FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Multi-Object Tracking in Video Sequences

Miguel Amável dos Santos Pinheiro

Master in Electrical and Computers Engineering

Supervisor: Luís Corte-Real (PhD)

Co-supervisor: Pedro Carvalho (MSc)

July 2010

Abstract

Object tracking in video sequences has found several different applications, such as surveillance, advanced interface, virtual reality, movement analysis, among others. However, the broadness of the spectrum of application is not caused by the existence of a general approach for object tracking. The inexistent generalization requires developers to build algorithms over several premises, in order to build robust solutions although in specific situations. To emphasize this issue, we present two approaches to object tracking, stating the premises of each method and the inoperability of one solution over the other context.

In one hand, the dissertation presents an approach for vibrating line detection and tracking. Devices like accelerometers or load cells are normally used for measuring of vibrations in cables, wires or generic line structures. However, the installation of such equipment can be troublesome or even impossible in some cases, therefore new ways of capturing vibrating characteristics were developed. Computer vision allows a contactless technique to gather these measurements. However, the state-of-the-art algorithms either require a user initialization or a reduced description of the vibration. In this dissertation, a method for automatic line detection is presented, integrating a stable path approach, followed by an optical flow method for tracking the line‘s displacement.

People tracking algorithms are an important subject within the object tracking field. These methods can usually be divided in several modules, consisting in initialization, tracking, pose estimation, among others. The tracking module matches people from one frame to another and therefore needs to make comparisons between two objects. This process often involves the use of an appearance model, *i.e.*, a representation of the person, and requires from the developer a trade-off decision between a reduced processing time and accurate characterization. An alternative method for people tracking is presented in this dissertation, consisting in the integration of an alternative appearance model which resorts to a local description using interest points, with the objective of an improvement in the description of a person.

Resumo

O seguimento automático em sequências de vídeo tem encontrado variados mundos para aplicação, tais como vigilância, *interface* avançada, realidade virtual, entre outros. Contudo, o largo espectro de aplicação não é causado pela existência de uma abordagem generalizada para seguimento automático de objectos. A inexistência desta generalização exige aos investigadores que construam algoritmos sobre algumas premissas, para que seja construídas soluções robustas apesar de em situações específicas. Para sublinhar este problema, apresentamos duas abordagens para seguimento automático de objectos, declarando as premissas de cada método e a inoperabilidade de uma solução sobre o outro contexto.

Por um lado, a dissertação apresenta uma abordagem para detecção e seguimento automático de linhas vibratórias. Aparelhos como acelerómetros ou células de carga são normalmente usados na medição de vibrações em cabos, fios eléctricos ou estruturas genéricas com forma de linha. Contudo, a instalação de tais equipamentos pode ser problemático ou até mesmo impossível em alguns casos, portanto novas técnicas para capturar características de vibração foram desenvolvidas. Visão por computador proporcionou uma técnica sem recorrer a contacto com a estrutura para obter estas medidas. Contudo, os métodos existentes ou requerem uma inicialização do utilizador ou proporcionam uma descrição reduzida da vibração. Nesta dissertação, um método para detecção automática de linhas é apresentada, integrando a abordagem de caminhos estáveis, seguido de um método de fluxo óptico para seguimento automático de deslocamento de linhas.

O seguimento automático de pessoas é uma parte importante dentro do campo de seguimento automático de objectos. Estes métodos podem, normalmente, ser divididos em vários módulos, que consistem em inicialização, seguimento automático, estimação de pose, entre outros. O módulo de seguimento automático faz a correspondência entre uma *frame* e a seguinte e, portanto, necessita de realizar uma comparação entre dois objectos. Este processo muitas vezes envolve o uso de um modelo de aparência, *i.e.*, a representação da pessoa, e requer do autor uma decisão de *trade-off* entre uma redução de tempo de processamento e uma caracterização precisa. Um método alternativo para seguimento automático de pessoas é apresentado nesta dissertação, que consiste na integração de um modelo de aparência alternativo que recorre a um descritor local que usa pontos de interesse, com o objectivo de uma melhoria na descrição de uma pessoa.

Acknowledgements

I‘d like to leave a brief note of appreciation for all the people at INESC Porto that provided me the possibility to work in a productive environment, always with good spirit and friendship. A special thanks to Pedro Carvalho who‘s been guiding my work since day one and, obviously to Professor Corte-Real for this opportunity to develop this dissertation. Also, I‘d like to mention all my friends for providing me the strength to endure harder times and also my faculty colleagues for challenging me and for bringing a good environment to all the hours of struggling throughout these past five years.

I’d also like to state my everlasting appreciation for all the opportunities my parents have given me, because I owe it all to their nurturing, care and attention they have given me all my life. Also, a special thanks to my better half who provided me all the strength to pursuit great opportunities and look at the future with optimism and willing to achieve great things.

Miguel Amável Pinheiro

"Vision is the art of seeing the invisible."

Jonathan Swift

Contents

List of Figures

List of Tables

Abbreviations and Symbols

2D	Two dimensions
3D	Three dimensions
BoV	*Bag-of-visterms*
CAVIAR	Context Aware Vision using Image-based Active Recognition
CVML	Computer Vision Markup Language
HSV	Hue, Saturation, Value
MSE	Mean Square Error
PETS	Performance Evaluation of Tracking and Surveillance
RGB	Red, Green, Blue
RMSE	Root Mean Square Error
ROI	Region of Interest
SIFT	Scale-invariant Feature Transform
VOT	Video object tracking
YCbCr	Luminance and Blue-difference and Red-difference chroma components

List of Symbols

Camera Calibration

A	Intrinsic Matrix
α_x	Horizontal picture size
α_y	Vertical picture size
γ	Skew coefficient
H	Homography matrix
M'	Vector for 3D points
m'	Vector for 2D points
r_{ij}	Rotation coefficient, i=1,2,3, j=1,2,3
s	Scale factor
t_i	Translation coefficient, i=1,2,3
u_0	Horizontal center point of a picture
v_0	Vertical center point of a picture

Line Detection and Tracking

$d_{a,b,i}$	Minimum distance from point i in line a, concerning all points in line b
$d_{a,b}$	Average distance between points in line a and b
$\tilde{d}$	Total average of the distance between lines in all frames of a sequence
D_f	Matrix of average distances between lines in frame f
E_f	Intersection matrix between D_f and H_f
H_f	Binary matrix of line matching for frame f
h	Height of frames
N	Number of frames in a sequence
$op(j)$	Value for optical flow for each point in a line
pen	Penalty associated with finding an extra line or not finding a line in matching
r_f	Reference value for the displacement in frame f
S	Number of points in a line
$v(i)$	Displacement for each point in a line
$\tilde{v}$	Median of $v(i)$ for all points
w	Width of frames
W_n	Number of points of a window

People Tracking

B	Number of elements in signature
$f1$	Feet position for reference 1
$f2$	Feet position for reference 2
$h1$	Head position for reference 1
$h2$	Head position for reference 2
H_{prev}	Signature for the position in the previous frame
H_t	Signature for the current considered position
$N_{H_{prev}}$	Sum of elements of H_{prev}
N_{H_t}	Sum of elements of H_t
r_1	Square root of the division between $N_{H_{prev}}$ and N_{H_t}
r_2	Inverse of r_1

Chapter 1

Introduction

1.1 Context and Motivation

Computer vision has received great attention over the last two decades. Computers have always been developed with the objective of performing human tasks. Though in early years, the goal could only be to try to make them think like humans, the technology has been evolving in so many ways that it is now enabling the simulation of other human skills. Processors are getting faster and smaller, digital cameras allow more and more image quality, sensors are becoming more accurate and so on.

This dissertation focuses on the field of computer vision. Because of this great evolution in technology and also the need for automating processes, computer vision has contributed with techniques that allow computers to gain visual perception, *i.e.*, they can now be aware of its camera‘s surroundings and therefore, compute algorithms according to actions visible to the camera. Nevertheless, there still hasn‘t been an optimal solution for some of the problems associated with the field.

One of the most important areas in computer vision is object tracking. This research field is important not only for security related software, but also in advanced interface between people and computers, advanced control methods and many other areas. Therefore, it becomes clear the wide range of operations in the computer vision field. The broad spectrum of these operations complicates the development of a unique solution for all issues that can be resolved resorting to this technology, due to the the complex adaptations that an algorithm would have to perform to be applied to all of these situations. In other words, artificial intelligence in computers is still not capable of adapting itself to the problems presented to this field of study. Because of this, developers started to include restrictions to their approaches, limiting the broadness of applicability, but also optimizing the performance for that range. If we know that we‘re dealing with only one particular type of object, then we can optimize the algorithm to handle that kind of object and although we‘re disregarding different types of objects, the method can succeed when it makes that premise. Emphasizing this issue, a decision was made to tackle two different subjects in the computer vision field, in order to demonstrate the differences between approaches and also the inapplicability

of one method over the other situation, due to the specificity in each method‘s assumptions. The work produced had, therefore, two main objectives: vibrating line and people tracking.

1.2 Video object tracking

By definition, video object tracking (VOT) is an application that tracks the displacement of several particular objects using cameras to capture a scene. The use of a particular view makes implicit the possibility of partial occlusion of the targeted object. This can be solved by the use of more cameras in the same scenario, which also widens the physical area of operability. Nevertheless, this requires the translation of the set to 3D measures and correlation between the cameras‘ positions. Such requirements can become expensive and computationally heavy for the algorithm‘s performance. Moreover, the knowledge of the position of the camera and its parameters, or camera calibration, is vulnerable to physical changes such as weather conditions (if outdoors) and even people vandalism.

The tracking of an object could be performed by detecting it in each frame, but only if one object was present in the video sequence, then computing its displacement. However, most of scenarios contain multiple objects, with the possibility of occlusion among them. To overcome this fact, other tracking techniques have to be used. Methods like optical flow, Kalman filters, texture matching and so on, are possible approaches or modules to be implemented in an object tracking algorithm.

1.3 Context of proposed methods

As said before, we want to emphasize the differences between object tracking algorithms by looking at two possible approaches. In the first method we propose a vibrating line detection and tracking and in the second we will work with a people tracking method.

1.3.1 Line detection and tracking

Nowadays, the monitoring of big structures like cable-stayed bridges or electricity distribution systems is vital for the extension of their life-span and for the reduction of system failures and consequent costs. An essential part for the monitoring of such structures is the vulnerability to weather variations and to other subjects such as trucks passing on a cable-stayed bridge. A common approach for the monitoring of such structures is to use sensors like accelerometers in order to receive information about the displacement taking place in that position. However, the utilization of such resources adds substantial costs to the structure monitoring when we wish to receive information from multiple points, not to mention that we will never get a complete information unless we install sensors consecutively throughout the line. Moreover, the installation and replacement of these devices can constitute a serious issue and might even be impossible when we consider structures such as high voltage power lines. Computer vision is considered a possible solution to

this problem, because it provides a contactless alternative. This approach also raises some issues such as the positioning of the camera, which might have to be placed away from the structure, the required high resolution for precise measurements and also the transmission of the data from the camera to the monitoring station. Moreover, one has to consider the implications in using a camera, such as the vulnerability to weather and obviously the illumination conditions during nighttime. Nonetheless, computer vision provides a valid monitoring technique in some situations usually with lesser costs associated.

1.3.2 People tracking

People tracking methods are widely associated to computer vision related applications. Surveillance has become a crucial matter of the society and also advanced interface methods using computer vision provided the field with important breakthroughs and techniques. Moreover, the increase in processing capabilities allowed developers to create people tracking algorithms with real-time interaction abilities. Nevertheless, these advances haven‘t yet allowed researchers to create a general method which would cover all range of applications and scenarios for people tracking. Still, authors are bound to establish some premises to increase the success rate in regards to specific situations. Therefore, most researches still focus in either the resilient, broad range or real time capabilities of their methods. The most successful state-of-the-art algorithms are the ones which show great value in more than one of these factors.

Developers of people tracking algorithms usually face some common issues, such as partial or full occlusion of an object, placement of the camera, weather conditions, sudden illuminance changes, among others. These problems are usually minimized by a few assumptions and by the camera‘s placement itself – if the camera is located indoors is not susceptible to weather conditions and possibly less conditioned by illuminance changes. Nonetheless, the developer has to work around these issues, trying to minimize them, trying to have an as complete as possible control over the scenario and the actions taking place.

1.4 Objectives

Methods for line detection and tracking have also been under research in some fields like robotics and 3D modeling in video sequences. We intend to analyze methods for tracking a line that is subject to oscillations. In practical situations, we can encounter this phenomenon in high voltage power lines, cables in cable-stayed bridges that support the weight of the bridge‘s deck and other similar structures, where the extraction of these characteristics can be otherwise hard or expensive. We propose a contactless method that will detect the lines present in each frame of a video sequence and then track its displacement throughout the sequence, by using the information provided by the line detection.

The second topic has caused over the time a lot of discussion in the computer vision field. Although some algorithms have proved to be successful in certain situations, it is settled that there still isn‘t a method that covers all of the people tracking scenarios. Therefore, there has been a lot

of studies concerning specific situations and also attempting to achieve some generalization. Furthermore, in recent years methods have become more robust and good results have been surfacing as we'll discuss further in this dissertation. The work in this field will consist in the integration of a new person appearance model in a state-of-the-art algorithm described in [Zhao and Nevatia, 2004]. This model originally resorts to a weighted average of the person's texture information, which can be scarce in some cases and also can be deformed throughout the object's track. The new model will rather rely on a technique described in [Teixeira and Corte-Real, 2009], which resorts to quantized local descriptors, or *bag-of-visterms*, to describe the object.

1.5 Structure of the Document

We'll begin by taking a look at some work related to the proposed methods in Chapter 2. We'll begin by presenting some work in line detection in computer vision and then go through some methods that were built with similar purpose as ours. We'll also look at some of the state-of-the-art algorithms for people tracking, where we'll also analyze the methods we worked with.

A brief description of camera calibration can be found in Chapter 3. This module is common in both developed methods, therefore a separate chapter was reserved for this topic, where we'll also describe some experiments done within this module.

Chapter 4 is reserved for the full description of the line detection and tracking algorithm and also the presentation of the respective evaluation and results. In Chapter 5, a description the integration of an experiment with a linear camera model and also the implementation of the new appearance model are presented. The conclusions extracted from the results gathered from the work developed can be found in Chapter 6.

Finally, concerning the appendixes, we have an Appendix A where we present samples taken from the dataset used for the validation of each method and present a complementary description of the dataset, to the one presented in both Chapters 4 and 5. A second appendix B is composed of a side experiment in camera calibration in the line detection and tracking context, which was not implemented in the presented approach.

Chapter 2

Related Work

2.1 Line Detection and Tracking

As mentioned in the introduction, this dissertation proposes a method for capturing the vibration parameters of line structures such as cables and wires. However, line detection and tracking is also used in computer vision for different purposes. In [Gambotto, 1989] and [Deriche and Faugeras, 1990], developers wanted to obtain characteristics present in a room, in order to develop an algorithm for a robot‘s path and they did this resorting to the edges on the walls, furniture and other objects present in the robot‘s visual space, having therefore the need to track the edges of these objects. They first extracted the lines in each frame, using edge detector algorithms. Afterwards, they tried to match the lines between frames, being aware of possible omissions in-between frames and also possible splitting of some lines. In the first method, Gambotto looked into the geometry of the lines to match them between frames. He stated that the occurrences of one line in consecutive frames have rigorous geometric differences between each other and looking at these rules, we could match such lines. In the latter algorithm, the authors decided instead to apply a Kalman filter to the lines‘ positions, in order to predict the following position.

Taking solely line detection into consideration, most of the developed algorithms use primarily edge detectors to detect lines. In [Canny, 1986], an edge detector is presented which detects the contours present in an image using a gradient filter and two separate thresholds to eliminate noise caused by the filter. However, this operation still causes false detections, specially in object detection cases where the background is not controllable. Line detection specifically targeted for video sequences is presented in [Cai and Tai, 2005]. Authors describe an algorithm for field line detection in football videos. It eliminates noise where an insufficient number of green pixels is not present, and clusters white pixels applying then a threshold to eliminate small clusters. This method relies however in high specificity cases, and doesn‘t show high promises in general situations. In [Rodrigo et al., 2006], authors look at line detection as an energy based detection rather than a purely edge detection. It first computes the Hough transform and then computes an energy based minimization over each obtained contour. This is a robust method for line detection, however it relies specially in straight line detection and energy values have to be reasonably high,

in order to take a line into consideration. A staff line detector is presented in [Cardoso et al., 2009a], used to detect lines in handwritten music scores. It treats the binary image as a graph of black and white nodes, establishing the shortest path between left and right margins of the image resorting to the paths with less cost, *i.e.*, the paths with less white nodes. Although effective in such premises, the algorithm is not fitted into other issues, as it requires lines to be straight and distributed from opposite sides of the image and also to use binary images.

