Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

SPATIO-TEMPORAL ALIGNMENT AND ANALYSIS OF VIDEOS FOR INDUSTRIAL APPLICATIONS

Allan Freitas da Silva

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientadores: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Rio de Janeiro
Abril de 2019

SPATIO-TEMPORAL ALIGNMENT AND ANALYSIS OF VIDEOS FOR INDUSTRIAL APPLICATIONS

Allan Freitas da Silva

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2019

Silva, Allan Freitas da

Spatio-temporal alignment and analysis of videos for industrial applications/Allan Freitas da Silva. – Rio de Janeiro: UFRJ/COPPE, 2019.

XVII, 193 p.: il.; 29, 7cm.

Orientadores: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 153 – 166.

1. Signal Processing. 2. Video Processing. 3. Video Alignment. 4. Cluttered Environment. 5. Moving Camera. I. Silva, Eduardo Antônio Barros da *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

We all change. When you think about it, we're all different people all through our lives, and that's okay, that's good, you gotta keep moving, so long as you remember all the people that you used to be. I will not forget one line of this. Not one day. I swear. I will always remember when the Doctor was me.

—The Doctor, Doctor Who

People's dreams... don't ever end!

—Marshall D. Teach, One Piece

Agradecimentos

Gostaria de agradecer primeiramente à minha família por todo o apoio e suporte que eu sempre recebi em todas as etapas da minha vida, sem os quais eu certamente não seria capaz de chegar onde eu cheguei hoje.

Agradeço à minha namorada e grande parceira Carolina por estar presente em todos os momentos (dos mais alegres aos mais desesperadores), por estar sempre disposta a escutar e nunca poupar palavras de incentivo. Sua atuação foi essencial até mesmo para a finalização deste texto, se dispondo a conferir e revisar diversos detalhes.

Reservo também um espaço para todos os colegas que eu encontrei durante minha vida acadêmica, em especial no SMT, que sempre me fizeram me sentir parte de uma família. Agradeço por toda ajuda que sempre recebi, assim como por todos os momentos de descontração, momentos culinários, momentos de cafezes. Agradeço em especial a todos aqueles com os quais eu trabalhei diretamente, não só me fornecendo mais conhecimentos teóricos como também me permitindo ser um profissional melhor.

Agradeço aos meus orientadores Eduardo e Sergio por sempre acreditarem no meu potencial e nunca pouparem esforços para me ajudar da maneira que fosse necessária. Apesar de tantos anos de conviência, ainda percebo que tenho muito a aprender com a presença deles, tanto profissionalmente quanto pessoalmente.

Agradeço aos meus orientadores Yannick e Guillaume na França, e todas as pessoas no IMS, que me receberam de braços abertos e me acolheram quando eu estava longe de casa. Sou grato por tudo que aprendi durante este período e por estarem dispostos a participar dessa minha jornada acadêmica.

Agradeço aos membros da banca, que se dispuderam a participar da defesa apesar de todas as adversidades. Certamente suas contribuições serão fundamentais para a continuação deste trabalho.

Por fim, agradeço às instituições de incentivo à pesquisa, em especial à Capes e ao CNPq, que me financiaram em todo o meu percurso acadêmico e que deveriam ser mais valorizadas pela sociedade. Espero um dia poder retribuir tudo que me foi oferecido.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ALINHAMENTO ESPAÇO-TEMPORAL E ANÁLISE DE VÍDEOS PARA APLICAÇÕES INDUSTRIAIS

Allan Freitas da Silva

Abril/2019

Orientadores: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Programa: Engenharia Elétrica

Neste trabalho, investiga-se o uso de técnicas de processamento de sinais e visão computacional com o objetivo de fornecer ferramentas para a análise de sinais em um contexto de deteção de anomalias com aplicação na indústria. Para este tipo de análise, é comum ser realizado algum tipo de comparação entre sinais diferentes, de modo que faz-se necessário o uso de técnicas de alinhamento de sinais.

A primeira parte deste trabalho lida com o alinhamento temporal de sinais. São considerados os casos em que os sinais possuem o mesmo comprimento, logo só é necessário estimar um atraso que alinhe os sinais, ou taxas diferentes, onde também é necessário o uso de técnicas de *warping*. Os sinais utilizados para alinhamento podem ser tanto sinais de interesse, como vídeo ou áudio, quanto informações auxiliares presentes no sistema, como por exemplo a medição de consumo de energia em uma plataforma móvel. Também considera-se o uso de técnicas que realizam o rastreamento da câmera a partir de imagens para a realização do alinhamento temporal.

A segunda parte deste trabalho estuda o alinhamento espacial de imagens. Um conjunto de técnicas para estimação do campo de movimento dos píxeis das imagens é aplicado tanto em uma bases de dados tradicional quanto na aplicação industrial considerada, o que envolve também um estudo da robustez dos métodos a diferentes condições.

Por fim, a terceira parte do trabalho lida com a identificação de eventos em uma nova aplicação. Durante um processo de escoamento de um fluido, obtêm-se imagens contendo um padrão de interferências de luz. Deseja-se estimar posições neste padrão em imagens, de modo a extrair características do fluido utilizado.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SPATIO-TEMPORAL ALIGNMENT AND ANALYSIS OF VIDEOS FOR INDUSTRIAL APPLICATIONS

Allan Freitas da Silva

April/2019

Advisors: Eduardo Antônio Barros da Silva
Sergio Lima Netto

Department: Electrical Engineering

In this thesis, we investigate the use of signal processing and computer vision techniques aiming to provide tools for signal analysis in a context of anomaly detection in industrial applications. In this kind of analysis, it is common to perform some sort of comparison between different signals, which requires the use of signal alignment techniques.

The first part of the thesis deals directly with the temporal alignment of signals. We consider the cases in which the signals have the same length, when is only necessary to estimate a delay that aligns the signals, or different lengths, which also requires the use of warping techniques. The signals employed in the alignment can be either a signal-of-interest, such as video or audio, or auxiliary information obtained from the system, for example the measurement of power consumption in a moving platform. We consider the use of techniques that perform camera tracking from images to obtain a temporal alignment.

The second part of this thesis studies the spatial alignment of images. Techniques for the estimation of the motion field of the image pixels are applied on a standard database and on the considered industrial application, which also involves a study of the method robustness to different image characteristics.

At last, the third part of the thesis deals with the identification of events in a new application. During a fluid flow process, we obtain images containing a light interference pattern. We aim to estimate positions on this pattern, in order to extract the characteristics of the fluid employed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Industrial processes have benefited from computer vision techniques to minimize costs and increase security. One example is the use of RGB and thermal cameras in surveillance systems to monitor an environment, recognize and track all kinds of events in the scene, and detect gas leakage or fire [1–3].

In several applications, static cameras are spread out to cover the entire environment. A more challenging approach uses cameras installed in a moving robotic platform, which facilitates the monitoring of difficult-to-access environments and also increases the range of the cameras without incurring a considerable cost increase [4]. However, these applications may result in an increased complexity by requiring methods to localize the camera and identify the recordings carried out in the same position, which can be a problem in applications with limited computational power.

The last decade also saw an increase in the number of devices that may simultaneously measure different data on an environment. Equipment such as smartphones or the Kinect provide several sensors accessible to the public at a low cost. For this reason, surveillance systems should be able to analyze multiple information from a large number of sources. The diversity of the signals acquired simultaneously and the easiness in their acquisition create a demand for techniques to properly align them, so that they can be compared. Moreover, it is important to identify the occurrence of different events in the signals, which provides a better understanding of the systems involved.

1.1 Objectives

This thesis deals with the problem of signal alignment and camera localization for a video surveillance application that employs a moving platform equipped with a camera and several other sensors. For this application, the platform moves in a trail installed in an environment and camera records it in at least two different moments: a reference recording that validates the normal condition of the environment and a target recording that may contain an unexpected event that must be detected. In this context, the signal alignment provides means for the reference and target signals to be compared properly. In addition, the camera localization allows the system to identify in which positions the anomaly may have occurred.

Another application considered on this thesis is the estimation of viscosity properties of polymer melts based on the analysis of images obtained during the flow of the polymer inside equipment designed for this purpose. In this analysis, one identifies the occurrence of dark fringes and estimates their position, which can be used to infer the stress levels on the polymer.

The contributions of this thesis are as follows. In the first part, we investigate the signals acquired by a surveillance system in a industrial environment (see Sections 1.2.1 and 1.2.2). Using these signals (which are not necessarily video signals), we develop methods capable of performing the temporal alignment between the signals in the cases when they have the same or different lengths. We also investigate an approach of the simultaneous localization and mapping (SLAM) algorithm in [5], which estimates the trajectory of a moving camera, in order to adapt this framework to the video surveillance application.

The second part of this thesis deals with the spatial alignment. After performing a temporal alignment step, we develop methods to perform a spatial alignment step (in this case specifically for the video signals), while considering different ways to extract features from the images. This study also takes into account several conditions that can jeopardize the computation of the spatial alignment, such as difference of illumination and occlusion.

The third and final part of this thesis studies the images obtained during the flow of molten polymers (see Section 1.2.3). We present a novel approach to automatically detect the position of dark fringes in these images, using mathematical morphology techniques to find the patterns that characterize the fringes and detect the center position of each one of them. The results are compared with other fringe detection methods, revealing the superior precision of our method.

1.2 Applications

This section describes in more details the main applications that are related to the techniques developed in this thesis.

1.2.1 Surveillance System Description

DORIS - Monitoring Robots for Offshore Facilities is a project which endeavors to design and implement a surveillance system for remote supervision, diagnosis, and data acquisition on offshore facilities. The proposed system is composed by a rail-guided mobile robot that moves inside a cluttered industrial environment [6–8]. The robotic platform is capable of carrying different interchangeable sensors, such as cameras, microphones, gas detectors, vibration and temperature sensors, which provide information about several properties of the robot and the environment. In that manner, the DORIS system presents a modular framework which allows performing several tasks such as: detection of audio anomalies [9], identification of gas leakage, detection of video anomalies [10, 11], and diagnosis of rotating machines [12]. A first prototype of the robotic platform was installed in a industrial environment as shown in Fig. 1.1, and runs in a circular track whose model can be seen in Fig. 1.2. Details on mechanics and control system of this prototype can be seen in [6–8].

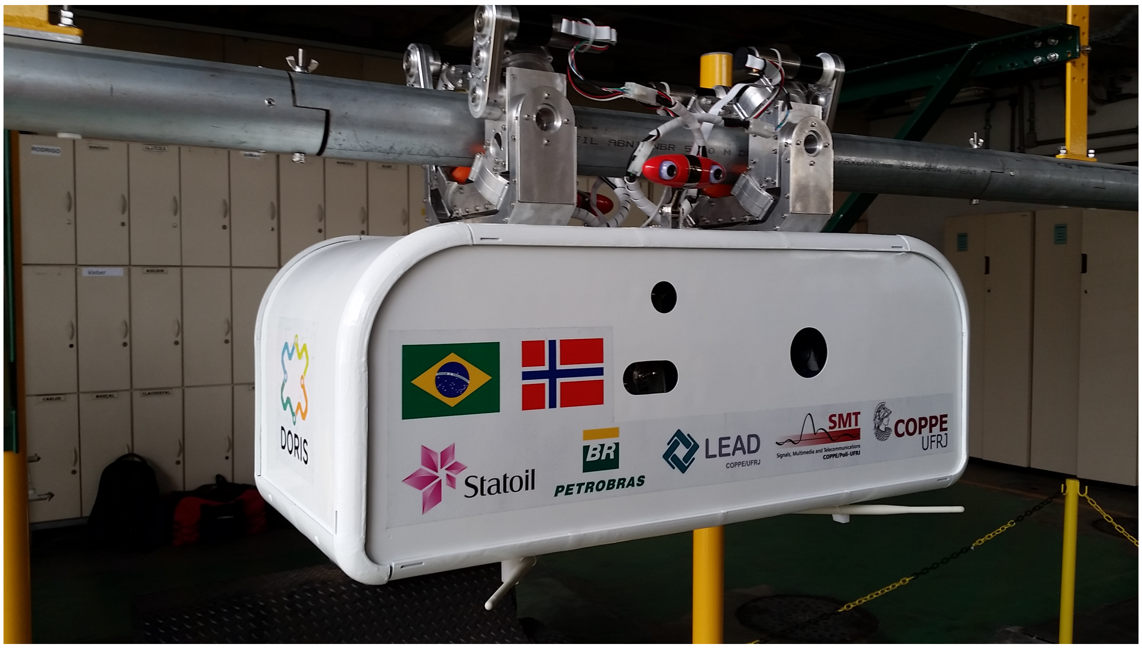


Figure 1.1: Robotic platform in a industrial environment.

The video-anomaly detection algorithm compares frames from a new video that may contain an anomaly with the most similar frames from a reference video (a previous recording which was validated by an operator as having no anomalies). During an alignment step, the algorithm computes a homography [13] between consecutive frames and uses it to estimate the horizontal displacement between frames, whose integral gives an estimate of the absolute position where each frame was acquired. Finally, the method uses a model-based maximum likelihood method to align frames based on their absolute position estimate. However, since this method computes only horizontal displacements, it is not compatible with non-rectilinear camera movements. Fig. 1.3 shows a frame from a video with an abandoned object and a frame with the same view from the reference video.

The system also has several sensors measuring orientation, speed, acceleration,

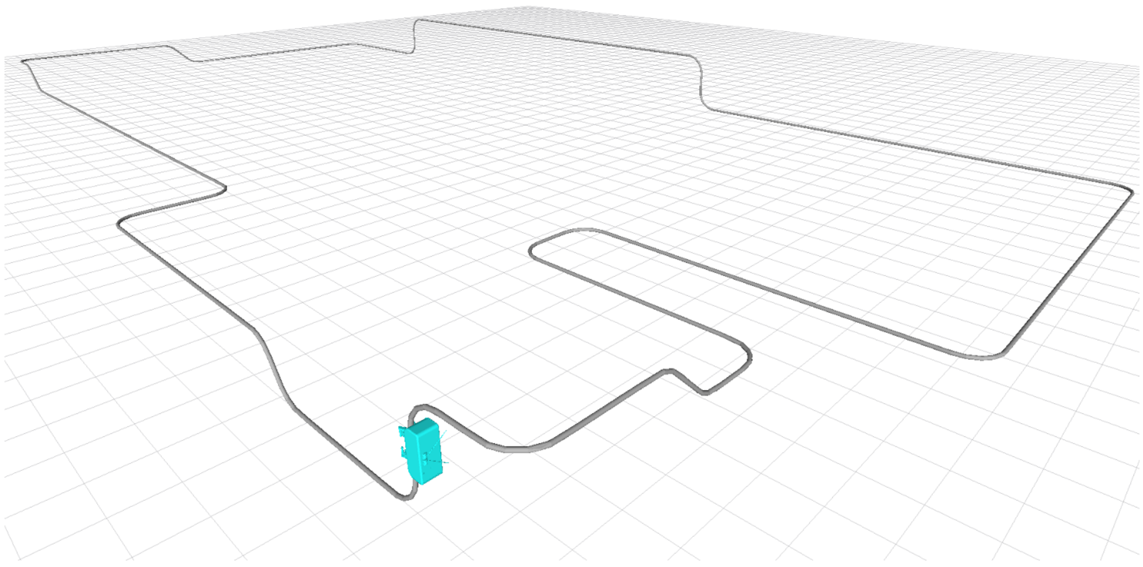


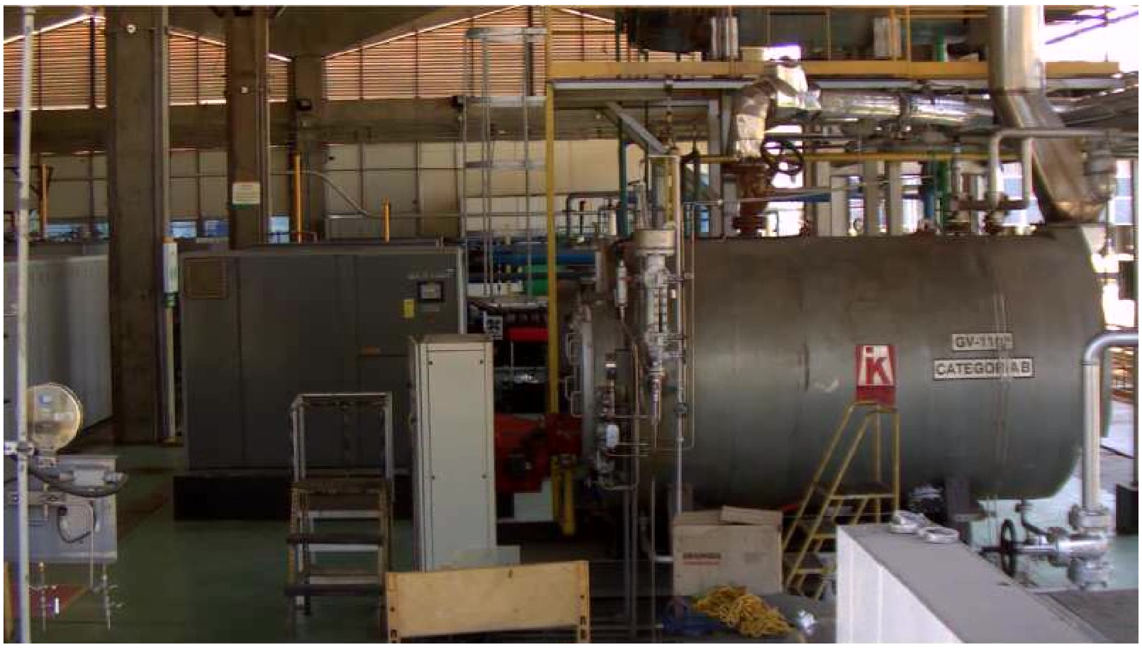
Figure 1.2: 3D model of the rail (gray) and the robotic platform (blue).

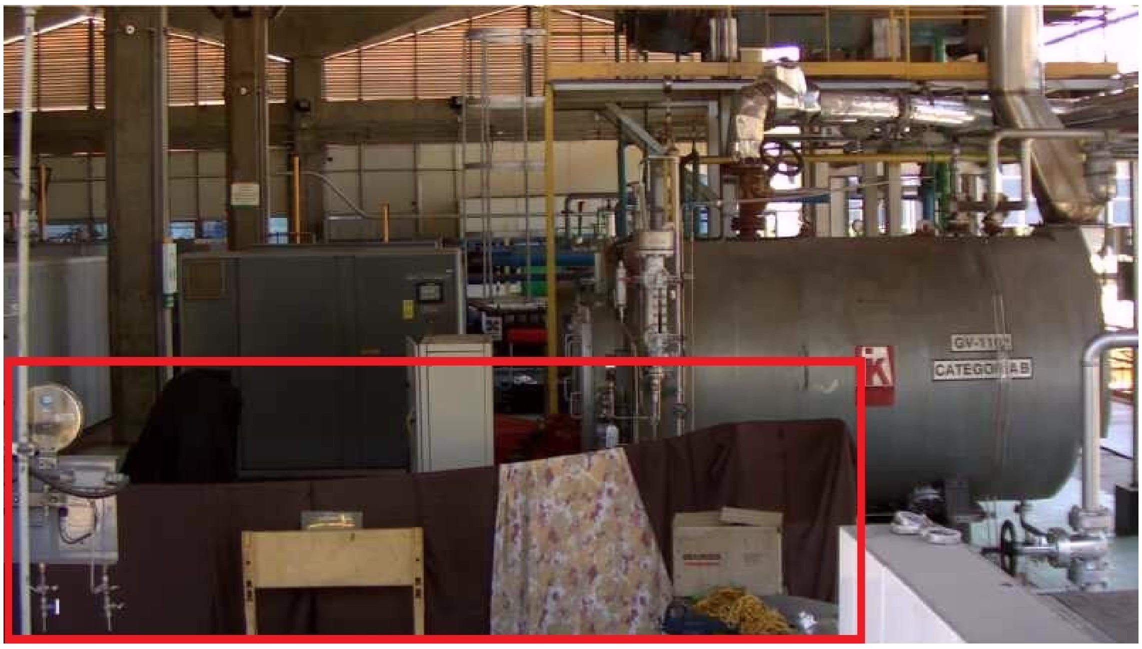
and power consumption. This information can be used, for instance, to estimate the current position and provide temporal and spatial alignment. The system has a native algorithm that uses information on the power consumption of the engines, along with a knowledge of their mechanical properties, to provide a rough estimate of the linear displacement between the current instant and the starting position, which can also be used to align any video with a reference video.

