highlighted with a red contour.

available foreground detection algorithm consists in using an adaptive Gaussian mixture model as described in [14, 15]. This method is also included in OpenCV's core framework, where it is known as MOG2 (Mixture of Gaussians 2), which makes it easier to test and configure it. OpenCV is a large library that provides many common computer vision procedures [16], so it was frequently used in this project.

There are many configuration options for this method, and all of them have been taken into consideration when testing it, but the history length, that is the number of frames which the approximation to mixtures of Gaussians is based upon, and the learning rate parameter α, which is a value for regulating how quickly the background model is updated, stand out among

the others. In fact, if the history is too long or α is too low then the model adapts too slowly to the changing scene, thus erroneously detecting groups of pixels as foreground, especially as a result of lighting changes due to passing clouds; on the other hand, if the history is too short or the learning rate is too high then many foreground objects strongly affect the background, resulting in people leaving 'trails' behind them, throwing off the detection process quite badly. It has been observed through several tests that, despite its shortcomings, a longer history is preferable to a short one, as pixels are misclassified much less frequently in this case. This is consistent with the test results reported in [17].

Applying this method to the bridge scene showed that foreground clusters often have 'holes' caused by some pixels being classified as background, and at the same time some isolated pixels in the background are classified as foreground. This situation is clearly visible in figures 2.2(a) and 2.2(b). A workaround for this problem consists in using morphological filtering, which will be discussed in a later section. Opening can be used for deleting stray foreground pixels and closing can be used for closing 'holes' in foreground objects. Even though applying these morphological operators contributes to obtaining more consistent results, there are still many cases of false negatives.

When running this background subtraction algorithm, it was quickly noted that its reliability is particularly sensitive to sudden changes in illumination. While these changes are not easily perceptible by the human eye, they seem to impact the algorithm significantly for brief periods of time. One such moment can be seen in figures 2.2(c) and 2.2(d), where a lot of noise is present in the foreground mask, while some pedestrians are not correctly detected. Even though these situations are relatively 'brief', they can go on for as long as 60 frames or more, which at 15 Hz frame rate corresponds to 4 seconds. Since the algorithm for estimating people flow (described in chapter 3) works at intervals of 90 frames, there is no way to sufficiently mitigate this effect.

Despite its shortcomings, the algorithm based on the mixture of Gaus-

sians is able to create a background image which is a visually believable representation of an empty scene (with no people). For this reason, this method was utilized to create an image used as a static background model for the method described in section 2.1.

2.3 Other Techniques

An extensive testing phase was necessary in order to find a good foreground detection method. A library[18, 19] containing the implementation of more than 40 background subtraction algorithms was used for this purpose. According to [20], some of the top performers should be: LBAdaptiveSOM[21], PBAS[22] and DPWrenGABGS[23].

LBAdaptiveSOM's performance seems to be very poor in the bridge scene: a lot of noise is included in the foreground; lighting changes, especially those due to the automatic camera adjustments, have a strong impact on the detection capabilities of this method, making it mostly useless. As suggested also by other sources[24], PBAS appears to be a very promising background subtraction algorithm in general, but it was found that its recall is rather low in the bridge scene, leaving many pedestrians undetected. Out of the best rated algorithms, DPWrenGABGS offers a somewhat more reasonable performance by usually including most people in the foreground. The problem with this technique is that it appears to be very sensitive to the camera lighting adjustments, resulting in a large quantity of noise and the inclusion of irrelevant areas such as part of the bridge steps into the foreground.

Although not listed here, all of the algorithms included in [18, 19] were tried out. It is also worth noting that all of the techniques mentioned so far are computationally expensive: this has been both reported in [20] and verified in practice.

2.4 Adaptive Median

Since many advanced algorithms for foreground detection were proven to be inaccurate, it has been necessary to fall back to a simple but effective method for maintaining a background model, that here will be referred to as Adaptive Median. This algorithm is a slight variation of the one used by McFarlane and Schofield in [25]. Each frame is converted to grayscale and differenced with a reference image which is updated over time, then the difference image is thresholded.

The reference image is initialized with the image of the empty scene or the first scene of a sequence; at each step every pixel's value is decreased by one unit if the corresponding pixel in the current frame is darker than it, or it is increased by one unit if the corresponding pixel in the current frame is brighter than it. After a sufficient number of frames has been seen, for each pixel value in the reference image half of the updating values are less than it and half of them are greater than it: this means that the reference image effectively converges to the median image of all the frames in the sequence. Only one image needs to be stored and each pixel is updated only once per frame, so this technique is also quite efficient.

In [25], pixels that are detected as foreground are excluded by the background model updating process in order to account for the foreground objects' tendency to stay still for several frames. In this way, the change in lighting has an effect on the reference image but the objects are not gradually incorporated into it. This behavior makes sense because the subjects' movements were constrained by the small room in which they were enclosed and because that simple indoor scenario is not influenced by shadows and other factors, making changes in lighting the only concern. In the case of *Ponte della Costituzione* however, people rarely stop in the ROI, while weather effects and the shadow of a nearby building sometimes cause the scenario to change significantly, thus lowering the risk of incorporating people into the reference image but making it important to include the aforementioned scenario changes in it. For these reasons, in the version implemented for

the analysis of the bridge scene every pixel is updated, regardless of it being classified as foreground or background.

Updating the reference image at every frame means that there are 15 updates per second; this rate is too high and leads to foreground objects leaving 'trails' in the background model and generally corrupting it. In order to avoid this, a sampling rate parameter x has been introduced: the model is updated one time for every x frames read from the video sequence. It has been found that $x = 12$ is good compromise between quickly adapting to environment changes and not incorporating foreground objects into the reference image.

Algorithm 1 shows the pseudo-code of the customized adaptive median algorithm which was eventually developed and used for estimating the people flow. Note that thresholding, represented by the function $Threshold$, will be discussed in the next section.

In general, quick lighting changes do not seem to adversely affect this algorithm as much as the others; in addition, it is able to react to environmental changes such as the appearance and disappearance of the shadows of nearby buildings. The relative success of this background subtraction method in the bridge scene can most likely be attributed to the properties of the median image. Camera lighting adjustments occur very frequently, potentially disrupting the effectiveness of many background modeling techniques, but as long as they balance themselves in terms of being darker or brighter, and assuming their variation is not too high, the median values should provide a good approximation of the background. It should also be noted that this method would not work if the people flow was very intense for a prolonged period of time, because in this case some pixels would represent foreground objects more frequently than the background, leading to the median being an incorrect estimation of the background. However, in all of the video sequences that were inspected this seemed not to be the case. Overall, given the particularly challenging scenario, the adaptive median algorithm proved to perform sufficiently well.

Algorithm 1: Adaptive median algorithm for background subtraction.

input : A bitmap image I at time n, of size $l \times w$;
a model of the background M;
a sampling rate s

output: A foreground mask F

for $i \longleftarrow 1$ **to** l **do**
  for $j \longleftarrow 1$ **to** w **do**
    $D_{ij} \longleftarrow |F_{ij} - M_{ij}|$
    if $n \mod s = 0$ **then**
      if $I_{ij} < M_{ij}$ **then**
        $M_{ij} \longleftarrow M_{ij} - 1$
      else if $I_{ij} > M_{ij}$ **then**
        $M_{ij} \longleftarrow M_{ij} + 1$
      end
    end
  end
end
$F \longleftarrow Threshold(D)$

2.5 Thresholding

Simple image differencing works only if the background can be assumed to be perfectly static: even the slightest change in lighting would cause the difference not to be null in background areas. For this reason the difference image has to be thresholded, so that pixels whose values are relatively similar are classified as background. Let C and B be respectively the current frame and the background model images and let τ be a threshold value, then the foreground mask F at time t can be calculated as follows:

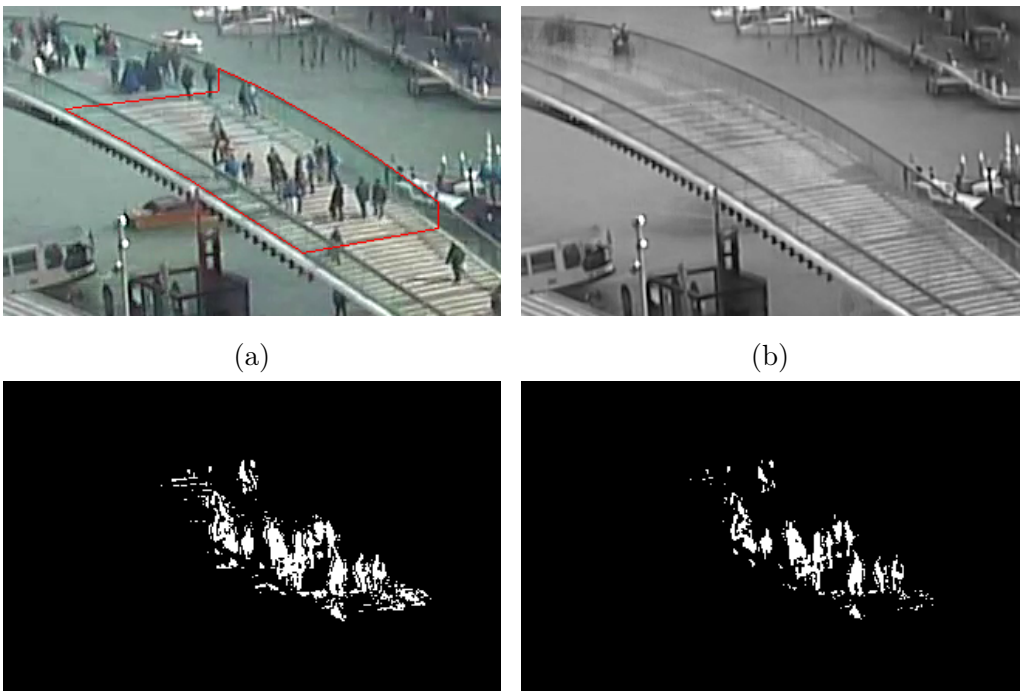
$$F(t) = |P[C(t)] - P[B(t)]| > \tau$$

where P denotes the pixel value of an image. However, thresholding works well only if a good value for τ is used. After trying several threshold values

Figure 2.3: The region used for improved thresholding is delineated in red.

it has been observed that there is not one good value that works well in all circumstances: either the value is too low, causing noise to be included in the foreground, or it is too high, resulting in foreground objects being mistakenly classified as background. Otsu's method[26] was used to work around this: at each step, a threshold is automatically chosen based on the image histogram: if the histogram is bi-modal, an optimal value is calculated.

Otsu's method becomes less effective when the foreground objects area is much smaller than the background area [27]. In the scene of *Ponte della Costituzione*, even if the frame is cropped to the minimum size which allows including all the regions of interest used in the people counting algorithm, the background is still many times as large as the foreground. A solution to this problem can be provided by leveraging the ROIs that are used for the counting algorithm. The total region considered for the histogram is delineated in figure 2.3, and consists of the union of the ROIs used for the algorithm. In other words, Otsu's method is not applied to the histogram of the whole image, but only to the histogram of the region of interest. This way the background area is significantly reduced, the threshold value is more



(a) (b)

(c) (d)

Figure 2.4: Otsu's method is applied to the difference image between the frame (a) and the background model (b). In (c) it operates on the bounding box of the ROI, while in (d) it operates only on the ROI itself.

accurate and the computation is also faster. A comparison of the foreground masks produced by applying Otsu's method to the whole image and to the ROI can be seen in figure 2.4.

2.6 Morphological Operations

Although the adaptive median algorithm proved to be the fastest and most effective among the scores of background subtraction methods that have been tested, the resulting foreground masks still left something to be desired. As can be seen in figure 2.5(c), under certain conditions the bridge steps are detected as foreground, thus forming undesirable thin stripes across a region of the image. More noise is also present in the form of small blobs

or thin stripes surrounding actual people. These problems can be solved or altogether eliminated by applying the right morphological operations, in particular erosion and dilation [28].