Line detection is an important module in the overall method, but when considering video sequences with more than one line, one needs to track the lines in consecutive frames. Line tracking takes some importance also is in 3D modeling. In [Shen et al., 2010], the authors propose an algorithm for 3D modeling of inextensible deformable surfaces. The method tracks the surface, modeling it in a triangulated 3D mesh as it is deformed manually. This is accomplished through tracking of edges of the surface resorting to an iterative L_2-norm approximation process.

Using other techniques, there has been some research about vibrating cables, but using techniques other than computer vision. In [Magalhães et al., 2008], a method for vibration measurement of the suspended roof of a football stadium which is connected by cables is proposed. They use seismographs placed on the roof in order to get readings on the roof's vibrating characteristics.

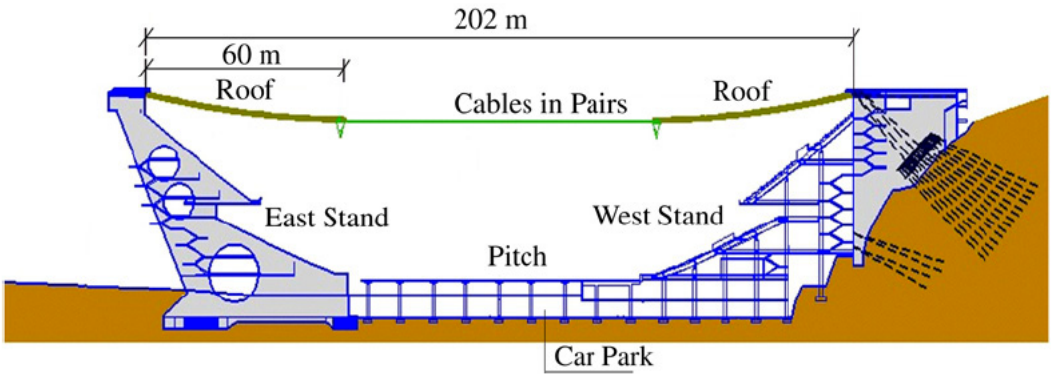


Figure 2.1: Side view of the Braga football stadium (extracted from [Magalhães et al., 2008])

A possible approach for measuring semi-rigid line vibrations is described in [Calçada et al., 2005]. In this paper, it is described a method for obtaining the vibration parameters of the cables from a cable-stayed bridge. They use load cells previously installed in the inferior anchorages of the stay cables, in order to get readings of these vibrations. Load cells are devices which translate force to an electrical signal. The force the cables were vulnerable to are therefore transmitted along the structure through wires, until it reaches the monitoring station. Other techniques use devices such as high-precision GPS antennas, in order to track the structure's position over time, obtaining its displacement. In [Çelebi and Sanli, 2002], the author presents a method of tracking the position of a tall building, in order to gather its vibrating characteristics using GPS antennas, at a sample rate of 10-20 Hz. Using this technique, the author could acquire the vibrating characteristics of a tall building, *e.g.* on a windy day. Using the same premise, the method described in [Roberts et al., 2003] uses GPS devices to obtain displacement attributes of a bridge. This proves that

such technique can be applied to different structures. However, its application to structures like cables and power lines is yet to be proven successful. The antenna‘s size doesn‘t allow an easy installation and the technique raises similar problems as accelerometers or load cells – they don‘t allow a complete description of the cable‘s displacement and its installation can, in some cases, cause high expenses and even in some cases it can be impossible, such as in high-voltage power lines. It is due to these issues, that a contactless method was thought of and so, the desire to develop a computer vision approach to this problem became reality.

The state-of-the-art computer vision techniques to obtain the characteristics of a line‘s displacement can be generalized into two different approaches. The first group of methods require the placement of a marker on the structure and the application will detect the position of the marker in each frame of the sequence. The second one requires the user to choose one specific point or a region of interest (ROI), to be tracked over the sequence of frames, establishing matches between consecutive frames.

The most immediate solution for tracking a structure‘s displacement using computer vision techniques is to place a marker on the exact point desired to track. Providing a reasonable resolution for the output of the camera, a basic pattern recognition algorithm will then identify the marker throughout the frames in the video sequence [Olaszek, 1999]. The knowledge of the marker‘s position allows the algorithm to compute its displacement over the sequence of frames. This technique is also used in different methods. In [Wahbeh et al., 2003], more than one marker were placed on the structure, in order to calibrate the camera, using the real distances between the markers, converting it to pixels in the sequence of frames. Although originally these methods track only bridge decks, one can assume the premises easily apply to different types of structures like cables in cable-stayed bridges or general lines with vibrating characteristics. In [Lee and Shinozuka, 2006], the author validated a similar approach of placing a marker on a structure using a lab test consisting in a table shaking with certain vibrating characteristics inserted by the user.

Nonetheless, this technique raises similar problems as the previous discussed approaches – although it presents a less expensive approach in relation to the use of accelerometers or load cells, it is not possible to obtain an overall characterization of the line‘s vibration and the placement of said markers on the structure can prove to be difficult or even impossible. Moreover, additional modules may be required if more than one similar marker is placed in the structure so the tracking algorithm can differentiate markers, not to mention that the aesthetics of the structure may be compromised when patterns are placed all over this structure.

Although the previous technique provided an almost contactless approach, since the only required contact was to place the marker, the desire of developing a totally contactless method was still present. With this in mind, other approaches using computer vision techniques were created, consisting on a user selecting a region of interest, normally a section of a line, that the program tracks using optical flow methods. In [Morlier, 2010], the user selects an exact point for the algorithm to track. From that information, the algorithm implements the pyramidal implementation of the Lucas-Kanade tracker [Bouguet, 2000] to track that same point over the subsequent frames. The method also allows the user to choose multiple points, keeping one track for each point over

the sequence of frames. Another approach, described in [Silva et al., 2007], uses the information of a region of interest (ROI), obtaining therefore, a more complete overview of the line‘s displacement. It computes the optical flow method described in [Horn and Schunck, 1981], for each pixel inside the ROI, which allows a more robust measurement of the vibration. Using the same premise, the method described in [Ji and Chang, 2008a] chooses arbitrarily the ROI, providing somewhat hands-free method, but not gathering all the line‘s characteristics. The same authors present a method in [Ji and Chang, 2008b] that uses stereo vision to build a more robust output of the vibrating attributes.

The technique of letting the user select a region of interest to track is the one which resembles our approach. However, it requires either the user to make a choice of the points or ROI to track or the algorithm chooses a small region arbitrarily, and unless the user chooses very carefully all the points of the line, the ROI is subject to noise on the background, such as clouds or even foreign objects passing by, such as birds or airplanes. Moreover, once the track is lost, the user has to reselect the points, or compare with trustworthy information such as the initial frame, which decreases the robustness of the method.

2.2 People Tracking

Research in the field of people tracking although being a particular case of object tracking, still finds lots of applications and in specific conditions, meaning that a general successful method for people tracking hasn‘t still been accomplished. Therefore, premises were taken, in order to create successful, although specific methods for people tracking. Techniques in this area involve not only surveillance applications, but also virtual reality, advanced interface or even motion analysis applications. Complete surveys about the several applications involving people tracking algorithms can be found in [Gravilla, 1999], [Moeslund and Granum, 2001] and [Moeslund et al., 2006].

Although usually people tracking is performed resorting to standard cameras, other more sophisticated techniques can also be used to track people. One of these solutions is to use an integrated system of laser sensors, as described in [Cui et al., 2006]. Here, the authors present a multiple laser scanner system positioned a little above ground level, where they track multiple people in crowd situations, tracking in particular each person‘s legs. Using this information, the algorithm is able to build a track and the respective path taken for each person. Another approach involving the use of multiple sensors is described in [Ikeda et al., 2006]. It uses a variety of sensors such as floor, vision and even wearable devices synchronizing the information to obtain a more robust description of the scene. These techniques require however higher costs and they are unpractical in most situations, because a general installation isn‘t always possible due to different scenarios and structures. Most practical approaches use strictly computer vision techniques, because it provides reasonable costs, due to the existence of methods using computer vision approaches which can be applied to a wide range of situations.

In the survey [Moeslund and Granum, 2001], a generalization of the people tracking algorithms is made. The author divides a general model for an algorithm of people tracking in four

phases: initialization, tracking, pose estimation and recognition.

Initialization usually involves separating the background from the foreground. The latter consists of the relevant objects to be treated. Depending on the method, this phase can also involve other modules like camera calibration.

The following steps contain many specific modules and strongly characterize the final algorithm. This specificity is strongly influenced by the environment of the scenario and the main goals of the algorithm. In [Siebel and Maybank, 2002], the method‘s purpose is to track people inside a metro‘s substation. This means that there will be an undetermined quantity of people in the image frame – either there is just a few people or a big crowd due to rush hour. In this case, pose estimation is not so crucial to the algorithm as in other situations such as movement analysis or virtual reality, because in situations of surveillance the main goal is to be aware of people‘s positions.

In [Haritaoglu et al., 2000], the authors present a method for people tracking, which shows good results in exterior environments, where there are many kinds of noise and sudden changes in luminosity. Furthermore, it is capable of identifying human body parts through the person‘s silhouette‘s contour, and its respective shape, obtaining pose estimation. The method has also the skill to recognize certain objects being carried by people and can detect when the object is being exchanged between people. For that, it is necessary that the object is clearly detected through the analysis of the person‘s silhouette, because it is through this method that the algorithm looks for silhouette‘s deformations in the person‘s usual symmetry, therefore identifying deformations as carried objects. There are, however, some limitations in the global algorithm. When a person crosses its path horizontally with another person (even with some distance between them), the program loses track of the person in front, mainly because it requires the camera to be situated at normal human eyesight level. This is a common issue in people tracking methods – because only a limited perspective of the scene is available, there‘s the possibility of occlusion and the recovery of the track after this occlusion isn‘t straight forward. Also, due to the camera‘s placement at ground level, the field of vision is narrowed comparing to the situation where the camera is positioned in high grounds.

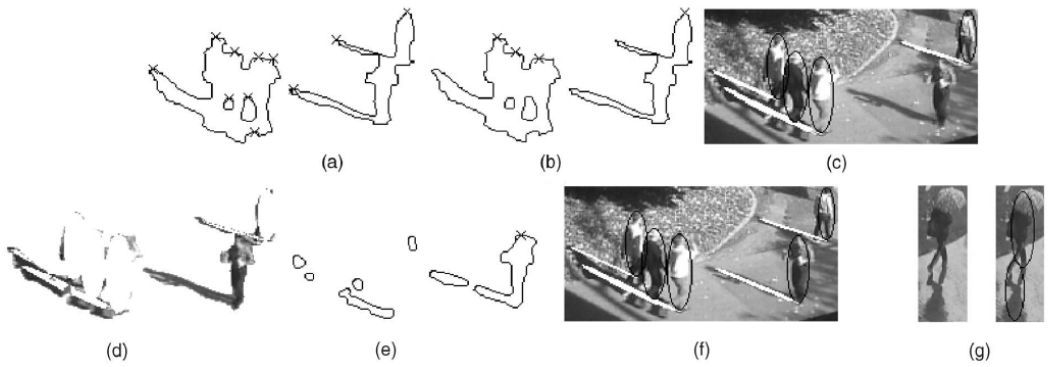


Figure 2.2: Iterative process of people identification (extracted from [Zhao and Nevatia, 2004])

In the algorithm developed by Zhao [Zhao and Nevatia, 2004], a three dimensional model of the scene is built, using camera calibration. This allows the algorithm to have a better knowledge of the space and also to help in situations of occlusion by restricting one determined place in the 3D space for only one object (two objects cannot occupy the same space in the three dimensional space). This tool is vital for obtaining the person‘s position in the scene. The process of detection consists in replacing the detected foreground by a model of a person, an ellipse. Iteratively, the method eliminates the detected people, allowing partially occluded people to also be detected. Unlike the previous analyzed method, this one requires that the camera is to be placed on high grounds, allowing a more peripheral, wider view of the space and reducing, therefore, the chances of occlusion. These possible occlusions prove to be harmful for the performance of this algorithm. If an object is occluded for a certain number of frames, the person‘s model is discarded, decreasing the overall performance of the method. Moreover, the tracking fails in case of sudden changes of illumination, since it relies in a appearance model based in color values. Furthermore, this appearance model requires the algorithm to go through every pixel of every considered position, in order to make a comparison with the weighted average model. This procedure is heavy, when the processing time is being concerned, since much of the area analyzed might not add more relevant information than what a few keypoints might.

A method for people detection and identification is presented to us in [Teixeira and Corte-Real, 2009]. Here the authors describe a method that uses a trained dataset of visual model objects. In other words, the algorithm gathers the object‘s characteristics using a *bag-of-visterms* – an histogram of local feature descriptors, learning therefore the so called vocabulary that defines the object. With this database, the algorithm can identify these people in different video sequences with high precision, resorting to stereo-view images to get more information from the scenario. The method also updates the model of every identified object, increasing its resilience. However, the SIFT transform of every object of every frame also in every view, requires a reasonable amount of processing time. Moreover, the need for a previously trained dataset makes it unpractical in many situations where random people are present in the scenario.

The appearance model for a person is an important module of a people tracking algorithm. This representation is used to match new possible positions for the person, refining its tracking. Many methods simply use a Kalman filter to predict the person‘s next position, but this prediction is often not precise in the first frames where the person appears and also if simple changes in direction and speed in the path of the person are considered.

An intuitive approach for the appearance model is described in [Zhao and Nevatia, 2004] and resorts to a weighted average of the texture values for the person‘s segment. It gives more weight to recent frames and also considers the probability that a certain pixel in the model is foreground or background. In the second part of the article, the authors present an alternative appearance model consisting in a stick-figure which allows to determine with a certain effectiveness the person‘s body pose and eliminate a person‘s shadow in outdoor situations, as the shadow does not fit this model. However, the method requires a great processing effort and high resolution to match body members to this appearance model. This implies a reasonable investment, which might not be

viable in many situations.

In [Ramanan et al., 2007], the authors propose an appearance model that consists of detecting the position of the person‘s body parts, like arms and legs, which allows a better background extraction and a better description of the person. However, the system handle only specific situations, where all the person‘s limbs are well identified. Therefore the system is expected to be inaccurate in multiple person and crowd situations.

Chapter 3

Camera Calibration

3.1 Introduction

As defined in [Tsai, 1987] "*camera calibration is the process of acquiring the intrinsic and extrinsic characteristics of a camera*". The intrinsic features are the geometric and optical parameters of the device. The extrinsic characteristics consist of the three-dimensional position of the camera and its orientation. The knowledge of these features enables us to convert 3D points to 2D and *vice-versa*. In Figure 3.1, the translation of these sets of points is represented.

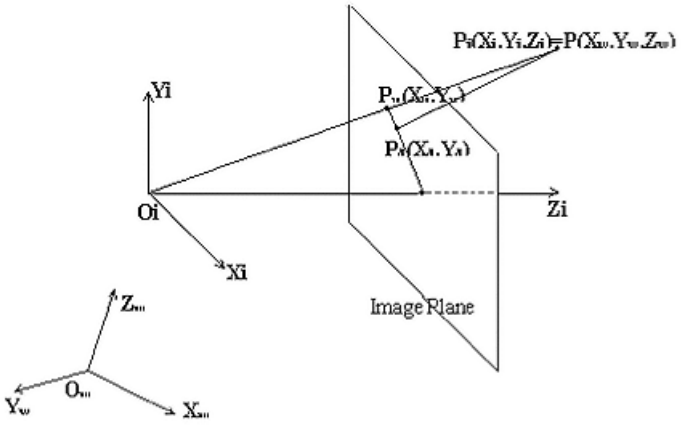


Figure 3.1: Geometrical concept for camera calibration

In [Tsai, 1987], it is presented a robust way for capturing these parameters. The calibration is achieved through the collection of a series of image frames where a standard pattern (for example, a checkerboard) is displayed as the one depicted in Figure 3.2. The algorithm detects the position of the pattern in different frames, obtaining the camera's intrinsic and also extrinsic characteristics. A similar technique is used in [Zhang, 2000] where camera calibration enables the elaboration of 3D models of the objects displayed in the video.

In the research conducted by Lv [Lv et al., 2002], it's demonstrated a self-calibration method which uses the horizon line to achieve calibration. Through these vanishing lines, and some objects

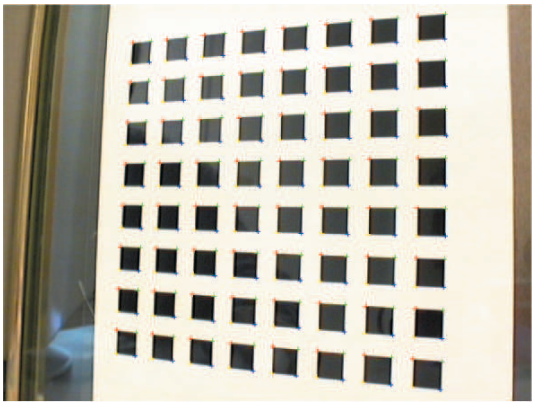


Figure 3.2: Possible pattern used for camera calibration

with known measures (like an electric pole), it‘s possible to gather the camera‘s characteristics, therefore obtaining camera calibration. A graphical demonstration of camera calibration using vanishing lines is made in Figure 3.3. Here, the vertical vanishing point of a reference object is depicted, as well as the computing of the horizon line, through the analysis of the ground plane.