1.2.2 Video Database of Abandoned Objects in a Cluttered Industrial Environment

The video database of abandoned objects in a cluttered industrial environment (VDAO) is a database created to promote the research on anomaly detection methods, containing several HD videos recorded in an industrial environment. A camera was coupled in a *Roomba*© robot, which was programmed to perform a back-and-forth movement in a rail, as exemplified in Fig. 1.4. For each recording, different objects were placed in the environment, simulating an anomaly to be detected, with two different lighting conditions.

The database possesses 4 videos containing only the original environment (as attested by an operator), considered the reference videos, 56 videos containing a single object placed in an arbitrary position in the environment, and 6 videos containing multiple objects placed in the environment, which adds up to 8 h of recordings with 24 objects. A manual marking of a bounding box for the position of each inserted object is also available. Figs. 1.5, 1.6, and 1.7 show parts of the environment recorded with the system shown in Fig. 1.4.


(a)



(b)

Figure 1.3: Example of a target frame with video anomalies to be detected and a similar reference frame. (a) Reference frame. (b) Target frame from the same scene with a few different objects marked by a red square.

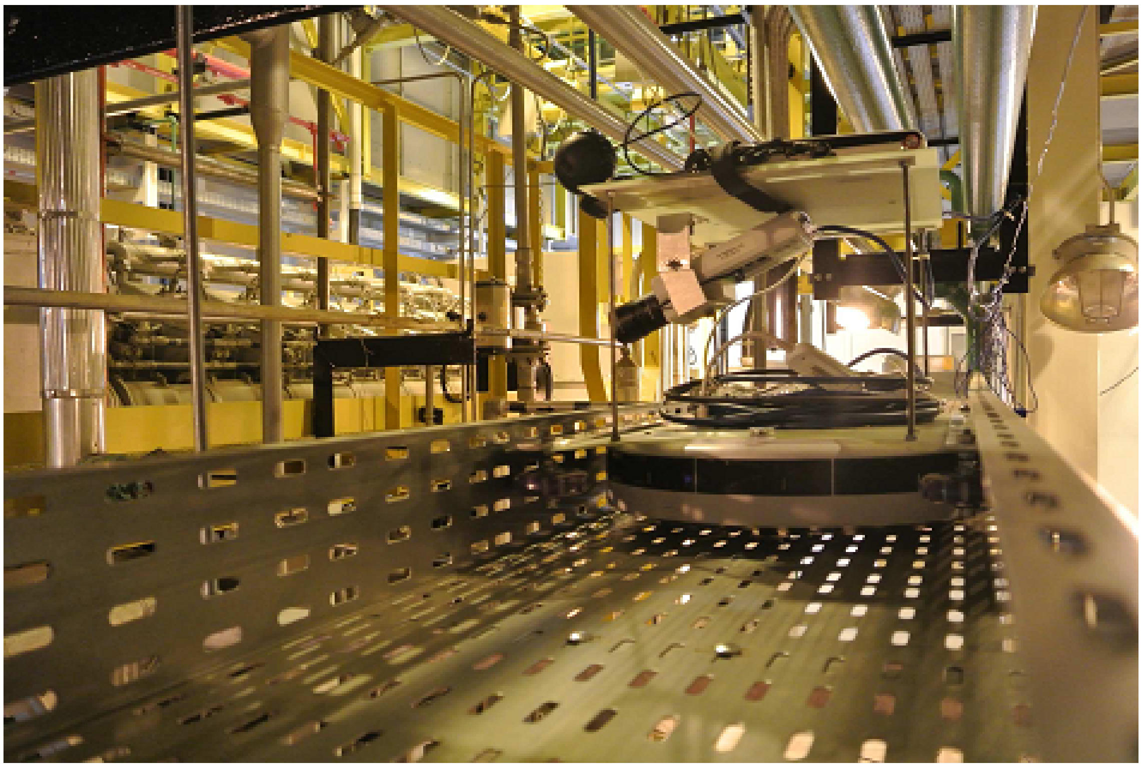


Figure 1.4: Prototype system to monitor an industrial environment with a camera mounted on a moving platform.

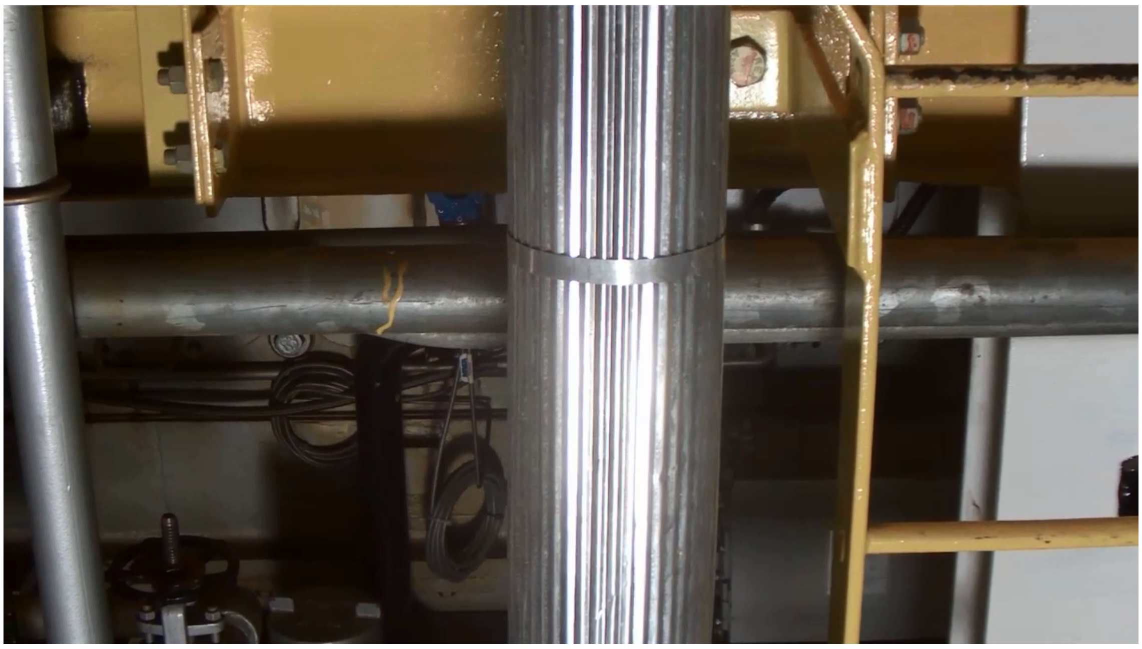


Figure 1.5: Portion to the right of the monitored environment.

1.2.3 Experimental Setup for Polymer Characterization

An application that can benefit from computer vision techniques is in the processing of polymers for industrial applications. Polymers are processed in a high temperature, which makes them subject to a high level of deformation. The final structure

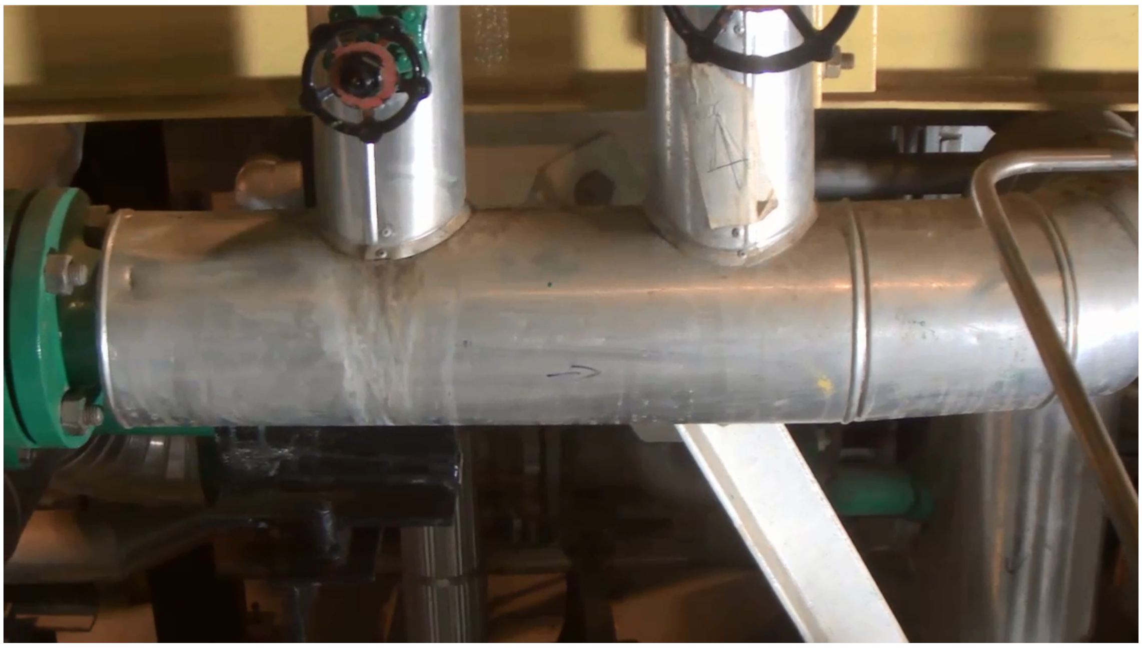


Figure 1.6: Mid section of the monitored environment.

Figure 1.7: Portion to the left of the monitored environment.

of the material depends directly on its rheological (i.e. flow-related) properties. Considering the high costs involved in trial and error procedures, due to the use of large industrial machinery, it is essential to define experiments to model the flow of viscoelastic fluids.

In an attempt to overcome this challenge, techniques for determining the flow birefringence stress patterns can be coupled with sophisticated experimental apparatus, such as the multipass rheomoter (MPR) developed by Prof. Mackley and co-workers [14–19]. In this device it is possible to map the stress field during the flow, when it is equipped with an optical cell designed to fit the different geometries.

The experimental procedure shown in Fig 1.8 is based on inducing a birefringence property in a fluid as a result of the orientation of the polymer chains during its flow. The polymer flows inside the MPR, which contains a specific geometry to induce the desired properties in the fluid. Using a light source positioned at one side and a camera on the other side, it is possible to measure a pattern of light interference, resulting in an image such as the one in Fig. 1.9. This pattern exhibits bright and dark fringes, whose positions depend on the geometry of the experiment and the properties of the flow. Therefore, the obtained patterns of bright and dark fringes provide information about the spatial evolution of the stress in the molten polymer.

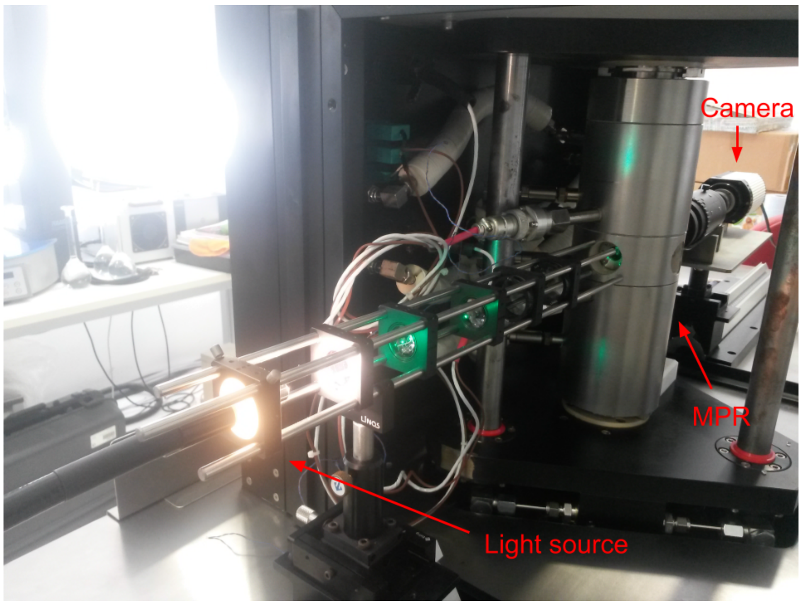


Figure 1.8: Experimental apparatus to induce birefringence properties in a polymer.

1.3 Related Work

In this section we review several areas of study with use in the aforementioned applications.

1.3.1 Video Alignment

The problem of video alignment appears repeatedly and in different ways in the literature. Some studies, such as [20–22], try to align video sequences that capture the same scene, that is, that have a considerable amount of visual overlapping.

In other applications, an alignment must be made between sequences acquired from the same position but at different times. In some cases, side information

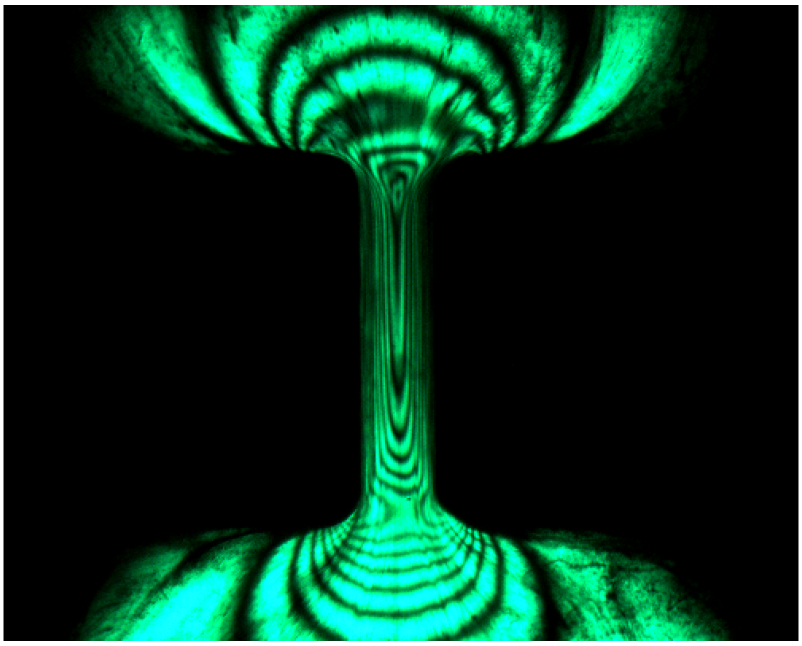


Figure 1.9: Example of birefringence image obtained with the experiment shown in Fig. 1.8.

related to the signal of interest can be used to improve the performance, as in [23], which combines the visual information and a global positioning system (GPS) signal. Yet other methods [21], besides using correspondences (via homography or even via fundamental matrix), profit from additional information available in the data stream, such as audio signals, to perform correlation measurements and obtain the desired alignment between distinct video sequences.

A field of research that has been neglected is the video alignment of videos acquired by moving single cameras following an approximately identical trajectory. Distinct methods can be applied to solve this problem. In [24] the video signals are transformed into one-dimensional signals through color histogram extraction from each frame and the serialization of the acquired data. To perform the alignment, correlation measures are taken from the one-dimensional data, which are used as a feature for a dynamic programming algorithm to compute the alignment.

The most relevant technique for aligning two signals is the dynamic time warping (DTW) [25, 26]. DTW considers that one sequence can be shrunk or stretched along the time axis to match the other. The problem of finding the best mapping can be described as the search for the path that minimizes the matching error between sequences. This technique has been initially used in the context of automatic speech recognition, but it has already been successfully applied to a wide range of applications, such as biomedicine [27], entertainment [28] and data-mining [29].

The standard DTW algorithm has a computational complexity that is quadratic with the number of elements of the sequences. To solve this problem, some constraints can be applied to save computation in the cost matrix. Sakoe-Chiba [26] bounds the path to lie inside a region around the diagonal of the cost matrix, while Itakura [25] restricts the path to be inside a parallelogram, requiring that one sequence never be more than a certain number of times faster than the other sequence. Other approaches, such as the one in [30], prune the DTW matrix if the cost reaches a given threshold. A DTW algorithm with linear computational complexity was developed by Salvador et al. [31]. It employs a multi-resolution approach to spare some computation. A drawback that is shared by these methods is the fact that the cost matrix must be computed beforehand, which creates difficulties for online processing.

An online DTW with linear complexity was developed by Dixon [32]. The proposed online DTW performs incremental alignment between two signals when one of them is being received in real time, so only a subsequence of one of the signals is currently known. This algorithm is applied in the alignment of audio signals to provide live analysis of musical performance.

A temporal alignment technique is frequently one of the steps of a video anomaly detection algorithm. In [33] a technique for detection of objects in a road was proposed. It applies a DTW algorithm to align present and past camera images using a similarity metric based on projective geometry information [13]. After aligning the frames, the algorithm applies a road registration, and objects are detected by image subtraction. A similar approach was developed by Mukojima et al. [34] in the context of railroad object detection. This algorithm performs time alignment between the reference and target videos by computing frame-by-frame correspondences and then applying a DTW algorithm using a similarity metric derived from the corresponding keypoints. After synchronizing both videos, the method performs spatial alignment between frames and computes image subtraction metrics to detect anomalies. Another method for detecting objects in a road was proposed in [35]. This approach applies a rough video alignment that uses only a GPS signal, that is followed by a geometric registration between frames, and objects in the road are detected by the computation of a correlation metric.

1.3.2 Motion Estimation

Recurrent problems in computer vision are camera localization and motion estimation. The field of simultaneous localization and mapping (SLAM) refers to algorithms that use any available information to estimate the movement of the camera as well as to reconstruct the environment in which the camera moves. For this purpose, several methods propose the use of auxiliary information, such as laser [36], radar [37] or infrared signals [38]. In [39], for instance, a method developed for low-powered devices uses a camera mounted in an aerial vehicle and pointed downwards along with a height sensor and estimates visual maps using a graph-based formulation.

Among such methods, the visual SLAM is composed of approaches that use a camera as their primary sensor. Davison [40, 41] uses a monocular camera and estimates the camera linear and angular velocities for each new frame, considering that between each measurement a random speed variation can occur. The work seen in [42] develops an object-oriented SLAM, which uses recognition algorithms to identify objects in the environment, which are used as features that are tracked along the frames.

Some of the most successful visual SLAM algorithms use stereo cameras [43], which are not widely spread. However, approaches using monocular cameras have several drawbacks since no information regarding the depth of the scene can be directly inferred from the images. To work around this problem, visual SLAM algorithms with monocular cameras usually use two main steps [5]: a visual odometry method to estimate camera poses and a loop closure step that prevents errors due to the uncertainty of the scale. The visual odometry step computes the epipolar geometry between frames, which is usually used to build submaps of the camera trajectory [44], and the results are refined with a bundle adjustment algorithm [45]. The loop closure step computes connections between submaps [44] or frames [46, 47] to detect loop closures, which are used to refine the results by minimizing a cost function.

A visual SLAM method that has state-of-the-art results in trajectory estimation of a monocular calibrated camera has been proposed in [5]. This method develops a new formalism using notions of Lie algebra [48] to efficiently estimate submaps of the camera trajectory, and uses graph optimization to combine different submaps and remove outliers.

1.3.3 Optical Flow

Optical flow consists of the calculation of the apparent motion of the image pixels. Using two images, the goal is to calculate a field of two-dimensional vectors that register the movement of points from one image to the other. For such calculation, an algorithm of optical flow must be able to overcome a few obstacles that can arise in real videos, such as outliers from the discontinuities or occlusions, variation in lighting and regions with big disparities between images.

Despite the fact that algorithms were established about 40 years ago [49], the calculation of optical flow is still in constant evolution, having reached significant advances in recent years. As viewed in [50], most of the advances in optical flow are still based on the basic structure proposed in [49]. They often include coarse-to-fine estimation [51], creating a pyramid of subsampled versions of the images, warping the images with the current flow, and optimizing it incrementally. Among the improvements to this algorithm, one can cite, for instance, the inclusion of texture decomposition [52, 53], a refinement step with a median filtering to remove outliers [50, 53], as well as the inclusion of robust penalty functions [50, 54, 55] along with a graduated-non-convexity scheme [56] to optimize them. One can also mention the study developed in [57], where in this work, several regularization techniques are reviewed, and a decoupling strategy for the inclusion of non-convex functions is proposed, along with a normalization factor for each term.