If a binary image were represented as a matrix, as most commonly is in typical computer applications, its points or pixels would have values of 0 or 1. An example of such a binary image is represented below:

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In mathematical morphology, a binary image is seen as a set of points belonging to the 2-dimensional set of integers $\mathbb{Z}^2$. These points are the coordinates of the pixels that would be set to 1 in the matrix-based representation of an image. For the sake of simplicity, let us assume that the origin of the coordinates lays at the center of the image. Then, a set $A \subset \mathbb{Z}^2$ that represents the same image as the one above would be $A = \{(-1,2),(1,2),(-1,1),(1,1),(-1,-1),(0,-1),(1,-1),(-2,-2),(2,-2)\}$.

Before defining the fundamental morphological operators, it is necessary to introduce the definition of reflected set and translated set. The reflected set of a set B, denoted as $\hat{B}$, is defined as

$$\hat{B} = \{w | w = -b, \quad b \in B\}$$

In other words, it is the reflection of a set with respect to the origin.

The translation of a set B by a point $z \in \mathbb{Z}^2$, indicated as $(B)_z$, is defined as

$$(B)_z = \{c | c = b + z, \quad b \in B\}$$

that is, all the elements of the original set are shifted by a certain quantity z.

Morphological operations are based on structuring elements: small sets or images that are used to explore a larger image, looking for particular

properties. The basic operations are erosion and dilation, while all the others can be formulated as a combination of these two.

2.6.1 Erosion

Let us consider two sets A and B in $\mathbb{Z}^2$ that represent binary images; the erosion of A with B, denoted by $A \ominus B$, is defined as such:

$$A \ominus B = \{z|(B)_z \subseteq A\}$$

This means that the erosion of A with B is the set of points z such that B translated by z is entirely contained in A. In this instance, B is a structuring element, and in most practical cases is chosen as being much smaller than A. In practice, eroding an image consists in sliding the structuring element over it, and selecting only the points for which all of the points in the structuring element match those in the image. If the image is represented as a matrix, such points can be denoted in the result by the value 1, while all the others are set to 0. The operation can be viewed as a logical *and* operation performed at each translating step: the result is true if the local subset of A matches B. If the structuring element is symmetric, eroding an image has the effect of shrinking it, because elements next or closed to the 'borders' with the background are effectively removed.

2.6.2 Dilation

The dilation of A with B, with A and B in $\mathbb{Z}^2$, is defined as

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \emptyset\}$$

that is the set of all shifts z of $\hat{B}$ such that $\hat{B}$ and A have at least one element in common. In practice, dilating an image consists in sliding the structuring element over it, and selecting only the points for which at least one of the points in the structuring element is contained in the image. As for the erosion, if the image is represented as a matrix, such points can

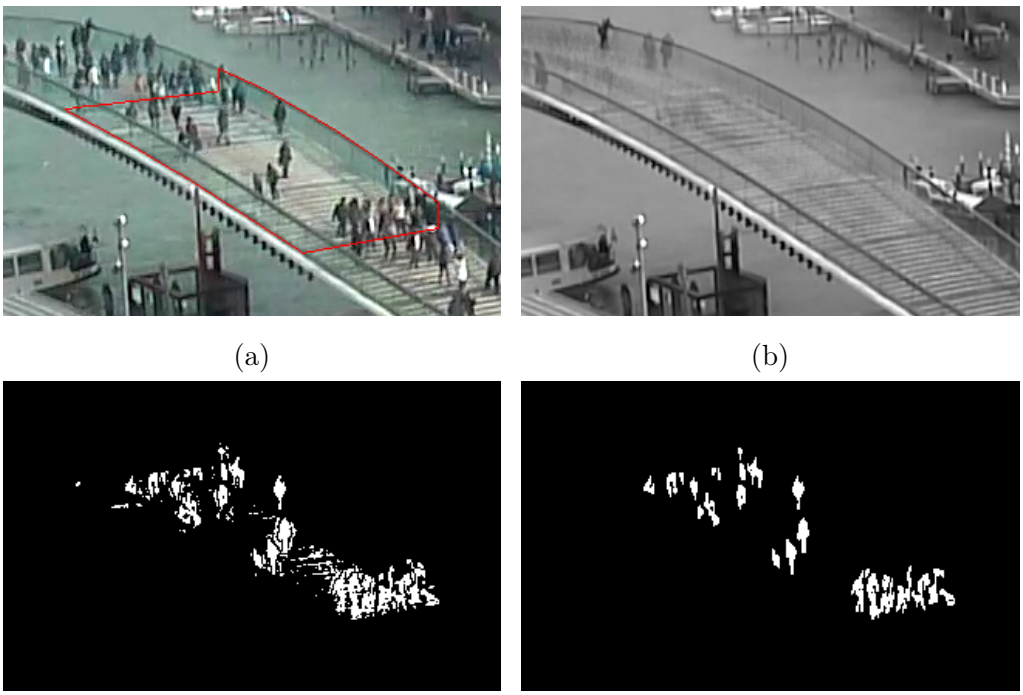
be denoted in the result by the value 1, while all the others are set to 0. The operation can be viewed as a logical *or* operation performed at each translating step: the result is true if at least one of all the elements matches. If the structuring element is symmetric, dilating an image has the effect of expanding it, because the background regions next or closed to the elements of the image are included in the result as foreground.

2.6.3 Foreground Mask Improvement

Figure 2.5(c) shows that most misdetected foreground areas are sufficiently thin to be removed by an erosion operation without deleting the actual foreground objects. Since the stripes are mostly horizontal, a vertical shaped structuring element was used:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

The structuring element is large enough to remove the noise which is erroneously detected as foreground, while its vertical shape tends not to damage actual foreground objects, because people have vertical shapes too, so they are tall enough to survive the erosion. It is also worth noting that this operation has the added benefit of removing most of the shadows projected from pedestrians, which look horizontal because of how the sun is oriented relative to the bridge and the camera. In the next step connected components are grouped and analyzed: if their size is less than 2 pixels they are most likely just noise or some bridge steps bits left over by the erosion, so they are discarded. Connected component filtering is not a morphological operation, but it is nonetheless useful. At this point, the stripes have been removed and as an added benefit the noise that sometimes surrounds the detected objects has also been eliminated. The remaining foreground areas are grown back with a dilation operation which uses the same structuring element previously used in the erosion operation. Figure 2.5(d) shows the resulting foreground mask.



(a) (b)

(c) (d)

Figure 2.5: The adaptive median algorithm results in the background model (b) for the challenging frame(a). The corresponding foreground mask(c) is improved with morphological and connected component filtering (d).

In conclusion it can be said that applying the adaptive median background subtraction technique, thresholding the foreground mask and correcting it with some morphological operations results in a satisfactory segmentation of the image. This is the basis for the algorithm that is able to estimate the pedestrian flow on the bridge, described in the next chapter.

Chapter 3

Algorithm

Analyzing the bridge scene is particularly challenging, making known approaches not completely applicable to this situation. In such harsh conditions, a high degree of accuracy would not be expected. Nonetheless, a semi-automatic working solution with sufficiently good accuracy has been developed and implemented.

This chapter first presents the concept of the solution, then goes on to describe in detail all of its components. The complexity of the algorithm is then analyzed, and finally the implementation choices are discussed.

3.1 Concept

The proposed solution consists in directionally counting people at regular time intervals, using only simple motion information such as the number of foreground pixels. Despite the challenging scenario, some background subtraction methods are still able to function properly. This task, described in the last chapter, basically consists in identifying which pixels belong to the background and which ones belong to foreground objects.

Ideally, the directed counts could be obtained by tracking individuals or groups of people. However, in the scenario that had to be analyzed all tracking techniques failed. The high density of pedestrians means that foreground

blobs could occasionally include a massive number of people walking in different directions, so tracking blobs does not produce meaningful results. Using optical flow information in order to estimate the pedestrian count based on their movement would have been another possibility, but optical flow calculation is known to be very inaccurate when the areas of moving bodies is small: this is unfortunately the case with the target scene, as the bridge (or the ROI) represents only a small fraction of the total camera field of view, causing people crossing it to appear even smaller.

Because of the inability to track individuals, it was decided to rely only background subtraction for counting purposes. By counting the number of foreground pixels it is possible to estimate the number of pedestrians present in the scene; in this sense this approach is similar to the one presented in [3], however unlike in [3] the color information is not sufficient to help in refining the estimate, so it has been ignored.

By detecting the foreground pixels it is possible to approximately count how many people are present at a certain instant of time. However, this count does not provide any information on the direction in which pedestrians are walking; since the final goal consists in counting how many people crossed the bridge in each direction, this is not enough. In order to identify the direction, it was necessary to develop a differential system that requires the user to input the initial directed count: based on this, by making some assumptions on the speed of pedestrians, the system is able to determine the direction of the counts. The bridge area is divided into multiple consecutive ROIs and people inside them are counted; the direction is calculated by observing how the quantities vary among the defined regions. This calculation is performed at successive regular intervals by solving a system of linear equations that models the situation.

3.1.1 Algorithm Steps

Let us see a more detailed scheme of how the whole algorithm works. At first the algorithm goes through an initialization phase, which includes

processing some parameters. The algorithm requires the user to provide, among other things, these fundamental items before running:

- A region map file that identifies the regions of interest. The regions are used for counting and for determining the direction of people. The number of regions is a user's choice, but it must be at least two.
- A pixel-people mapping file created by manually counting people for a set of frames. The system performs a linear regression on this data, obtaining a function that can estimate the number of people based on the number of foreground pixels.
- A directed people count (how many people are going in one direction or the other) for each region in the starting frame. This is used to initialize the system in order to solve the linear system for the first time.

After initialization, these are the main steps that are iterated until the end of the sequence:

1. Advance by a fixed number of frames, thus reaching the next counting instant.
2. Detect the foreground, creating a foreground map. Update the background model.
3. Compute the number of foreground pixels in the current frame.
4. Map the number of foreground pixels to the number of people in each region.
5. Solve the linear system, based on the previous solution and the newly computed people count, in order to find out how many new people entered the scene and from which side.
6. Store the new solution for use in the following iteration.

7. Store the partial results, that is the number of people that entered the scene from each side, along with the frame number, in a results file.

Eventually, the results file will be filled with directed counts coupled with the time at which the pedestrians were detected. The counts are added in chunks relative to the time interval (90 frames, or 6 seconds), so the time precision is limited by the interval length. This is not a concern, because the precision required for studying the pedestrian flow is in the order of several minutes.

The sequence above sums up the algorithm steps well enough, but it leaves out some details. In particular, instead of counting the number of people for each region only in one frame (before moving on to the next time instant), the operation is performed on multiple frames, then the average is computed and used for solving the linear system. This is helpful because in this way the risk of counting the wrong number of people due to instantaneous and momentary problems such as sudden noise or occlusions is lessened.

The frames considered in this calculation are taken before and after the reference time that is used in the linear system. It was thought that the average count for a total of 5 frames would be a good compromise between the error reduction and the risk of having people move into the wrong region during this interval; it is a reasonable value considering that the interval between solving the linear system is 90 frames. Experimental results showed that this was indeed a good choice.

3.2 Linear System

At each step, a linear system must be solved to get the directed counts, which are the goal of the whole analysis process, so well defined equations are crucial to the overall accuracy of the algorithm. The linear system will now be illustrated.

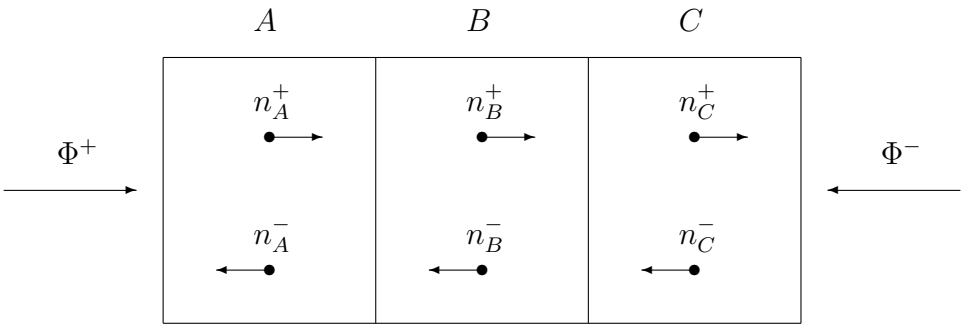


Figure 3.1: Naming scheme for the equations. A, B, C denote the three regions; $n_x^{\pm}$ are the directed counts for each region x; Φ^+ and Φ^- are the two flows.