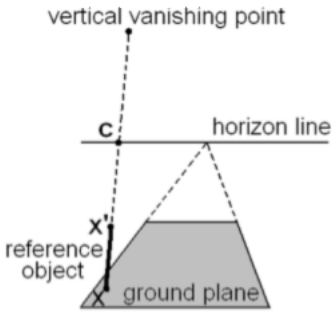


Figure 3.3: Camera calibration using vanishing lines (extracted from [Lv et al., 2002])

In some situations, camera calibration has been used not as a unique module, but as a contributing part for a system. In [Zhao and Nevatia, 2004], the method uses camera calibration to build a 3D model of the scene, in order to get a better perception of space and of the people in the scene.

3.2 Camera‘s parameters

In this section, a brief technical description of camera calibration will be presented. A more detailed discussion of the subject can be found in [Forsyth and Ponce, 2002].

As mentioned before, camera calibration is the process of obtaining both the intrinsic and extrinsic parameters of a camera installed in a certain position. The intrinsic values consist of the internal parameters of the camera and they are represented by:

$$A = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where α_x and α_y are the horizontal and vertical pixel size respectively - also known as the focal length in pixels, (u_0, v_0) is the principal point or center of the image in pixels and γ is the skew coefficient - the camera may have a manufacturing error, allowing the angle between the two axes of the image not to be 90° (the camera is skewed). Matrix A is called camera matrix and represents the overall intrinsic parameters.

The camera‘s position and rotation relatively to the scene or the extrinsic parameters are represented by a rotation matrix and a translation vector,

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \qquad t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{3.2}$$

R and t, respectively.

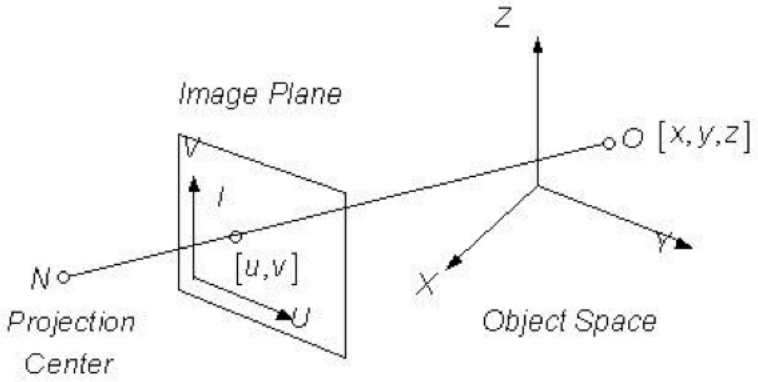


Figure 3.4: Model of 3D to 2D conversion

Considering the three dimensional space where the camera is inserted, we can build a model of this scene as depicted in Figure 3.4.

This model is called the pinhole camera model and it makes the conversion between 3D points in the scene and 2D points in the image. This conversion is accomplished using equation 3.3 that gathers both intrinsic and extrinsic parameters of the camera:

$$s\, m‘ = A[R|t]M‘ \tag{3.3}$$

where s is a scale factor, m‘ is the point (in pixels) in the image and M‘ is the point in 3D (in distance scale) in the image. From equation 3.3 we can derive

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

Note that if, for example, we impose $Z = 0$, *i.e.* we fix one projection plane eliminating one variable, then we can eliminate one column from the $[R|t]$ matrix, obtaining

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.5}$$

The derived matrix is called homography matrix, which can be used to convert points from 2D to 3D and *vice-versa*, when considering one constant projection plane,

$$s\, m\text{‘} = H.M\text{‘}, \qquad H = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \tag{3.6}$$

This equation is used to obtain object characteristics in tracking algorithms, such as line parameters and scene characteristics in people tracking in our case. Obviously as discussed, this equation only allows to get information when one variable from the 3D space is constant. Therefore, in case of the experiment with line detection and tracking, the immediate solution would be to consider the plane common to all the lines in the sequence, since all lines are in most cases parallel between themselves. In people tracking situations, this equation might be useful to get a description of the 3D measures of the floor in the scene, in order to get positions of people in the various frames.

3.3 Experiments in camera calibration

For a better understanding of the method of camera calibration, some experiments were conducted using a set of images from the CAVIAR project [CAVIAR, 2004]. The objective here was to get to know how this information could be used to our advantage, in both algorithms. Moreover, we wanted to know how precise was the camera calibration, when a few points were inserted to compute the homorgraphy matrix.

In order to translate world coordinates, or in 3D, to image coordinates we need to calibrate the camera, as we've seen before. One of the possibilities is to use the homography matrix which makes this translation but considering only one plane in the world coordinates. Using the example of a typical scenario for people tracking, if we consider Figure 3.5, the homography matrix could provide us a translation from the image coordinates to the 3D coordinates in the ground floor ($Z = 0$) and *vice-versa*. In this Figure we can see a few drawn crosses. The black crosses represent a set of points provided by [CAVIAR, 2004], that represent the ground truth from which we can extract the homography matrix. These points are also depicted in Table 3.1.

Figure 3.5: Image from [CAVIAR, 2004] with crosses from the camera calibration method

Table 3.1: Given Points

2D (*u*,*v*)	91,163	241,163	98,266	322,265
3D (*x*,*y*,*z*)	0,975,0	290,975,0	0,-110,0	290,-110,0

As a result of the extraction of the homography matrix, we obtain:

$$H = \begin{bmatrix} 8.95994 & -0.608928 & -716.099 \\ -0.197069 & -51.6092 & 12978.4 \\ -0.000202122 & 0.0225961 & 1 \end{bmatrix} \tag{3.7}$$

We can now test the precision of the gathered result, inputting some 3D coordinates and observing the result. In Table 3.2, we can see the tested coordinates followed by a mapping for reference values, *i.e.*, the correct translation composed by two values representing its height (u_p) and width (v_p) position in the image. These coordinates were collected, based on the patterns drawn in the floor, which contain a certain specific measure. In the third column, we can see

the results represented by a pair of values representing its height and width, u and v respectively, also represented by the white crosses in Figure 3.5, and finally in the last column, the comparison between the mapping and gathered coordinates, computed with the mean square error.

Table 3.2: Set of computed points for camera calibration test

3D points - input (x,y,z)	**Mapping** (u_p,v_p)	**Computed points** (u,v)	**Mean Square Error**
(0,0,0)	95,251	96,251	0.5
(290,0,0)	305,247	310,250	17.0
(145,-110,0)	208,265	210,265	2.0
(145,975,0)	166,161	166,162	0.5
(145,0,0)	201,250	204,250	4.5
(0,1085,0)	88,152	90,156	10.0
(290,1085,0)	230,151	235,156	25.0
(0,2060,0)	85,111	87,111	2.0
(290,2060,0)	197,111	200,111	4.5

3.4 Conclusion

From the results presented in Table 3.2, we can conclude that the gathered coordinates are well translated by the homography matrix, as the maximum obtained error was a mean square error (MSE) of 25, correspondent to a displacement of ≈ 7.1 pixels in that particular case and a MSE average of approximately 7.3. Therefore, this matrix provides a successful correspondence between image and world coordinates, when considering points in the ground level. As we‘ve discussed before, this provides some measuring possibilities in both line and people tracking algorithms. In line detection and tracking, if the lines are parallel between themselves, we can consider the plane where all lines are inserted. However, as it will be presented further on, most of experiments were performed with synthetic images, the decision was made not to include this module, as we couldn‘t compare the results with real measures. As for the people tracking method presented, a full calibration module was already implemented. However, as we were not provided with the calibration parameters for all sequences, an alternative camera model was implemented discussed in Chapter 5. The knowledge presented in this chapter provided crucial information for the development of this module.

Chapter 4

Line Detection and Tracking

4.1 Introduction

In this chapter, we present a method for measuring line or cable‘s movement characteristics, in video sequences featuring one or more lines. As mentioned in section 1.3.1, an alternative to the methods which use accelerometers, load cells – that required installation of devices in or on the lines was desired, in order to decrease costs or even solve some situations where the installation of such equipment was impossible. Therefore, computer vision became a possibility for a solution to this issue, because it provided a contactless approach which reduced costs. Current state-of-the-art methods were discussed in Chapter 2 and can be generalized into two approaches: (1) the use of a marker which the algorithm will track and (2) requires the user to select a region to track. The method proposed here implements an automatic line detector, followed by a line tracking that outputs the displacement of one or more lines, *i.e.*, the vibration characteristics. In Figure 4.1, we can see the system diagram that illustrates the general steps the method goes through. Having captured the images, the first step is to enhance the lines in the frames – a pre-processing module, in order to apply the line detector method, obtaining the output of binary images of the lines for each frame. With the information of the location of the lines we can apply a line tracking method on those points, in order to obtain the displacement of the points and the line from one frame to another and with this information we can compute the line‘s characteristics.

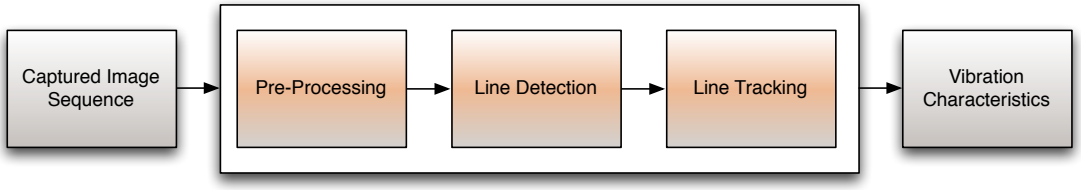


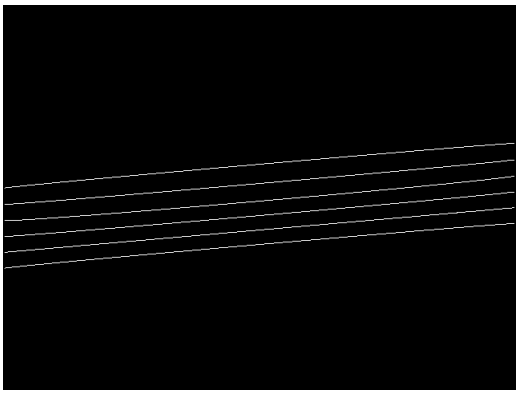
Figure 4.1: Diagram of the operations that are involved in the line tracking method

4.2 Approach Description

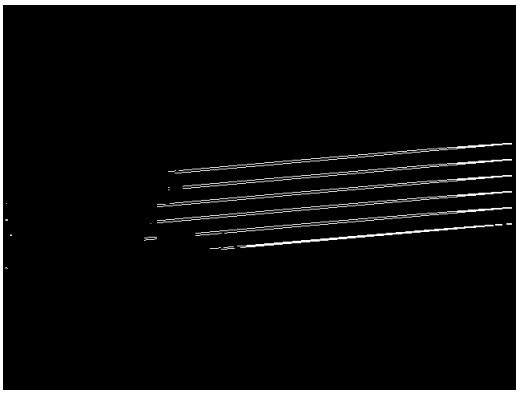
4.2.1 Stable Path

In order to understand the reasons why the gradient is used, we need to understand the stable path method and its importance in the system.

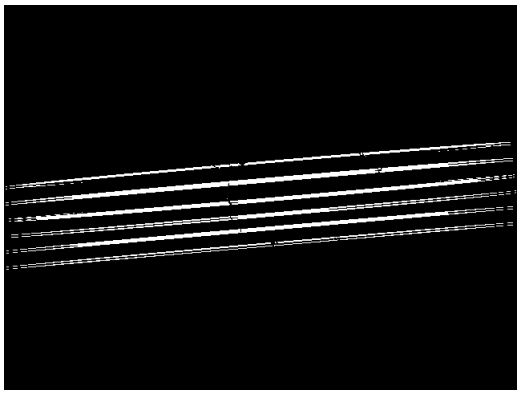
First, it is necessary to extract the location of each line in every frame, only then we can think about a method that extracts its movement's characteristics. We considered statistical segmentation algorithm to separate the foreground (the lines) from the background at one point. However, the extracted information was seldom either insufficient or provided too much residual information from older frames, as we see in the examples in figure 4.2.



a)



b)



c)

Figure 4.2: Line Segmentation: a) desired result; b) result with insufficient information; c) result with too much information

We then considered the stable path algorithm, which is a method for connecting two distant points, taking the route with less cost. The original method, described in [Cardoso et al., 2009a] and [Cardoso et al., 2008], was developed with the intent of recognizing staff lines in hand-written music score images. However, we cannot apply this algorithm directly for our purposes. As the original method relies on a pre-processing module that rather binarizes the music score sheet

producing the necessary information for that specific application, there's the need to modify this module in order to fit our premises' needs. Indeed, we could binarize each image, but the common low contrast between the line and the background doesn't allow a direct binarization of the frames. In order to eliminate this background noise, a simple gradient filter was applied, obtaining a grayscale image of the edges present in the frame, composed mainly by the lines. The other edges present in the image can be eliminated by the subsequent modules of the stable path algorithm.

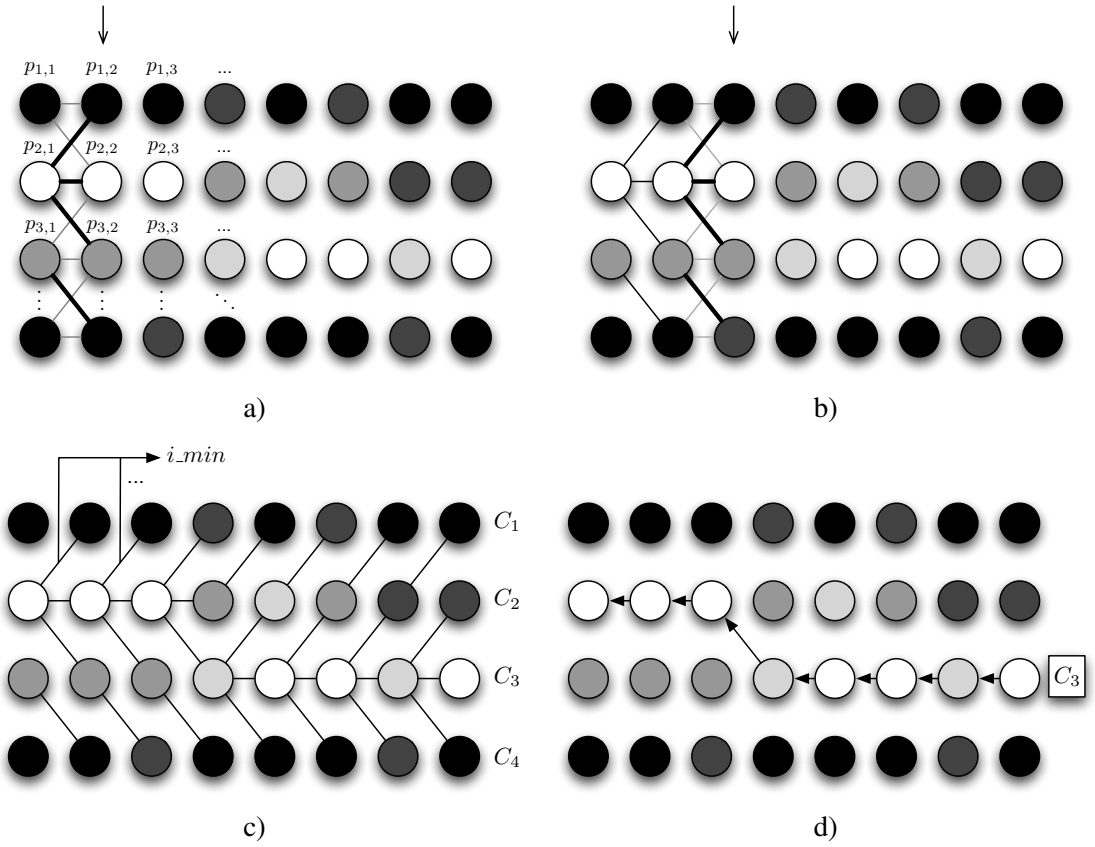


Figure 4.3: Stable path and its various phases: a), b), c) and d)

The stable path method considers the image as a graph, where pixels are the nodes and a cost is associated with each path between nodes. In our case, the cost is the value of the gradient and our line extremities would be the pixels on the left side and the pixels on the right margin, or *vice-versa*. Consider an 8x4 pixel frame of a gradient image (grayscale) as illustrated in Figure 4.3, where each circle represents one pixel and its color represents its respective gradient value. Let each pixel have the value $p_{i,j}$, where i is the number of the row, j the column number and the value is in the interval [0,1], where 0 represents the highest gradient and 1 represents the lowest. The objective is to construct a steady line between the left side pixels and the right side. We start by going through every pixel of each column as depicted in 4.3(a) and 4.3(b), and for each pixel we associate one pixel from the previous column (*previous_node*) that has a direct contact with the current pixel, *i.e.* the pixel on the previous, same or next row and on the previous column. This

decision is made according to the lowest cost (c_{min}) associated with the previous pixel's gradient value. Having made this decision, the current pixel will then inherit the previous pixel's total cost and add the cost associated with the new decision ($c_{i,j}$) and also the information of the starting point ($init\_point_{i,j}$) of that route and, obviously, the chosen row of the previous column (a), *i.e.*, a is either $i-1$, i or $i+1$ according to the lowest cost. Figure 4.4 tries to summarize this process.

```
for each column j:
    for each row i:
        c_min ← min(p_{i-1,j-1}; p_{i,j-1}; p_{i+1,j-1});
        previous_node ← c_min = p_{a,j-1};
        c_{i,j} ← p_{a,j-1} + c_{a,j-1};
        init_point_{i,j} ← init_point_{a,j-1};
    end.
end.
```

Figure 4.4: Algorithm summarizing the stable path method applied to line detection

At the end of this process, we will have a map of decisions as portrayed in Fig. 4.3(c), where the last column will have for each pixel an associated total cost, that represents the least cost of going from that pixel to the other side of the image, here represented by C1, C2, C3 and C4. Because we also inherited the pixel's starting point, we can see that all of these pixels have the same starting point, which means that the minimum cost between C1, C2, C3 and C4, represents the path with the minimum path between the two sides. The final step is to do the inverse path, from the last pixel to the first, bearing in mind the saved decisions throughout the map (Figure 4.3(d)).