A recent trend in the development of new methods is the robustness to different characteristics. An example is when the images have distinct lighting conditions. Such case can happen, for instance, if the images were acquired in separate moments, and the weather or the sunlight is different during each recording, or if the movement of the camera and the objects create a different pattern of shadows cast upon the scene. This creates a difference of content in the images, which violates the basic assumption made in [49], that the only difference in the intensity values between the two images is due to the motion of the objects in the scene. To solve this issue, several methods replace the brightness information for each pixel by a much richer feature. In [58] it is assumed that the brightness information can be modeled as a combination of the reflectance of the objects and the illumination provided by a light source. By decomposing the brightness in these two components, a method that takes into account each component as separate cues is proposed, enforcing the use of the reflectance component, which should have reduced lighting influences. A similar approach is developed in [58], by decomposing a color image in illumination and chromaticity components. In [59], several strategies for the color decomposition are discussed and tested, each one with a different invariance to illumination changes.

A family of methods propose different illumination-robust descriptors. In [60] the authors propose dividing the images in patches centered on each pixel, normalizing the variance and the mean of the intensity values, stacking the patch in a vector, and using it as a descriptor for each pixel, thus showing that the sum of square differences (SSD) between two descriptors is equivalent the zero-normalized cross-correlation (ZNCC) between the original patches. A binary descriptor is defined in [61], by associating the neighborhood around each pixel to an eight-bit string which depends on the edges direction. Since the descriptors do not rely on the edges magnitude, they become more robust against illumination changes. Another neighborhood descriptor is proposed in [62]. They define a different set of displacements, depending on the size of the neighborhood, and compute the SSD between each neighborhood and its n-th displaced version. The result of this operation is associated with the n-th component of a descriptor, along with a normalization, which they show to be more robust to illumination changes than [60] and [61].

Difficulties in the estimation of the optical flow can also arise when estimating the flow for large displacements. In this case, the traditional coarse-to-fine heuristic is expected to fail, since the dimension of the objects could be smaller than the displacement they perform. Methods such as [63, 64] draw ideas from deep learning concepts. Those methods perform pre-processing in the images based on the dense SIFT [65], which is rearranged to become similar to the architecture of a convolutional neural network (CNN). With this step, they find candidates for the correct flow, which are used both as initialization and as an additional term during the optimization. In [66] it is proposed a different initialization step based on patch matching techniques.

One of the first successful methods to develop a deep learning architecture for this problem is the FlowNet described in [67]. It defines two different architectures: the FlowNetSimple (also called FlowNetS), which concatenates the two images prior to the application on the network, and the FlowNetCorr (or FlowNetC), which concatenates the features obtained for each image in an intermediate layer. Since the known optical flow databases are too small for the training of CNNs, it also proposes the Flying Chairs database, containing over 20 thousand synthetic sequences with ground truth. A continuation of this method, the FlowNet2 [68], stacks several FlowNetC and FlowNetS networks to compute the incremental flow, creating a model with a large number of parameters.

In [69], it is proposed an architecture with three levels of CNNs to compute the optical flow, having results similar to the FlowNetC with much fewer parameters. Another architecture that cascades CNNs is proposed in [70], which sometimes outperforms the FlowNet2 while having fewer parameters to train.

In a recent work, [71] obtains state-of-the-art results with an architecture inspired by traditional optical flow methods. They include a pyramid of features obtained with a CNN, warp the features of the second image using the current flow estimate, use a CNN to find the optimum flow for the current level and refine the estimate using a feed-forward CNN. This architecture also allows them to have almost 20 times fewer parameters than the FlowNet2.

1.3.4 Fringe Detection

Several aspects related to the processing of different types of polymer tested in the MPR rheometer have been discussed and presented during the last years [14–16, 19, 72, 73]. In general, many methods studied fringe detection, specially in the context of light interference fields [74–83]. For example, a method based on image binarization creates a centerline of each fringe by employing morphological approaches like thinning or skeletonisation [74]. In other works, the periodic characteristics of the fringes are modulated by a signal [77].

Some of the most successful techniques to detect light interferometric fringes [76, 84] rely on the estimation of an orientation map for each image, which is used with an adaptive median filter to remove noise. By measuring the changes in the gradient component along the direction given by the orientation map, one can define the positions of maxima and minima in the image, which characterize the centerline for each light or dark fringe.

However, most of the methods developed for light interference fields tend to fail when applied to the case of birefringence images. In this scenario, the fringe patterns in the images are related to the flow and geometry of the experiment and some assumptions about the periodicity of the image do not hold. In addition, due to experimental problems such as impurity in the material and difficulty in the image acquisition, the resulting images are often corrupted by noise.

For the specific case of MPR optical images, [85] mentioned the application of a morphological analysis that considers a skeleton birefringence pattern to detect the position of the fringes. However, no further details are provided about the adopted approach.

Recently, it was proposed in [86] a group of mathematical morphology techniques to find the patterns that characterize the birefringence fringes obtained in the MPR [14, 17, 18, 87]. Their approach to detect the center position of each fringe contains five steps: (i) input image enhancement, (ii) minima contour detection with watershed, (iii) skeleton creation and center detection, (iv) post-processing and (v) inflection points detection. The authors showed that the method has great potential for detecting dark fringes in birefringence images with accuracy prediction comparable to a manual marking while minimizing the need for human interaction with the images.

In [88] a semi-automated methodology was presented for the principal stress difference (PSD) analysis from flow-induced birefringence images using the GNU Image Manipulation Program (GIMP) open-source software. The position of the centers of dark fringes obtained through the flow-induced birefringence images of two polystyrene samples processed in the MPR4 was determined with greater accuracy and with shorter processing time when compared with the standard manual technique. The main advantage of using the cited approach is that it does not require any prior knowledge of advanced image processing techniques nor the use of expensive computational packages. However, in such scheme, the user still performs the fringe processing and analysis one image at a time, resulting in a time-consuming process.

1.4 Text Organization

The remainder of this text is organized as follows. Chapter 2 presents a method to temporally align signals acquired in the surveillance system, assuming that during the recording the moving platform had the same speed. Since this condition is hardly met, Chapter 3 deals with the alignment problem in the case where there are distinct signal lengths between each recording. In Chapter 4, a SLAM algorithm is described and tested in several databases, including videos from the DORIS surveillance system. Chapter 5 deals with the problem of the spatial alignment of two images, using the optical flow algorithm, which can be used to complement the temporal alignment performed in Chapters 2 and 3. Chapter 6 describes the proposed methodology to handle the fringe-estimation problem detailed in Section 1.2.3. Finally, Chapter 7 summarizes the work developed in this thesis and devises proposals for future works.

Chapter 2

Signal Alignment Using Sensor Ensemble

Anomaly detection problems often have a reference signal that saves information about the normal condition in a process or environment. In this scenario, a new target signal must be aligned to the reference signal in order to be properly compared and analyzed. However, the alignment using only the information intrinsic to the desired signals can be costly and imprecise.

In this chapter, we propose an alignment algorithm using correlation measures obtained from sensor ensembles which have a time stamp that enables a synchronization with another signal of interest, such as the ones in Section 1.2.1. For this approach, the signal of interest was acquired with a moving sensor (e.g., a moving camera), and several different sensors were also available, which measure other information that may be correlated with the movement and position of the moving sensor. Angular and linear velocity sensors are good examples of sensors that may be related to the robot movement.

It is assumed that the signals were acquired using a robotic platform in a closed-loop trajectory that moves at the same speed for any recording, but there may be an unknown delay between the signals obtained from different recordings. Since the robotic platform moves in a closed-loop trajectory, whenever it is in the same position along the trajectory, the sensor ensemble should output a similar set of samples, indicating that there is a periodic behavior in the acquired signals. The proposed algorithm also includes an initial step to identify the fundamental period of the signals, which simplifies the computation of the correlation. The algorithm is applied to signals acquired using the current prototype of the DORIS system, which has several sensors measuring physical and electrical properties of the robotic platform, such as orientation, angular velocity and power consumption.

This chapter is organized as follows: in Section 2.1, the proposed algorithm that aligns a signal of interest using other synchronized information is described. It is composed of a first step that pre-processes a reference set of signals and a second step that aligns the pre-processed reference to a target set of signals. Section 2.2 shows the performance evaluation of both steps of the proposed methodology.

2.1 Alignment Algorithm for Curvilinear Tracks

Given the surveillance system described in Section 1.2.1, it is assumed that two surveillance rounds were performed. For each round, several pieces of information were acquired: some auxiliary signals used to monitor the current state of the robot and some signals that a surveillance algorithm must monitor, for instance, a video signal or an audio signal. For each run, it is assumed that all this set of information was available in a synchronized manner.

The first round, called reference, contains the normal behavior of the environment (for instance, having no video anomalies) as validated by a human system operator. A second run, called target, may have an abnormal behavior in a given sensor, for instance, it may contain a video that registered a fire or a leakage.

The alignment between a target and a reference signal of interest, acquired during different rounds of a robotic platform, can be obtained through the auxiliary signals, by calculating the relative delay $\hat{\delta}$ that maximizes the cross-covariance between the reference and target signals obtained with the same sensor, in the following manner:

$$\hat{\delta} = \underset{\delta}{\operatorname{argmax}} \sum_{i} \sum_{n} (\mathbf{r}_{\mathrm{i}}(n+\delta) - \bar{r}_{\mathrm{i}})(\mathbf{t}_{\mathrm{i}}(n) - \bar{t}_{\mathrm{i}}), \tag{2.1}$$

where $\mathbf{r}_{\mathrm{i}}(n)$ and $\mathbf{t}_{\mathrm{i}}(n)$ represent the signals acquired by the i^{th} sensor during the recording of the reference and target videos, and $\bar{r}_{\mathrm{i}}$ and $\bar{t}_{\mathrm{i}}$ are their respective means.

The computation of the cross-covariance seen in Eq. (2.1) can be efficiently performed in a simple step with the help of the two-dimensional covariance between two images. Considering two matrices $\mathbf{R}$ and $\mathbf{T}$, the cross-covariance can be written as:

$$c_{ij} = \sum_{m} \sum_{n} (\mathbf{R}(m+i, n+j) - \bar{R})(\mathbf{T}(m,n) - \bar{T}). \tag{2.2}$$

The signals from the reference and target sensors can be grouped in such a way that they compose the matrices $\mathbf{R}$ and $\mathbf{T}$ whose i^{th} rows contain, respectively, data from sensors $\mathbf{r}_{\mathrm{i}}$ and $\mathbf{t}_{\mathrm{i}}$. Using these matrices, one arrives at the following cross-covariance equation:

$$c_{ij} = \sum_{m} \sum_{n} (\mathbf{r}_{\mathrm{m+i}}(n+j) - \bar{R})(\mathbf{t}_{\mathrm{m}}(n) - \bar{T}). \tag{2.3}$$

This equation shows that, if one normalizes the signals by removing the mean value, Eq. (2.3) becomes identical to the argument of Eq. (2.1) for a null i. Therefore the optimization problem becomes:

$$\hat{\delta} = \underset{j}{\operatorname{argmax}}\ c_{0j}. \tag{2.4}$$

A computationally efficient way to obtain the cross-covariance matrix is to use the two-dimensional DFT:

$$\mathbf{C} = [c_{ij}]_{M\times N} = \mathrm{DFT_{2D}}^{-1}[\mathrm{DFT_{2D}}[\mathbf{R}] \circ \mathrm{DFT_{2D}}[\mathbf{T}]^*], \tag{2.5}$$

where $\circ$ is the element-wise multiplication operator and * is the complex conjugate operator.

Nevertheless, it is necessary that both vectors have the same dimension, for this computation to be possible. If this is not true, the zero-padding technique can be used to allow the computation. In the general case, considering N_1 the length of vectors $\mathbf{r}_\mathrm{i}$ and N_2 the length of vectors $\mathbf{t}_\mathrm{i}$, the vectors are filled with zeros until their length is $(N_1 + N_2 - 1)$.

Alg. 1 summarizes the procedure of obtaining the relative delay between two vectors ensembles.

Algorithm 1 Alignment algorithm from the maximum cross-covariance between the two vector ensembles.

```
    Input: Data from reference sensors ($\mathbf{r}_\mathrm{i}$) and target sensors ($\mathbf{t}_\mathrm{i}$)
    Output: Delay $\hat{\delta}$ that maximizes the cross-covariance between reference and target data
1: $N_1 = \mathrm{length}(\mathbf{r}_\mathrm{i}),\ N_2 = \mathrm{length}(\mathbf{t}_\mathrm{i})$
2: $\mathbf{r}_\mathrm{i} = \begin{bmatrix}\mathbf{r}_\mathrm{i} & \mathrm{zeros}(1, N_2 - 1)\end{bmatrix}$
3: $\mathbf{t}_\mathrm{i} = \begin{bmatrix}\mathbf{t}_\mathrm{i} & \mathrm{zeros}(1, N_1 - 1)\end{bmatrix}$
4: $\mathbf{R} = \begin{bmatrix}\mathbf{r}_1 \\ \vdots \\ \mathbf{r}_\mathrm{M}\end{bmatrix},\ \mathbf{T} = \begin{bmatrix}\mathbf{t}_1 \\ \vdots \\ \mathbf{t}_\mathrm{M}\end{bmatrix}$
5: $\mathbf{C} = [c_{ij}]_{M\times N} = \mathrm{DFT_{2D}}^{-1}[\mathrm{DFT_{2D}}[\mathbf{R}] \circ \mathrm{DFT_{2D}}[\mathbf{T}]^*]$
6: $\hat{\delta} = \underset{j}{\operatorname{argmax}}\ c_{0j}$
```

2.1.1 Reference Video Fundamental Period Estimation

The procedure shown in Alg. 1 increases the length of every vector $\mathbf{r}_\mathrm{i}$ and $\mathbf{t}_\mathrm{i}$ in order to determine the cross-covariance matrix using the two-dimensional DFT. Since it is assumed that the signals were acquired in a robotic platform that moves in a closed-loop trajectory, the signals transmitted by the sensors will naturally have a periodic behaviour. The estimation of the lap period, given by the amount of time needed for the robot to perform a complete lap of the rail, allows the smallest representation of the environment and simplifies the computation of the cross-covariance.

To find the lap period, the cross-covariance technique can be applied. This step requires a set of signals acquired during at least two consecutive rounds of the robot. The whole set containing all acquired data from the multiple consecutive rounds is considered to be a reference data ensemble that has a periodic behavior with an unknown number of periods, therefore composing the vector $\mathbf{r}_\mathrm{i}$. The initial samples, which should be composed of at most half the total number of samples, compose the vector $\mathbf{t}_\mathrm{i}$.

It is expected that the correlation will entail a high value for at least two different values of delay, indicating the position from which the data were copied from the vector $\mathbf{r}_\mathrm{i}$ to the vector $\mathbf{t}_\mathrm{i}$ and the samples from the subsequent rounds that are similar to the first one. These samples are generally obtained from the same position of the robot in the rail. In this way, two consecutive peaks in the value of the covariance function mark the time the robot takes to complete a lap around the rail, therefore indicating the fundamental period of the signals.

An example of correlation obtained in this step is shown in Fig 2.1. The peaks indicate that if one applies a delay of 0, 200 or 400 samples, the correlation is high, which indicates that this signal has a fundamental period of 200 samples.

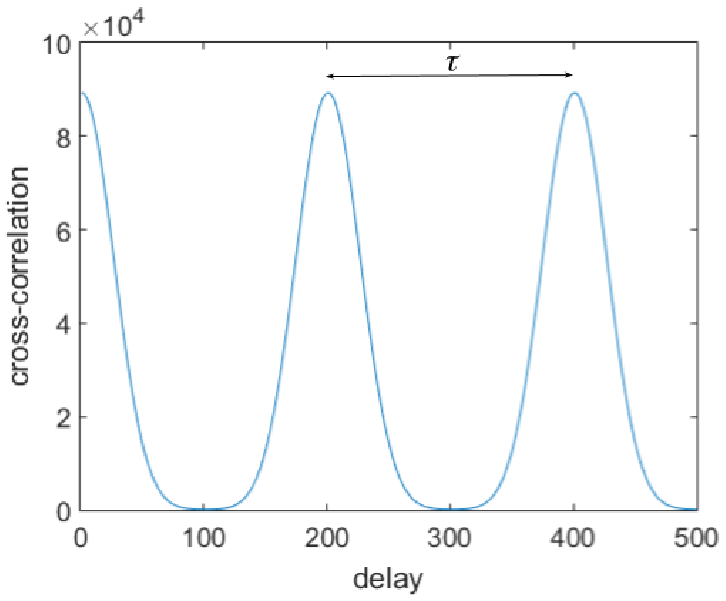


Figure 2.1: Example of fundamental period estimation. The distance between the peaks, denoted as τ, indicates the estimated period of the signals.

By the end of this step, the algorithm generates a new reference video containing an exact complete round of the robotic platform on the rail and the associated data from the sensors. Alg. 2 summarizes this step.

Algorithm 2 Obtention of the reference data fundamental period.

```
    Input: Reference data from the sensors ($\mathbf{r}_\mathrm{i}$) in two consecutive rounds, interval ($N_2$) of samples to be used in the covariance computation.
    Output: Fundamental period ($\tau$), vectors ensemble ($\mathbf{r}_\mathrm{i}$) and reference videos with exact one lap around the rail.
1: $N_1 = \mathrm{length}(\mathbf{r}_\mathrm{i})$
2: $\mathbf{t}_\mathrm{i} = \left[\mathbf{r}_\mathrm{i}(1, \cdots, N_2) \quad \mathrm{zeros}(1, N_1 - N_2)\right]$
3: $\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_\mathrm{M} \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_\mathrm{M} \end{bmatrix}$
4: $\mathbf{C} = [c_{ij}]_{M \times N} = \mathrm{DFT_{2D}}^{-1}[\mathrm{DFT_{2D}}[\mathbf{R}] \circ \mathrm{DFT_{2D}}[\mathbf{T}]^*]$
5: $\tau$ = difference between the positions of the two largest peaks of $c_{0j}$
```

2.1.2 Target Video Alignment

After processing the reference data, an approach similar to the one seen in Alg. 1 is used to align the target data from a new recording to the processed reference data. For this step, it is not necessary to wait for the system to obtain the data from a complete lap around the rail to perform the alignment. It is necessary, however, that there are enough data to perform the covariance computation.

The target vectors are filled with zeros (zero padding) if needed, until they have the same length τ of the reference vectors, which after the pre-processing step contain an exact complete round of the robotic platform and are assumed to be a periodic signals. The algorithm, then, computes the cross-covariance between the reference and target data using a DFT as given in Alg. 1.

The resulting cross-covariance should present a peak that indicates the position around which the information in the reference data are similar to the target data. This will occur every time the data are acquired from the same position along the rail. In this way, the results indicate the delay between the reference and target signal. Alg. 3 describes the process of obtention of the delay between the reference and target video. Since for this algorithm it is assumed that the reference data $\mathbf{r}_\mathrm{i}$ contain an exact period of the signals, it is only necessary to perform a zero-padding in the target data $\mathbf{t}_\mathrm{i}$, in comparison with the Alg. 1.

2.2 Experimental Results

For recording the database, a camera was configured to record videos at a framerate of 25 Hz and a resolution of 800 × 450 pixels. Several signals were synchronously acquired in a real industrial environment with a robotic platform moving along a rail having an average speed of 0.1 m/s. The signals are transmitted to a central computer via a wi-fi network, along with a time stamp used to synchronize them.