3.2.1 Three Regions

In the following, directions are denoted by the + and − signs, such that + can be interpreted as the direction going from left to right and − can be interpreted as the direction going from right to left. The bridge is divided into three consecutive ROIs of equal physical size. Let A, B, C be three such regions. Let $n_x(t)$ be the people count at time t in region x. Each individual is assumed to walk at a constant velocity v^+ or v^-, depending on his direction of movement. We denote by $\Phi^{\pm}$ the incoming flows from left and right in the considered area. This situation is depicted in figure 3.1. In a time interval Δt, $n_A^+(t)p(t) \quad (0 \leq p(t) \leq 1)$ is the number of individuals that move from region A to B after a time Δt; $p(t)$ is the probability that people managed to cross the region boundary within that time. One can then express this

situation with the following equations:

$$
\begin{aligned}
n_A^+(t+\Delta t) &= n_A^+(t)(1-p(t)) + \Phi^+(t+\Delta t)\Delta t \\
n_B^+(t+\Delta t) &= n_B^+(t)(1-p(t)) + n_A^+(t)p(t) \\
n_C^+(t+\Delta t) &= n_C^+(t)(1-p(t)) + n_B^+(t)p(t) \\
n_A^-(t+\Delta t) &= n_A^-(t)(1-p(t)) + n_B^-(t)p(t) \\
n_B^-(t+\Delta t) &= n_B^-(t)(1-p(t)) + n_C^-(t)p(t) \\
n_C^-(t+\Delta t) &= n_C^-(t)(1-p(t)) + \Phi^-(t+\Delta t)\Delta t
\end{aligned}
$$

The equations mean that people who were seen walking towards a certain direction in a particular region at the previous time point are now (Δt time units later) expected either to have moved to the next region or to have stayed in the same region. The situation in the two regions at the extremes is slightly different, because new people may have entered the scene from either side: these quantities, which are a consequence of the flow, are $\Phi^+(t+\Delta t)\Delta t$ and $\Phi^-(t+\Delta t)\Delta t$.

Now one can add three more equations, which mean that the directed counts must be consistent with the undirected counts obtained by detecting the foreground:

$$
\begin{aligned}
n_A(t+\Delta t) &= n_A^+(t+\Delta t) + n_A^-(t+\Delta t) \\
n_B(t+\Delta t) &= n_B^+(t+\Delta t) + n_B^-(t+\Delta t) \\
n_C(t+\Delta t) &= n_C^+(t+\Delta t) + n_C^-(t+\Delta t)
\end{aligned}
$$

In other words, people seen in a certain ROI are the sum of people walking in either direction in that ROI.

Using the equations above, one can get a system of nine equations with nine unknown quantities: $n_{A,B,C}^{\pm}(t+\Delta t)$, $\Phi^{\pm}(t)$ and $p(t)$. The other variables are known: the directed counts at the previous time step $n_{A,B,C}^{\pm}(t)$ and the current undirected counts $n_{A,B,C}(t+\Delta t)$. The system can be solved step by step by knowing the initial counts $n_{A,B,C}^{\pm}(0)$, that have to be manually obtained. At each step $n_{A,B,C}(t)$ must be measured somehow. In the proposed solution, they are computed by counting the number of foreground

pixels obtained by subtracting the background from the current frame and converting it into a corresponding number of pedestrians.

The probability $p(t)$ can be related the pedestrian velocity:

$$p(t) = \frac{v(t)\Delta t}{L} \tag{3.1}$$

where L is the length of a region.

3.2.2 Two Regions

The setup with a variable p, which is dependent on the time t, acknowledges the fact that the speed of people might vary throughout the day. However, running some tests on a sample video sequence showed that in practice the calculated value of p varies wildly between successive time steps. This behavior is unreasonable, because p is not expected to change so drastically in the span of a few seconds. Moreover, the solution of the system more often than not attributes values outside the (0, 1) range associated with a probability; this problem could be somewhat alleviated by truncating the value of p, thus forcing it to be in the expected range, but the meaning of a such a probability parameter would be unclear, and the system would not be any more stable.

All of these problems led to a simplification of the system: rather than considering a variable p, it could be fixed beforehand to a reasonable value. Assuming that the speed of pedestrians stays constant throughout the day does not cause a great loss of information: it is not expected to vary too much, so studying it was not considered interesting. A fixed probability value also grants another benefit: rather than assuming the people walking in different directions to have the same speed, it can be split into two different values dependent on the direction of movement. Note that this capability is especially important when considering a scene with a bridge (or a sloped road), where walking uphill is harder, thus slower, than walking downhill. Instead of $p(t)$, one would have p^+ and p^-. In this case, the equation 3.1 can

be rewritten as

$$p^{\pm} = \frac{v^{\pm}\Delta t}{L} \tag{3.2}$$

that is, $p^{\pm}$ can be bound to the pedestrian velocities $v^{\pm}$ and the length L of each region.

The lack of a variable causes another problem though: the system which was described previously would now be overdetermined, having nine equations and eight variables. Nonetheless, an approximated solution can still be found, for example by using the least squares method*; tests have shown that indeed such a setup produces much better results. However, the uncertainty can be removed completely by removing one of the ROIs. Using the same notation as before and only two regions A and B one would get the following system of equations:

$$\begin{cases} n_A^+(t+\Delta t) = n_A^+(t)(1-p^+) + \Phi^+(t+\Delta t)\Delta t \\ n_B^+(t+\Delta t) = n_B^+(t)(1-p^+) + n_A^+(t)p^+ \\ n_A^-(t+\Delta t) = n_A^-(t)(1-p^-) + n_B^-(t)p^- \\ n_B^-(t+\Delta t) = n_B^-(t)(1-p^-) + \Phi^-(t+\Delta t)\Delta t \\ n_A(t+\Delta t) = n_A^+(t+\Delta t) + n_A^-(t+\Delta t) \\ n_B(t+\Delta t) = n_B^+(t+\Delta t) + n_B^-(t+\Delta t) \end{cases} \tag{3.3}$$

which has six equations and six variables, so it is well determined.

The algorithm was run on a video of approximately 75 minutes using three and two regions, to compare the two solutions with the previously generated ground truth data. In the system with three regions, p^+ and p^- are fixed, so it is overdetermined, while the system with two regions is not. The results of this experiment are shown in figure 3.2, which depicts the flows from the two sides and their absolute error in relation to the ground truth. The characteristics of the ground truth and the method employed for processing the results are explained in great detail in chapter 4, which

*Given a linear system $ax = b$, with a and b known, the least squares method finds a solution that minimizes the squared Euclidean norm $||b - ax||^2$.

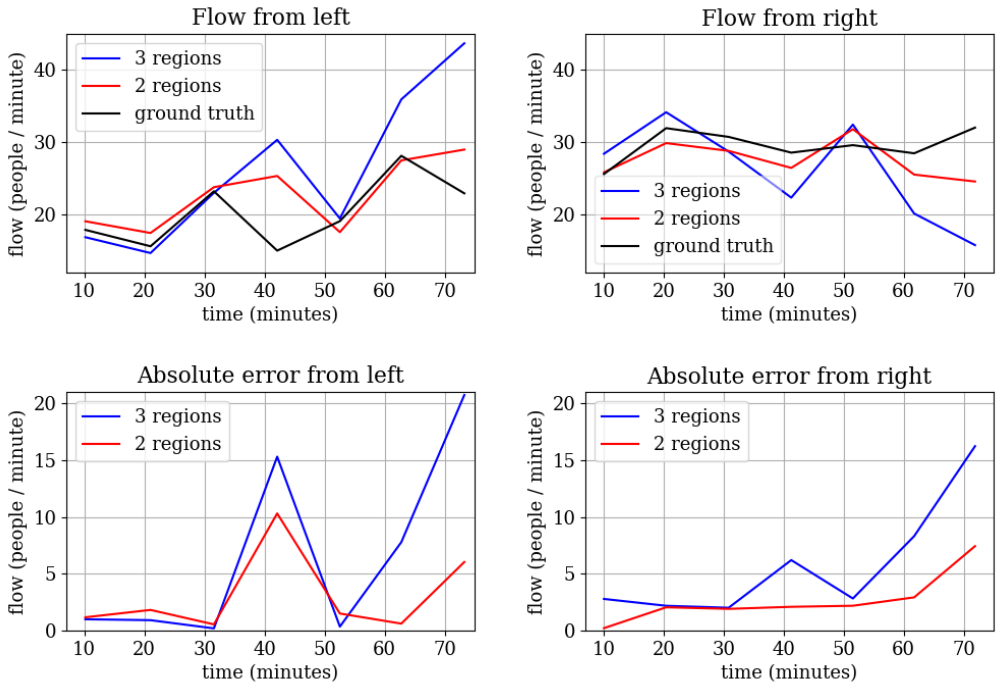


Figure 3.2: Comparison between results obtained using two and three regions, with fixed probabilities $p^{\pm}$. Using only two regions clearly yields more accurate results.

includes more results. In short, the flow values reported at a certain time are computed by summing the counts collected during the last time interval and dividing by the length of the interval (approximately 10 minutes). This means that there is no information on the flow value at time 0.

The test results have clearly shown that the accuracy of the system which uses only two regions is superior, so it was eventually selected for analyzing the scene of *Ponte della Costituzione*.

3.3 Assumptions

Since the proposed solution is based on pedestrian average speed, there are thee implicit assumptions:

1. People crossing the bridge do not go back and reverse their path.

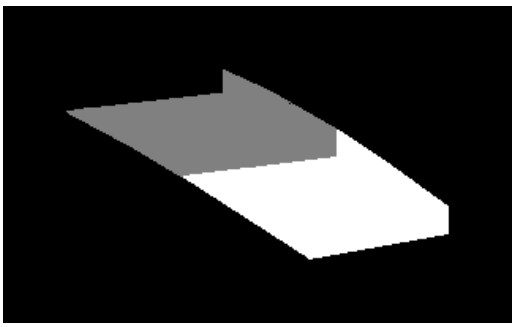
2. People do not stop for long periods of time.

3. The walking speed does not vary too much throughout the day.

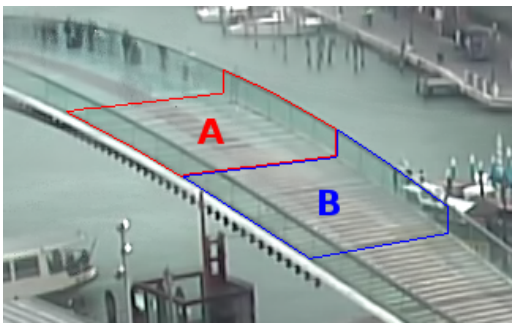
If these were not true, the average speed would be too rough an estimate of the real speed. However, it is worth noting that even though some exceptions can occur, they should not completely throw off the system, so there is a certain degree of tolerance. These cases must just be sufficiently infrequent for the system to be stable, and this has been observed to be true.

In the specific case of *Ponte della Costituzione*, the velocity v of a person has been initially taken as being about 1.2 m/s, based on known pedestrian speed measurements [29]. The length L of each region is 10 m and the time interval Δt was chosen to be 90 frames, which is equal to 6 s since the video has a 15 Hz frequency. The value of 6 s for Δt was chosen such that, based on a reasonable estimate of v (1.2 m/s, as previously said) pedestrians have enough time to move from one region to the next one, but generally not enough to get to the one after the next. Knowing Δt and v, a first estimate of the probability p for either direction was calculated by solving equation 3.2, which gives a value of 0.72. This initial value was later refined by comparing the counting algorithm results.