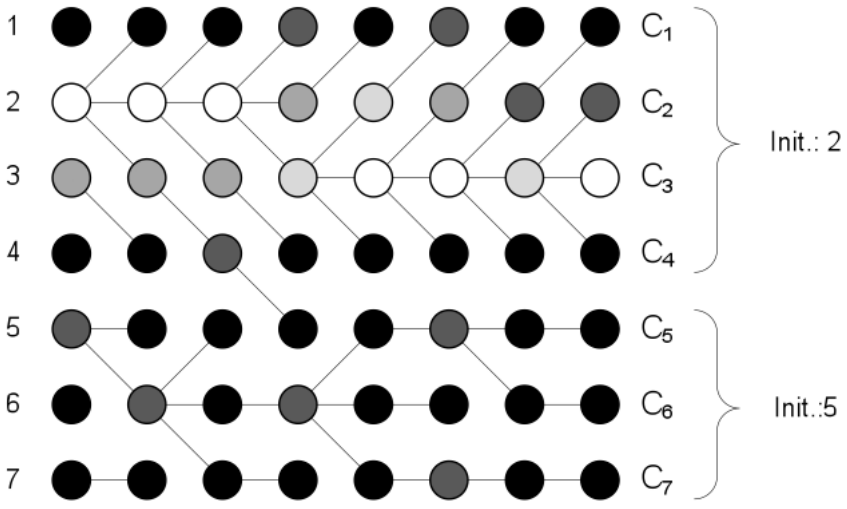
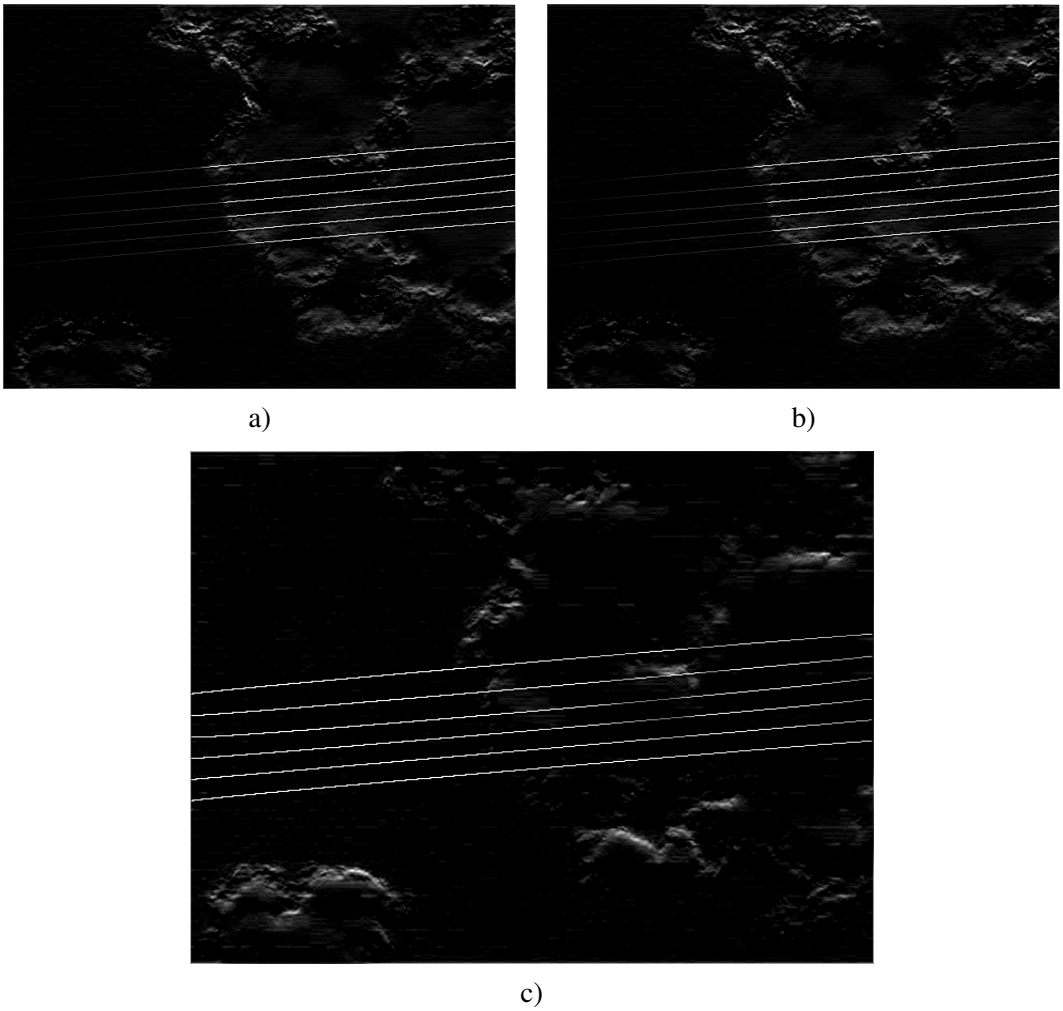


Figure 4.5: False positives in the stable path algorithm applied to the method

However, we can have various lines in one frame, and in this case we will only consider the minimum cost of each set of pixels from the last row with the same starting point.

A problem caused by this algorithm related to our specific case is depicted in Figure 4.5. In situations where a large area of the image is without any line, the algorithm detects one where there shouldn‘t be one. However, the final cost will be very high, due to the low values of gradient in that area. Therefore, to overcome this problem a threshold is applied to the final costs, eliminating the false positives.



a) b)

c)

Figure 4.6: Various spaces of colors applied to a synthetic frame, followed by a gradient filter: (a) grayscale space; (b) luminance channel of the YCbCr space and (c) saturation channel of the HSV space

In order to apply the stable path, a gradient filter needs therefore to be applied and this comes associated with some other options: (1) over which space of colors or which channel of a colorspace to apply to the gradient filter and (2) which specific filter should be used.

The immediate attempt for the first option issue to use was grayscale. However, the grayscale space causes some deficiencies in the luminance values, as we can see in Figure 4.6(a). These deficiencies obviously cause the gradient to break down in those regions. Moreover, it will cause malfunctions in the stable path method, therefore not detecting correctly every line.

Because of these problems, it was decided to try other spaces, to improve the performance of the upcoming modules. The next test was made with the luminance value of the YCbCr color space. However, this option proved also to be inefficient for our purposes, as we see in Figure 4.6(b).

Another map of values tested (now with more successful results) was the HSV color space. In particular, the Saturation channel provided robust information for the stable path, depicted in Figure 4.6(c). A few line breaks were visible, but sufficiently small for the stable path to recover them.

4.2.2 Optical Flow

Now that we have the information of all the lines present in each frame of the sequence, we need to: (1) find the corresponding line in each frame and (2) find the displacement of each line in two consecutive frames. In order to do this, we resorted to the optical flow method.

Optical flow "*is the distribution of apparent velocities of movement of brightness patterns in an image*" [Horn and Schunck, 1981]. The function of the optical flow method is to find matches between patterns in two frames, more precisely between windows of pixels. The method has some handicaps, as it doesn't handle occlusions, or great deformations in objects. However, one is not expecting the lines to be occluded and also they are not expected to suffer great deformations. Furthermore, the optical flow doesn't handle what one might refer as optical illusions. Considering, for example, a rotating barber's pole in Figure 4.7, the optical flow method will tell you the stripes in the poll are traveling vertically, but in reality the pole is rotating.

Figure 4.7: Spinning barber's pole with rotation movement but vertical optical flow

Other methods could be considered – image block matching is possible, however the texture of a line is usually uniform, which would complicate this process. Nonetheless, other algorithms could be considered as some of the future work for this project, as it is discussed further on.

Bearing in mind the discussed definition for optical flow, a single point in the line should give us the overall behavior of the line. However, optical flow provides yet another problem commonly referred as 'aperture problem'. In Figure 4.8, we can see an illustration of the problem. A rectangle with homogeneous color is moving according to two motion components: one strictly vertical, pointing downwards and another one, tangential to its length, as we see in the first image of the

illustration. If we are observing the moving rectangle through a small window, as illustrated in the second image, the observed information would not be complete, as the tangential component is 'hidden'. This is what happens when the optical flow uses only the information from the neighborhood of one point adding to the fact that the object is homogeneous. Taking information from more points doesn't, obviously, solve the problem entirely, but could minimize some noise in the tangential component of the line's movement. Some of the results presented, were performed in order to validate this statement, as it will be presented the comparison between using a single and several points to measure the displacement of the lines, which will be presented and discussed further on.

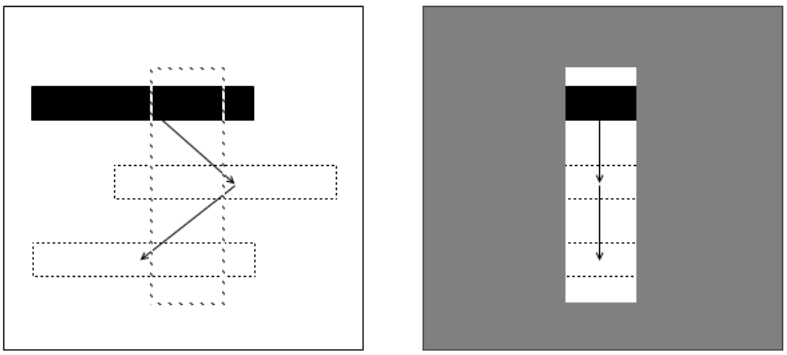


Figure 4.8: Illustration of the aperture problem

There are a few approaches for the optical flow method, that have reasonable differences and some of them will be discussed and considered. Three different methods of optical flow were tested using the information from the line detector: the Lucas-Kanade tracker, the Horn-Schunck method and the pyramidal implementation of the Lucas-Kanade tracker.

The Horn-Schunck optical flow method presented in [Horn and Schunck, 1981] presents a more global description of optical flow as the gathering of the displacement of brightness patterns. The Lucas-Kanade tracker, described in [Lucas and Kanade, 1981], presents a faster technique than previous methods, using a faster algorithm for comparing pixel windows in different frames, which consists of a minimization of the L_2 norm measure of error. Strongly based on the previous method, the pyramidal implementation of the Lucas-Kanade tracker portrayed in [Bouguet, 2000], uses a pyramid of different levels of coarseness and uses each level of the pyramid in different stages of iteration, providing more accurate results. These three approaches were tested in our method, and results from these experiments are presented further on.

4.3 Experiments with line detection and tracking

For experimenting with line detection and tracking, due to the current lack of a set of images with appropriate characteristics, a set of synthetic images consisting of six lines, each of them

vibrating at a certain frequency and with a certain peak was created. We can see one frame from this developed sequence in Figure 4.9.

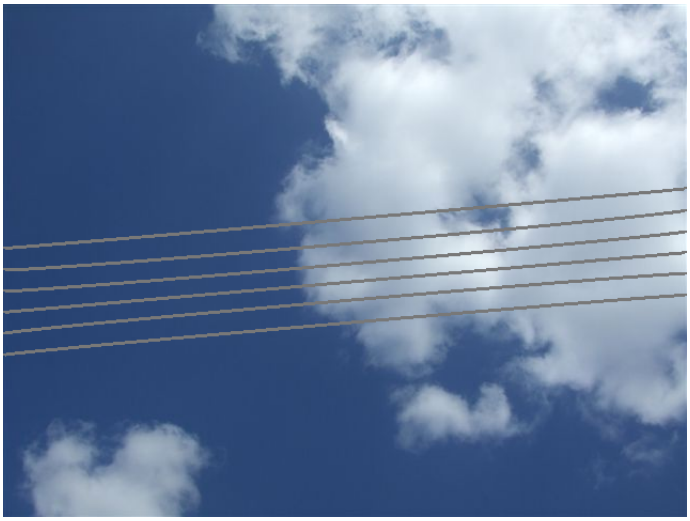


Figure 4.9: Frame from the sequence of synthetic images developed for line tracking experiments

A background was added so it could resemble a real situation, where the background is not uniform and will have to be taken in account. Moreover, each line vibrates with different peaks and maximum amplitudes and each synthesized sequence has 500 frames. In our experiments and tests, six specific sequences were used. A sample for each sequence can be seen in appendix A, in figure A.1 and also, the inserted values for each sequence are presented in table A.1. We wanted to test line detection where the lines were subject to curvature, as in the example of high-voltage power lines, where the lines are hanging between two electricity pylons, therefore present some curvature. With that objective, sequences 1 and 2 feature curved lines vibrating at different frequencies. As mentioned before, the synthetic sequences feature six different vibrating lines, and for sequences 1 and 2 the inserted values for the lines are depicted in sequence a) of Table 4.1 (in this Table, three different sets of values are described and each of the six tested sequences is described by one of this category). Moreover, the lines in the two sequences have different thicknesses, as in sequence 1 lines have the thickness of 1 pixel and in sequence 2 they have the thickness of 2 pixels.

Although sequences 1 and 2 provide a good test for line detection, the validation of the tracking method is rather problematic, as the lines are curved therefore the position of each point of the line in each line is hard to compute, so we need a set of frames that allows to compare displacements using all points of the lines. Therefore, sequences 3 and 4 were synthesized and feature six vibrating straight lines. The use of straight lines allow the knowledge of the position of each point and the displacement of all points will be similar throughout the line. In sequence 3, the lines vibrate according to sequence a) in Table 4.1, as the lines in sequence 4 vibrate according to sequence b). This allows the measure of a larger number of frequencies and amplitudes. Also, the direction of the displacement has just one direction, allowing an easier measurement and result

Table 4.1: Sets a), b) and c) of frequencies inputted in the synthetic sequences (unit of results are pixels)

	Sequence a)		Sequence b)		Sequence c)	
Line	Amplitude (pixels)	Frequency (rad/s)	Amplitude (pixels)	Frequency (rad/s)	Amplitude (pixels)	Frequency (rad/s)
1	5	0.8	5	0.8	5	0.8
2	4.7	0.85	2	0.85	2	0.3
3	4.3	0.9	4.3	0.9	4.3	0.9
4	4	0.95	1.2	0.95	1.2	0.95
5	3.7	1.0	1.5	0.05	1.5	0.1
6	3.3	1.05	3.3	1.5	3.3	1.5

analysis. As in sequences 1 and 2, these sequences possess lines with different thicknesses, where the lines in sequences 3 have the thickness of 1 pixel and in sequence 4 have 2 pixels of thickness. Sequences 1–4 have the same background as in Figure 4.9.

Sequence 5 has a background different from the previous ones. It features heavy clouds, which causes noise in the gradient image. Moreover, the lines‘ color is more similar to the background, causing low perception of the lines. This is intended to test the line detection and tracking robustness. The lines in this sequence are also straight, which allow the testing in line tracking and also possess only one direction in its displacement. The values inputted in this sequence correspond to sequence c) in Table 4.1. The last sequence (6) consists of a set of frames extracted from real live footage of a cable-stayed bridge and is intended to test line detection in real situations. In Figure 4.10, it is depicted samples from sequences 5 (4.10(a)) and 6 (4.10(b)). It becomes, therefore clear the differences between both sequences – sequence 5 is intended for testing robustness as the lines are not so perceptible and sequence 6 although composed of a few number of frames, can be used for line detection testing.