Algorithm 3 Efficient alignment using the maximum cross-correlation.

```
   Input: Reference data ($\mathbf{r}_\mathrm{i}$) containing an exact period and target data ($\mathbf{t}_\mathrm{i}$)
   Output: Delay ($\hat{\delta}$) that maximizes the cross-correlation between reference and target data
1: $N_1 = \mathrm{length}(\mathbf{r}_\mathrm{i})$, $N_2 = \mathrm{length}(\mathbf{t}_\mathrm{i})$
2: $\mathbf{t}_\mathrm{i} = \begin{bmatrix}\mathbf{t}_\mathrm{i} & \mathrm{zeros}(1, N_1 - N_2)\end{bmatrix}$
3: $\mathbf{R} = \begin{bmatrix}\mathbf{r}_1 \\ \vdots \\ \mathbf{r}_\mathrm{M}\end{bmatrix}$, $\mathbf{T} = \begin{bmatrix}\mathbf{t}_1 \\ \vdots \\ \mathbf{t}_\mathrm{M}\end{bmatrix}$
4: $\mathbf{C} = [c_{ij}]_{M\times N} = \mathrm{DFT_{2D}}^{-1}[\mathrm{DFT_{2D}}[\mathbf{R}] \circ \mathrm{DFT_{2D}}[\mathbf{T}]^*]$
5: $\hat{\delta} = \underset{j}{\mathrm{argmax}}\ c_{0j}$
```

Initial recordings with consecutive rounds allow the testing of the fundamental period estimation of reference signals. Afterwards, a new set of signals was recorded to be used as target signals. The algorithms were implemented in Matlab [89] and C++ programming languages.

The estimation of the signals period is compared to the positioning estimate of the DORIS system, which allows the estimation of the time the robot takes to perform a full lap around the industrial plant (130 m). It is also possible to use this position estimate to perform a rough video alignment between the reference and target sequences, which is compared to the estimated delay that aligns reference and target data.

2.2.1 Reference Signal Fundamental Period Estimation

Alg. 2 was tested in the estimation of the fundamental period of the reference signal using a data set with two complete laps of the robot in the rail, as given in Fig. 2.2. In this figure, the signal (a) defines the vector $\mathbf{r}_\mathrm{i}$. An excerpt of this signal is copied into another vector, creating signal (b), associated to the vector $\mathbf{t}_\mathrm{i}$. The correlation between the vectors creates the signal (c), whose peaks indicate the same spot in the rail. The distance between the correlation peaks yields an estimate of the time the robot takes to complete a full lap in the rail. The signal (d) contains an exact period of the signal (a).

Tab. 2.1 presents a fundamental period obtained for a reference video. This value is compared to the one obtained from the positioning system of the robot. The method is also tested on a recording containing signals acquired with a sample rate of 10 Hz with the robot moving at the speed of 0.2 m/s.

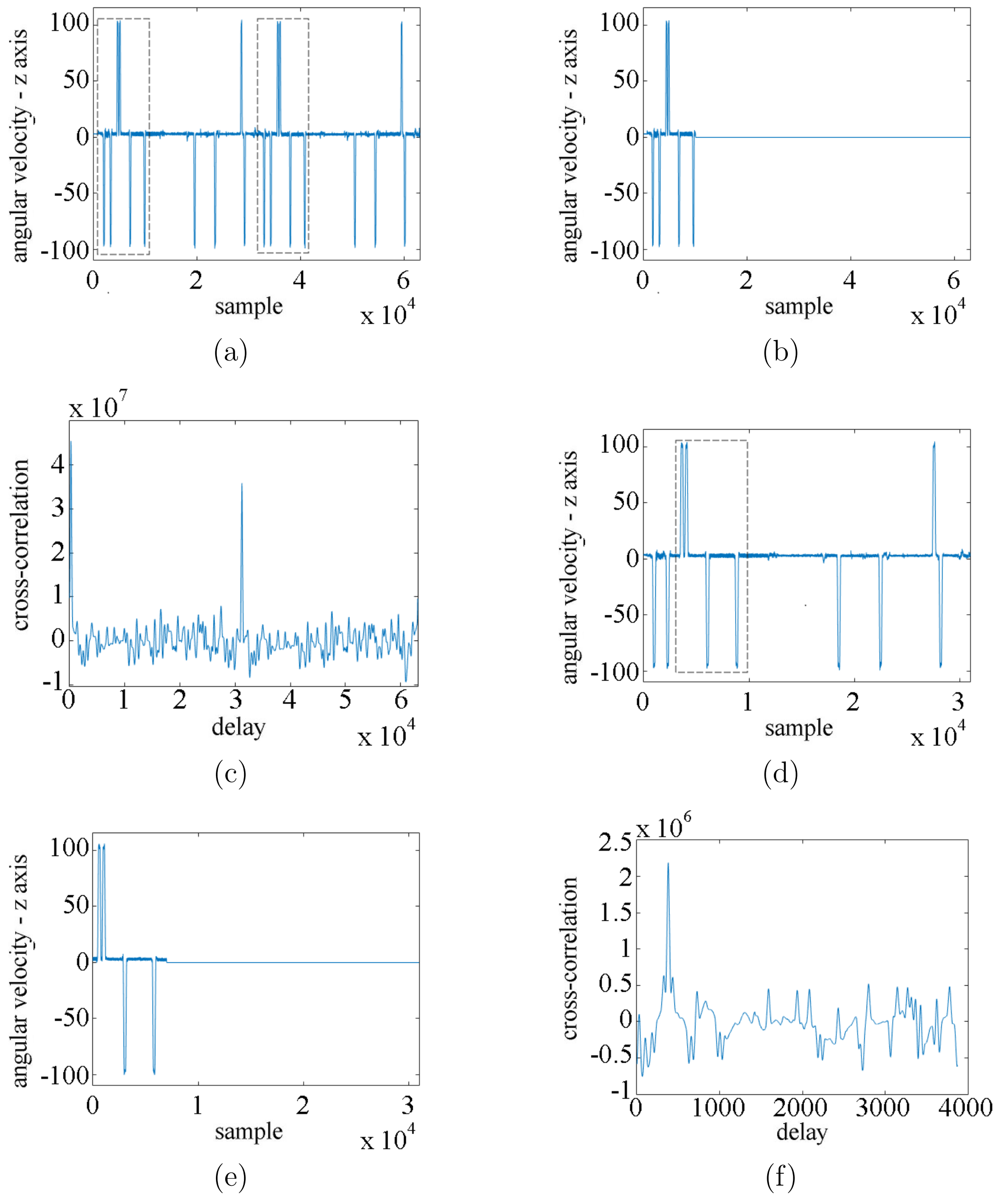


Figure 2.2: Example of the fundamental period estimation and relative delay between two signals. The dashed lines indicate the probable regions of the reference signal that are similar to the target signal. (a) Reference signal of a sensor of the ensemble during two robot laps, assigned to the vector $\mathbf{r}_i$. (b) Samples of the signal (a) used as target vector $\mathbf{t}_i$. (c) Cross-covariance between $\mathbf{r}_i$ and $\mathbf{t}_i$ used to obtain the fundamental period of the reference signal, according to Alg. 2. (d) One lap of the reference signal (a). (e) Samples in the data set of the target signal of the same sensor. (f) Cross-covariance of the sensor ensemble to obtain the relative delay between the signals, according to Alg. 3.

Table 2.1: Comparison between the fundamental period obtained by the algorithm and the robot positioning system estimate. Configuration 1: sampling rate of 25 Hz and average speed of 0.1 m/s. Configuration 2: sampling rate of 10 Hz and average speed of 0.2 m/s.

	Fundamental period (number of samples)	
	Algorithm	**Position estimate**
Configuration 1	30985	30947
Configuration 2	6042	6033

2.2.2 Alignment of the Reference and Target Signals

The synchronization test was performed using an exact period of the reference signals obtained after the application of Alg. 2. Fig. 2.2 also presents an example of the computation of the relative delay between the videos. The signal (d) contains 8000 samples acquired in a posterior recording. The cross-correlation between the reference data and target data, shown in image (f), presents the delay that maximizes the similarity between the signal samples, that is, the delay needed to perform the synchronization.

In another test, one varied the number of samples of the target signal and the position where the samples were acquired, in order to analyze how the algorithm behaves when the target signal has less meaningful information. For this test, target signals with 2000, 5000, 10000 or 15000 samples were used, and the acquisition could start in a straight or in a curve section of the rail. Figs. 2.3 and 2.4 present the cross-covariance obtained for two signal ensembles, obtained at distinct positions and with distinct sample amounts.

The results show that the number of samples used in the target signals has a significant impact on the algorithm performance. Having only a small amount of samples, there may not be enough information for the algorithm to find a single region with enough similarity. This fact occurs in the signal used in the computation of image (d), whose recording started in a straight section of the rail. With few samples, the correlation metric indicates more than one position in the reference signal that has samples similar to the target sequence, and the algorithm is not able to decide in which straight section of the rail the signal was acquired (see Fig. 1.2 for a model of the rail). The signal used in the computation of image (a) started in a curve section of the rail, with a very distinct pattern. In this case, even with few samples there is enough information to detect the correct corresponding section of the rail.

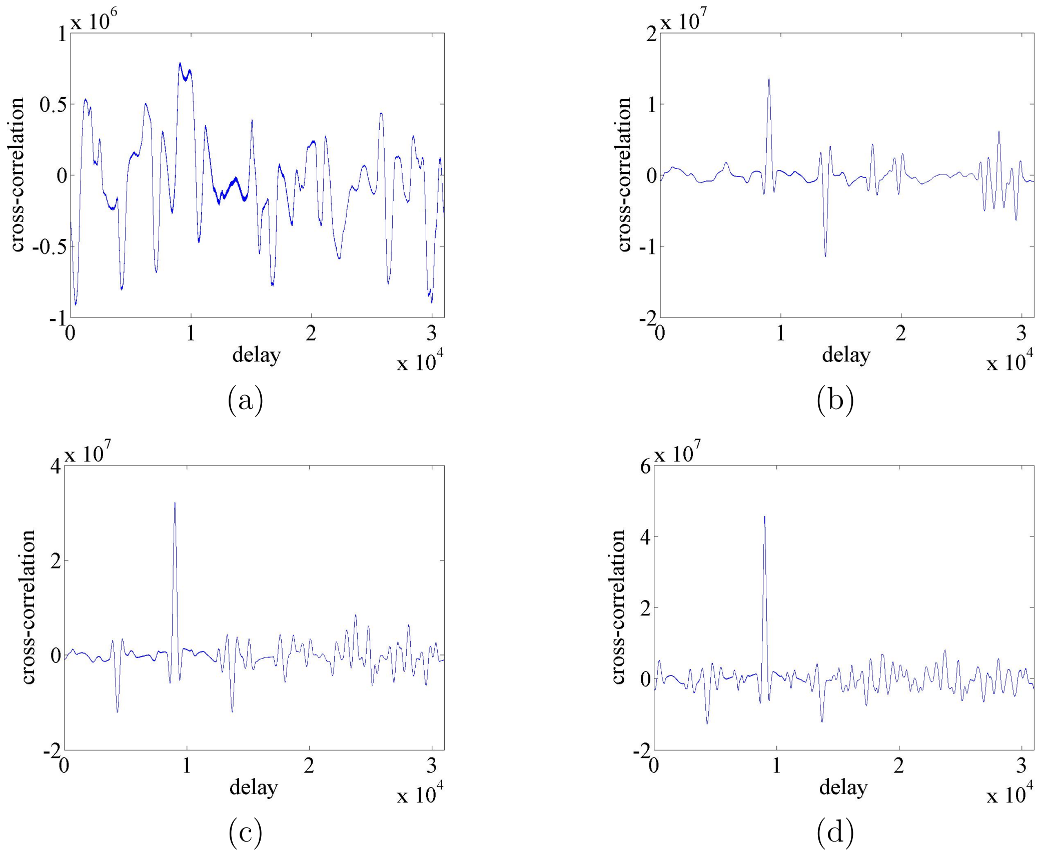
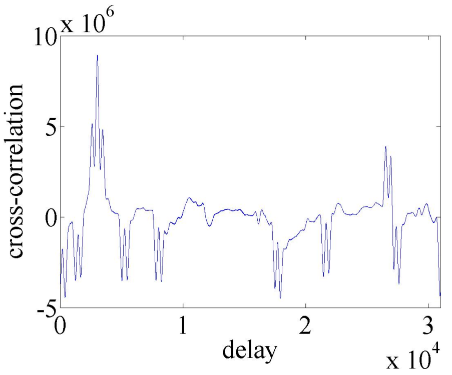


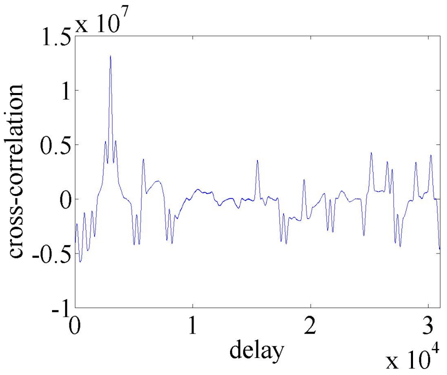
Figure 2.3: Cross-covariance using data ensembles acquired with start in a straight section of the rail. (a) Signal with 2000 samples. (b) Signal with 5000 samples. (c) Signal with 10000 samples. (d) Signal with 15000 samples.

2.3 Summary

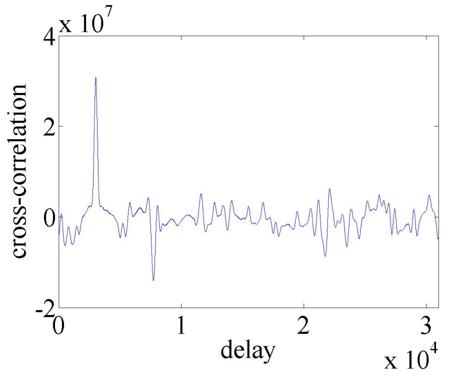
This chapter presented a method to perform the alignment of any signals of interest that are synchronized to an ensemble of sensors that have a periodic behavior and have no speed variation between multiple recordings, such as the ones present in a robotic platform moving in a closed-loop trajectory along a rail. Through the maximization of a similarity (measured by the cross-correlation function) of the captured signals, one is able to obtain the fundamental period of the signals and their alignment delay. The method was tested and compared with an odometry system present in a surveillance system, and the results suggest that, for a data set with enough samples, the method shows a similar behaviour to the method natively present in the robot. The next chapter depicts an alignment method that is able to compensate the case when the signals exhibit a difference in the lengths. In addition, the algorithms from Chapter 2 were adapted to directly use the signal of interest information (in this case, the video content) to perform the alignment.



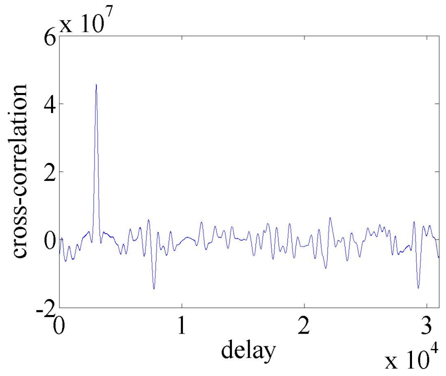
(a)



(b)



(c)



(d)

Figure 2.4: Cross-covariance using data ensembles acquired with start in a curve section of the rail. (a) Signal with 2000 samples. (b) Signal with 5000 samples. (c) Signal with 10000 samples. (d) Signal with 15000 samples.

Chapter 3

Online Sequence Synchronization Based on Dynamic Time Warping

A common problem in anomaly detection is the case where two signals have different lengths, which may occur, for instance, due to different sampling rates or if the signals were recorded using sensors moving with different speeds.

In this chapter, we investigate the alignment between two video sequences that register the same scene. It is considered that a camera goes multiples times through an environment. During each recording, the camera follows approximately the same trajectory but its speed along the trajectory may differ among recordings, thus generating time warping between the videos.

This work proposes a video alignment algorithm based on the dynamic time-warping (DTW), that can be used in anomaly detection systems. An online DTW approach is adapted and optimized in the context of real-time video alignment. The algorithm was tested with several image distance metrics using data acquired in a robotic platform that moved at different speeds. The videos were acquired using the robotic system described in Section 1.2.1, which also includes the simultaneous recording of several signals from various auxiliary sensors.

This chapter is divided as follows: in Section 3.1, a traditional method to align and compare signals with different lengths is described. Section 3.2 shows an online version of the traditional method that is able to compute a similar result with less computational complexity, and Section 3.3 adapts these methods to perform video alignment in the context of the moving-camera object detection. Section 3.4 that may have a difference in their lengths, and also shows tests to assess the method robustness Section 3.5 shows an application of the same side information employed in Chapter 2 for the alignment of signals.

3.1 Dynamic Time Warping

Dynamic time warping (DTW) is a technique that aligns two time series $\mathbf{X} = [x_1, x_2, \cdots, x_N]$ and $\mathbf{Y} = [y_1, y_2, \cdots, y_M]$ with coincidental beginnings and ends by warping one of the sequences in a nonlinear fashion to match the other. The DTW aims to find the minimum-cost path $W = [w_1; w_2; \cdots; w_L]$ which is a sequence of the ordered pairs $w_k = (i_k, j_k) \in [1 : N] \times [1 : M]$ such that x_{i_k} and y_{j_k} are aligned. This path W should satisfy some constraints:

- Boundary: $w_1 = (1, 1)$ and $w_L = (N, M)$;
- Monotonicity: $i_1 < i_2 < \cdots < i_L$, $j_1 < j_2 < \cdots < j_L$;
- Continuity: $w_{k+1} - w_k \in \{(1, 0), (0, 1), (1, 1)\}$.

To find the optimal warping path that aligns the time series $\mathbf{X}$ and $\mathbf{Y}$, one can create a cost matrix $\mathbf{d}$ of size $N \times M$ where each element $d(i, j)$ represents a similarity measurement between the samples x_i and y_j that is also the cost of their misalignment. The optimal warping path is the one that minimizes the sum of the costs along the path:

$$\mathrm{DTW}(\mathbf{X}, \mathbf{Y}) = \min \sum_{(i,j) \in W} d(i, j). \tag{3.1}$$

This problem can be easily solved by dynamic programming, creating an accumulated-cost matrix $\mathbf{D}$ with elements $D(i, j)$ using the following recursive formulation:

$$D(i, j) = d(i, j) + \min(D(i - 1, j), D(i, j - 1), D(i - 1, j - 1)). \tag{3.2}$$

The path is obtained by starting at the element $D(N, M)$ and testing each previous element $D(N-1, M)$, $D(N, M-1)$ and $D(N-1, M-1)$ in the recursion. For whichever has the smallest value, the corresponding index $(N - 1, M)$, $(N, M - 1)$ or $(N - 1, M - 1)$ is added to the path and the recursion continues from it until the element $D(1, 1)$ is reached, as described in Alg. 4.

3.2 Online Dynamic Time-Warping

The classical DTW, described in Section 3.1, requires that all samples from both sequences are known at the start of the execution of the algorithm, since it aligns the initial and final samples from each sequence beforehand. One of its drawbacks is that when one of the sequences is only partially known the boundary conditions

Algorithm 4 Dynamic time-warping algorithm.

```
    Input: Time series ($\mathbf{X} = [x_1, x_2, \cdots, x_N]$ and $\mathbf{Y} = [y_1, y_2, \cdots, y_M]$).
    Output: Warping path ($W = [w_1; w_2; \cdots; w_L]$).
 1: for $i \in \{1, ..., N\}$ and $j \in \{1, ..., M\}$ do
 2:     Compute $d(i, j) = f(x_i, y_j)$
 3:     Compute $D(i, j)$ using Eq. (3.2)
 4: end for
 5: Compute the warping path $W$ using Alg. 5
```

Algorithm 5 Dynamic programming to find the optimal warping path given an accumulated-cost matrix.

```
    Input: Accumulated-cost matrix ($\mathbf{D}$).
    Output: Warping path ($W = [w_1; w_2; \cdots; w_L]$).
 1: Initialize an empty warping path $W$
 2: Assign $i = N$ and $j = M$
 3: while $i > 0$ and $j > 0$ do
 4:     Assign $w = (i, j)$
 5:     Append $W = [w; W]$
 6:     if $D(i-1, j) < D(i-1, j-1)$ and $D(i-1, j) < D(i, j-1)$ then
 7:         Assign $i = i - 1$
 8:     end if
 9:     if $D(i, j-1) < D(i-1, j-1)$ and $D(i-1, j) < D(i-1, j)$ then
10:         Assign $j = j - 1$
11:     end if
12:     if $D(i-1, j-1) \leq D(i-1, j)$ and $D(i-1, j) \leq D(i, j-1)$ then
13:         Assign $i = i - 1$ and $j = j - 1$
14:     end if
15: end while
```

cannot be satisfied. The online DTW proposed by [32] seeks the best alignment of a partially unknown target sequence and a subsequence of the reference, restricting the search to a prealigned window so that the algorithm has linear complexity.