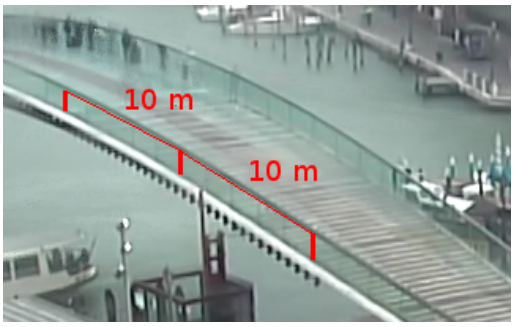
Eventually, it has been found that 0.66 and 0.43 are good values for p^+ and p^- respectively. This makes sense, as the value for pedestrians going up was expected to be lower than that for the pedestrians going down ($+$ denotes the left-to-right direction, which people go in by descending the bridge steps, while $-$ denotes the opposite). Moreover, it has been observed that pedestrians who cross the bridge are sometimes holding a suitcase, which slows them down considerably because they are forced to climb the steps either up or down.



(a) Grayscale region map image.



(b) Frame with highlighted regions A and B.



(c) Regions are 5 panels long.

Figure 3.3: Two regions were defined. The regions were selected so that they are of equal length, and troublesome areas have been avoided.

3.4 Regions of Interest

The visible part of the bridge surface is divided into at least two adjacent consecutive ROIs. The different regions are used for solving the linear system 3.3. At each step, an estimate of the number of people in each region is created. Counting people in a ROI is significantly easier than tracking them, and it has been proven to work even in lowly detailed scenes such as the one to be analyzed.

The regions of interest were manually defined with the intent of creating areas of equal physical length along the bridge's longitudinal axis. The length has been estimated by counting the glass panels which compose the railings, since the glass panels are known to be of equal size (2 m long). Each region

comprises 5 panels, so it is 10 m long (see figure 3.3(c)). The regions' location has been chosen so that people are clearly visible while crossing them, therefore distant places such as the center of the bridge have been ignored. It was also observed that tourists have a preference for standing in the central area of the bridge: they often lean over the railings and look at the scenery. This behavior is undesirable because the system assumes that pedestrians do not stop very often, so it is one more reason to avoid extending the ROIs over the central part. Similarly, the entrance of the bridge was not included in the ROIs because the steps cause problems to the background subtraction algorithm, although this area can be used for defining a third ROI if needed.

The regions are specified through a gray-scale image in which the background is black and each region has pixels of a different intensity. The intensity values are also used for ordering, meaning that the leftmost region has the lowest intensity while each additional region has a greater value. Figure 3.3(a) shows the grayscale image representing the ROI; figure 3.3(b) highlights the regions in a video frame.

3.5 People Count Regression

Background subtraction produces a foreground mask, that is an image of the same size as the original image, in which foreground pixels are white and background pixels are black. This information alone can be used to provide an estimate of the number of people present in the scene. For example, in [3] each foreground blob was analyzed separately on the basis of the number of pixels. Chen, Chen, and Chen suggested that blobs representing only one person occupied a typical area, but each additional person included in the blob contributed a smaller number of pixels. In the case studied in the paper this behavior can be explained by the partial occlusions caused by people that walk in close groups. The authors managed to analyze blobs of up to 5 people in this way. A similar principle was applied to the analysis of the foreground masks produced in the scene of *Ponte della Costituzione*, but

there are some important differences that must be accounted for.

First, in this case the camera is positioned at the side of pedestrians, rather than on top of the passageway, so the occlusions play a more important role and can also be total, meaning that people could be hidden behind other people at any time. Second, the camera is distant from the region of interest and the pedestrian density is at times very high, meaning that foreground blobs can potentially include a large number of people walking in both directions. Third, the pedestrian count must be used for solving the linear system (see 3.2), which means that the counts must be eventually attributed to each region of interest (see 3.4) separately; in other words, it is necessary to count how many people are present in each region individually. All of this lead to not counting people per blob; instead, people are counted on the basis of the number of foreground pixels contained in each region. If counting was performed on a per blob basis it would be difficult to split the count when a blob occupies more than one region. Moreover, since blobs can reach massive sizes at times, analyzing them separately would not really add any benefit.

Let $n_{A,B}$ be the undirected counts for the regions A, B and let F be a foreground mask. We denote by $F[A]$ the foreground mask restricted at region A. Then, $n_{A,B}$ is calculated as follows:

$$n_i = f_i(|F[i]|) \quad i = A, B$$

Therefore a function f_i for each region i is needed to convert the number of foreground pixels to the number of people. It was decided to compute this conversion differently for each region because this allows for a better accuracy, overcoming the problems associated with perspective: people occupy a smaller area of the image plane when they are farther from the camera, so it can be assumed that performing a different conversion based on distance results in a better estimate. Moreover, this method allows using different techniques in the foreground mask optimization, because each ROI operates independently. Counting on a per region basis introduces a problem though: if a person is crossing a region boundary, he/she cannot be counted as one in

either region. This is easily solved by converting the number of foreground pixels to a fractional number of people.

The two functions were built automatically by performing a linear regression on a manually generated set of pixel-people associations. The data was created by manually analyzing 150 frames belonging to a 10 minutes video. At 15 Hz, a 10 minutes video corresponds to a total of 9000 images, so the sample frames were taken at 60 frames apart from each other. A comma-separated text file was automatically generated.

The file is organized in rows, where each row corresponds to a frame. The frame number is listed, followed by the foreground pixel count of each region as computed by running the adaptive median background subtraction algorithm (see section 2.4). Successively, the file was manually completed by adding the actual people counts corresponding to the the pixel counts previously reported. In order to fill the file, the frames had to be manually inspected: this was done by advancing the video and automatically stopping every 60 frames. The choice to scan the video rather than inspect a static set of previously saved frames is due to the fact that in this way it is possible for a human viewer to detect even total occlusions, thus allowing to count also the completely hidden pedestrians in a frame. This behavior is desirable because by taking possibly hidden people into account, it can be assumed that on average the estimated people count will be closer to its real value.

The pixel-people association file needs to be generated only once for a scenario. At the directed people count analysis initialization phase, the file is read into memory and all information on frame numbers is discarded, as it is only useful for users who need to manipulate the file. The frame numbers also allow for reusing the same user collected data with a different background subtraction technique; should the need arise, the associations can be automatically modified by substituting the number of foreground pixels but leaving the people counts unaltered for each frame.

After discarding the frame numbers, association data is grouped based on region and fitted to create the conversion functions. Fitting is performed by

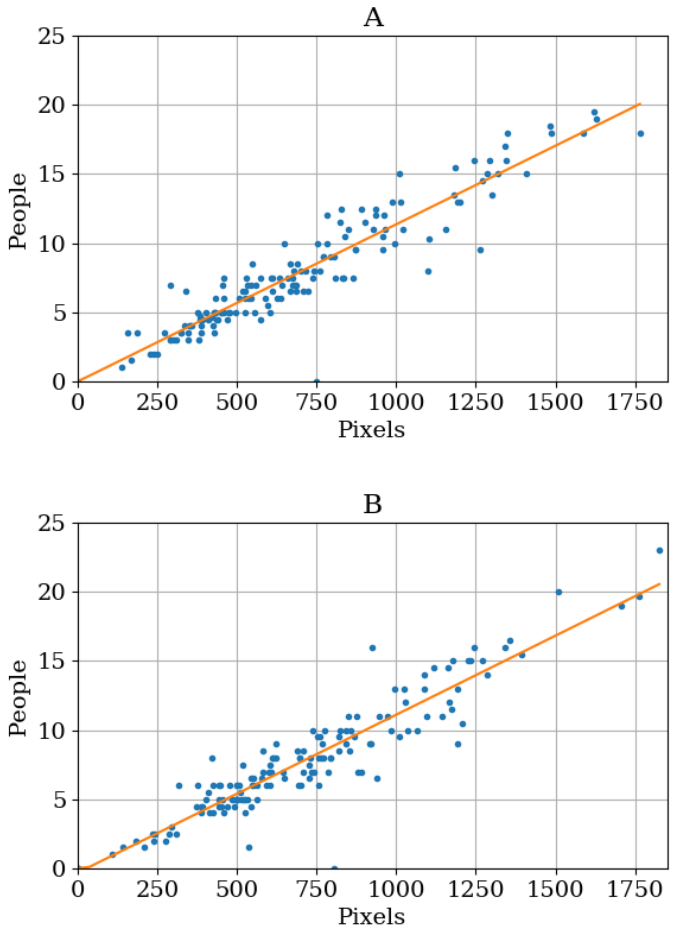


Figure 3.4: Linear least squares polynomial fitting. Empirically obtained associations are represented by blue dots, while the fitted line is represented by an orange line.

using the least squares method on the data. Figure 3.4 shows the pixel-people association data and the corresponding fitted line. Initially the data was fitted by a polynomial of degree 2, but due to the low number of observations in the higher ranges (i.e. when there are more than about 15 people in the same region) some fitted polynomials used to be concave, which is not expected: with a higher concentration of people occlusions should usually be more numerous, leading to a lower average number of pixels per person. For

this reason, it was decided that a polynomial of rank 1 would be a better representation of data, especially in cases of high density. The lines depicted in figure 3.4 have a very similar inclination, but the one corresponding to the rightmost region (denoted by B) is actually slightly more inclined, which means that each person is composed of a larger number of pixels. This was expected, because the rightmost region is also the closest to the viewer.

3.6 Complexity

Although speed was not the main goal of the project, the proposed algorithm is quite fast, provided that the implementation is efficient. This will now be proven by analyzing the the complexity of the new algorithm.

Let n and m be respectively the height and width of a single frame, and let ℓ be the number of frames in a video, that is its duration. The total size of the input video is therefore $n \times m \times \ell$. It will be shown then that the asymptotic computational complexity of the algorithm is $O(nm\ell)$, while its space complexity is $O(nm)$.

3.6.1 Time

Decoding the file requires some computation which depends on the format employed, but since this step is independent of the proposed algorithm for estimating the people flow, it can be ignored when calculating the complexity. Besides, the format which is used by the security cameras does not include any compression mechanisms.

Before starting the main loop, which involves solving the linear system and actually counting people, all parameters must be loaded and some of them must be further processed. In particular, a pre-generated set of pixel-people associations must be fitted in order to find a function that is able to convert the number of foreground pixels in each ROI to an estimate of the number of people present therein. This step depends on the number of observations that must be fitted; if such a number was very high, the

complexity might depend on it. In the case that was studied, only 150 associations were provided, which are orders of magnitude fewer with respect to the video size. If this were not the case, the required fitting function could be precomputed and directly given as input, making the process less expensive if the same scene had to be analyzed several times. The complexity of initializing other parameters is limited by the size of a single frame, that is nm; for example loading the ROI and creating their respective boolean masks requires that each pixel of the region map image is processed.

Once the main loop is entered, there are some steps that are repeated for each frame, that is ℓ times. However, even though each frame must be read, many of them are actually discarded by the algorithm, without further processing them. This happens because, as previously explained, people are counted only after a predefined period of time, such that they have had a chance to move from one ROI to the next. The scene is then analyzed multiple times consecutively in order to increase the accuracy of the estimate. This means that, for a 15 Hz video and a 6 s period, if the average people count is computed over 5 consecutive counts, 85 frames are discarded for every 90 frames read. Despite this, the actual rate of processed frames could be higher depending on the input configuration, because of how the background subtraction method, necessary for computing the foreground mask and thus for counting people, operates. In fact, in order to adapt to various changes in the scene, the background model has to be regularly updated. It was found that for the scene of *Ponte della Costituzione* it was sufficient to update the model only once every 12 frames.

Even though discarding frames clearly reduces the overall processing time, it should be noted that an effective sampling rate parameter for the task of background subtraction would never have a high value, at least for an outdoor scenario. For example, in a 15 Hz video a sampling rate of 30 would already make the background model not reactive to lighting changes, in addition to increasing the chance of including foreground objects in it. Since this parameter's value would be limited in practice, it can safely be assumed to

have a constant value, thus not influencing the asymptotic complexity. A similar approach can be applied to the time interval parameter: it would be reasonable to assume that the ROI length would not exceed a certain value, let us say 30 meters, in a real life scenario; the time period would be limited by this value, since the pedestrian speed is also limited.