(a) (b)

Figure 4.10: Sequences 5 and 6

4.4 Evaluation Methodology

4.4.1 Line Detection

In order to validate the approach, we need a methodology to evaluate the performance of line detection. How do we measure the successfulness of the algorithm? For this, we have two sets of images for each sequence, one being the reference for the evaluation, *i.e.*, the masks of the lines synthesized together with the original synthetic image. The second is simply the set of images that feature the lines extracted using the stable path algorithm. Using this pair of sequences an evaluation can be performed, which will be now described.

Because we‘re working with a multiple line detector, first it‘s necessary to establish matches between the obtained and reference lines, so we can know which line in the reference corresponds to which one in the stable path output. Let‘s consider the reference image R, and the output from the stable path image I. Let R have m lines and image I, n lines, and D_f a $m \times n$ matrix of distances between lines in frame f. For each point in each line a of image R,

$$d_{a,b,i} = min(p_{b,j}) \tag{4.1}$$

where $d_{a,b,i}$ is the minimum distance from point i in line a and all the points (j) in line b. Let then,

$$d_{a,b} = \frac{1}{\sum_i p_{a,i}} \sum_i d_{a,b,i} \tag{4.2}$$

where $d_{a,b}$ is the average distance between a and b, $\sum_i p_{a,i}$ the number of points in line a. With the average distances we can build matrix D_f:

$$D_f = \begin{bmatrix} d_{1,1} & \dots & d_{1,n} \\ \vdots & \ddots & \vdots \\ d_{m,1} & \dots & d_{m,n} \end{bmatrix}. \tag{4.3}$$

In D_f, we can see all the average distances between every reference and detected line in frame f. Resorting to these average distances, we can now match the lines in images R and I. This is accomplished using the Hungarian algorithm, which finds the matches between lines in matrix D_f, using the minimum possible cost for the available distances. If $m \neq n$, or in other words if the number of lines in the reference image is different than the number of lines in image I, a penalty (pen) is given to the evaluation and multiplied according to the difference between m and n. Let then H_f be a $m \times n$ binary matrix, which contains the output of the Hungarian algorithm and, therefore only has one not-null element per row and column. Let E_f also be a $m \times n$ matrix of the result of the evaluation with its coefficients given by equation 4.4.

$$E_f(x,y) = D_f(x,y) \cdot H_f(x,y) \tag{4.4}$$

The not-null elements of matrix E_f will represent the distances between matched lines in images R and I. From matrix E_f, we can take two measures: (1) the average distance between matched lines and (2) the maximum possible distance considering the average distances between points of the reference and obtained lines, *i.e.*, the Hausdorff distance.

$$\bar{d} = \frac{1}{N} \sum_f \frac{sum(E_f) + |m-n| \cdot pen}{min(m,n)} \tag{4.5}$$

where $\bar{d}$ is the total average distance, N is the total number of frames in the sequence, $sum(E_f)$ is the sum of the elements the matrix E_f and pen is the penalty given for each line not found or extra line found in the frame. If this happens, then $m \neq n$ and the minimum of the two elements will be the divisor of the fraction, increasing the error. As for the Hausdorff distance, it is simply given by

$$d_H = max(max(E_1), max(E_2), \ldots, max(E_N)) \tag{4.6}$$

where each $max(E_x)$ is the maximum element in each E_x matrix.

4.4.2 Line Tracking

For the validation of the line tracking, we can only resort to sequences 3, 4 and 5 of the dataset, as mentioned in section 4.3. Using these sequences, every point in each line has the same movement characteristics, *i.e.*, the same characteristics as the inputted values. We want, therefore, to compare the inserted values to the ones gathered by the tracking procedure. The number of points to consider could be chosen by the user, as the line in the sequence could not have the same behavior throughout its length. As mentioned in section 4.2.2, the effects caused by the aperture problem in optical flow is greater when less points are considered. Also, because the line has a uniform texture, the obtained displacement for a single point is often pointing to a distant point in the second line, as there is no perceptible difference between the two points for the optical flow method. For this reason, we computed the displacement for each point in a line ($v(i)$):

$$v(i) = \begin{cases} \frac{1}{W_n} \sum_{j=i-(W_n-1)/2}^{i+(W_n-1)/2} op(j) & \text{if } i - \frac{W_n-1}{2} > 0 \text{ and } i + \frac{W_n-1}{2} < S \\ op(i) & \text{otherwise} \end{cases} \tag{4.7}$$

where W_n is a small window of neighbor points (with an odd number of elements) intended to smooth the value of the optical flow for each point ($op(j)$). Then, we take the median ($\tilde{v}$) of

the vector v – ordering from lowest to highest value into vector v‘, as the displacement of the considered segment S of the line, *i.e.*:

$$\tilde{v} = \begin{cases} v‘(S/2), & \text{if } S \text{ is even} \\ (v‘(\frac{S-1}{2}) + v‘(\frac{S+1}{2}))/2 & \text{if } S \text{ is odd} \end{cases}, \quad v‘ \rightarrow v \text{ sorted from lowest to highest value.} \tag{4.8}$$

Having this value $\tilde{v}$ for each line and each pair of frames (displacement from one frame to the other), we can now compare it with the inserted values. Using the Root Mean Square Error, we can compare both reference r_f and obtained $\tilde{v}_f$ vectors considering the line‘s displacement from frame $f-1$ to frame f, *i.e.*,

$$RMSE = \sqrt{\frac{\sum_{f=1}^{N-1} (\tilde{v}_f - r_f)^2}{N-1}} \tag{4.9}$$

4.5 Results

Having performed the evaluation described in the previous section, we will now present the results extracted from this process. As with the experiments, the results are two-fold, *i.e.*, for line detection and line tracking.

4.5.1 Line Detection

In Table 4.2, we can see the results computed from the line detection experiments. The results are presented in percentages, where the reference distance, *i.e.*, the image‘s diagonal corresponds to 100%. There, we can see a very small variance for a non-null value, *i.e.*, most of the maximum distances are close to the average distance. This is due to a residual error that is expected. With appropriate post-processing, it is expected that this error factor can be reduced or even eliminated. Sustaining this affirmation is the value for variance obtained – ≈0.00%.

Table 4.2: Results for line detection

	Average Distance	**Hausdorff Distance**
Seq. 1	0.13%	0.13%
Seq. 2	0.13%	0.15%
Seq. 3	0.13%	0.15%
Seq. 4	0.14%	0.15%
Seq. 5	0.20%	1.21%
Seq. 6	0.037%	0.067%

Sequences 2 and 4 have a slight increase, due to the fact that we are considering reference lines with a larger thickness. Sequence 5 has a higher value for both average and Hausdorff

distance, however, it should be taken into consideration the high noise in the background and the low contrast present in the frames, which complicates the line detection. A direct conclusion is however harder to take from sequence 6 (the sequence containing real images), due to the manual drawing of the reference masks. Nonetheless, the success of line detection in real footage becomes clear. All obtained values are below the average results obtained for the synthetic sequences, as the maximum distance was 0.067%.

4.5.2 Line Tracking

First of all, we present the results used to make the decision between the three considered optical flow methods. In Table 4.3, we can see a clear reduction of the root mean square error when using the pyramidal implementation of the Lucas-Kanade Tracker. These results are an average of results taken from sequences 3 and 4 and obtained when all the points in a line are considered. The presented values are composed by the average root mean square error (RMSE) for each line and, in the last column, the average of all lines.

Table 4.3: Comparison between different optical flow methods

RMSE Averages (pixels)	Line 1	Line 2	Line 3	Line 4	Line 5	Line 6	Total Average
Lucas-Kanade	2,698	1,885	2,737	1,602	1,389	2,862	2,196
Horn-Schunck	2,671	1,847	2,709	1,573	1,334	2,838	2,162
Pyr. Lucas-Kanade	0,374	0,382	0,425	0,422	0,271	0,466	0,390

When taking this table in consideration, the immediate decision was to use the pyramidal implementation of the Lucas-Kanade tracker, because of its clear improvement to the line tracking, when compared with the other approaches.

As mentioned before, the experiments in line tracking involved testing different number of points involved in the process. In Table 4.4, we have the results gathered from the tests. Here, we can see the results using different windows of points over the line, which increase from the left to right and also the results for the different components of the displacement (vertical and horizontal). The horizontal values should be considered noise, as the direction of displacement of the lines in these sequences were established as vertical. A great part of the improvement visible in the line tracking is due to a better approximation of the horizontal component of the signal. This is a consequence of the problem we‘ve discussed in section 4.2.2 – the aperture problem and the homogeneity of the lines. With a greater number of points taken into consideration, a decrease in the RMSE is visible. Nevertheless, a general improvement of the vertical component of the displacement is also present.

In Figure 4.11, we can see a clear decrease of the total root mean square error when we consider more points in the line.

Although having a good approximation to the line‘s displacement is already a good validation of the method, the computation of the frequency of vibration is often taken as the main output to

Table 4.4: Results for line tracking

		Window Size					
		1 points	3 points	11 points	101 points	301 points	all line
	Direction	**RMSE**	**Error gain relatively to the 1 point approach (%)**				
Seq. 3	horizontal	1,2162	-2,45%	-13,72%	-54,52%	-87,16%	-90,72%
	vertical	0,7935	0,03%	4,32%	-7,10%	-39,01%	-60,26%
	total	1,7332	-1,39%	-7,46%	-35,74%	-68,98%	-80,48%
Seq. 4	horizontal	1,8527	-6,34%	1,05%	-44,75%	-69,94%	-91,23%
	vertical	0,4632	-2,65%	-12,98%	-28,38%	-42,18%	-29,43%
	total	1,9421	-5,97%	-0,42%	-42,59%	-66,79%	-80,22%
Seq. 5	horizontal	0,8947	-0,60%	-4,45%	-39,32%	-61,24%	-81,86%
	vertical	1,1645	-0,13%	0,93%	-15,76%	-10,78%	-7,43%
	total	1,8051	0,22%	-0,79%	-25,97%	-31,18%	-38,90%

take from this type of methods. In Table 4.5, we can see the calculated frequencies, based on the displacement values presented before. These values were calculated using the average of frames between each positive peak of optical flow for each line. Although the results presented are only gathered from single frequency vibrations in synthetic sequences, the small RMSE presented in the previous results leaves reasonable assumptions that multi-frequency vibrations would also lead to successful results in frequency retrieval.

Table 4.5: Frequency outputs (in rad/s) from sequences 3, 4 and 5

Seq. 3	
Reference	Obtained
0,8	0,800
0,85	0,848
0,9	0,901
0,95	0,951
1	1,00
1,05	1,051

Seq. 4	
Reference	Obtained
0,8	0,801
0,85	0,848
0,9	0,902
0,95	0,950
0,05	0,0502
1,5	1,502

Seq. 5	
Reference	Obtained
0,8	0,800
0,3	0,300
0,2	0,200
0,95	0,950
0,1	0,100
1,5	1,501

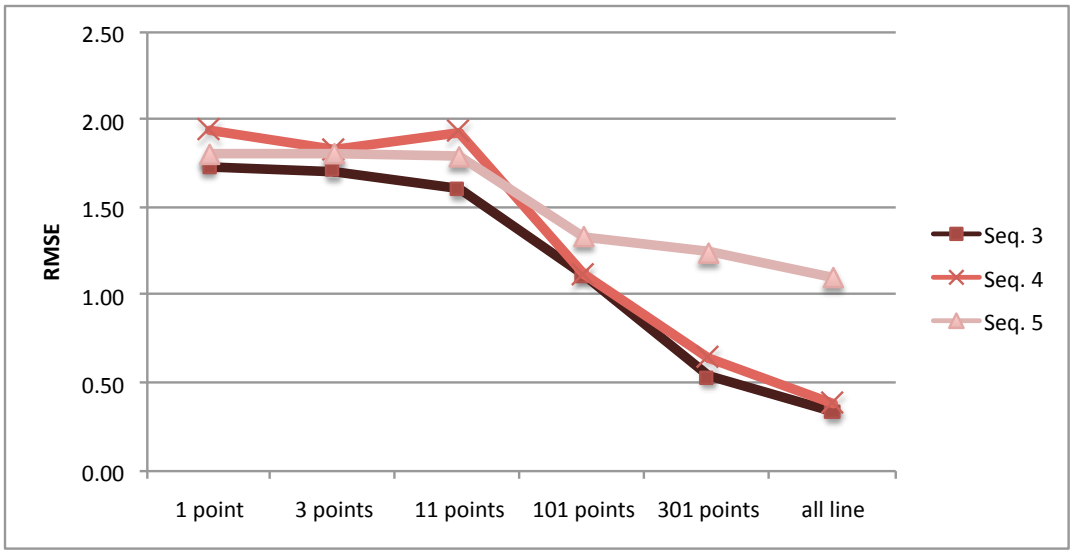


Figure 4.11: Total error using different window sizes

Chapter 5

People Tracking

5.1 Introduction

As mentioned in section 1.4, the research in people tracking conducted in this dissertation focused on an algorithm previously developed by Tao Zhao described in [Zhao and Nevatia, 2004]. More specifically, the work consisted of experimentations with different appearance models. In the original method, this model was based on a weighted average of the object‘s texture, together with a foreground probability. This requires the algorithm to go through every point of both this model and every point of every possible position, therefore causing a possible inefficiency and respective a increase of the processing time. Furthermore, as the texture values for the object can be very similar (*e.g.*, the person has clothes with the same color), this high processing might not be advantageous as an appearance model.

To keep track of an object, an algorithm has to learn some of its characteristics and have enough tools to be able to successfully identify this person throughout the video sequence, while dealing with possible full or partial occlusions. Although the original algorithm by Zhao handles object identification through position prediction, texture templates and foreground probability, we believe this appearance model could be substituted so that the success rate is increased and there would be a decrease in processing time. We therefore propose to adapt the method described in [Teixeira and Corte-Real, 2009], where it is proposed an algorithm for object matching in video sequences using multiple views, to work in this single camera algorithm.

The method presented in [Teixeira and Corte-Real, 2009] focuses on object matching across multiple cameras and constructs a model for a person based on a local descriptor and a *bag-of-visterms* (BoV) representation, building a vocabulary for each person and subsequently, a visual object model. This vocabulary is built resorting to an interest point extractor and SIFT descriptor, being the latter presented in [Lowe, 2004]. This allows the algorithm to use a lower number of points, but better defining the object rather than requiring to compute every point. However, this proposed method does not perform tracking, as it only identifies every person in every frame resorting to a database of previously trained objects. In Figure 5.1, we can see an example of the visual object model updating and matching functions of the method described in [Teixeira and

Corte-Real, 2009]. Note how an object can be partially occluded and then merged with a complete model, updating the information.

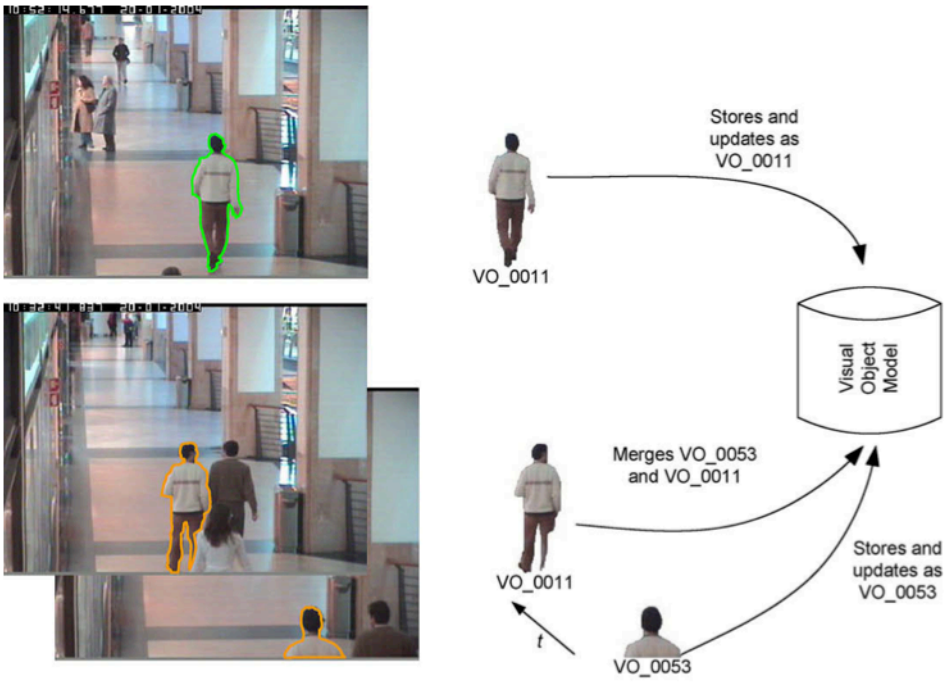


Figure 5.1: Matching and updating of the visual object model (extracted from [Teixeira and Corte-Real, 2009])

We propose an integration of the appearance model presented by Teixeira in Zhao‘s method, in order to experiment with the approach‘s resilience and performance. Instead of computing the appearance matching using every point of the object, the proposed integration would extract a signature from both previous position and possible next position, comparing values and making a decision upon them.