Starting with reference and target subsequences of the size of the search window c, the algorithm applies the standard DTW to find an initial warping path, inserting a weight 2 for diagonal steps in the definition of the accumulated cost matrix of Eq. (3.2), so that there is no bias for diagonal movements:

$$D(i,j) = \min \begin{cases} D(i-1,j) + d(i,j) \\ D(i,j-1) + d(i,j) \\ D(i-1,j-1) + 2d(i,j) \end{cases} . \tag{3.3}$$

In the first iteration, the algorithm considers that the reference and target sequences are composed of c values, so only the elements of the cost function $d(i,j),\ i,j = 1,\cdots,c$ are computed. Eq. (3.3) is used to determine the values of the elements $D(i,j),\ i,j = 1,\cdots,c$ of the accumulated-cost matrix $\mathbf{D}$.

For each new iteration, the algorithm uses the values of the accumulated-cost computed up to a given point to decide whether to increase the size of the reference or the target subsequences. Afterwards, instead of re-computing all costs between samples from both sequences, it uses the cost matrix found in the previous iteration and only updates it with the costs associated with the new sample. In addition, in order to spare computation, it only computes the values of the cost function between the new sample from one sequence and the last c samples from the other sequence (instead of all of them, as in the original DTW). The accumulated-cost matrix $\mathbf{D}$ is also updated by applying Eq. (3.3) only for the positions where the value of $d(i,j)$ is currently known, and a new warping path between the reference and target subsequences is found. Alg. 6 describes these ideas.

In Fig. 3.1, an example of the evolution of the cost matrix computation is presented. In the figure, the white squares represent pixels not yet computed in the cost matrix, the light gray squares represent the initial elements computed in the cost matrix, and the dark gray square represents the warping path computed in the forward direction. During each step, the algorithm checks the forward path to decide if it should include in the matrix a new sample from the target sequence (Figs. 3.1(b) and 3.1(c)), reference sequence (Figs. 3.1(e) and 3.1(f)) or both (Fig. 3.1(d)), and computes the similarity metric between the new sample from one sequence and the last 4 samples from other sequence. In this figure, it can be seen that, starting in Fig. 3.1(d), a few samples from the cost matrix were not computed, remaining as white squares in the figure.

Algorithm 6 Online dynamic time-warping algorithm.

```
    Input: Time series ($\mathbf{X} = [x_1, x_2, \cdots, x_N]$ and $\mathbf{Y} = [y_1, y_2, \cdots, y_M]$), search window size ($c$).
    Output: Warping path ($W = [w_1; w_2; \cdots; w_L]$).
 1: for $i \in \{1, ..., c\}$ and $j \in \{1, ..., c\}$ do
 2:     Compute $d(i, j) = f(x_i, y_j)$
 3:     Compute $D(i, j)$ using Eq. (3.3)
 4: end for
 5: Assign $updateType$ = "$BOTH$", $previousUpdate$ = "$BOTH$"
 6: Assign $updateCount = 0$, $ref = c$, $tar = c$
 7: while $ref \leq N$ and $tar \leq M$ do
 8:     Define $updateType$ using Alg. 7
 9:     if $updateType$ == "$REF$" or $updateType$ == "$BOTH$" then
10:         for $i = ref + 1$ and $j \in \{tar - c + 1, ..., tar\}$ do
11:             Compute $d(i, j) = f(x_i, y_j)$
12:             Compute $D(i, j)$ using Eq. (3.3)
13:         end for
14:     end if
15:     if $updateType$ == "$TAR$" or $updateType$ == "$BOTH$" then
16:         for $i \in \{ref - c + 1, ..., ref\}$ and $j = tar + 1$ do
17:             Compute $d(i, j) = f(x_i, y_j)$
18:             Compute $D(i, j)$ using Eq. (3.3)
19:         end for
20:     end if
21:     if $updateType \neq previousUpdate$ and $updateType \neq$ "$BOTH$" then
22:         Assign $updateCount = updateCount + 1$
23:     else
24:         Assign $updateCount = 1$
25:     end if
26:     Assign $previousUpdate = updateType$
27: end while
28: Compute the warping path $W$ using Alg. 5
```

Algorithm 7 Algorithm to determine the next update in the online DTW.

```
Input: Current number of consecutive updates of same type ($updateCount$), maximum number of consecutive updates ($maxCount$), previous update type ($previousUpdate$), search window size ($c$), accumulated-cost matrix ($\mathbf{D}$), last reference position ($ref$), last target position ($tar$).
Output: Type of update used in the current iteration ($updateType$).
 1: if $updateCount > maxCount$ then
 2:     if $previousUpdate ==$ "$TAR$" then
 3:         Assign $updateType =$ "$REF$"
 4:     end if
 5:     if $previousUpdate ==$ "$REF$" then
 6:         Assign $updateType =$ "$TAR$"
 7:     end if
 8: else
 9:     Find $i$ such that $D(i, tar) = min(D(1, tar), ..., D(ref, tar))$
10:     Find $j$ such that $D(ref, j) = min(D(ref, 1), ..., D(ref, tar))$
11:     if $D(i, tar) < D(ref, j)$ then
12:         Assign $updateType =$ "$TAR$"
13:     end if
14:     if $D(i, tar) > D(ref, j)$ then
15:         Assign $updateType =$ "$REF$"
16:     end if
17:     if $D(i, tar) == D(ref, j)$ then
18:         Assign $updateType =$ "$BOTH$"
19:     end if
20: end if
```

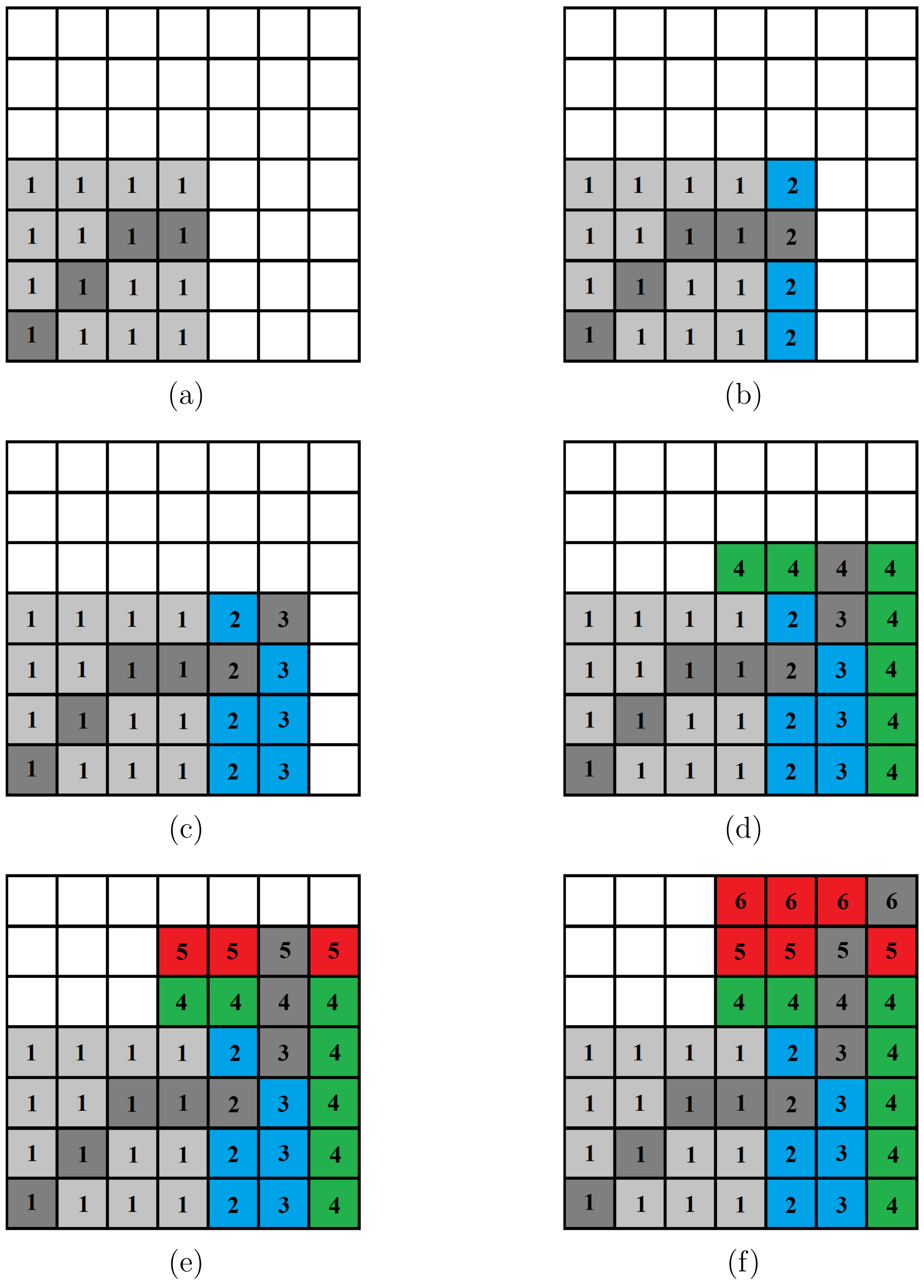


Figure 3.1: Example of the cost matrix computation performed by the online DTW algorithm. The white square represents pixels not yet computed in the cost matrix, the light gray squares represent the initial elements computed in the cost matrix and the dark gray square represents the warping path computed in the forward direction. The number inside each square represents the iteration where that element of the cost matrix was computed. (a) Initial elements. (b) and (c) New sample from the target sequence included. (d) New sample from both the target and reference sequence included. (e) and (f) New sample from the reference sequence included.

3.3 Video Alignment for Moving Camera Object Detection

The framework of surveillance systems with moving camera object detection imposes several constraints that must be satisfied by the alignment algorithm. In this application, one of the sequences, the reference signal, is fully known and the other sequence, the target sequence, is being received in real-time and must be aligned and processed on-the-fly, which makes the online DTW a suitable approach. However, in order to be used in this framework, some innovations had to be made to the online DTW algorithm.

Since the original algorithm was developed for a music application, a new cost function must be applied in order to align the video frames. Furthermore, when dealing with videos acquired in a surveillance operation, one can often deal with frames recorded in the same position that have regions with different information. As can be seen in Fig. 1.3, the frames from the target video may have regions with video anomalies. In this case there may be objects that did not exist or were in a different position during the reference recording. Thereby, the alignment algorithm must be able to align frames even when one of them has small regions that do not match the ones in the other. We have performed several tests to determine the best cost function to be used in this application and propose the use of a simple metric, the mean square error (MSE) between subsampled frames, showing in Tabs. 3.2 and 3.3 that it produces the best compromise between error rate and processing time.

The original algorithm proposed by Dixon [32] performs a warping between two videos which can include repetitions of any of the frames of the videos. However, in an object detection application, the real concern is, for each new frame in the reference video, finding a frame that is equivalent to each new frame in the target video. Therefore, the proposed algorithm computes the optimal warping path and, for each target frame, finds the aligned reference frame with the minimum cost.

In addition, the online DTW algorithm computes the path in the forward direction, incrementally computing the optimal warping for each new frame. In the proposed system, a latency in the warping path computation is introduced, by computing the alignment for a given target frame only after the k subsequent frames were received. This approach was discussed in [32] and was deemed unnecessary in the context of music alignment. However, since this work deals with a different application, this approach with latency is also considered.

3.4 Experimental Results

Using the robotic platform described in Section 1.2.1, six runs of the robot were performed and a video and its equivalent sensor information were acquired. From these runs, two videos were used as reference videos and four as target videos in which at least one anomalous object was placed in the environment. For three target videos, the robotic platform was programmed to vary its speed between 0.2 m/s and 0.4 m/s along the trajectory. A reference video for these three target videos was recorded with a constant speed of 0.2 m/s, which generates a time-warping between the reference and target videos to be aligned. The fourth target video was recorded with a constant speed of 0.1 m/s, but containing regions with larger anomalous objects than the other three target videos, as the one seen in Fig. 1.3, and a reference video for this target was also recorded at the constant speed of 0.1 m/s (having no time warping). All videos have a spatial resolution of 800×450 pixels and a frame rate of around 2.5 fps. Tab. 3.1 summarizes some of the properties of the videos. The proposed algorithm was implemented in C++ and tested with several configurations. The tests were made in a computer with an Intel Core i7-3630 QM processor with 2.4 GHz clock and 16 GB of RAM running Windows 10 ©.

Table 3.1: Properties of the videos used in the tests.

	Duration (s)	**Total frames**	**Camera speed (m/s)**
Target 1	560	1400	0.2 to 0.3
Target 2	486	1215	0.2 to 0.4
Target 3	488	1218	0.2 to 0.4
Target 4	1329	3176	0.1
Reference 1	649	1622	0.2
Reference 2	1346	3050	0.1

3.4.1 Tests with Different Cost Functions

To test the robustness of the DTW algorithm in this application, several cost functions were considered. In [28], a DTW is developed which uses a subsampled version of the frame as the frame descriptor, and uses as cost function the L_1-norm between frame descriptors. In this work, we subsample the frames to the size 16×9, stack their lines in a descriptor vector and consider both L_1 and L_2 norms.

The moving-camera background-subtraction algorithm proposed in [34] employs the normalized vector distance (NVD) [90] to compare frames and detect anomalies, which can also be applied as a cost function in a DTW algorithm. For this test, the original frames are subsampled to the size 80×45 and each frame is divided into 25 image patches. Other common metrics for comparing frames include the structural similarity (SSIM) [91] and distance between the histogram of oriented gradients (HoG) descriptors [92]. The HoG descriptor is based on the implementation given by [93] and the SSIM is applied in the comparison of the frames after downsampling them to the size 32×18.

The original DTW described in Sec. 3.1 was also tested in the videos for comparison purposes. Since it has a computational complexity that is quadratic on the length of the videos, this method was only tested with the cost function that computes the L_2-norm of the error between subsampled frames. In Fig. 3.2, one can see an example of the cost matrix computation using the original DTW and the online approach. In Fig. 3.2(b), the white regions in the upper right and lower left represent all similarities that were not computed by the online DTW, which did not result in any alignment error (Figs. 3.2(c) and 3.2(d)).

For the sake of comparison, the sensor data information obtained from the DORIS system (the set of auxiliary signals obtained during each recording) is also adapted to a DTW algorithm. This information is the same used for the video alignment discussed in Chapter 2, but in this new experiment, it was imposed that the reference and target sequences can have a different speed, so the premises of the algorithm described in Chapter 2 were not satisfied. Considering $\mathbf{r}_i$ the set of sensor outputs from the reference recording at time i, and $\mathbf{t}_j$ the ones from the target recording at time j, we applied the DTW with cost function:

$$d(i,j) = \|\mathbf{r}_i - \mathbf{t}_j\|^2 . \tag{3.4}$$

It is also important to emphasize the alignment that was actually computed by the online DTW algorithm. This algorithm computes an online warping path that only considers the current samples available and is used to control the comparisons that must be computed or that can be ignored. This path is represented as the dark gray squares in Fig. 3.1. However, this warping path was used only during the execution of the algorithm and not analyzed in this first experiment.

For this experiment, the online algorithm decides which comparisons are or not necessary, creating the cost matrix depicted in Fig. 3.1. After this procedure, a warping path is found using Alg. 5, which may or may not coincide with online warping path depicted as the dark gray squares in Fig. 3.1. Thereby, this experiment strictly analyzes the impact of the several cost functions and the reduced cost matrix on the alignment between the target and reference sequences. In the second experiment, we perform an analysis of the quality of the incremental path estimated by the online approach.

Tab. 3.2 presents the alignment error between the several cost functions tested in the DTW algorithm. The positioning estimate of the DORIS system, which was

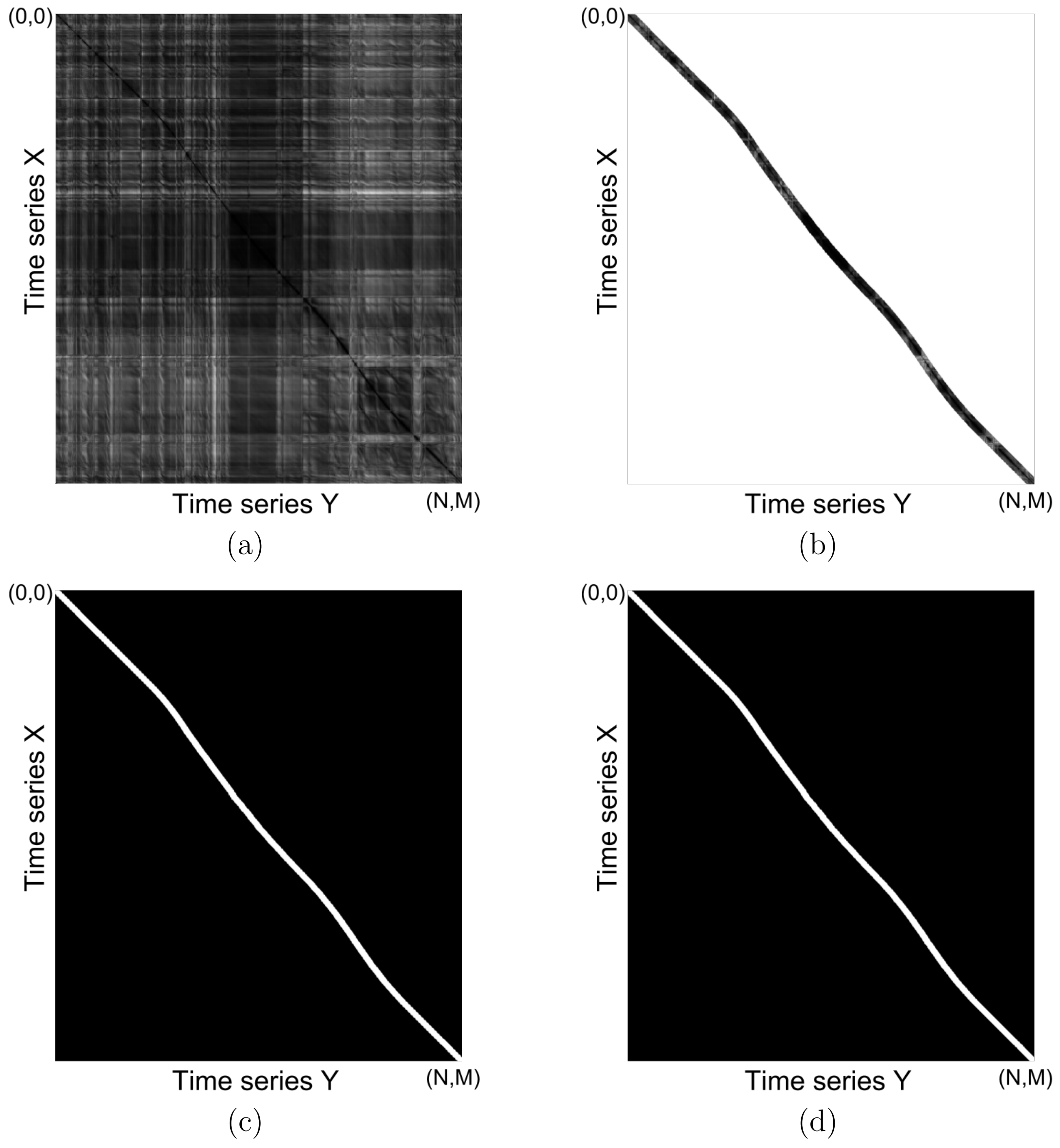


Figure 3.2: Example of the cost matrix computation in the DTW algorithm. In the figures, the warping path was too thin and was dilated for a better visualization. (a) Original DTW - The full matrix is computed. (b) Online DTW - Only a region of the cost matrix inside a given search window is computed. (c) Original DTW - Warping path. (d) Online DTW - Warping path.

also used in Chapter 2, was employed in the computation of a video alignment between the reference and target videos, and its results were considered as ground-truth. For each cost function, the DTW alignment was computed, which gives for each frame of the target video the corresponding frame of the reference video. The alignment error is computed by taking the average of the absolute difference between the frame position given by the estimated alignment and the frame position given by the ground truth.