Updating the background model and detecting the foreground require a constant amount of operations for each pixel. The particular technique employed, namely adaptive median, consists of several steps: converting the current frame to grayscale, increasing or decreasing the intensity of each pixel of the model, computing the difference between background and current frame, applying Otsu's method to obtain a binary image, and performing a few morphological operations. All of these steps have a complexity of $O(nm)$, the number of steps is constant, and they are repeated approximately ℓ times, so the algorithm complexity for the whole video cannot be lower than $O(nm\ell)$ because of the background subtraction task.

Once the number of foreground pixels has been obtained, it must be converted to a number of people for each region. This is done by calling a different function for each of them, created by fitting manually created association data; each function corresponds to a polynomial of degree 1, that is a line, so it requires a constant amount of operations. The people counts are then used for solving the linear system with the least square method, a procedure that depends exclusively on the number of variables and equations. While the system could be more complicated than the ones described in section 3.2, in practice the number of equations q would always be much smaller than the size of a frame: $q \ll nm$, so the total complexity of the algorithm is still determined by the background subtraction operations.

With all things considered, after the initialization phase, the algorithm's time complexity depends on the background subtraction procedure, which is $O(nm\ell)$. It can therefore be concluded that the algorithm has a linear complexity.

3.6.2 Space

At any time, the algorithm requires only a constant number of matrices of size nm to be stored in memory.

The region map file is loaded and elaborated only once during the initialization phase: at this stage r masks are created, where r is the number of regions. These masks are used to extract only a ROI from a frame; in particular, they allow the foreground mask to be easily split into regions, in order to count the number of foreground pixels separately for each of them. It follows that storing the masks increases the space complexity to $O(nmr)$. It should be noted though that r is always a very small number in practice, and could safely be assumed to be limited by a small constant, so the actual space complexity is effectively $O(nm)$.

In the main loop, after loading an input frame, it is converted into grayscale and the original is discarded. This grayscale frame is compared with the background model, also of size nm. The background subtraction algorithm maintains only one image in memory: the model's pixels are calculated from their history, which does not need to be stored explicitly. The difference image between the current frame and the model (also of size nm) is then thresholded, without requiring additional memory.

When the numbers of foreground pixels have been computed for each region, they are converted to their corresponding number of people, requiring only a total of r entries. The linear system must then be solved: it requires enough space to store the coefficient matrix and the known terms matrix, both of which have constant size. The solution of the system is kept in memory only until the next iteration.

In conclusion, the space complexity is dominated by the size of a single frame, since a constant number of images needs to be stored, therefore the algorithm's space complexity is $O(nm)$.

3.7 Implementation

Finding a good solution required some experimentation and application of different techniques, so the research included a good deal of coding; in addition, a usable final product was developed. Because of this, choosing the right technologies was important for successfully completing the project.

3.7.1 Language

The two most important factors in the choice of language and libraries were code readability and speed of development; efficiency was taken into account, but it was considered a secondary concern. The path to finding a working solution in a research context often includes many failed attempts. Sometimes, especially in the field of computer vision, some experimentation is required for assessing the effectiveness and applicability of certain techniques. For example, as seen in chapter 2, various background techniques were tested in order to find one which was suitable for the analysis of the scene of *Ponte della Costituzione*.

This large amount of experimentation called for compact code that is easy to modify and replace. Python was considered to be a suitable language for this task. While lacking in efficiency compared with other languages such as C++, Python helps in increasing the speed of development and readability, especially thanks to its high level features. Performance is also enhanced by the extensive use of efficient libraries.

3.7.2 Libraries

Both the experiments performed during the research phase and the final product included extensive use of well known computer vision techniques. Rather than reimplementing many standard procedures, a popular library was employed for this task: OpenCV[16]. It is implemented in C++ and includes Python wrappers, so it is both fast and compatible with the chosen language. In addition, in the event of the project being ported to an embed-

ded platform, it would be easy to rewrite the code in C++ while keeping the same library calls.

OpenCV's Python wrappers are designed to make use of NumPy [30], an optimized Python library for efficiently handling n-dimensional arrays. The NumPy array data structure is used to represent images, which can be thought of as arrays of values (for gray-scale images) or triplets (for color images). Using typical Python loops for accessing each pixel in an image would have been very costly, so they were completely avoided and NumPy's array methods were used instead. This library also includes some useful linear algebra and numerical methods. For example, it was used for solving the linear system described in section 3.2 and for fitting the pixel-people association measurements with the least squares method as described in section 3.5.

As mentioned in section 2.3, BGSLibrary [18, 19] was used for the task of testing background subtraction techniques. At the time of writing the library implements 43 background subtraction algorithms, and provides an executable program for trying them out. Like OpenCV, this library is written in C++ and includes Python wrappers. However, it was quickly found that the Python wrappers had a memory leak problem, so it could not be used in the main Python program. Despite this, it was used for testing various background subtraction algorithms using the executable program bundled with the library. When a suitable algorithm had been found (the one described in [25]) it was reimplemented in Python with NumPy. The algorithm is simple and mostly straightforward, and since NumPy was used for the computing-heavy tasks the performance loss associated with the language was not significant.

3.7.3 Configuration

The algorithm needs a large number of parameters to function properly. They were not hard-coded, because in a different scenario they would have to be set to different values. Moreover, the program includes a method for just playing the sequence, which requires additional parameters if used. It

was decided to group all of these parameters into a configuration file, rather than using other input methods such as command line or a graphical user interface. This choice allows for better handling of the large number of entries and for easier automation through scripts.

The configuration format follows the informal INI file standard. It basically consists of a textual list of keys and values, along with a slightly more complicated section structure. A sample configuration file was also created. It provides a skeleton which can be modified in order to appropriately set the parameters, but also serves as documentation thanks to the explanatory comments contained within it. Configuration items are explained in the list below and an example is given for each of them.

- Path to video file to operate on. This is always required.

  ```
  video_path = /path/to/video.mkv
  ```

- Path to region map image file. It must be a gray-scale image with black background. Foreground regions are ordered from darkest to brightest.

  ```
  region_map_path = /path/to/region_map/image.png
  ```

- Path to background image. The region of interest must be empty (without people). This is the image used for initializing the background model.

  ```
  background_path = /path/to/background/image.png
  ```

- Path to file that maps the number of foreground pixels to the number of people. This file must have previously been generated and manually updated, so that it contains a list of pixel-people associations for each region.

  ```
  pix_people_path = /path/to/pixel_to_people_/data.csv
  ```

- Path to results file. Results will be written here in csv format.

  ```
  results_path = /path/to/results.csv
  ```

- Path to reset data. It can be used to periodically reset the system.

  `reset_path = /path/to/reset_data.csv`

- The old directed count is reset after this period, expressed as number of frames. This parameter is used to skip entries in the reset file specified with `reset_path`. By default the count is reset for each entry in that file.

  `reset_period = 900`

- Sub-frame region to work on. Each frame is read then cropped to this region before further processing. It is a rectangle defined by its top-left and bottom-right corners. Image coordinates are ordered from left to right (horizontal axis) and from top to bottom (vertical axis). In other words, if x_l and x_u are the minimum and maximum values on the horizontal axis and y_l and y_u are the minimum and maximum values on the vertical axis, then the format is $((x_l, y_l), (x_u, y_u))$. Note that this parameter is useful for playing, but it is not necessary for analyzing the video, because in that case the region can be automatically calculated as the bounding box of the non-background regions defined in `region_map_path`.

  `box = ((240, 150), (580, 345))`

- Initial directed counts for each region. A directed count is the number of people walking in a certain direction. There must be r couples where r is the number of foreground regions. The first value of each couple denotes the number of people walking in direction $+$, that is from the first region towards the last, while the second value denotes the number of people walking in direction$-$, that is from the last region towards the first. In the case of the bridge direction $+$ is left to right and direction $-$ is right to left.

  `init_directed_count = ((3, 5), (5.5, 2), (4.5, 3))`

- Percentage of people that is assumed to have moved from one region to the next one in the specified time interval. See equation 3.2 for a way to calculate p. The first value is for direction $+$, the second for direction $-$.

  ```
  p = (0.666667, 0.42)
  ```

- Time interval Δt as number of frames. The algorithm will skip this many frames at each step. A reasonable value should allow people to move from one region to the next, but not to the one after the next.

  ```
  frame_interval = 90
  ```

- Number of frames that the average people count will be computed over. Rather that estimating the people count only at one instant, more consecutive frames are analyzed and the results are averaged, thus increasing the counting accuracy.

  ```
  averaging_interval = 5
  ```

- Start working from the specified frame number.

  ```
  start_at = 0
  ```

- Stop and quit at the specified frame number.

  ```
  stop_at = 18000
  ```

- Highlight people contours when playing the video (only for playing).

  ```
  show_contours = False
  ```

- Wait for this number of milliseconds before showing next frame (only for playing). This does not include the additional required processing time.

  ```
  play_delay = 1
  ```

- Wait for this number of milliseconds before showing next frame (only for rewinding). This does not include the additional required processing time.

  `rewind_delay = 30`

- The number of frames that are replayed when pressing the r key. A longer history requires more memory and is more computationally expensive. About 30 frames, that is 2 seconds, should be a sufficient time for a user to understand the direction movement of pedestrians.

  `rewind_history = 30`

- Pause video at every frame that is a multiple of this number (when playing). Useful for creating a reset file, because the video is periodically stopped, then directional counting can be performed by replaying the last `rewind_history` frames.

  `pause_frame_period = 900`

Chapter 4

Results

The algorithm presented in chapter 3 was intended to run on a large dataset which includes video records spanning several days. This chapter describes how the results were produced and how they were compared with simulated and ground truth data to test the algorithm accuracy.

4.1 Results Format

When run over a long video, the algorithm produces a large amount of data. The output was designed to be informative and easily manipulatable; even though the most interesting piece of information about the video sequences is the pedestrian flow, it was decided not to directly output it. A more basic or 'raw' format was deemed to be more suitable, because the flow can be easily calculated from the results and it is easier to obtain other kinds of information, such as the total number of people that have crossed the bridge over a certain amount of time. The output simply states how many people have entered one of the defined ROI from a certain direction in a specified period of time (6 seconds in the case that was studied).

This information is stored in a CSV file; its format is more or less standard (there is not a unified CSV standard), so that external programs should be able to open it without much trouble. Each entry is composed of three

comma-separated columns: the first shows the frame number at which the algorithm analyzed the scene, while the other two indicate how many new people were detected in each direction between the time instant of the entry and the one before. The first row contains the heading, which is a list of strings. An example of a results file is shown below:

```
"frame","from_left","from_right"
0,0,0
90,0.0,4.86813935099
180,6.11236058661,1.79947363876
270,9.51028243321,0.0
360,1.1731975269,3.06610531039
450,1.88824672883,7.45171237788
540,0.0,2.10095832907
630,2.46236018695,1.01593009628
720,3.94547343164,0.0
810,4.94366136759,2.6409891184
900,4.92604152219,0.0
990,2.7365398621,0.0
1080,3.54235038442,0.0
1170,1.69777450428,0.0
1260,0.0,3.50756618194
1350,0.0,3.6076959037
1440,1.16369315675,0.0
1530,2.33162888802,1.82287975544
1620,3.20303038802,0.0
1710,4.15346991495,4.19140690285
1800,3.8985913416,2.24051849038
```

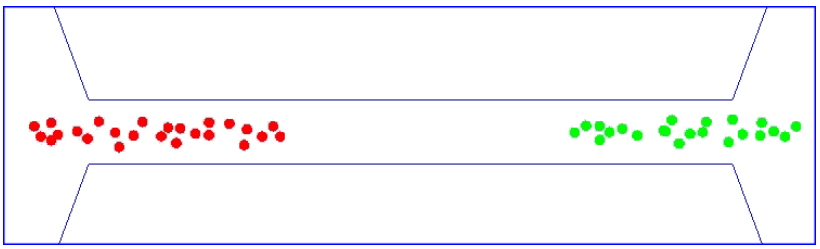
The flow can be calculated by adding the counts and dividing by the corresponding interval of time. For example, the flow of people going from left to right over the first minute (assuming a frame rate of 15 Hz) is computed by

summing the contents of the second column of the first 10 entries (excluding the first, because at time 0 nothing happened), then diving this quantity by the time elapsed, that is 1 minute. Referring to the results file above, the flow would then be 34.96 people/minute. This can be repeated for the following entries, thus finding out how the flow varies over time. With reference to the same example, during the second minute the flow from left would be estimated to be 22.73 people/minute.