First, we‘ll describe the dataset used for the experiments and validation of the methods, then we‘ll take a look at some experiments made involving Zhao‘s original method and finally we‘ll go through the steps taken to implement Teixeira‘s appearance model into Zhao‘s algorithm.

5.2 Dataset

For the validation of the proposed method, some sequences extracted from [CAVIAR, 2004] (the CAVIAR project) and the PETS 2006 workshop [PETS, 2006] were used. These sequences are often used by the computer vision community for method validation, therefore providing a wider comparison to other developed approaches.

The first set is comprised of two sequences from the CAVIAR project. One sample extracted from one of these sequences is depicted in 5.6(a). Both sequences present the same scenario, with different actions taking place. They are labelled as Caviar1 and Caviar2 in the results section.

The second set is composed of two sequences used in [PETS, 2006], which consist of the same scene recorded in different angles. These sequences have a higher resolution, therefore requiring a higher processing effort. The labels for this set are PetsCam3 and PetsCam4.

In appendix A, a description of the available information is presented in table A.2 and a sample for each sequence used in Figure A.2.

5.3 Alternative model to camera calibration using a linear camera model

Part of the dataset available didn't provide camera calibration parameters required by the original algorithm, but only a few 2D points together with the corresponding 3D points for the ground plane. Therefore, as the points provided only enough points to compute the calibration for the ground plane, an alternative model should be implemented. Moreover, a new camera model could provide a better processing time, as a lot of computation is involved in the initialization of the original model, and also for some of the procedures taken throughout the sequence. For all these reasons, the first part of work with the people tracking method was devoted to some experiments that were performed in order to compare the camera calibration approach to a simpler linear camera model.

In the original method, camera calibration was essentially used to compute the height of a person in pixels, based on the person's position in the image and the height of the foreground object, gathered from the segmentation and head detection modules of the algorithm. Therefore, we argued that a method that could also do this, resorting to other means, could also be a valid technique for the module. Our approach is based on a linear description of the scenario where the people are inserted.

In Figure 5.2, we can see a representation of this camera model. In image a), we take the position of the woman in the right upper corner, as well as in image b) for the same woman, now in the lower center position. With this information we can compute a linear transformation represented in image c). In this image, we can see the same woman in the two positions and it becomes clear the change in the person's height that depends of the position in the image. Indeed, for this method to be functional, the camera should be at a certain height, so the position of the person in the vertical axis is in direct proportion with the person's height.

So, if we have the person's feet positioned at the height $f1$ and its head at the height $h1$ in the first frame, and then in the second frame with its feet at $f2$ and head at $h2$, we can compute the height in pixels at any point y using the following linear equation:

$$height = \begin{cases} \frac{|f2-h2|-|f1-h1|}{f2-f1}(y-f1)+|f1-h1| & \text{if } y \text{ is at the feet of the person} \\ \frac{|f2-h2|-|f1-h1|}{h2-h1}(y-h1)+|f1-h1| & \text{if } y \text{ is at the head of the person} \end{cases} \tag{5.1}$$

a) b)

c)

Figure 5.2: Representation of the approximation taken in the new camera model approach: a) frame 105, b) frame 371 and c) overlap of frames 105 and 371, extracted from [PETS, 2006]

where we‘re basically finding the slope of the red lines of Figure 5.2c), and applying it to the position y, whether it is represents the feet of the person or its head.

In Table 5.1, we can see the differences in processing times when using each of the discussed methods. Although we‘re dealing with a linear model, the computational time required proved to be 60% to 160% higher. Nonetheless, because we don‘t have camera calibration information for all sequences, we need to prove that this model is feasible, in order to use it to the remaining sets of frames.

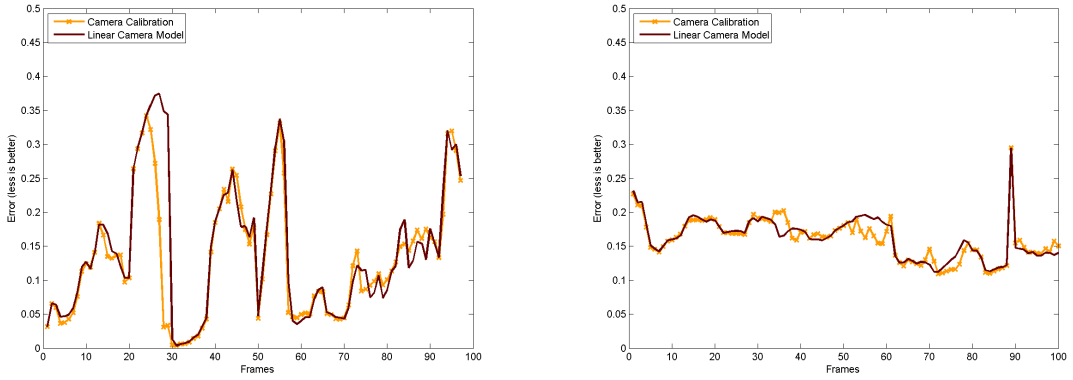
Table 5.1: Processing times using Camera Calibration vs. Linear Camera Model

Sequence	Camera Calibration $(sec/frame)$	Linear Camera Model $(sec/frame)$
PETS Cam3	0.460	0.718
PETS Cam4	0.373	0.972

In Figure 5.3, we can see the error produced in a few number of frames, using the sequences

extracted from [PETS, 2006]. This error is computed using the metrics described in [Cardoso et al., 2009b].

Although there are significant differences in some frames, overall the two models have an approximate behavior. Therefore, although there‘s an increase in the processing time, this method can be used in sequences where we are not provided with camera calibration information.



a) b)

Figure 5.3: Error comparison between linear camera model and camera calibration for sequences: a) PETS Cam3 and b) PETS Cam4

Therefore, for the results, the original baseline to which the gathered results will be compared is comprised of the information obtained when using the method described in [Zhao and Nevatia, 2004] for the sequences extracted from [PETS, 2006]. As for the sequence from [CAVIAR, 2004], it‘ll be used Zhao‘s algorithm but now using the new linear camera model.

5.4 Implementation of local descriptors as an appearance model

The main assignment for the work involving the people tracking algorithm was to implement an alternative appearance model as an attempt to improve the person‘s description and processing time. This appearance model would be inspired in the one described in [Teixeira and Corte-Real, 2009]. The differences between the premises of the methods and the problems associated with the implementation of this new method are discussed in section 5.1.

First, we should identify where the modification is going to be performed. The tracking module of the original method consists of a Kalman filter to predict the next position basing itself on previous ones. The module then searches for an optimal match in a neighborhood of the estimated position. The best match is found comparing each possibility with a template that represents the texture model of the person, based on previous positions and a foreground probability based on the segmentation values also of the previous positions. Instead, the implemented module takes every possible position around the position predicted by the Kalman filter and instead of comparing it to a template, it would compare its signature with the signature of the previous frame. This signature is a description of the person‘s image, which consists on a *bag-of-visterms*, or a histogram

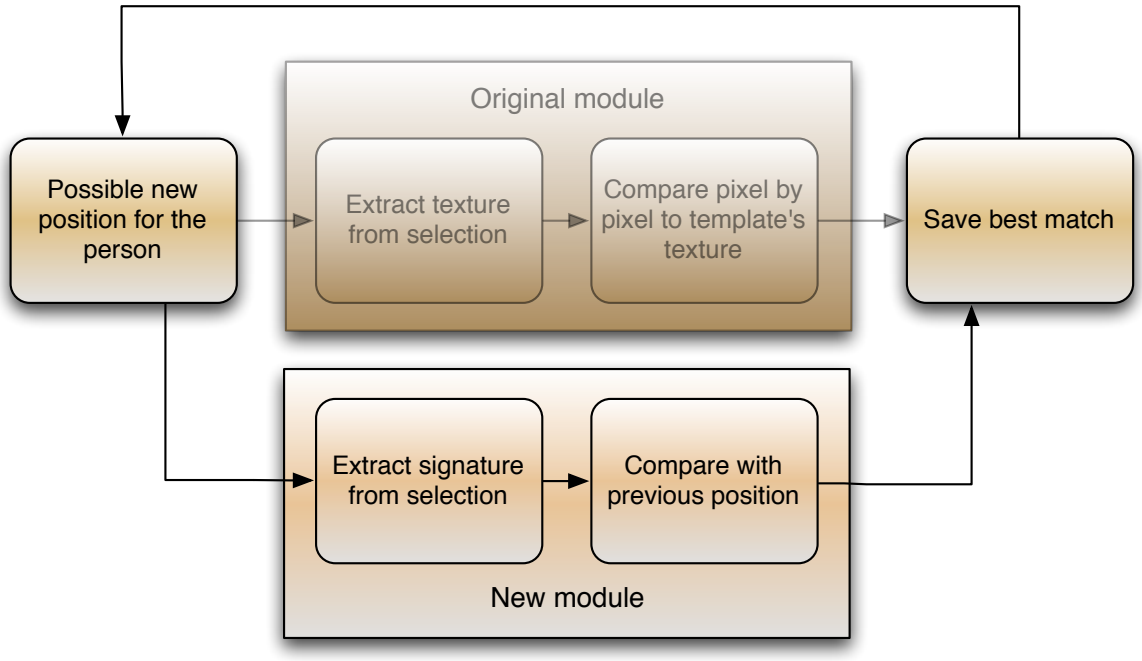


Figure 5.4: Implementation of the new signature extractor module

of quantized local descriptors. Basically, a BoV is a vocabulary based on SIFT descriptors [Lowe, 2004]. In our case, there's no previously trained object dataset, as opposite of what happens in the case presented in [Teixeira and Corte-Real, 2009], the objects to be tracked are not previously trained, therefore a generic vocabulary is required.

In Figure 5.4, we can see a diagram explaining this procedure. The comparison between the two signatures, which are represented by histograms, is made using a χ^2 metric [Erdem et al., 2004], where the histogram of the previous position H_{prev} and the tested histogram H_t are compared using the following expression:

$$\chi^2(H_{prev}, H_t) = \frac{\sum_{j=1}^{B} \frac{[r_1 H_t(j) - r_2 H_{prev}(j)]^2}{H_t(j) + H_{prev}(j)}}{N_{H_t} + N_{H_{prev}}} \tag{5.2}$$

where

$$N_{H_t} = \sum_{j=1}^{B} H_t(j), \quad N_{H_{prev}} = \sum_{j=1}^{B} H_{prev}(j), \quad r_1 = \sqrt{\frac{N_{H_{prev}}}{N_{H_t}}} \quad \text{and} \quad r_2 = \frac{1}{r_1}$$

and where H_t and H_{prev} have B elements.

Now that we have the methods to compare signatures and respective histograms, we need to decide which part of the ellipsoid we should input to the method. The signature extractor uses a rectangle window as the input. It was decided to use the middle part of the person, as the upper and lower part often contains a reasonable amount of background information because the legs and head don't occupy totally the ellipsoid. Therefore, the essential information relative to one person is in the middle part of the ellipsoid, as the upper and lower part won't give us resilient

information, as they have lower probability of being foreground, therefore providing less vital information.

As a result, we experimented with different sizes of windows, similar to what is described in Figure 5.5. In this figure, we can see the ellipsoid, and the region extracted from it. Part of the outside of the ellipsoid is also inserted to the signature extractor, which can behave as noise. For this reason, we experimented the algorithm with different sizes for this window, where this size would be related to the ellipsoid height, which is represented by the dashed line in the figure.

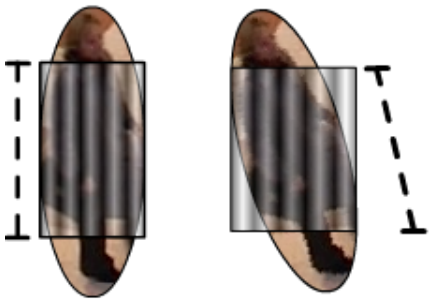


Figure 5.5: Extracted part of ellipsoid for signature extraction

Another inserted value to the signature extractor module is the number of interest points to be used by the same module. This value is initialized as a percentage of the points (P) in the rectangle, as the ellipsoid's height varies with its position in the image. The objective here is to get to know how many points are necessary to get a reasonable description of the person. Obviously, a trade-off has to be made between the processing time needed and the performance obtained when a certain number of points is used. A small number of points might decrease the time needed for processing but it also can prejudice the algorithm's performance. Moreover, in order to provide a valid description of the object, the signature extractor requires a minimum of points (pt_{min}) to be analyzed. Therefore, a threshold was set in order to comply with this premise, depicted in equation 5.3.

$$P = max(P, pt_{min}) \tag{5.3}$$

5.5 Evaluation methodology

For the evaluation of the modifications tested, the results were compared to reference information, also known as ground-truth. Specifically, two types of ground-truth were considered: coarse position and dimensions of the objects in the form of bounding boxes; exact information of the objects silhouette, also known as reference segmentations. The reference information or tracking results represented in the form of bounding boxes is conveyed using the CVML [List and Fisher, 2004] information, which is a language that translates the position of each person as its bounding box

(a) (b) (c)

Figure 5.6: Information of the position of each person: a) original frame, b) position information and c) intersection of both

for every frame. For the CAVIAR and PETS sequences considered, a subset of frames with reference segmentations exists. However, only the CAVIAR sequences possess the CVML reference information. Hence, two evaluation methodologies were considered to take advantage of the most information available. For the sequences, and for the existing reference segmentations, the metrics described in [Cardoso et al., 2009b] were computed. Moreover, for the CAVIAR sequences, the CVML information was used to compute the metrics described in [Black et al., 2003]. A review of tracking metrics is presented in [Carvalho et al., 2010].

5.6 Results

In this section, the results that were gathered from the experiments described in the previous sections are presented. We'll mainly take a look at the results from the integration of the new appearance model and how they change when imposing different window sizes and number of interest points for the region of interest of the person. These values are in percentage, translated into the portion of the ellipsoid considered in the case of the window size, and the fraction of points considered inside that window in the case of the number of points. These results will be compared with the ones observed for the baseline, *i.e.*, the original method.

Table 5.2: Processing times results and respective gain relative to the original processing time (0.46 sec/frame)

	PetsCam3								
Window size (%)	25			50			75		
No. of points (%)	1	5	10	1	5	10	1	5	10
Processing Time (sec/frame)	44.4	46.3	54.4	50.7	95.0	107.3	54.5	125.5	254.8
Gain	96.5	100.7	118.2	110.2	206.5	233.3	118.4	272.8	553.8

In Table 5.2, the processing times are presented in seconds per frame. In Table 5.1, the average processing time for a frame was already presented for the original algorithm – 0.460 s/frame. Observing these values, we can conclude that the fastest algorithm is still more than 96 times

slower than the processing time obtained for the original algorithm and far from considered to be functioning in real-time. This, obviously represents a serious factor in considering the validation of the algorithm, as people tracking methods should present a close to real-time performance. A closer look into the distribution of processing time over the modules, led to the conclusion that the signature extractor module in consuming approximately 91.1% of the processing time per track and frame, therefore leading to the conclusion that the code implemented from [Teixeira and Corte-Real, 2009] and not optimized, is the module consuming the most processing time. Nonetheless, comparing the obtained processing times, the decisive factor seems to be the number of points, as we see a higher increase when changing that parameter, comparing to the window size.

Table 5.3: Error obtained in relation to the baseline

Window size (%)	25			50			75		
No. of points (%)	1	5	10	1	5	10	1	5	10
PetsCam3 (%)	22.24	23.64	9.15	8.67	7.62	5.17	7.89	9.03	10.12
Caviar2 (%)	10.96	10.96	7.84	6.49	5.39	4.77	5.64	-	-

Nevertheless, not only the processing time is essential, as optimizations can be conducted, but also the results concerning the performance of the algorithm should be considered. These are presented in Table 5.3 for sequences PetsCam3 and Caviar2. The values displayed are relative to the results gathered from the original method using the evaluation method described in [Cardoso et al., 2009b]. So, for instance, concerning the PetsCam3 sequence, for a window size of 50% and using 1% of the number of points, the error obtained is 8.67% higher than the original result. It's visible a slight decrease in the error for both sequences, as the window size increases, which is caused by the use of more information.