Tab. 3.3 shows the average processing time for each cost function. As can be seen from the results in Tabs. 3.2 and 3.3, the cost function based on the L_2-norm (which represents, up to scale, the MSE between subsampled frames) is only outperformed, in terms of alignment error, by the cost functions based on NVD and SSIM. However, it is at least two times faster than both of them. Given that it is advantageous that the alignment step be as simple as possible due to the very complex nature of the anomaly detection step, this indicates that the MSE is the recommended cost function to be used in a real-time application.

Table 3.2: Alignment error (in frames) for several cost functions used in the DTW algorithm. The best 3 results are marked in blue.

		Average error (frames)			
		Target 1	**Target 2**	**Target 3**	**Target 4**
Online DTW	L_1	0.95	0.92	1.45	0.44
	L_2	0.58	**0.48**	**0.80**	**0.37**
	SSIM	**0.41**	**0.39**	**0.66**	**0.36**
	NVD	**0.48**	**0.60**	**0.78**	**0.32**
	HoG	**0.45**	0.61	0.95	0.38
	Sensor	4.88	6.08	5.86	4.84
Original DTW	L_2	0.58	0.48	0.80	0.37

Table 3.3: Processing time for several cost functions used in the DTW algorithm. The best 3 results are marked in blue.

		Processing time (s)			
		Target 1	**Target 2**	**Target 3**	**Target 4**
Online DTW	L_1	**76**	82	89	**283**
	L_2	**83**	**80**	**87**	308
	SSIM	189	187	189	516
	NVD	243	233	236	671
	HoG	89	**79**	**83**	**280**
	Sensor	**19**	**18**	**20**	**76**
Original DTW	L_2	41311	52954	40567	132571

3.4.2 Online Warping

In a real-time anomaly-detection application, a frame from the target video must be synchronized to a frame from the reference video in order to produce an output without prior knowledge of any posterior target frames. If the system can allow a fixed latency by producing a detection output for the target frame N only after K new target frames have been received, the optimal alignment for frame N can be estimated with a fixed view into the future, which can make it more stable and reliable. In this case, the optimal warping path used to align the target frame N is computed between a subset of the reference sequence and the target sequence up to the frame $(N + K)$.

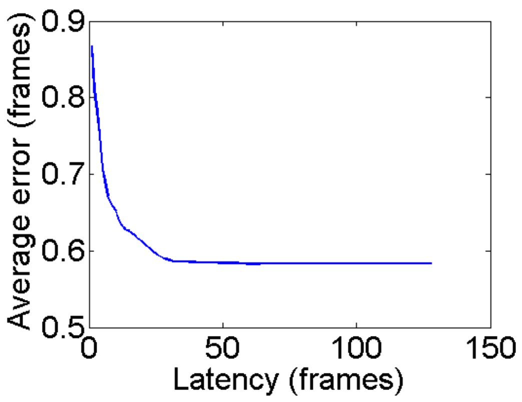
In this test, we analyze how the DTW algorithm behaves when providing a frame alignment in a real-time application. Using as cost function the MSE between subsampled frames, we vary the allowed latency in the system. Using the online DTW algorithm, the cost matrix is gradually filled and whenever it includes elements computed from a new target frame $(N + K)$, a warping path is computed and the algorithm outputs the best aligned reference frame for the target frame N. Note that for a zero latency, the path computed is the one shown in a dark gray color in Fig. 3.2. The results can be compared to the ones shown in Tab. 3.2, by considering that it represents the case when all information from the target video is available in the computation of the video alignment, so it is equivalent to the alignment error with infinite latency.

Fig. 3.3 shows the alignment error obtained when using several values of latency in the warping computation. The results show that, contrary to what is stated in [32], the use of latency can reduce the alignment error up to a third of the one obtained when using only the current target frame. Fig. 3.3 also shows that with a latency of 50 frames, which corresponds to approximately 20 s, the error becomes close to the minimum, reaching a value similar to the one in Tab. 3.2. For all cases, a good trade-off is found when using a latency of around 15 frames, which represents a delay of approximately 6 s and is not prohibitive for the considered application.

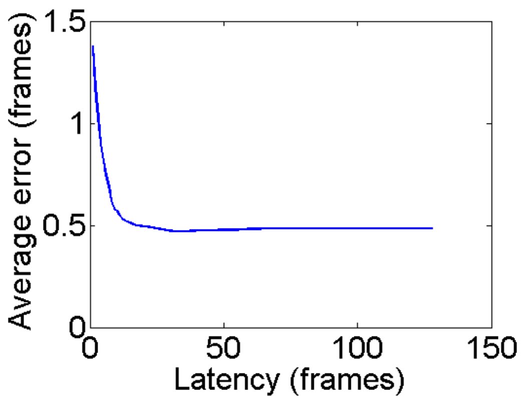
3.4.3 Tests of Robustness

Due to the lack of a larger amount of video samples, since the recording demands a displacement of people and equipment, and also require prior scheduling of a recording in an industrial environment, our video database does not have much content variability. To test the robustness of the DTW algorithm in the alignment of surveillance videos, we simulated some conditions that can occur in the acquisition of videos by the DORIS system.

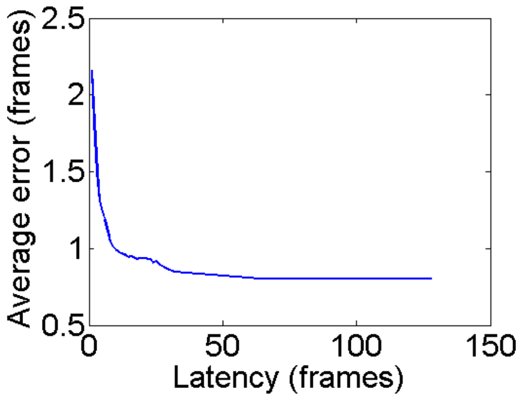
Among the problems that can occur during video acquisition, one of the most common is a change in the field of view. Since, in this application, a camera is mounted on a robotic platform hung on a trail with a closed-loop trajectory, the traction can sometimes make the camera shake or even make the robot slightly twist around the rail, which can create differences in the camera field of view during the reference and target recording. To simulate this effect, a random crop was performed by fixing a window size (proportional to the frame size) and randomly selecting a window of the given size inside each frame of the reference and target videos. Although this problem was not so common in the videos acquired using the



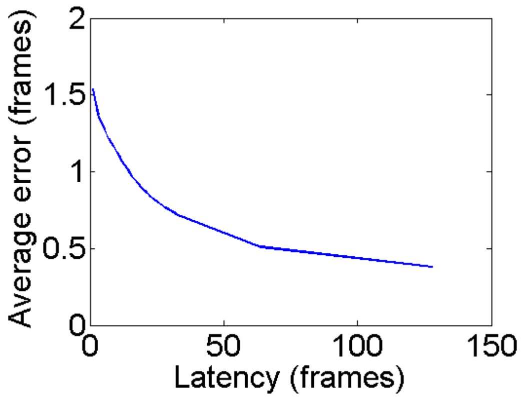
(a)



(b)



(c)



(d)

Figure 3.3: Average alignment error (in frames) using as cost function the MSE between subsampled frames. (a) Video 1. (b) Video 2. (c) Video 3. (d) Video 4.

DORIS system, it occurred quite frequently for videos from the VDAO database. An example of a frame mismatch from this database can be seen in Fig. 3.4.

Another common problem is a difference in illumination in the videos. Since the recordings took several hours in the same day and there are regions in the video that show the external environment, the movement of the sun and clouds can create differences in the average illumination, regions of shadow, and/or sunlight reflection. An example of the difference of illumination in videos acquired using the DORIS system is depicted in Fig. 3.5. To create a variation between the target and reference illumination, the target frames were processed in order to increase their contrast.

The last problem considered for this test is the noise. During the recordings, the videos were acquired by a camera and transmitted, via wi-fi, to a receiver that is always at least 10 m away. However, since there is a lot of electrical machinery inside the industrial plant, the interference created by these objects can result in a loss of data packets and frames, and the resulting video often has repeated frames and reduced quality. In order to assess the algorithm performance when dealing with this problem, we induced a difference between reference and target frames by compressing every target frame with lower quality.

In Tab. 3.4, the results of the alignment algorithm for one target video after applying the several deformations in the frames are presented. The results indicate that among all conditions that can affect the video acquisition, the shaking or twist in the robot is the main responsible for the loss of performance of the DTW algorithm. This result is somewhat expected since these deformations can create videos with little or no content in common between the reference and target videos, so there is no equivalent information in both videos to be corresponded. However, often this effect is temporary: it is possible that some frames from the target video do not have much content in common with the reference frames, due to a difference in the field of view caused by a shaking or a twist in the robot position, and there is no synchronism between the videos, but after a while the robot can stop shaking or enter a straight section of the rail and correct any abnormal twist that occurred

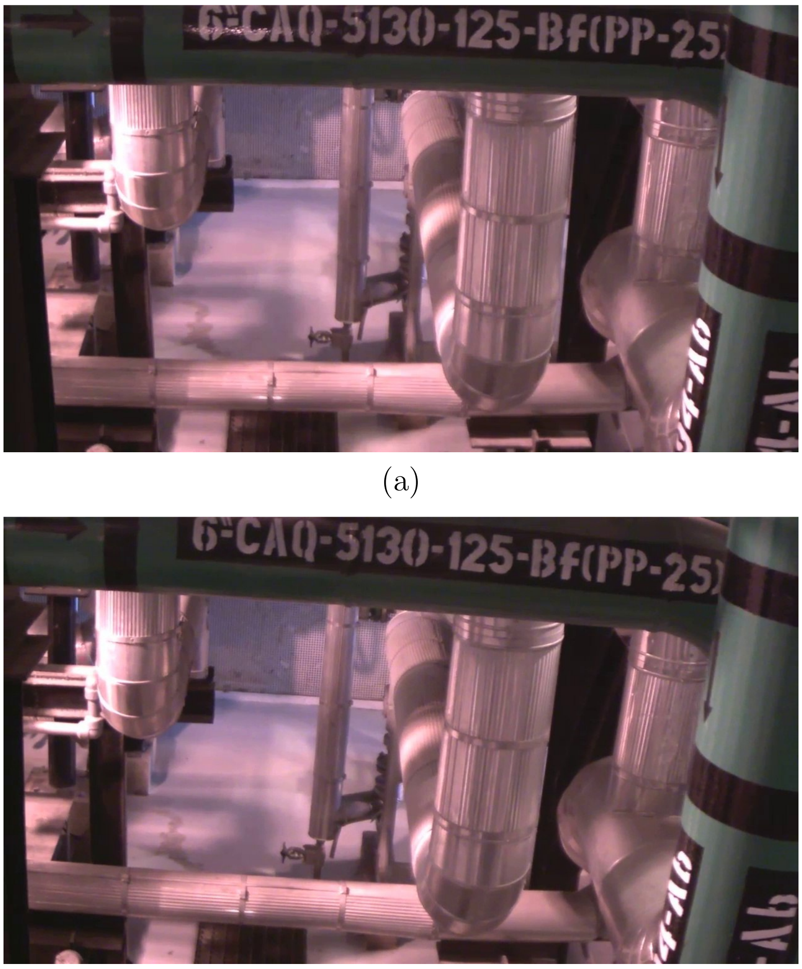
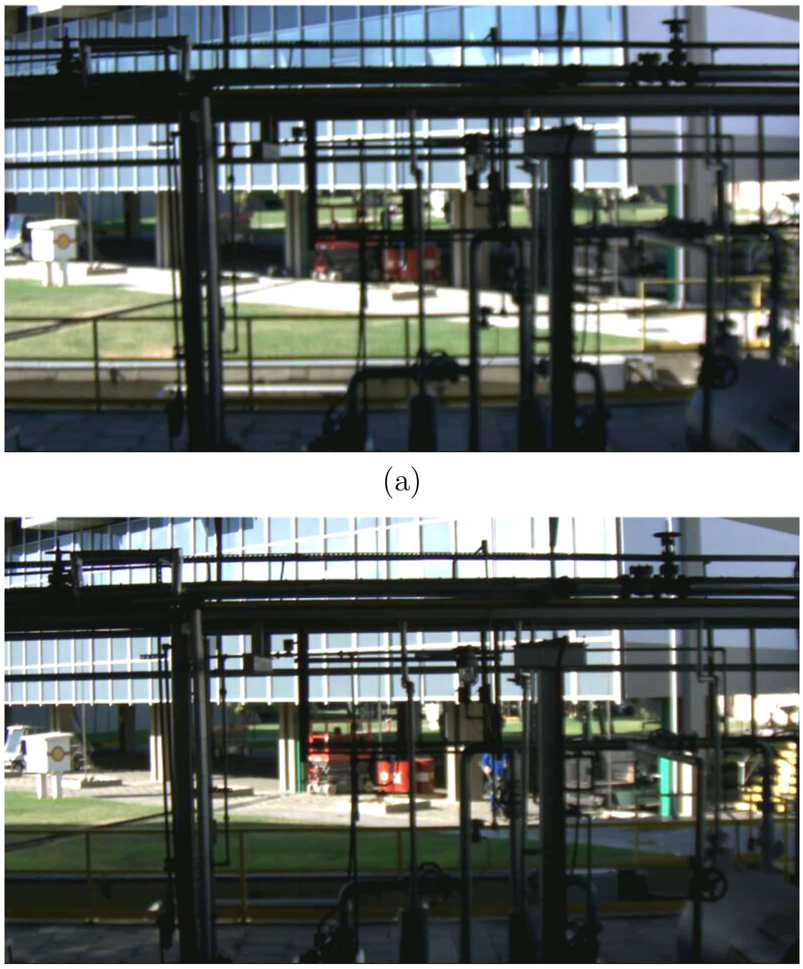


Figure 3.4: Example of frame mismatch from the VDAO database [94]. (a) Frame from a reference video. (b) Equivalent frame in another video with a mismatch. Due to the movement, the camera was rotated with respect to the first recording.

during a curve. Therefore an algorithm should be able to recover the synchronism when it is available.

Table 3.4: Alignment error (in frames) for several effects applied to the video 1.

	Average error (frames)
Original	0.58
Crop (90 % of the frame)	1.46
Crop (70 % of the frame)	2.88
Crop (60 % of the frame)	21.41
Contrast	0.72
Compression	0.97


(a)

(b)

Figure 3.5: Example of the difference of illumination in videos acquired with the DORIS system. Two videos that were recorded at different times during the same day, and may present a different light pattern. (a) Frame of a recording where the lawn is under sunlight. (b) Equivalent frame from another recording where a shadow covers the lawn.

3.5 Analysis Based on the Sensor Data

The sensor data can provide valuable clues to complement the information given by the video content. However, as can be seen from the results of Tab. 3.2, applying the DTW algorithm to the whole set of sensor data yielded worse results than any method that uses solely the video content. A thorough analysis of the sensor data available in the DORIS system must be performed in order to determine which sensors can actually contribute to the robot localization and how this information can be extracted.

The set of sensor data is shown in the Fig. 3.6 for a target video and in Fig. 3.7 for its reference video. An analysis of these plots shows that these sensors have two distinct behaviors. For most of the sensors (plots (a), (b), (c), (d), (f), (g), (h), (i) and (l)), the signals are composed of a background noise and some sparse peaks that, if compared to the model of the rail in Fig. 1.2, appear in positions in which the robot moves along a curve. The other sensors (plots (j), (k), (m) and (n)) create signals that are more elaborate than only a sequence of peaks followed by noise. Due to this distinct nature, the sensors were split in two sets (sparse and non-sparse signals), and experiments were performed in each set. Moreover, a visual inspection revealed two signals that seem to be more influenced by the background noise (plots (e) and (o)), which represent the acceleration in the x axis and the average current consumption. These signals were disregarded for these tests, since it is likely that are not suited for this algorithm.

3.5.1 Alignment of Non-Sparse Signals

The set of sensors that generates non-sparse signals is shown in Figs. 3.8 and 3.9. For these sensors, the approach of using a DTW algorithm in the whole set of sensors ensemble, using Eq. (3.4) as cost function, was once again employed.

Fig. 3.10 shows the alignment error for each frame in the target sequence. The result shows that the DTW algorithm is not able to properly align the signals. This fact may occur because the signals acquired from these sensors change significantly for different recordings, and the DTW algorithm is not robust to this effect.

3.5.2 Alignment of Sparse Signals

For the set of sensors producing signals that have sparse peaks added to a background noise, the DTW algorithm cannot be applied directly. As can be seen from the Figs 3.11 and 3.12, the cost function computed for these signals has a few sparse small regions of local minima in the cost matrix. This behavior is caused by the fact that most of the comparisons are made between samples from the reference and

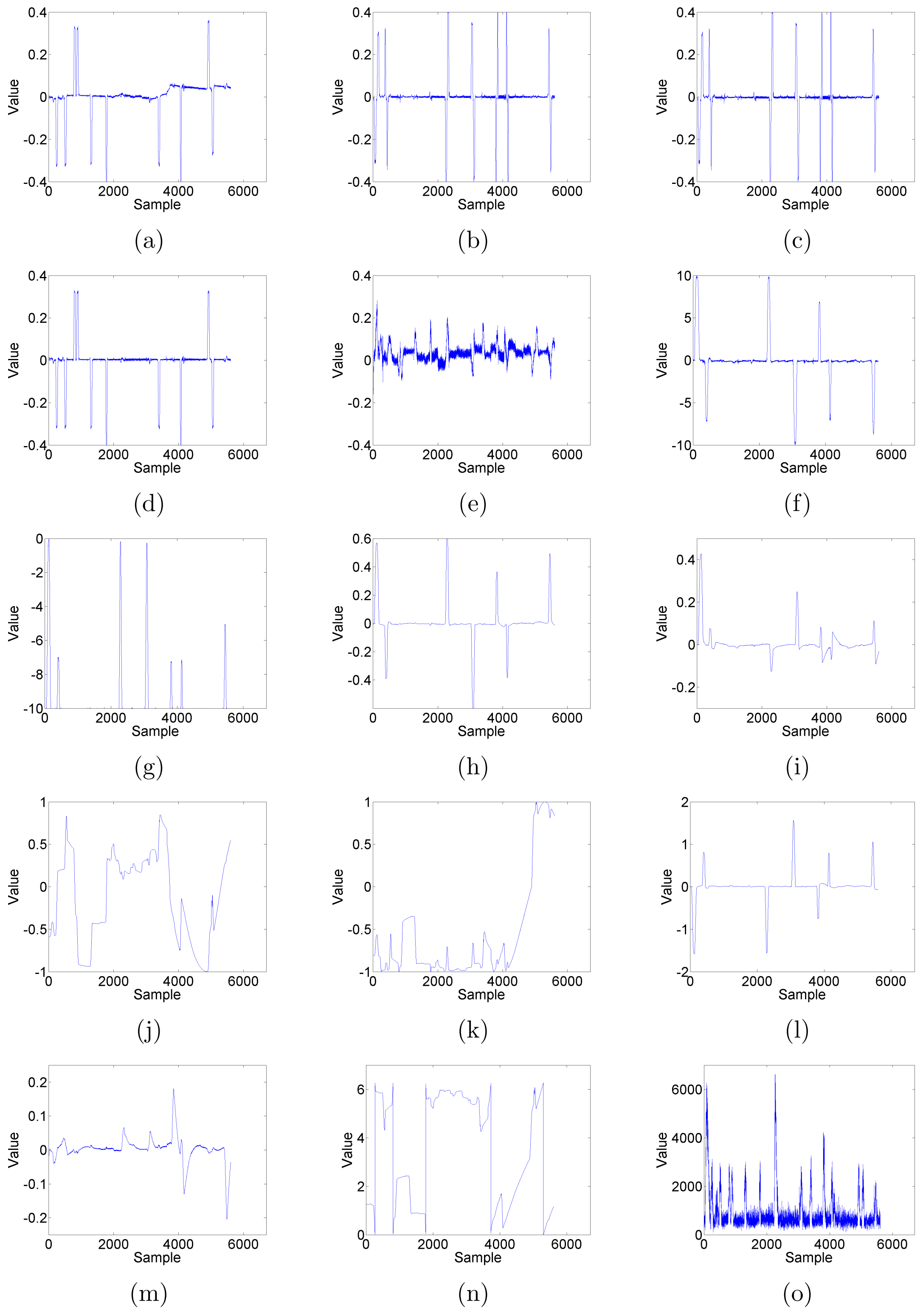


Figure 3.6: Set of sensors from a target recording. The visual information for each plot was reduced in order to treat each signal as a generic signal in the set. (a) Yaw velocity. (b) Roll velocity. (c) Angular velocity - x axis. (d) Angular velocity - y axis. (e) Acceleration - x axis. (f) Acceleration - y axis. (g) Acceleration - z axis. (h) Orientation quaternion x. (i) Orientation quaternion y. (j) Orientation quaternion z. (k) Orientation quaternion w. (l) Roll. (m) Pitch. (n) Yaw. (o) Average current consumption.