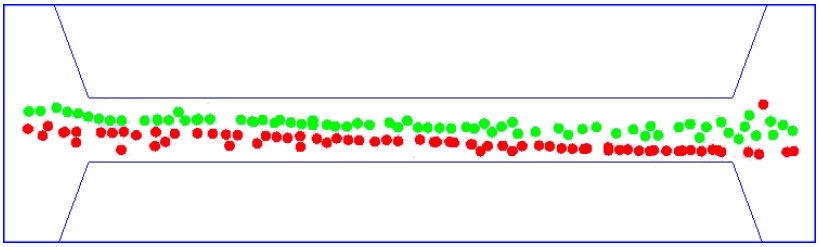
While analyzing the flow on a per-minute basis is useful for simulated flows, it would not be of much interest in real videos, since it would be subject to sudden changes due to people crossing the bridge at uneven intervals. Therefore the flows were computed with a time span of approximately 10 minutes when comparing the results with the ground truth. Note that this flexibility in the choice of the time interval is due to the results format: if the flows were given directly, the interval would have to be chosen before running the algorithm and it could not be changed afterwards.

4.2 Simulation

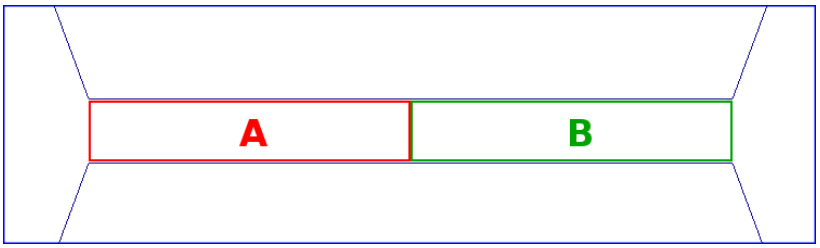
Before executing the algorithm on real videos, it was tested on simulated data. The simulation consists of a video of two very dense flows of people passing through a constrained passage. Each person is represented by a colored circle, which spawns on one side of the passage and moves through it until it disappears at the other side. Note that the color identifies the direction of movement, as red circles come from the left and green ones from the right, for visualization purposes, but this kind of information is not used by the algorithm when the video is analyzed. Simulated people tend to avoid running into each other, just as in real life. Despite this, the circles are allowed to overlap, which, to a certain degree, represents the occlusions that occur in the real videos. Both flows are very intense, because the algorithm was designed to work in high density scenarios. They are asymmetrical, so that it is possible to verify that the algorithm is able to separate them



(a) Initial flow.



(b) Flow at full intensity.



(c) Highlight of regions A and B.

Figure 4.1: Simulation scene and the regions defined over it. The central corridor is densely populated and some circles overlap each other, resembling the characteristics of the algorithm's intended scenario.

correctly. Figure 4.1(a) shows a frame where the first circles are starting to cross the passage, and figure 4.1(b) shows the two flows at full intensity.

The video is 10 minutes long and has a frame rate of 30 Hz. The simulation data is reported every 101 frames. The corridor or passage that the circles run through is 594 pixels wide and 56 pixels high. Two regions, both 297 pixels wide, were defined over it: they are highlighted in figure 4.1(c). The circles' speeds are constant in both directions, so the mean velocity required for solving equation 3.2 and calculating the probabilities $p^{\pm}$ is known. Since in this case the speed does not depend on direction, p^{+} and p^{-} are

equal, so they can be referred to as p: $p = p^+ = p^-$. A circle crosses the whole passage in 404 frames, so it takes 202 frames to move from one region to the next one. The time interval of the algorithm was set at 101 frames to match the simulation data. The probability p can be calculated according to equation 3.2 as follows:

$$p = \frac{v\Delta t}{L} = \frac{\frac{297}{202}101}{297} = 0.5$$

The function to convert the number of foreground pixels to number of people was manually defined, knowing the size of a circle and ignoring the variations caused by occlusions at higher densities.

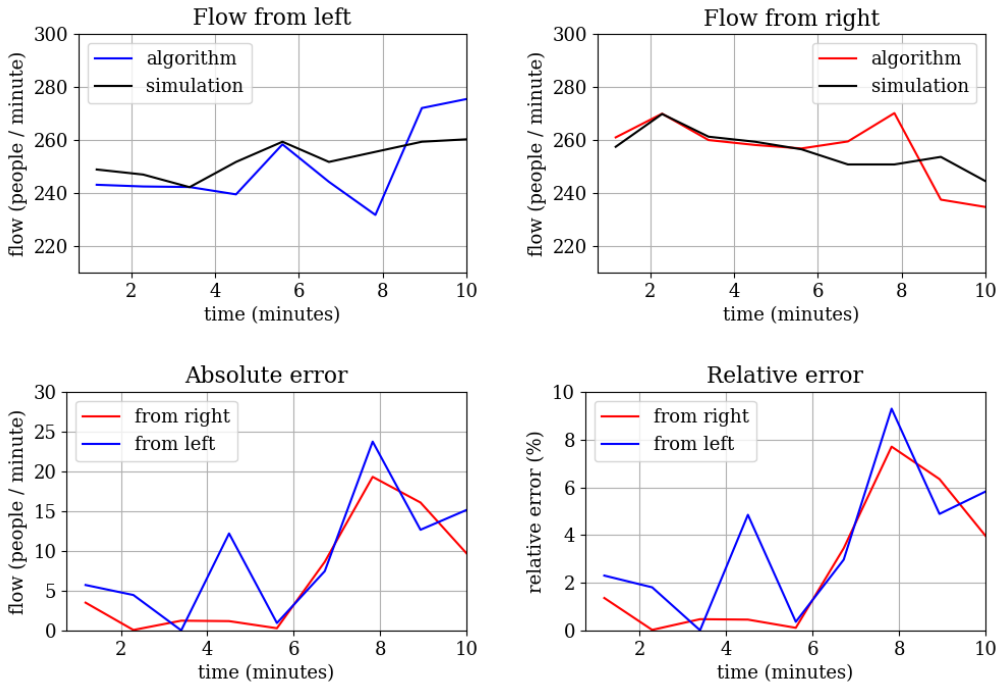


Figure 4.2: The charts show a comparison over time between the simulated flow and the one computed by the algorithm. The low relative error means that the algorithm provides an accurate estimate of the flows, despite the extremely high density.

Figure 4.2 shows charts of the two computed flows compared with the simulated flows over time, and charts of the absolute and relative errors of

the computed flows over time. The results of the algorithm and the simulation are compared at about every minute. Note that, since the flows are computed at every time point using the preceding counts, they are not defined at time 0. The figure shows that the computed flows tend to follow the same trend as the real ones. Overall, the results are very good, because even with this extremely high density the computed flows stay within 9% of the real ones. Such promising results mean that, provided that the assumptions about the pedestrian velocity hold and that foreground detection is reliable, the algorithm should work well on real world videos too.

4.3 Ground Truth

Since the algorithm can not provide a perfect estimate, it was important to compare its output with a large set of ground truth data from a real video in order to test its accuracy. This data was manually generated by visually analyzing a video record that shows the bridge from 11:05 to 12:20 of Feb 28th 2017. Because of the unfavorable characteristics of the scene, described in chapter 1, the manual analysis was not a trivial task.

4.3.1 Counts

The video was played with the play function which was added to the program; this function allows the user to see the region boundaries, so it makes it possible to understand when new people have entered the scene which should be detected by the algorithm. The video was played twice, once for each direction, so that it was possible to focus on one end of the area and count pedestrians crossing the bridge in each direction more accurately. Whenever a group of people stepped into the external region, the video was paused and the frame number the number of people was noted in a file (the *ground truth file*) with a simple textual format. For each entry the number of people refers to the pedestrians that entered the scene from a specific side between the last recorded time and the one which is part of the entry.

This approach was chosen because it is manageable for the user, although it leads to the time intervals which the counts refer to not to be exact, because sometimes they do not take into account long periods during which no pedestrians enter the scene. Overall, all pedestrians were counted with reasonably accurate timings, given that the average time interval between groups of counts is 198 frames, that is about 13 seconds.

The file format consists of two lists of rows, one list which records people coming from the right and one list which records those coming from the left. Each row or entry contains a frame number and the people count, separated by a comma. The beginning of each list is marked by a tag, either `right_to_left` or `left_to_right`. This format makes it easy for a human to create the file and also to later read or check it. The following is an excerpt from the ground truth file which shows only the first two minutes:

```
right_to_left:
59, 1
145, 6
233, 4
455, 10
545, 0
701, 4
929, 1
1345, 7
1633, 8
1866, 3

left_to_right:
218, 7
422, 4
605, 4
706, 2
951, 8
```

```
1222, 5
1603, 5
1858, 7
```

4.3.2 Reset

As mentioned in 3.1, the algorithm requires the user to input the initial condition, that is the directed people counts in each region of interest. For each analysis step, the solution of the linear system might become less accurate, leading to progressively inconsistent results. This would mean that a good solution could be obtained only by periodically resetting the system. For this reason, apart from comparing the algorithm results with those obtained by hand, it was also necessary to test the robustness of the algorithm by estimating the impact of periodical resets of the directed counts. The reset information was extracted at regular intervals of one minute; resetting every minute should have a greater impact on the final analysis results, but with this information available it is also possible to downsample it and reset the algorithm after longer periods of time (e.g. every five minutes) by setting `reset_period` to an appropriate value.

The information necessary for each reset was collected in a second ground truth file, which will be referred to as the *reset file*. The directed counts were obtained by playing the video with the custom play function built into the program: every `pause_frame_period` frames the video is paused, then the user presses the r key to replay the last `rewind_history` frames in order to understand the pedestrians' direction of movement (see section 3.7.3 for a description of the parameters). The frame number and the directed counts are then manually written in a custom CSV (comma-separated values) file.

The file format allows it to be easily read by either a human or a computer. It contains two columns: one for the frame number and one for the directed counts. The columns are marked by a header and separated by a semicolon. The directed counts are represented by a set of nested parenthe-

sis: for example `((1, 4), (3, 2))` means that in the first region one person was moving in direction 0 (towards the right) and 4 people were moving in direction 1 (towards the left), and so on. Also note that the counts can be fractional if one or more persons were crossing a region boundary when the video was paused. This is the same format of the `init_directed_count` described in section 3.7.3. The following is an excerpt from the reset file which shows the information relative to the first five minutes:

```
"frame_num"; "directed_count"
900; ((9.5,7), (1,2))
1800; ((6,3), (4,5.5))
2700; ((2,4), (1,7))
3600; ((2,3), (3,4))
4500; ((5,3), (6,6))
```

4.4 Verification

The algorithm was tested to measure its accuracy and robustness. As previously described, the ground truth was collected in order to compare it with the results computed by the algorithm. The ground truth is relative to an hour and fifteen minutes long video, starting at 11:05 of Feb 28th 2017. The algorithm operates at regular intervals, wheres the ground truth does not, so a problem arises when trying to compare two points: no exact data is available at a shared time instant. This has been solved by adapting the algorithm results to match the time points at which the ground truth is defined. In particular, before calculating the flows, each people count detected in a regular time interval (6 s) was assumed to be evenly distributed in that interval; in this way it is possible to split any interval into two or more parts and leverage this flexibility to match the time intervals for which the ground truth is defined. The manually collected data was not modified in the process, in order to avoid introducing additional errors or distorting the truth. Since the time intervals of the counts are irregular, the flow was

also calculated at irregular intervals of approximately 10 minutes; however the time intervals of the counts are short enough to allow for having only small variations. In any case, the correctness of the estimated flows is not influenced by the regularity of the intervals, because their duration was taken into account in the calculation.