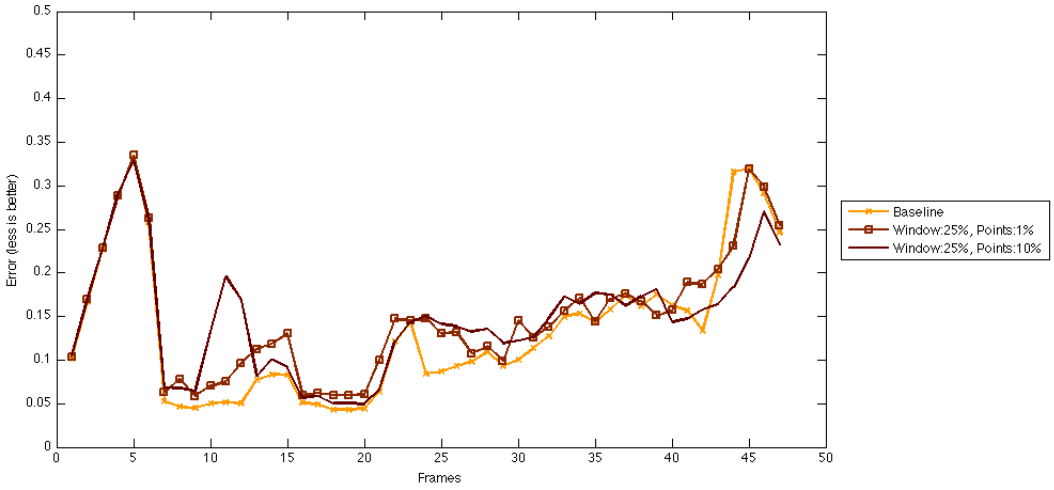


Figure 5.7: Error variation when using different number of points (PetsCam3)

In Figure 5.7, the output provided by the evaluating method are presented, concerning the results when using a window size of 25% in the sequence labelled PetsCam3. It becomes clear

that the results are quite similar, although a slight decrease is also visible. A similar error function is also depicted in Figure 5.8. In this Figure, the various results for the number of points at 1% are presented, for the same sequence.

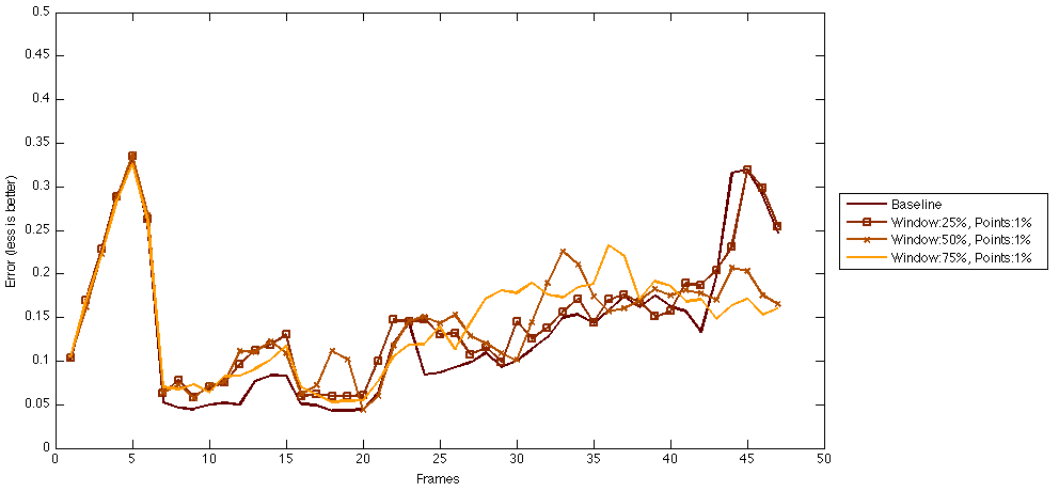


Figure 5.8: Error variation when using different window sizes (PetsCam3)

Analyzing the processing times together with the error function obtained, both number of points and window sizes are important for the good performance of the algorithm although a good performance is translated into a increase in processing time. However, the decisive factor seems to be the number of points used for the object description, since the window size increase does not correspond to a direct decrease in the method‘s error.

Table 5.4: Results from CVML information for the Caviar1 sequence

Window Size	Number of Points	Detection Rate	Tracker Detection Rate	Accuracy	False Negative Rate	False Alarm Rate
Original algorithm		69.4%	0.792	2.61	30.6%	10.1%
25%	1%	62.1%	0.758	2.74	37.9%	24.7%
25%	5%	61.9%	0.754	2.73	38.1%	23.9%
25%	10%	67.7%	0.834	3.02	32.3%	16.3%
50%	1%	76.2%	0.909	3.29	23.8%	5.4%
50%	5%	77.4%	0.886	3.21	22.6%	3.1%
50%	10%	76.9%	0.923	3.34	23.1%	4.5%
75%	1%	70.5%	0.836	3.03	29.5%	13.4%
75%	5%	78.6%	0.854	3.09	27.8%	9.5%

In Tables 5.4 and 5.5, the results using the evaluation method described in [Black et al., 2003] are presented for sequences Caviar1 and Caviar2 respectively. A clear improvement is visible for both sequences, as the number of points and window size are higher. For the first sequence, the best results are for a window size of 50%, where an accuracy of 3.34 is obtained when using 10% of the number of points, showing an improvement of 28.0% relative to the baseline and a detection

Table 5.5: Results from CVML information for the Caviar2 sequence

Window Size	Number of Points	Detection Rate	Tracker Detection Rate	Accuracy	False Negative Rate	False Alarm Rate
Original algorithm		78.4%	0.988	5.11	21.6%	24.7%
25%	1%	54.1%	0.693	3.49	45.8%	49.1%
25%	5%	54.1%	0.693	3.49	45.8%	49.1%
25%	10%	58.8%	0.755	3.80	41.2%	44.3%
50%	1%	71.4%	0.972	4.12	28.1%	27.3%
50%	5%	75.3%	0.990	4.98	24.7%	25.3%
50%	10%	75.3%	0.998	5.02	24.7%	25.6%
75%	1%	78.8%	1.057	5.32	21.2%	21.8%

rate of 77.4% is obtained when using 5% of the number of points, resulting in an increase of 8.0% when comparing to the original approach. This validates the experiments with a reduced window size, since the results improve when using only a window size of 50% of the height of the ellipsoid. Nonetheless, the overall results for the new technique improve relatively to the baseline approach. The exceptions are the results gathered for a window size of 25%, which show very similar values to the baseline. The results for the second sequence also present an improvement as the parameters are increased. In this case, the best results are even when a window size of 75% is used, proving that the parameters can be optimal depending on the sequence they‘re applied to. However, in this case most results don‘t show an improvement to the original approach, although for the best parameters, tracker detection rate is increased 7% and accuracy improved 4%.

Chapter 6

Conclusions

The overall purpose of this thesis was to demonstrate the specificity of each video object tracking algorithm and the difficulties it poses to the application of a generic tracking method. As said before, one needs to look at the objectives and needed outputs to construct a computer vision algorithm and build a object tracking method. Contributions were provided in two different contexts: vibrating line detection and tracking and an appearance model for single camera people tracking.

In this chapter, we'll take a look at the objectives proposed in the beginning of the work and the respective conclusions that we can take from the results.

6.1 Object Tracking

As mentioned in Chapter 1, a unique approach for computer vision hasn't still been achieved, therefore developers make assumptions and define restrictions, in order to build a robust solution although in specific terms. In order to stress this issue, we presented two separate methods for object tracking in video sequences. The line detection module presented is obviously very specific to line detection, as it searches for stable paths of minimum energy between two margins of an image and, therefore cannot be applied to people detection. The same goes for the tracking module – the optical flow applied in line tracking requires a low or inexistent deformation of the object and the absence of occlusion, and ergo applying it to people tracking could be unsuccessful as a person can deform severely from one frame to the next, specially if rotating. Hence, the application of line detection, as a first step to tracking, or of tracking method to track multiple people in a scene would fail. On the other hand, an appearance model is also not viable for a line detection and tracking approach. Usually a line presents itself having a uniform and homogeneous texture, and therefore to analyze its appearance would not be enough to track several lines over a video sequence.

6.2 Line Detection and Tracking

Concerning the proposed method in Chapter 4, we presented a new method for line detection in color images and their tracking as they vibrate throughout the sequence. The approach was

strongly based in a previous method used in another field, which provided successful results for the distance between the original and the detected line, in the range of 0.125% to 0.20% of the image‘s diagonal. For the image size considered, these values are between 1 and 1.5 pixels, which allow to say that lines were successfully detected. Furthermore, even for a sequence with a difficult background, the maximum distance found was 1.21%, which makes clear the robustness of the algorithm. We also proved that using information from the line detection and considering a larger number points in a line we could get a better approximation of the line‘s displacement (less 40%-80% of the approach using solely the information from 1 point). This decrease in the error, specially in the tangential component, represents a minimization in the aperture problem mentioned in the description of the optical flow, visible if we would just consider one point on the line.

As a result of the development and respective results provided by this method, an article entitled ‘*A Stable Path Approach for Vibrating Line Detection and Tracking*’ was submitted to the Electronics Letters journal and is currently under revision.

6.2.1 Future Work

Some of the future work for this subject has been presented throughout chapter 4, such as tests using lines provided with multi-frequency vibration characteristics and also using real footage. The latter is the most crucial for a more robust validation of the line tracking system. The comparison between the data gathered and information provided by other devices like accelerometers, or other techniques discussed in the state-of-the-art (chapter 2) would help to get an even better understanding of the system‘s performance.

The possibility of pre-processing done to the sequence is also something to take under consideration. The premises of the line detection algorithm state that the lines should be distributed from one margin of the image to the other, and in some situations, this could reduce the method‘s ability to operate. Therefore, an option of letting the user indicate where the lines begin and end, could improve the previously mentioned problems, or applying some rotation to the sequence, in order to distribute the lines according to our restrictions. Also some post-processing could be applied to the detected lines, to eliminate the effects of using the gradient filter – the line is detected slightly separately from the actual line, as the gradient marks the edge of the line.

As mentioned in section 3.4, a module for camera calibration could also be added to the method, to provide translation between world and image coordinates. This module would be essential to get information such as vibration amplitude of the lines.

Concerning the implemented stable path method, some experiments could be performed in order to find the critical point where the lines become too imperceptible for the algorithm to recover them.

6.3 People Tracking

The objectives described in Chapter 1, were to experiment with a new appearance model as an alternative to the original one. However, the integrated module led to an increase of the processing time of approximately 96 times, considering the result which uses the less processing time. This is obviously considered a serious issue in the algorithm as people tracking algorithms should perform close to real-time in order to be considered in practical situations. Furthermore, the gathered results from the first evaluation method used did not imply an increase of the performance, since the values extracted from the approach described in [Cardoso et al., 2009b] presented an error between 4.8% and 23.6% above the error obtained for the original method. Nonetheless, it became clear the increase of the performance as the number of points analyzed are increased, although the processing time also increases. Concerning the results extracted from the second evaluation method, described in [Black et al., 2003], two sequences, both extracted from [CAVIAR, 2004] were tested and evaluated, considering various pairs of parameters. With this evaluation method, we obtain overall better results when comparing with the baseline, with the exception of the parameters with a small window size and therefore also considering a small number of points. However, as it was observed in the first method, results can be improved using a 50% window size instead of 75%. Moreover, when considering the processing times when using a 75% window, this fact benefits the approach. Looking at the values for the sequence labelled Caviar1, the accuracy improves from 5.0% to 28.0% of the original value, which proves the overall better performance. As for sequence Caviar2, the use of a 25% window size produced worse results, making the detection rate decrease 24.3%. However, using a larger window size and consequently a larger number of points, the results improved, and for a window size of 75%, using 1% of the number of points, accuracy increased 4% and the detection rate improved 7%. This is a proof for the better performance of the integrated method over the CAVIAR sequences, rather than the PETS sequences. The sequences extracted from PETS have a better resolution, providing a better texture description of a person and a more controlled people flow, where there are practically no occlusions. This is a possible reason for the worse performance when comparing to the CAVIAR sequences, where we have a low resolution, less controlled scenario, where the appearance of a person proves to be worse, when using texture information.

6.3.1 Future Work

Some modules of the implementation are targeted for improvements, specially as a result of the high processing times obtained from the experiments. The immediate proposed work is to perform code optimization to reduce considerably this parameter. As the algorithm has a reasonable size, the processing time could be reduced drastically. Another way of reducing processing would be to consider less values in the signature vector. Although obtaining a less accurate description, this could mean a great decrease in processing and should be taken into consideration. If a reduction of the processing time is possible, then the increase of the number of points could be viable, since this could translate in an increase of the performance, as seen in the results where more points

were considered. Tests using much more variations in the parameters are therefore advised, as it might be viable to consider, for instance, 20% of the number of points.

An interest point detector could be an interesting module to integrate in the method, as sometimes it provides a better description of the person, and could translate in better results. An alternative local descriptor could also be used (such as the SURF descriptor) instead of the SIFT transform, which could allow a more effective signature extraction and therefore a better performance of the overall method. An average of the signature could also be produced over the frames instead of comparing the current signature with the previous one. This could allow a more robust characterization, increasing the precision of comparison.

Appendix A

Datasets

This is an appendix to Chapter 4 – line detection and tracking and Chapter 5 – people tracking. Some of this content is also described in those chapters.

Table A.1 shows a description of the data inputted to the sequences featuring vibrating lines. Figure A.1 depicts samples from the six different sequences used for the validation of the line detection and tracking method described in Chapter 4. The description of this dataset can be found in section 4.3 of the same chapter.

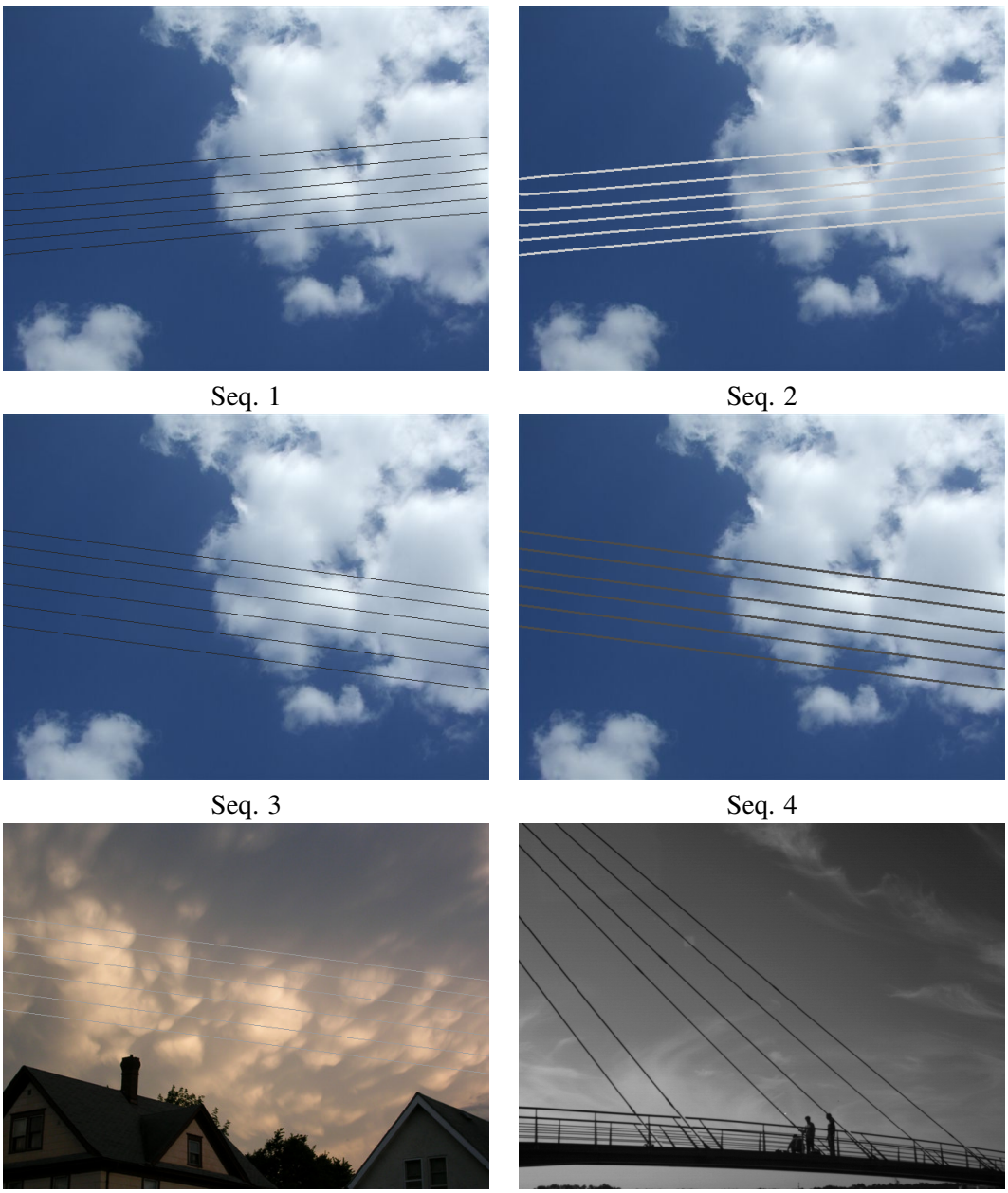
As for the people dataset, Table A.2 shows the available information for each sequence. The CVML information is an XML-based markup language described in [List and Fisher, 2004], which tells us the position of the bounding box of a person in each frame. In Figure A.2, we can see a sample of some sequences used for validation and experiments of the method. A further description can be found in section 5.2.