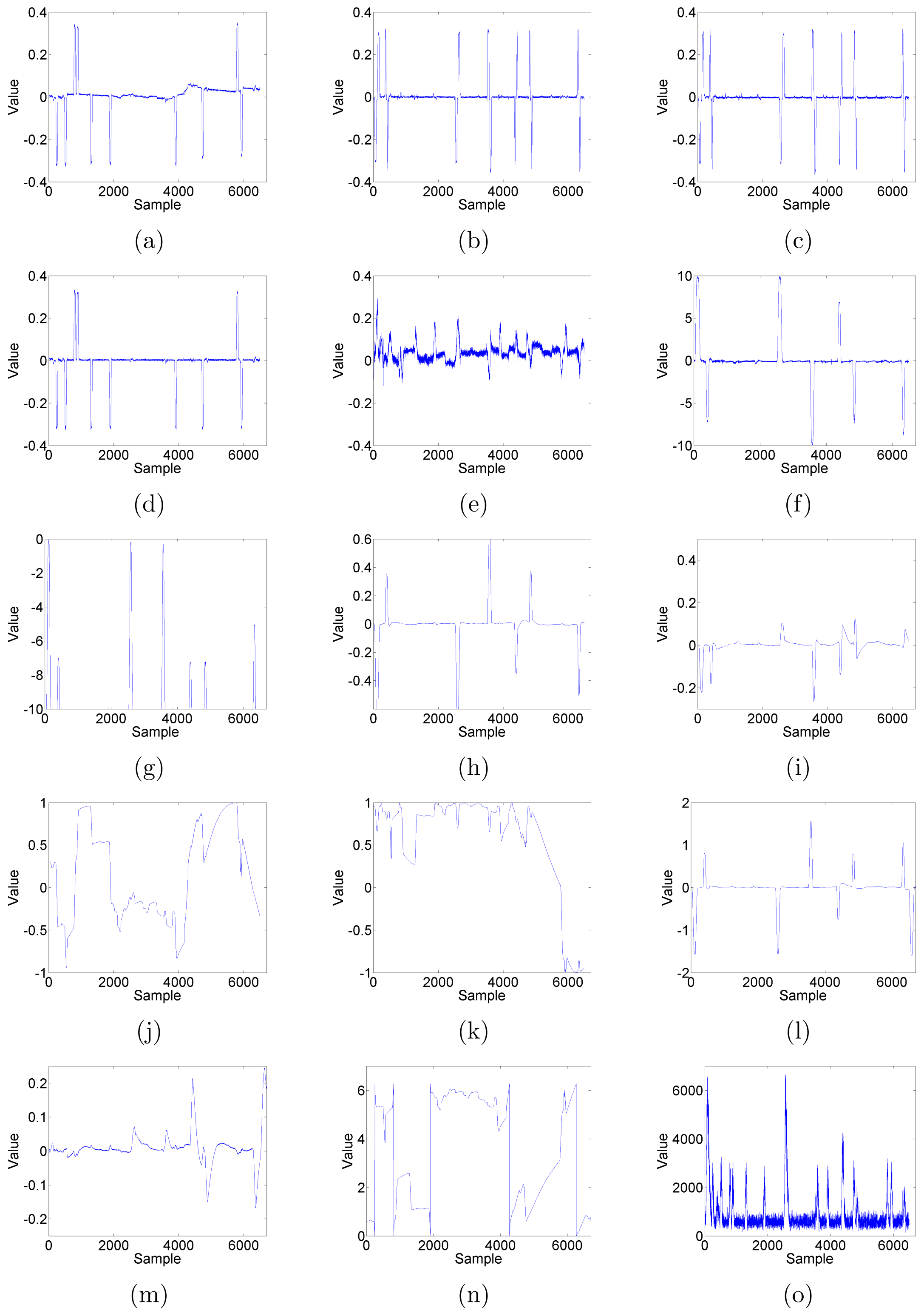


Figure 3.7: Set of sensors from a reference recording. The visual information for each plot was reduced in order to treat each signal as a generic signal in the set. (a) Yaw velocity. (b) Roll velocity. (c) Angular velocity - x axis. (d) Angular velocity - y axis. (e) Acceleration - x axis. (f) Acceleration - y axis. (g) Acceleration - z axis. (h) Orientation quaternion x. (i) Orientation quaternion y. (j) Orientation quaternion z. (k) Orientation quaternion w. (l) Roll. (m) Pitch. (n) Yaw. (o) Average current consumption.

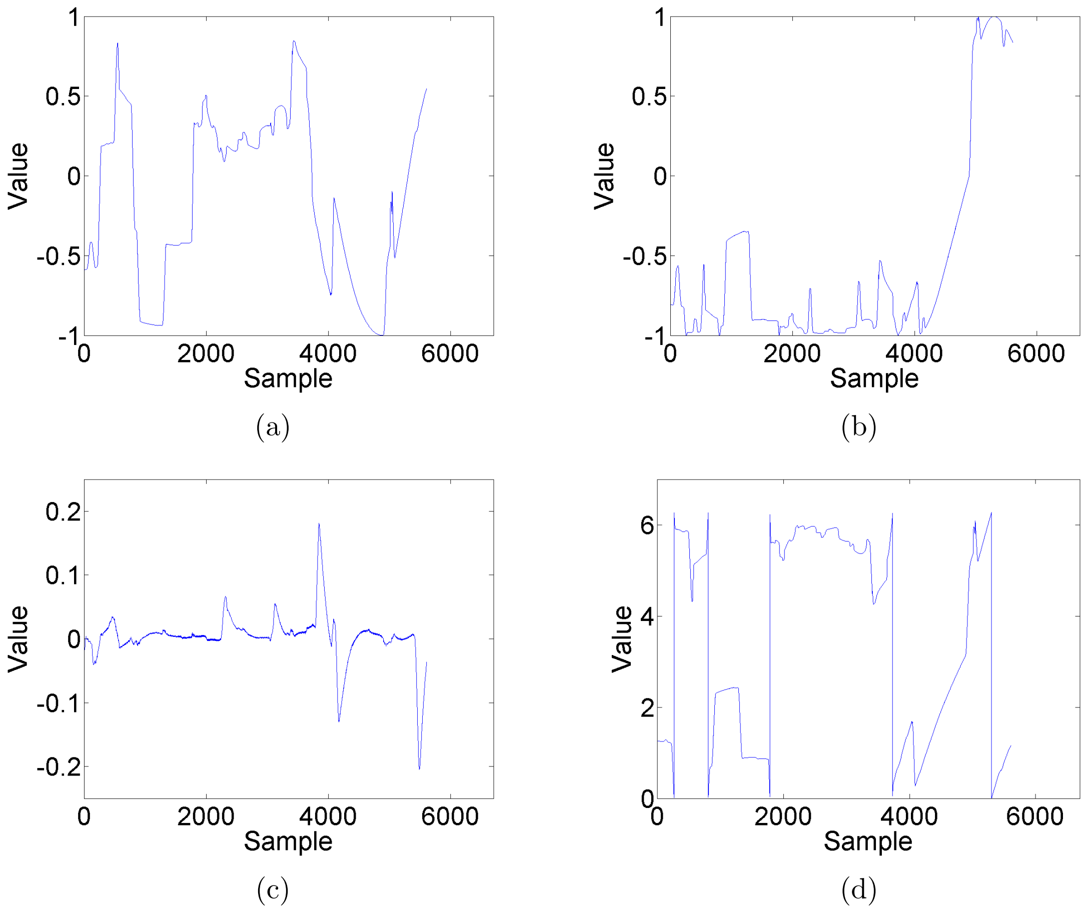


Figure 3.8: Set of non-sparse signals from a target recording used in the alignment. The visual information for each plot was reduced in order to treat each signal as a generic signal in the set. (a) Orientation quaternion z. (b) Orientation quaternion w. (c) Pitch. (d) Yaw.

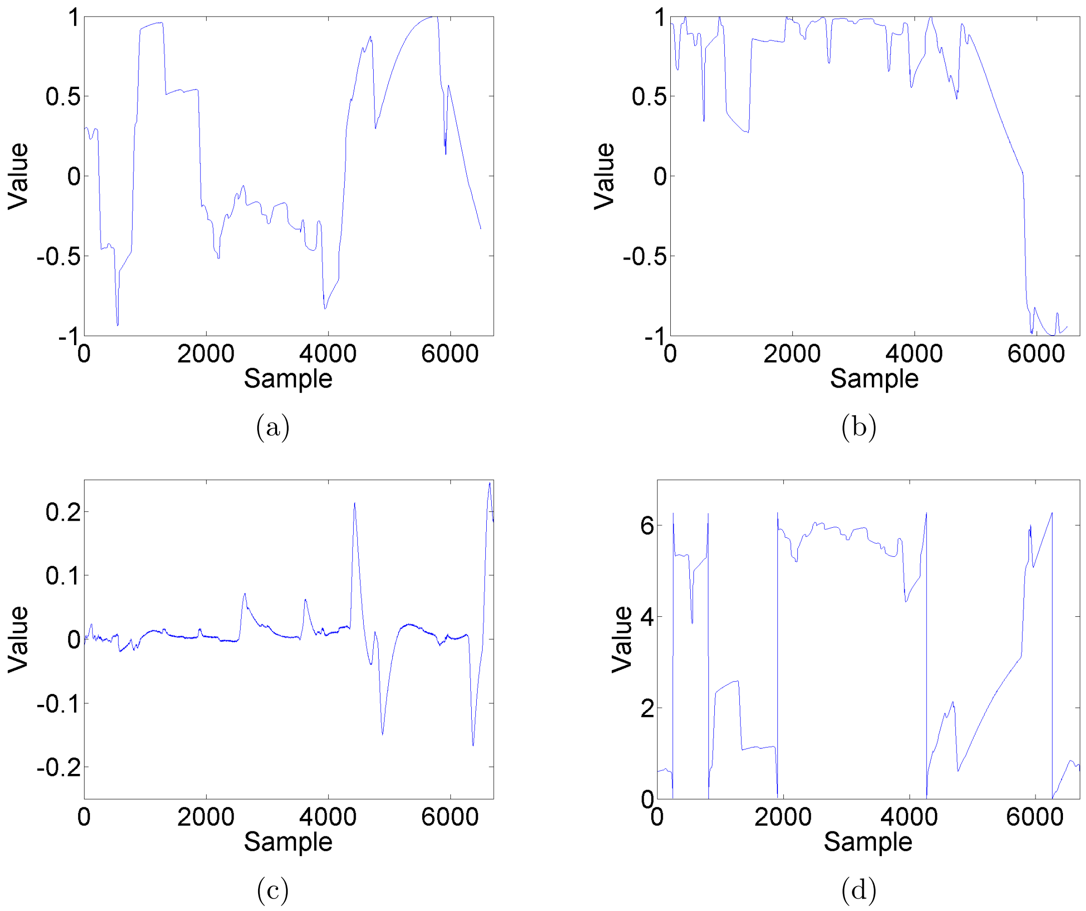


Figure 3.9: Set of non-sparse signals from a reference recording used in the alignment. The visual information for each plot was reduced in order to treat each signal as a generic signal in the set. (a) Orientation quaternion z. (b) Orientation quaternion w. (c) Pitch. (d) Yaw.

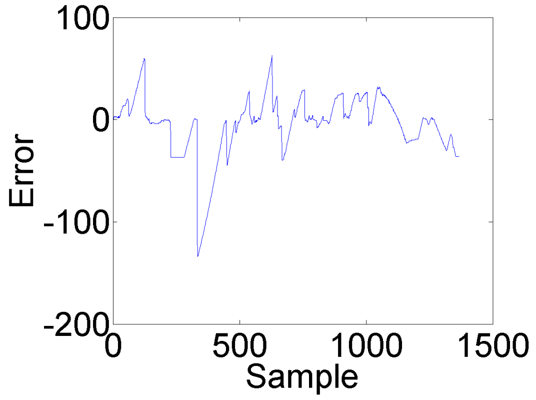


Figure 3.10: Alignment error (in frames) using the non-sparse sensor ensemble corresponding to the run 1. One can see that the algorithm loses track of the correct alignment in the regions with background noise.

target signals that just have background noise. Therefore, not enough information is provided for the DTW algorithm to properly align the sequences.

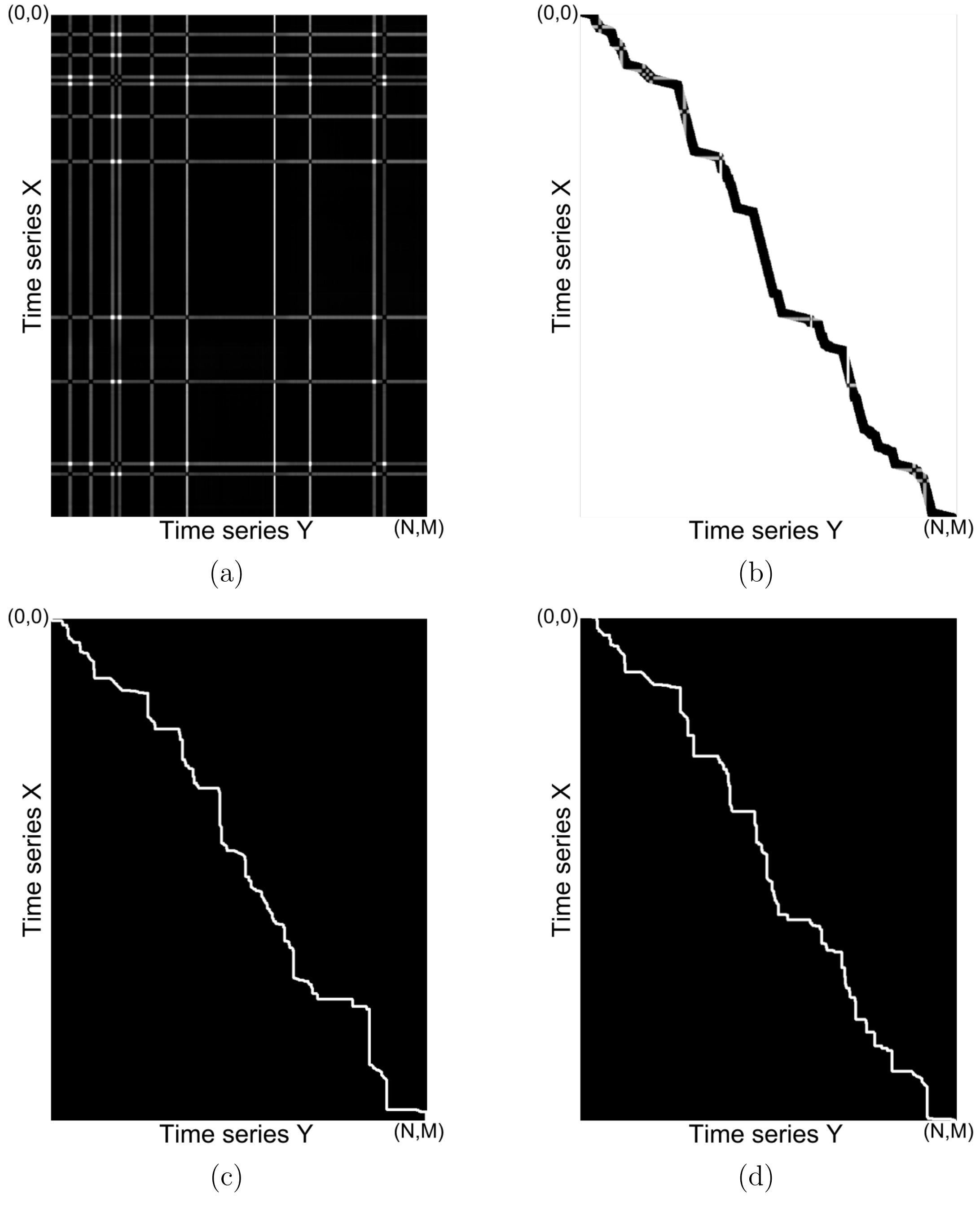


Figure 3.11: Example of the cost function computation for the sensor that measures the yaw velocity. (a) Original DTW - The full matrix is computed. (b) Online DTW - The algorithm fails to detect the region that contain the optimum alignment between the signals, when compared to Fig. 3.2. (c) Original DTW - Warping path. (d) Online DTW - Warping path.

For this experiment, a threshold was applied in the reference and the target signals to extract the interval of samples that contain each peak, which are the only samples that provide meaningful information to align the signals. Then, the

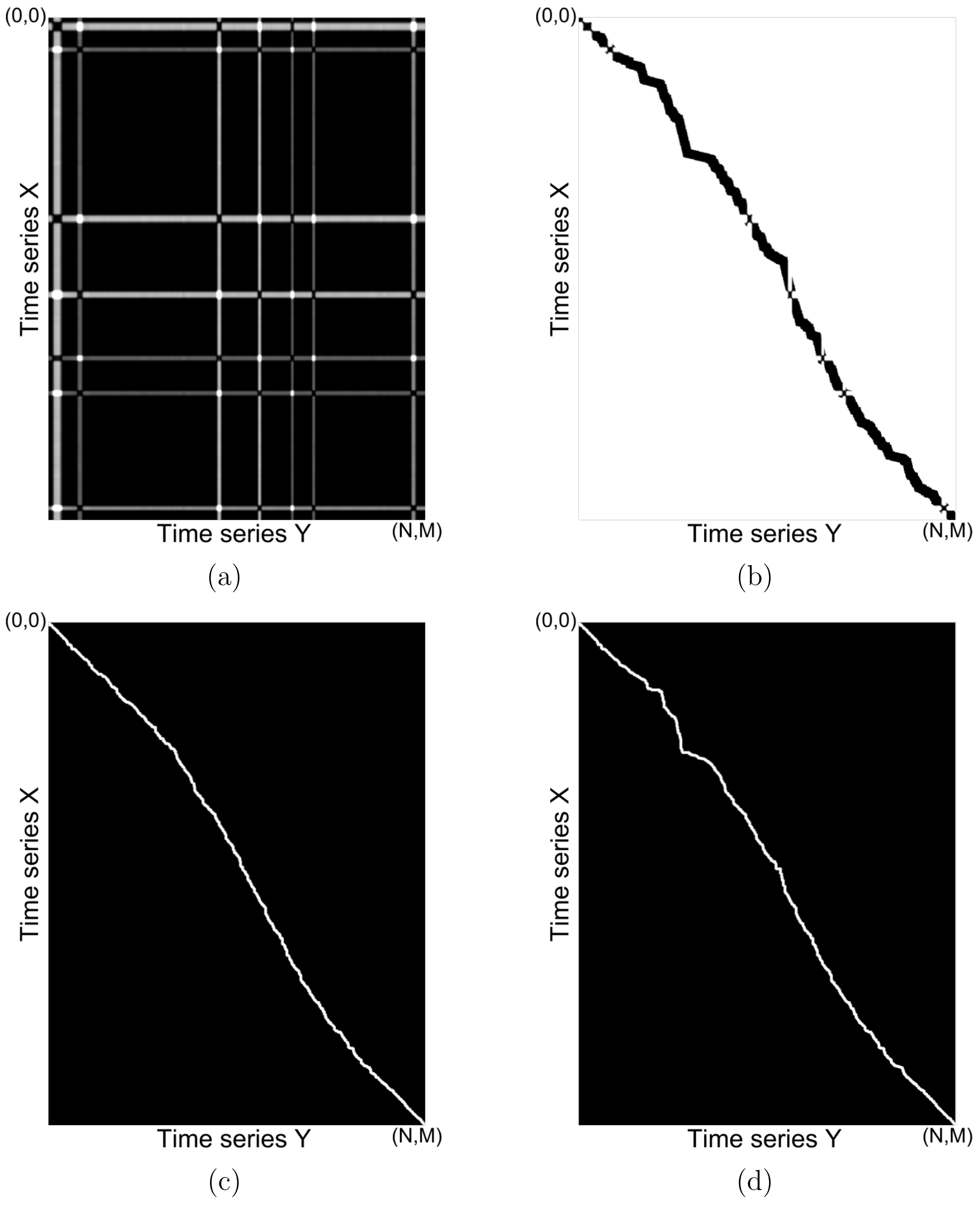


Figure 3.12: Example of the cost function computation for the sensor that measures the acceleration in the y axis. (a) Original DTW - The full matrix is computed. (b) Online DTW - The algorithm fails to detect the region that contains the optimum alignment between the signals, when compared to Fig. 3.2. (c) Original DTW - Warping path. (d) Online DTW - Warping path.