The algorithm was run on the test sequence with different configurations. In the first one, the initial directional counts were given, but they were never reset: at each and every step the algorithm had to solve the system 3.3 using its self-calculated directional counts. In the second and third the directional counts were periodically reset using the reset file described in section 4.3.2, at intervals of respectively 5 and 1 minutes. The results are presented with charts and tables at the end of the chapter. The resulting flow coming from the left side is reported in table 4.1, and the one coming from the right in table 4.2. Figure 4.3 shows how the flows evolve over time according to the ground truth and the three test runs.

It can immediately be observed that resetting the system every 5 minutes, which equates to one time every 50 steps, has little to no effect, because the results are very similar to the ones one would get by never resetting. In order to get a noticeable effect, one has to increase the frequency of resets to one per minute, which corresponds to one time every 10 steps. However, resetting so frequently is not acceptable in practice, because it would require a great amount of human work, comparable to manually counting every pedestrian. Moreover, it appears that resetting frequently does not always improve the accuracy of results, and indeed in many cases it reduces it. This situation is clearly visible in figure 4.4, which represents the absolute error of the computed flows over time.

On one hand, the small impact of resets means that it is not possible to really keep the algorithm under control through human intervention, so this is not a viable way to improve the results in particularly challenging situations. On the other hand, since the results proved to be relatively accurate and stable over time (i.e. the error does not increase as a function of time), it

also means that the algorithm can keep running without human intervention for a very long time. It follows that the algorithm would be able to operate in an on-line manner after an initial setup by an operator, even without remote managing. Moreover, thanks to the low computation complexity, porting the system to an embedded platform could be a concrete possibility.

The charts of figure 4.3 show that when the video was recorded the bridge saw a higher flow of people coming from the right than from the left. The latter also looks more variable than the former, and this might be a consequence of its lower volume. If people cross the bridge at uneven time intervals it is more likely that they concentrate only in some of the 10 minutes intervals considered in the analysis and this is more noticeable if the total flow is lower.

Despite the higher volume of people coming from the right, the flow computed by the algorithm is remarkably accurate, having a mean relative error of only 9% and a maximum of 23% in a run with no resets. Considering the challenging scenario described in chapter 1, this is considered an excellent result. Figure 4.5 depicts the relative error of the various computed flows as a function of time: it can be seen that for both directions the error does not increase over time, which reinforces the idea that the algorithm can run unattended for a long time without losing its accuracy.

The situation of the flow from left is slightly worse. Both figure 4.4 and figure 4.5 show that the estimated flow has a high error at minute 42. The error relative to the ground truth amounts to up to 68% for the run without resets, which might seem worrisome at first. However, it should be noted that it jumps up only once, while most of the time it stays within reasonable values lower than 20%. The mean relative error, including the high spike, is limited to 18%. Another thing that should be taken into account is that the relative error is very high when the flow is low: at minute 42 it is only 15 people per minute.

The algorithm was studied for working with especially intense flows, and results show that it accomplishes this task quite well. It is sometimes inac-

curate with lower flows, but if a better precision was desired, the analysis could be improved by combining this new algorithm with another one that works better with lower volumes but fails at higher ones, such as one of those presented in chapter 1.

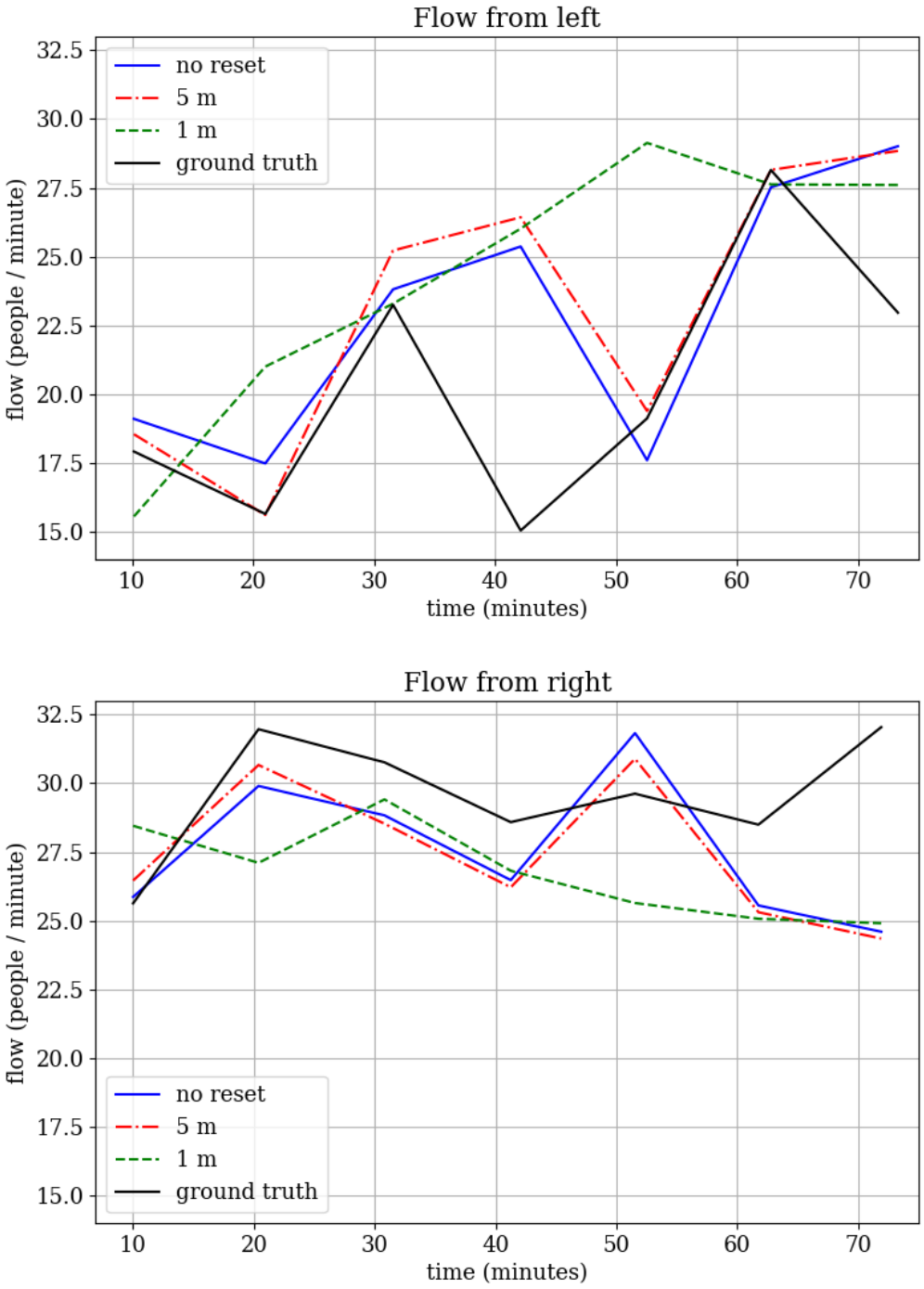


Figure 4.3: Comparison between computed flows and ground truth. Despite a few exceptions, the actual flows are estimated well.

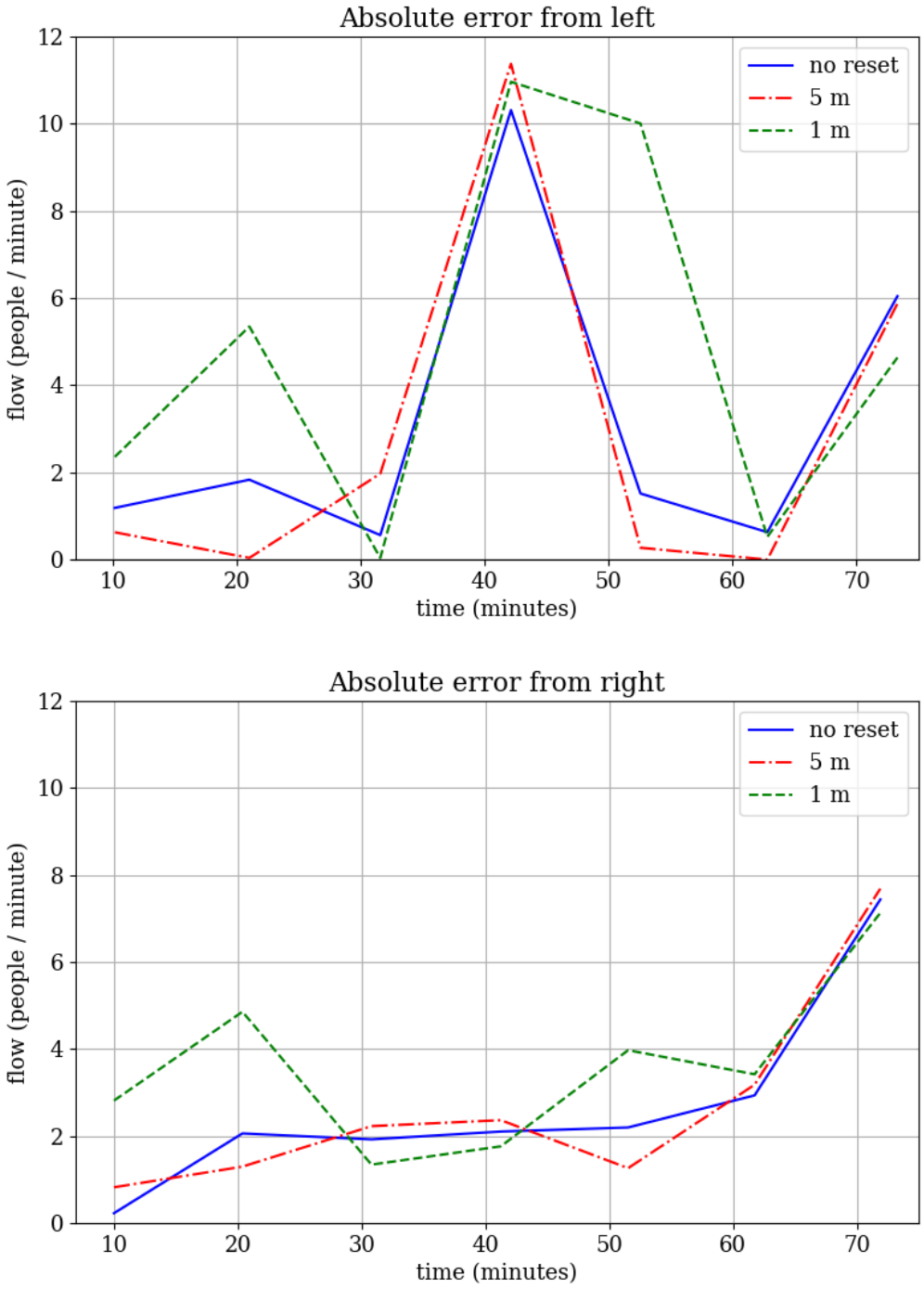


Figure 4.4: Absolute error of computed flows. As seen in the charts, resetting the algorithm frequently does not improve its accuracy.