Table A.1: Description of content for each synthetic sequence of vibrating lines

Sequences	Line	Frequency (rad/s)	Amplitude (pixels)	Thickness	Grayscale value
1	1	0.8	5	1	45
	2	0.85	4.7		
	3	0.9	4.3		
	4	0.95	4		
	5	1.0	3.7		
	6	1.05	3.3		
2	1	0.8	5	2	205
	2	0.85	4.7		
	3	0.9	4.3		
	4	0.95	4		
	5	1.0	3.7		
	6	1.05	3.3		
3	1	0.8	5	1	40
	2	0.85	4.7		
	3	0.9	4.3		
	4	0.95	4		
	5	1.0	3.7		
	6	1.05	3.3		
4	1	0.8	5	2	75
	2	0.85	2		
	3	0.9	4.3		
	4	0.95	1.2		
	5	0.05	1.5		
	6	1.5	3.3		
5	1	0.8	5	1	155
	2	0.3	2		
	3	0.2	4.3		
	4	0.95	1.2		
	5	0.1	1.5		
	6	1.5	3.3		

Table A.2: Properties of each sequence used in the dataset for the people tracking method

Sequence Label	**CVML info**	**Segmentation info**	**Camera calibration**
Caviar1	x	x	
Caviar2	x	x	
PetsCam3		x	x
PetsCam4		x	x



Seq. 1 Seq. 2

Seq. 3 Seq. 4

Seq. 5 Seq. 6

Figure A.1: Samples from the dataset for line detection and tracking validation

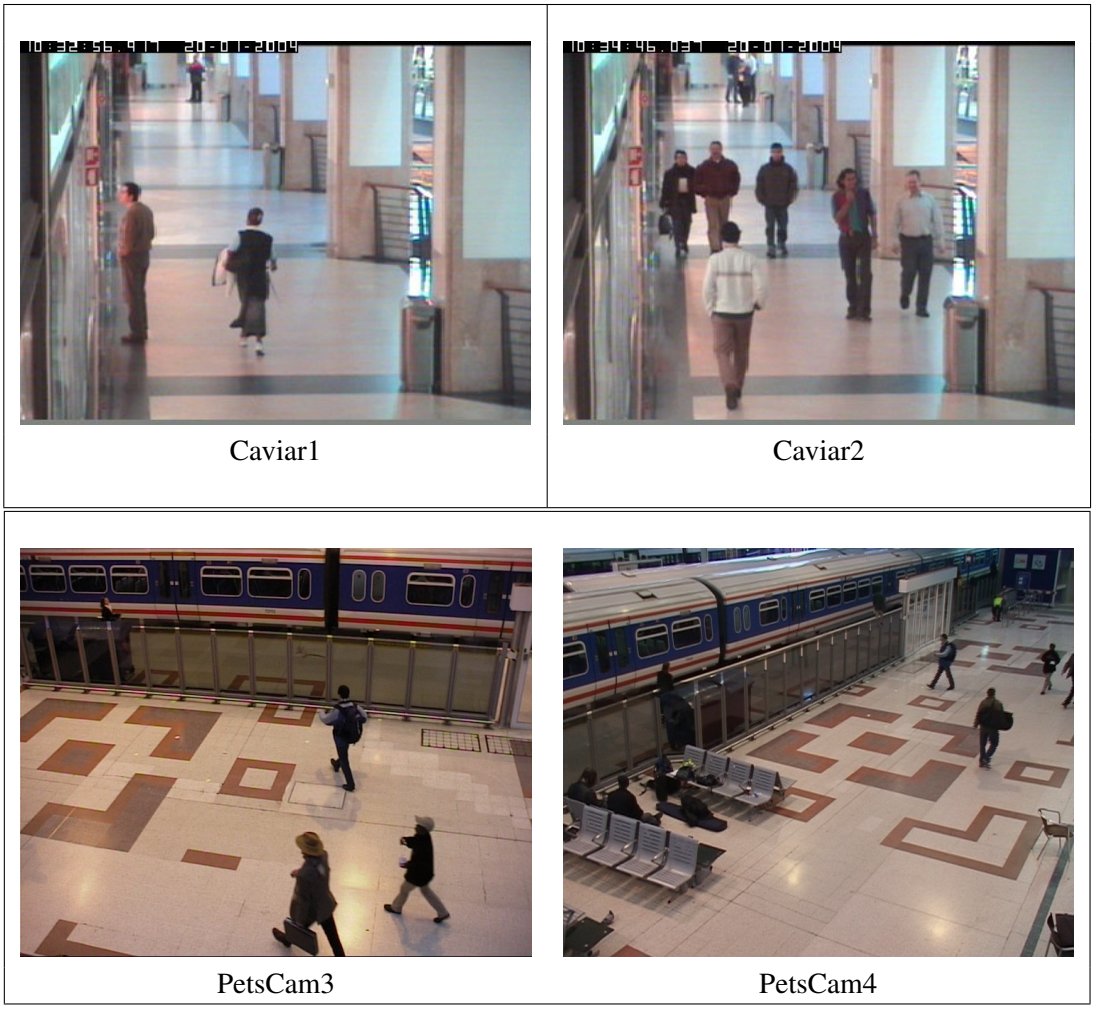


Figure A.2: Samples from part of the dataset for people tracking validation, presented with its respective label

Appendix B

Experiments in Line Detection and Tracking with Camera Calibration

In order to conduct some experiments with camera calibration using the line detection and tracking dataset we need to assume that all lines are parallel and inserted on the same projection plane. Doing so allows us to use the homography matrix, discussed in chapter 3.2.

For this initial experiment, we set some coordinates for the ends of three of the lines, depicted in Table B.1.

Table B.1: Set points for the line tracking image sequence

2D points (pixels)	**3D points**
(1,328)	(0,0,0)
(1,287)	(0,40,0)
(1,247)	(0,80,0)
(639,272)	(622,0,0)
(639,231)	(622,40,0)
(639,192)	(622,80,0)

With this set of points, we can calculate our homography matrix, which results in:

$$H = \begin{bmatrix} 0.964 & 0.000780 & -1.19 \\ -0.870 & -0.988 & 324.1 \\ -1.89e-5 & 2.51e-6 & 1 \end{bmatrix} \tag{B.1}$$

We assumed the first, third and fifth line from the bottom up were 40cm from each other and having a length of 622cm. We now want to follow the fifth line, iterating the 3D-2D conversion to the points (20i, 80, 0) where i=1, 2, 3,... until we reach the end of the line.

In Figure B.1, the black crosses represent the assumed points and the white ones represent the computed ones. As we can see, the crosses follow the trail of the line from the beginning until the end, apart from a small deviation, result of a little bend on the line. It is this deviation that

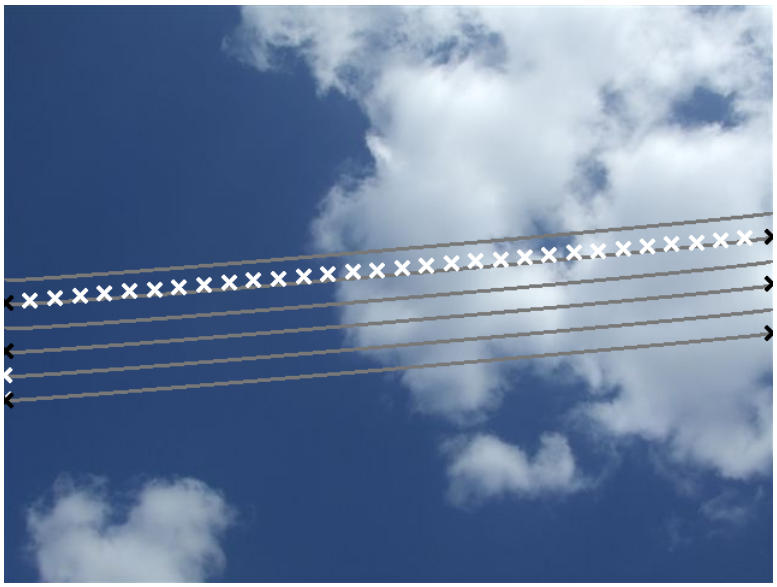


Figure B.1: Camera calibration test for line tracking

we want to calculate further in our experiments, as well as how often it occurs. In Figure B.2, the results gathered from this experiment are presented, as the error obtained when comparing the points tested to the respective reference values. From these results, the average Root Mean Square Error is 2.57, which allows to conclude that the camera calibration was successful and could be implemented in such situation.

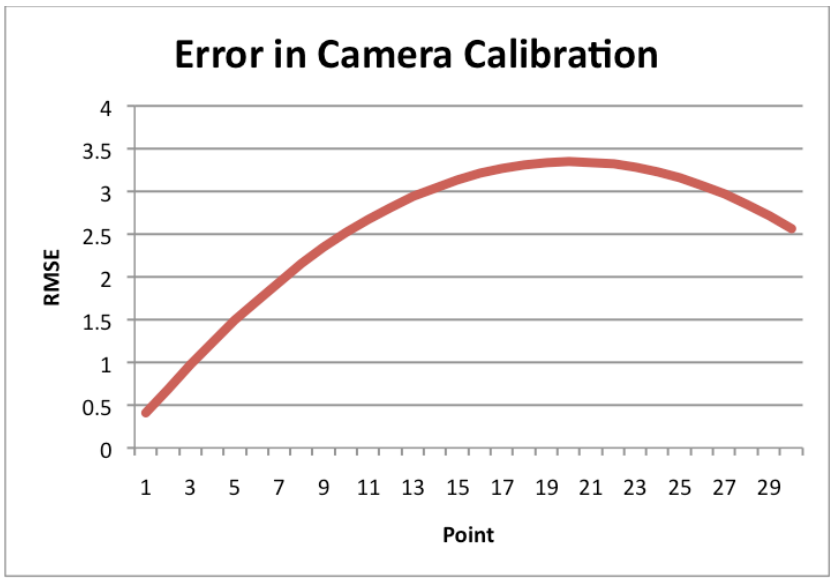


Figure B.2: Gathered RMSE results of the 30 analyzed points

References

D. Aouada and H. Krim. Squigraphs for fine and compact modeling of 3-d scenes. *IEEE Trans. Image Processing*, 19(2):306–321, February 2010.

F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. In *Proceedings of the IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, pages 7–14, 2006.

J. Black, T. Ellis, and P. Rosin. A novel for video tracking performance evaluation. *Joint IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 125–132, 2003.

J.Y. Bouguet. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*. Microprocessor Research Labs, Intel Corporation, 2000.

Z.Q. Cai and J. Tai. Line detection in soccer video. In *Fifth International Conference on Information, Communications and Signal Processing*, pages 538–541, 2005.

R. Calçada, A. Cunha, and R. Delgado. Analysis of traffic induced vibrations in a cable-stayed bridge. *Journal of Bridge Engineering, ASCE*, 10(4):370–385, 2005.

J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

J.S. Cardoso, A. Capela, A. Rebelo, and C. Guedes. A connecter path approach fot staff detection on a music score. In *Proceedings of the International Conference on Image Processing (ICIP 2008)*, pages 1005–1008, 2008.

J.S. Cardoso, A. Capela, A. Rebelo, C. Guedes, and J.F. Pinto da Costa. Staff detection with stable paths. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31:1134–1139, 2009a.

J.S. Cardoso, P. Carvalho, L.F. Teixeira, and L. Corte-Real. Partition-distance methods for assessing spatial segmentations of images and videos. *Computer Vision Image Understanding*, July 2009b.

P. Carvalho, J.S. Cardoso, and L. Corte-Real. Hybrid framework for evaluating video object tracking algorithms. *Electronics Letters*, 46, 2010.

CAVIAR. Ec-funded-caviar-project, i. 2001-37540., 2004.

M. Çelebi and A. Sanli. Gps in pioneering dynamic monitoring of long-period structures. *Earthq Spectra*, 18(1):47–61, 2002.

J. Cui, H. Zha, H. Zhao, and R. Shibasaki. Robust tracking of multiple people crowds using laser range scanners. In *Proceedings of the IEEE 8th International Conference on Pattern Recognition*, pages 857–860, Hong Kong, China, 2006.

R. Deriche and O.D. Faugeras. Tracking line segments. In Inc. Springer-Verlag New York, editor, *In ECCV 90: Proceecings of the first european conference on Computer Vision*, pages 259–268, New York, NY, USA, 1990.

C.E. Erdem, B. Sankur, and A.M. Tekalp. Performance measures for video object segmentation and tracking. *IEEE Trans. Image Processing*, 13(7):937–951, 2004.

D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

J.-P. Gambotto. Tracking points and line segments in image sequences. *Workshop on Visual Motion*, pages 38–45, March 1989.

D. Gravilla. The visual analysis of human movement: A survey. *Computer Vision Image Understanding*, 73(1), 1999.

I. Haritaoglu, D. Hardwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8), 2000.

B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

K.P. Horn. Tsai's camera calibration method revisited. *Massachusetts Institute of Technology*, 2000.

T. Ikeda, H. Ishiguro, and T. Nishimura. People tracking by fusing different kinds of sensors, floor sensors and acceleration sensors. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 530–535, 2006.

Y. F. Ji and C. C. Chang. Nontarget image-based technique for small cable vibration measurement. *Journal of Bridge Engineering, ASCE*, 13(1):34–42, 2008a.

Y. F. Ji and C. C. Chang. Nontarget stereo vision technique for spatiotemporal response measurement of line-like structures. *Journal of Engineering Mechanics*, 134(6):466–474, 2008b.

S.C. Kim, H.K. Kim, C.G. Lee, and S.B. Kim. A vision system for identifying structural vibration in civil engineering constructions. In *SICE-ICASE Int. Joint Conference*, 2006.

J.J. Lee and M. Shinozuka. A vision-based system for remote sensing of bridge displacement. *NDT and E International*, 39(5):425–431, 2006.

T. List and R.B. Fisher. Cvml – an xml-based computer vision markup language. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 789–792, Cambridge, 2004.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

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*, pages 674–679, 1981.

F. Lv, T. Zhao, and R. Nevatia. Self-calibration of a camera from a walking human. In *Proc. Int'l Conf. Pattern Recognition*, volume 1, pages 562–567, 2002.

F. Magalhães, E. Caetano, and A. Cunha. Operational model analysis and finite element correlation of the braga stadium suspended roof. *Engineering Structures*, 30:1688–1698, 2008.

T.B. Moeslund and E. Granum. A survey of cumputer vision-based human motion capture. *Computer Vision Image Understanding*, 81:231–268, 2001.

T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture a survey of advances in vision-based human motion capture and analysis. *Computer Vision Image Understanding*, 104:90–126, 2006.

J. Morlier. Virtual vibration measurement using klt motion tracking algorithm. *Journal of Dynamic Systems, Measurement, and Control*, 132(1):011003–011011, 2010.

P. Neubert, P. Protzel, T. Vidal-Calleja, and S. Lacroix. A fast visual line segment tracker. In *IEEE International Conference on Emerging Technologies and Factory Automation*, pages 353–360, September 2008.

P. Olaszek. Investigation of the dynamic characteristic of bridge structures using a computer vision method. *Measurement*, 25:227–236, 1999.

PETS. IEEE International Workshop on performance evaluation of tracking and surveillance, 2006.

D. Ramanan, D.A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(1):65–81, January 2007.

G. Roberts, X. Meng, M. Meo, A. Dodson, E. Cosser, E. Iuliano, and A. Morris. A remote bridge health monitoring system using computational simulation and gps sensor data. In *Proceedings, 11th FIG Symposium on Deformation Measurements*, Santorini, Greece, 2003.

R. Rodrigo, W. Shi, and J. Samarabandu. Energy based line detection. In *Canadian Conference on Electrical an Computer Engineering*, pages 2061–2064, 2006.

G.M. Schuster and A.K. Katsaggelos. Robust line detection using a weighted mse estimator. In *Proceedings of the 2003 IEEE International Conference on Image Processing*, 2003.

S. Shen, W. Shi, and Y. Liu. Monocular 3-d tracking of inextensible deformable surfaces under l2-norm. *IEEE Trans. Image Processing*, 19(2):512–521, February 2010.

N.T. Siebel and S. Maybank. Fusion of multiple tracking algorithm for robust people tracking. In *Proc. European Conf. Computer Vision*, pages 373–387, 2002.

S. Silva, J. Bateira, and E. Caetano. Development of a vision system for vibration analysis. In *2nd Int. Conf. on Experimental Vibration Analysis for Civil Engineering Structures*, 2007.

L.F. Teixeira and L. Corte-Real. Video object matching across multiple independent views using local descriptors and adaptive learning. *Pattern Recognition Letters*, 30(2):157–167, 2009.

R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE J. Robotics and Automation*, 3(4):323–344, August 1987.

A.M. Wahbeh, J.P. Caffrey, and S.F. Masri. A vision-based approach for the direct measurement of displacements in vibrating systems. *Smart Mater Struct*, 12(5):785–794, 2003.

C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7), July 1997.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:1330–1334, November 2000.

T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(9), September 2004.