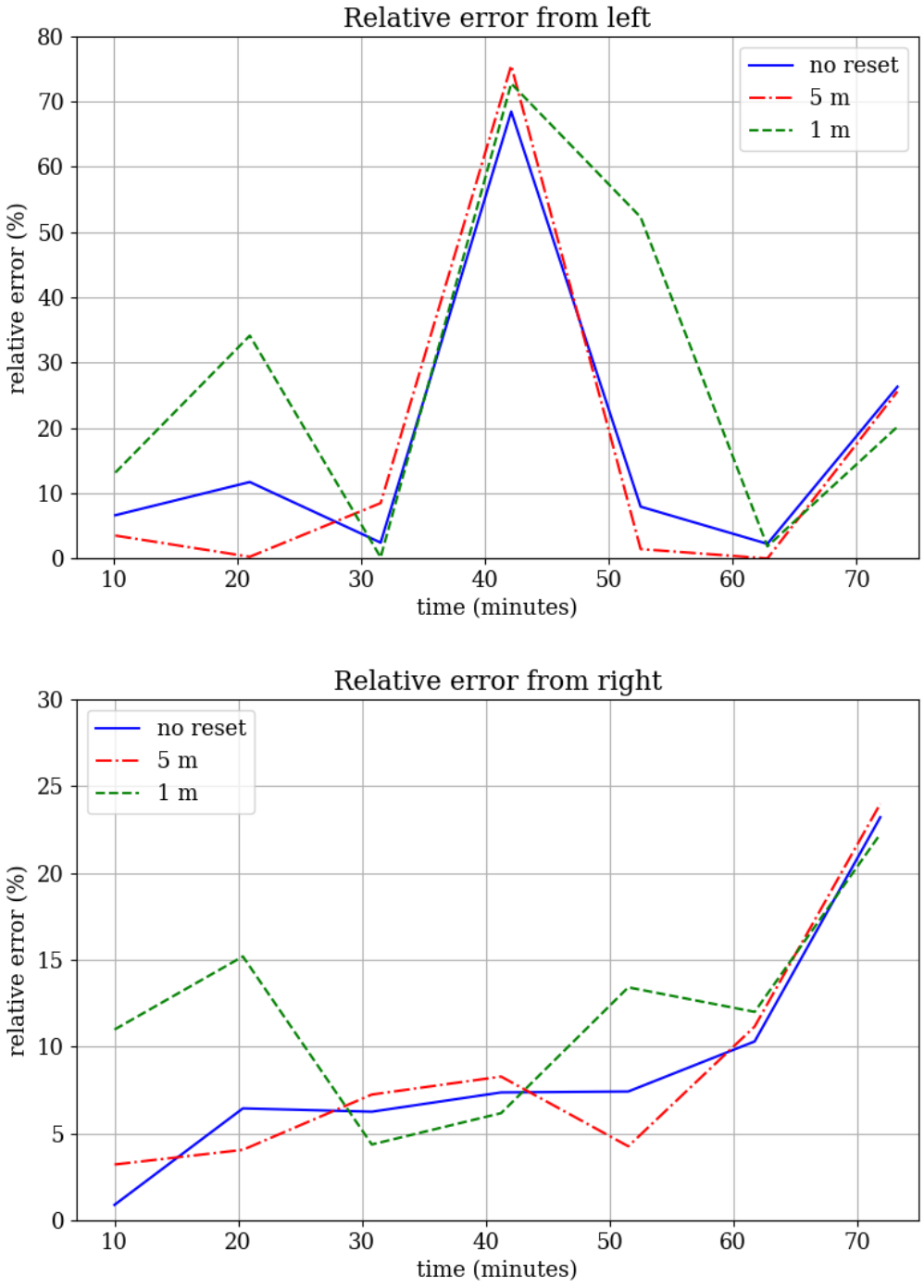


Figure 4.5: Relative error of computed flows. It stays within reasonable values most of the time, with an exception at minute 42 in the flow from left.

	Flow from left			
Time	G. truth	No reset	Reset 5m	Reset 1m
10.15	17.93	19.11	18.56	15.57
21.01	15.66	17.50	15.62	21.01
31.57	23.25	23.82	25.23	23.29
42.11	15.06	25.37	26.43	26.02
52.56	19.13	17.61	19.40	29.13
62.80	28.15	27.52	28.15	27.63
73.29	22.97	29.01	28.84	27.60

Table 4.1: Flow from left. Time is expressed in minutes and flow in people per minute.

	Flow from right			
Time	G. truth	No reset	Reset 5m	Reset 1m
10.00	25.64	25.87	26.46	28.45
20.38	31.97	29.90	30.66	27.11
30.80	30.76	28.83	28.53	29.41
41.25	28.59	26.48	26.22	26.82
51.53	29.62	31.82	30.89	25.65
61.75	28.50	25.56	25.31	25.07
71.90	32.04	24.60	24.35	24.91

Table 4.2: Flow from right. Time is expressed in minutes and flow in people per minute.

Conclusions

The project required to analyze a series of video sequences extracted from a security camera with a view of *Ponte della Costituzione* in Venice. The camera was already in place and the sequences were already recorded, so it was not possible to choose an optimal image acquisition setup. Although the bridge is in the scene, it is only a small part of it due to its distance. Furthermore, the camera is not placed in a very high place, which, coupled with intense flows of pedestrians, causes many occlusions. Existing people counting techniques are based on tracking, but they are not able to work correctly with a combination of low detail and a high density of people.

For this reason, a new algorithm had to be developed and implemented. The solution is based on a simple but effective background subtraction technique. Background subtraction is a well known topic in computer vision, so several methods could be studied and tested. Once a suitable method, based on computing an adaptive median background image, was found, it was used for estimating the number of people present in certain regions of the bridge. The conversion from number of foreground pixels to number of pedestrians was achieved by fitting manually collected pixel-people association data.

The movement of people along a path was modeled with a linear system of equations, which was used for obtaining the directional people counts, that is for separating the two flows that go through the bridge. Given the initial directed counts, the algorithm works by computing the number of people present in two adjacent regions of interest, then solving the linear system at regular intervals to find how many pedestrians crossed the bridge in each

direction over a period of time.

The ground truth was collected over an approximately 75 minutes long sequence and it was used for verifying the accuracy of the algorithm. Running some tests showed that the algorithm is sufficiently accurate, as the relative error of the results usually stays lower than 20%, even after a long time. It was also observed that resetting the system had no significant impact on its precision, unless it is done very frequently. It was therefore concluded that, after an initial setup phase, there is little to no need for manual intervention to ensure the consistency of the results.

In addition to being independent of a human operator, the algorithm is in theory very efficient, because its computational complexity is dominated by the background subtraction phase, and the particular technique that was employed for this task is much faster than many alternatives. All of this contributes to the viability of porting the algorithm to an embedded platform in the future, which would allow running an on-line analysis of the scene at a cheap price. In other words, the pedestrian flow could be monitored *during* public events that attract large crowds, rather than afterwards.

Although the system of equations which was used models the flow of people walking at constant speed in a constrained passage, a variant which includes the possibility of a variable pedestrian speed was also studied. The system might be the basis for more complicated models that take into account more than two directions of movement and several entry points.

In conclusion, despite the particularly challenging scenario, a solution was found by developing a new algorithm which is fast, autonomous and relatively accurate. While a better accuracy would be achieved by properly placing a high-quality camera, the new algorithm allows for analyzing heavy pedestrian flows using an existing setup, without requiring additional expensive equipment.

Bibliography

[1] D. Helbing, A. Johansson, and H. Z. Al-Abideen. "Dynamics of crowd disasters: An empirical study". In: *Phys. Rev. E* 75 (4 Apr. 2007), p. 046109.

[2] M. Achille. "Sistema semi-automatico per il tracciamento di pedoni da video digitali in condizioni di alta densità". MA thesis. Alma Mater Studiorum, Università di Bologna.

[3] T.-H. Chen, T.-Y. Chen, and Z.-X. Chen. "An intelligent people-flow counting method for passing through a gate". In: *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*. IEEE. 2006, pp. 1–6.

[4] P. KaewTraKulPong and R. Bowden. "A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes". In: *Image and Vision Computing* 21.10 (2003), pp. 913–929.

[5] M. Bozzoli, L. Cinque, and E. Sangineto. "A statistical method for people counting in crowded environments". In: *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*. IEEE. 2007, pp. 506–511.

[6] J. Shi and C. Tomasi. "Good features to track". In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE. 1994, pp. 593–600.

[7] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'81. Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[8] J. Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[9] H. Xu, P. Lv, and L. Meng. "A people counting system based on head-shoulder detection and tracking in surveillance video". In: *Computer Design and Applications (ICCDA), 2010 International Conference on*. Vol. 1. IEEE. 2010, pp. V1–394.

[10] O. Masoud and N. P. Papanikolopoulos. "A novel method for tracking and counting pedestrians in real-time using a single camera". In: *IEEE transactions on vehicular technology* 50.5 (2001), pp. 1267–1278.

[11] V. B. Subburaman, A. Descamps, and C. Carincotte. "Counting people in the crowd using a generic head detector". In: *Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on*. IEEE. 2012, pp. 470–475.

[12] J. García et al. "Directional people counter based on head tracking". In: *IEEE Transactions on Industrial Electronics* 60.9 (2013), pp. 3991–4000.

[13] J. F. Henriques et al. "Exploiting the circulant structure of tracking-by-detection with kernels". In: *European conference on computer vision*. Springer. 2012, pp. 702–715.

[14] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction". In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 2. IEEE. 2004, pp. 28–31.

[15] Z. Zivkovic and F. Van Der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". In: *Pattern recognition letters* 27.7 (2006), pp. 773–780.

[16] G. Bradski. In: *Dr. Dobb's Journal of Software Tools* (2000).

[17] C. Kamath and S. Cheung. *Robust techniques for background subtraction in urban traffic video*. Tech. rep. Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2003.

[18] A. Sobral. "BGSLibrary: An OpenCV C++ Background Subtraction Library". In: *IX Workshop de Visão Computacional (WVC'2013)*. Rio de Janeiro, Brazil, June 2013. URL: `https://github.com/andrewssobral/bgslibrary`.

[19] A. Sobral and T. Bouwmans. "BGS Library: A Library Framework for Algorithms Evaluation in Foreground/Background Segmentation". In: *Background Modeling and Foreground Detection for Video Surveillance*. CRC Press, Taylor and Francis Group., 2014.

[20] A. Sobral and A. Vacavant. "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos". In: *Computer Vision and Image Understanding* 122 (2014), pp. 4–21.

[21] L. Maddalena and A. Petrosino. "A self-organizing approach to background subtraction for visual surveillance applications". In: *IEEE Transactions on Image Processing* 17.7 (2008), pp. 1168–1177.

[22] M. Hofmann, P. Tiefenbacher, and G. Rigoll. "Background segmentation with feedback: The pixel-based adaptive segmenter". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE. 2012, pp. 38–43.

[23] C. R. Wren et al. "Pfinder: Real-time tracking of the human body". In: *IEEE Transactions on pattern analysis and machine intelligence* 19.7 (1997), pp. 780–785.

[24] T. Bouwmans. "Traditional and recent approaches in background modeling for foreground detection: An overview". In: *Computer Science Review* 11 (2014), pp. 31–66.

[25] N. J. McFarlane and C. P. Schofield. "Segmentation and tracking of piglets in images". In: *Machine vision and applications* 8.3 (1995), pp. 187–193.

[26] N. Otsu. "A threshold selection method from gray-level histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[27] J. Kittler and J. Illingworth. "On threshold selection using clustering criteria". In: *IEEE transactions on systems, man, and cybernetics* 5 (1985), pp. 652–655.

[28] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 2nd ed. Upper Saddle River: Prentice Hall, 2002, pp. 519–527.

[29] R. Knoblauch, M. Pietrucha, and M. Nitzburg. "Field studies of pedestrian walking speed and start-up time". In: *Transportation Research Record: Journal of the Transportation Research Board* 1538 (1996), pp. 27–38.

[30] S. v. d. Walt, S. C. Colbert, and G. Varoquaux. "The NumPy array: a structure for efficient numerical computation". In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.

Ringraziamenti

In primo luogo vorrei ringraziare con grande affetto i miei genitori, non solo per avermi cresciuto ed educato, ma anche per aver sempre supportato le mie scelte, inclusa quella di proseguire gli studi.

Grazie ai miei amici e compagni, con cui ho condiviso pranzi, progetti, serate e giornate: Antonio, Fabio, Federico, Giovanni & Giovanni, e Stefano Pisciella. Se non fosse stato per voi, studiare a Bologna sarebbe stato triste e noioso.

Ringrazio Ilenia, per essere una persona meravigliosa, e per aver reso così piacevoli i miei giorni.

Grazie a Stefano Sinigardi, per la sua attenta guida ed il suo aiuto nello sviluppo di questa tesi. Grazie anche a Nico e a tutto il team di Sistemi Complessi per i preziosi consigli.

So long, and thanks for all the fish.