DTW algorithm was applied to the alignment of each peak of target signal to the equivalent peak of the reference signal.

After an analysis of the signals, it could also be observed that for some signals the peaks in the reference and target runs can have a significant difference of amplitude. Therefore, those signals were also disregarded and the tests were performed only in the signals from the sensors shown in Figs. 3.13 and 3.14. The results of the alignment of the peak regions are presented in Fig. 3.15. The results show that, on average, the alignment error is smaller than the one seen in Tab. 3.2, when using the sensor information, and even comparable to the error obtained when using only the video information.

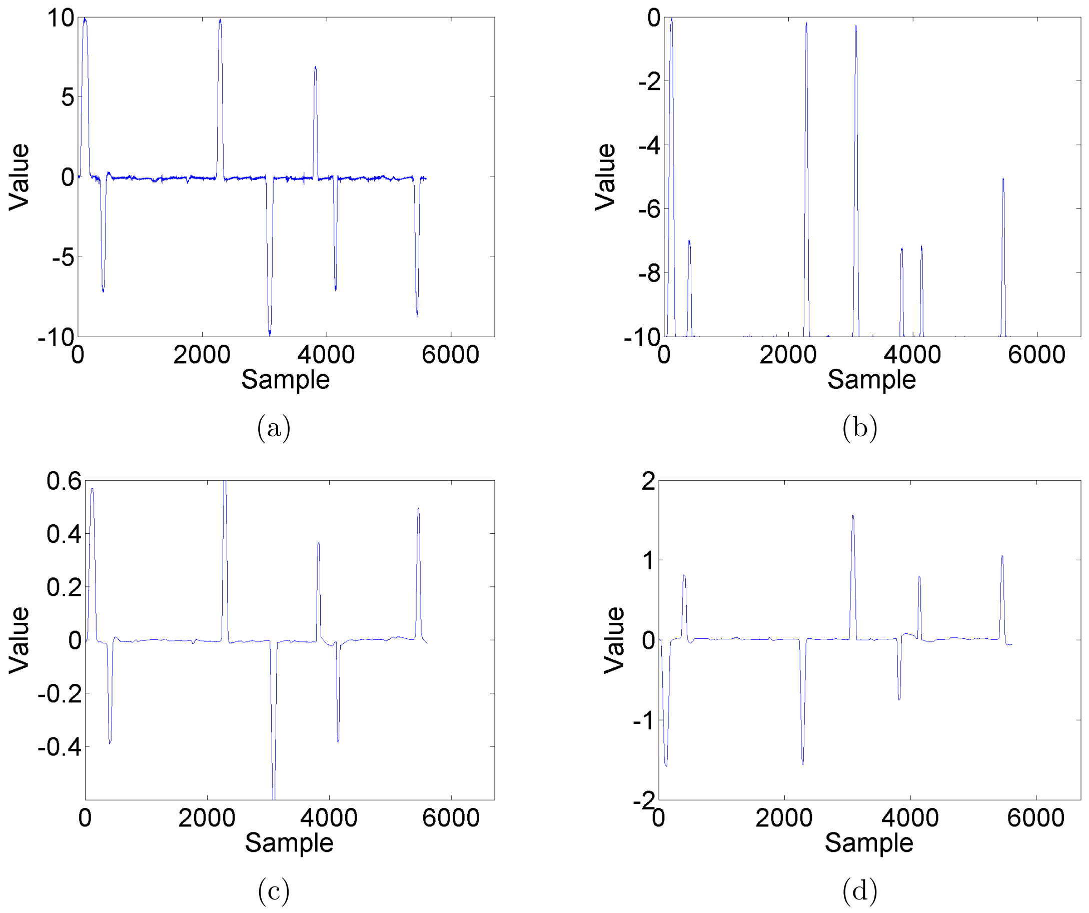
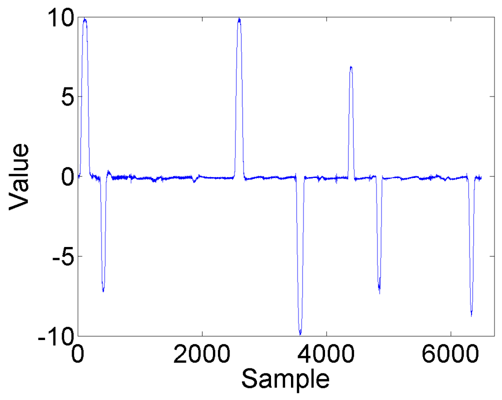


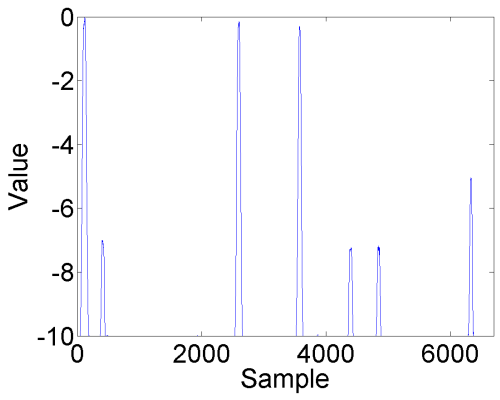
Figure 3.13: Set of sparse signals from a target recording used in the alignment. The visual information for each plot was reduced in order to treat each signal as a generic signal in the set. (a) Acceleration - y axis. (b) Acceleration - z axis. (c) Orientation quaternion x. (d) Roll.

Table 3.5: Alignment error of the peak regions corresponding to the run 1.

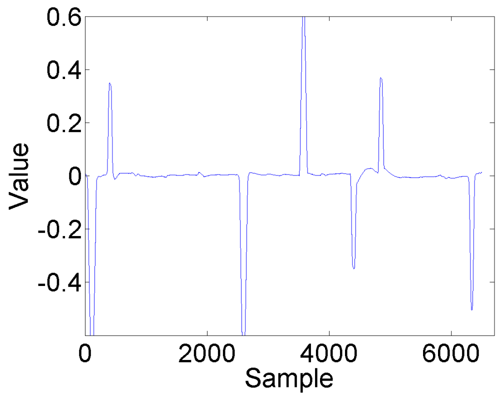
Sensor	**Average error (frames)**
Acceleration - y axis	1.27
Acceleration - z axis	0.62
Orientation quaternion x	0.55
Roll	0.49



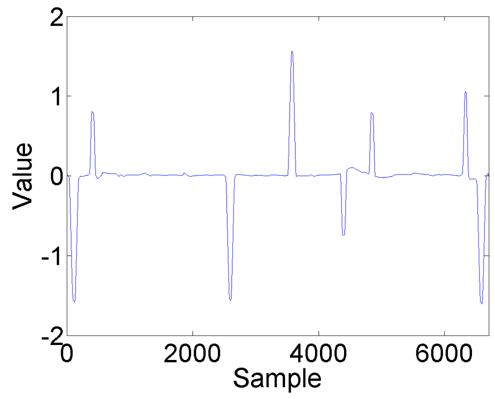
(a)



(b)



(c)



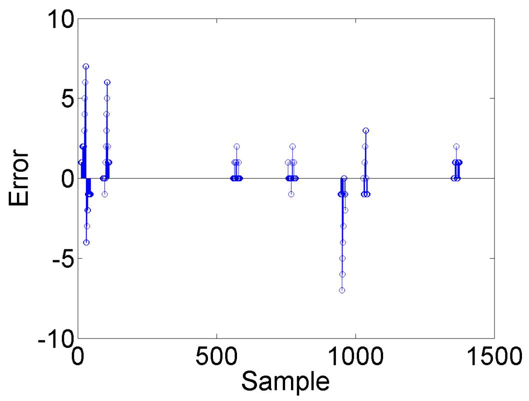
(d)

Figure 3.14: Set of sparse signals from a reference recording used in the alignment. The visual information for each plot was reduced in order to treat each signal as a generic signal in the set. (a) Acceleration - y axis. (b) Acceleration - z axis. (c) Orientation quaternion x. (d) Roll.

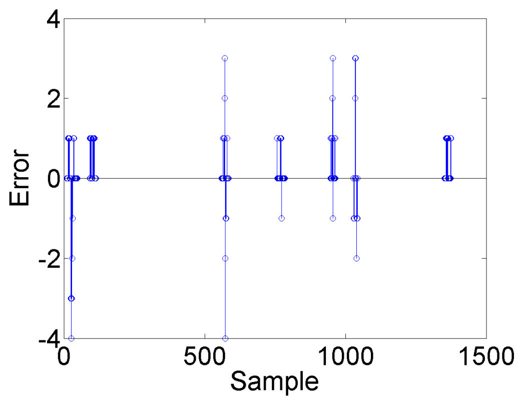
3.6 Summary

This chapter presented a video-based temporal alignment algorithm for surveillance systems based on a dynamic time-warping approach. The algorithm was tested with videos acquired in a real application using several metrics to be used as cost function for the DTW algorithm, which culminated in the choice of the mean square error between frames. Additional experiments showed conditions that can make the alignment algorithm fail, and the other available signals were investigated more deeply in the alignment problem.

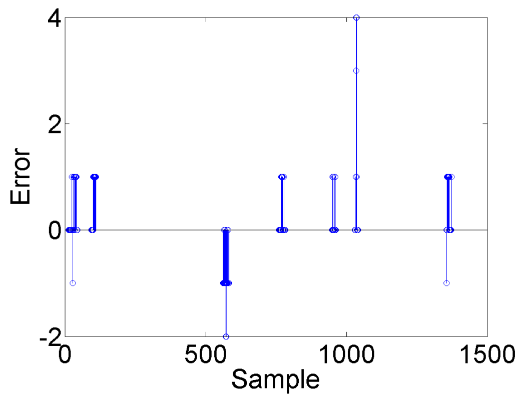
The next chapter studies a method that uses video information to reconstruct the camera trajectory, which can be used to extract various other properties from videos. For instance, the video alignment can be performed by searching the frames whose cameras are in a similar position and orientation.



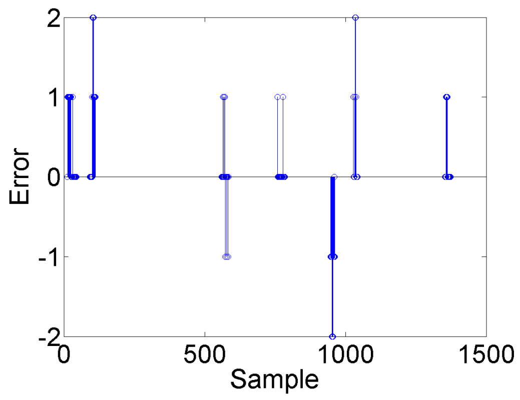
(a)



(b)



(c)



(d)

Figure 3.15: Alignment error (in frames) of the peak regions corresponding to the run 1. (a) Acceleration - y axis. (b) Acceleration - z axis. (c) Orientation quaternion x. (d) Roll.

Chapter 4

Camera Trajectory Estimation

In applications involving a moving sensor, it may be important to estimate the actual position of the sensor, which can be used for example to help a robot navigate through an environment, to synchronize signals, or to identify patterns. In computer vision, the problem of estimating the position of a moving sensor, as well as a mapping of the environment, is named the simultaneous localization and mapping (SLAM) problem.

This chapter investigates an approach that solves the SLAM problem using only camera information. The studied work, proposed by [5], employs notions of Lie groups with a graph-based optimization to estimate the trajectory and has shown significant advances for camera trajectory computation. The videos obtained in Section 1.2.1 are tested in this algorithm and the results are discussed.

The chapter is organized as follows. Section 4.1 details the SLAM algorithm of interest. In Section 4.2, some tests are performed using this algorithm for different databases. The results for the DORIS videos are analyzed and the limitations of the algorithm are discussed.

4.1 Robust Large Scale Monocular Video SLAM

The work developed in [5] presents an algorithm for the trajectory estimation of a monocular calibrated camera evolving in a large unknown environment. This work develops a SLAM algorithm that employs the concept of Lie groups to robustly align trajectories estimated in multiple submaps. To align a larger number of submaps, the work proposes a graph-based optimization algorithm, which also employs an efficient outlier-removal step.

This SLAM algorithm is composed of four main modules, which are depicted in Fig. 4.1. To reduce the computational complexity and also to ensure that pairs of frames have a minimum camera displacement between them, a keyframe selection step is employed. The keyframes are split in submaps and inside each submap the algorithm estimates the camera trajectory along the frames. In order to align all submaps, three-dimensional similarities between pairs of submaps, which transform the coordinates of one submap to another, are computed. The recovered submaps and the 3D similarities between submaps are used in the relative similarity averaging step, that computes the three-dimensional similarities that take each submap to a common global coordinate.

For a better understanding of the algorithm, the reader is invited to read Appendix A, which describes notions of projective geometry, and Appendix B, which shows theoretical concepts of Lie algebra.

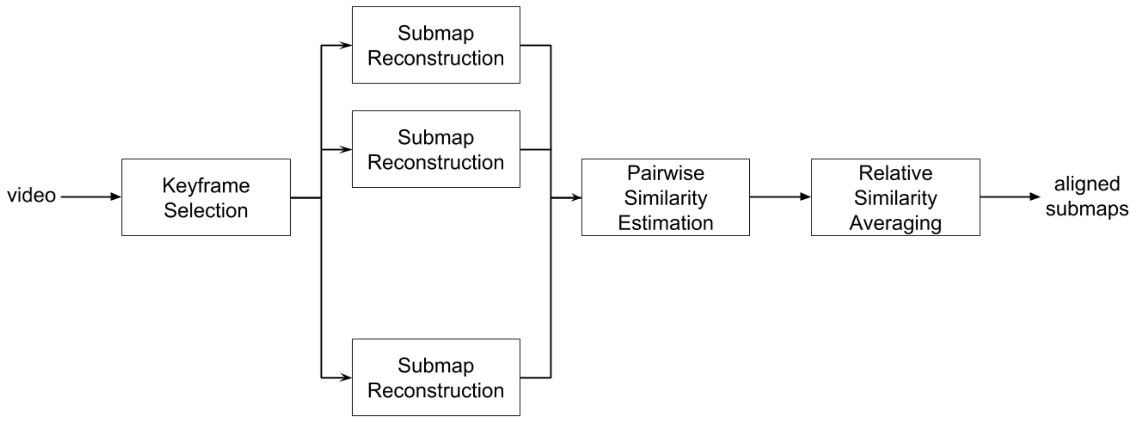


Figure 4.1: Block diagram of the SLAM algorithm proposed in [5].

4.1.1 Keyframe Selection

To perform a keyframe selection, it is necessary to use a fast method that will be applied in the whole set of frames. Thereby, the algorithm applies a Lucas-Kanade tracker [95], which detects and tracks Harris points of interest (PoI) [96] in the video frames. A frame is selected as a keyframe when the Euclidean distance between the corresponding PoI of the current frame and the previous keyframe is bigger than a given threshold (which is typically 5% of the image width).

Fig. 4.2 exemplifies the keyframe selection step. The method starts with the first frame being considered a keyframe. The ensuing frames are tested and only the one whose content displays a substantial difference with respect to the previous keyframe, which is represented in the figure as the one where the black circle moves a minimum amount of pixels, is defined as another keyframe.

4.1.2 Submap Reconstruction

The set of keyframes selected in the previous step is split in clusters of L consecutive frames with overlap factor of 50% and, for each keyframe, SURF keypoints [97] are

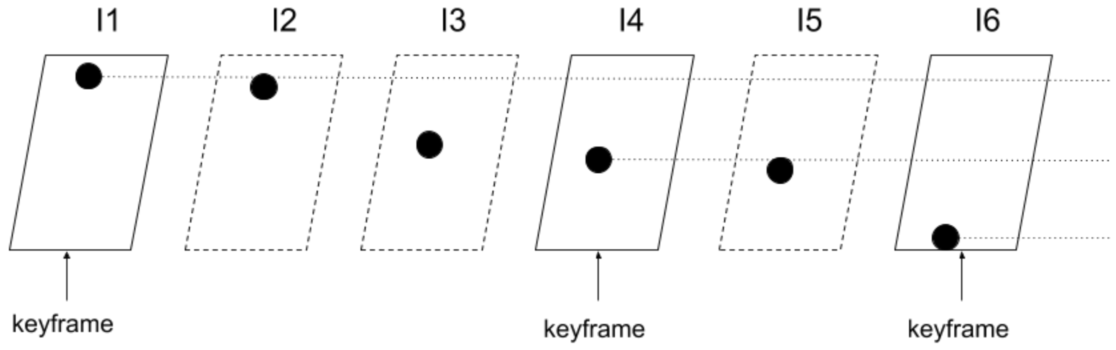


Figure 4.2: Example of the keyframe selection step. The sequence of frames is represented as the dashed parallelograms and the ones considered as keyframes are displayed with solid lines. Significant difference from the previous keyframe is used to classify the next keyframe.

computed. For each cluster (or submap), SURF descriptors are matched and used in the estimation of corresponding points between pairs of keyframes. In order to increase the number of connections among frames, reducing the occurrence of incremental errors, this step is performed for all pairs of consecutive frames and also for some pairs of non-consecutive frames. The epipolar geometry of each of these pairs of frames is computed through the estimation of the essential matrix [13] using the five point algorithm [98], combined with a RANSAC algorithm [13] and a bundle adjustment optimization [45].

Using the essential matrix computed for a pair of frames, one can estimate the rotation between the camera coordinate systems of each frame of the pair [13], that is, the relative rotation between the orientation of the camera for each frame. As a result of this calculation, several relative rotations between frames are estimated. These relative rotations estimated for all pairs of frames are then employed in the computation of a global orientation for each frame, in relation to a reference common to all frames. For this computation, the relative similarity averaging algorithm described in the following sections can also be employed (which is defined for a general transformation).

After estimating a global orientation for each frame in the submap, the position of the camera for each frame still needs to be determined. In order to estimate the camera pose for each frame, keypoints are tracked among the frames and a linear programming is employed in the computation of the *Known rotation problem* [99], which is described below.

Known rotation problem: For a camera matrix $\mathbf{P} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} = \begin{bmatrix}\mathbf{R_1} & t_1 \\ \mathbf{R_2} & t_2 \\ \mathbf{R_3} & t_3\end{bmatrix}$,

$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ is an image point with corresponding three-dimensional point $\mathbf{X}$. The reprojection error is given by:

$$E(\mathbf{X}, \mathbf{R}, \mathbf{t}) = \left\| \left(x - \frac{\mathbf{R_1}\mathbf{X} + t_1}{\mathbf{R_3}\mathbf{X} + t_3}, y - \frac{\mathbf{R_2}\mathbf{X} + t_2}{\mathbf{R_3}\mathbf{X} + t_3} \right) \right\|^2 . \quad (4.1)$$

For the reprojection error to be less than a given threshold γ, this condition can be written as:

$$\|((x\mathbf{R_3} - \mathbf{R_1})\mathbf{X} + xt_3 - t_1, (y\mathbf{R_3} - \mathbf{R_2})\mathbf{X} + yt_3 - t_2)\|^2 \leq \gamma(\mathbf{R_3}\mathbf{X} + t_3)^2. \quad (4.2)$$

If $\mathbf{R}$ is known, this condition is a convex constraint, and linear programming can be used to solve simultaneously for $\mathbf{t}$ and $\mathbf{X}$.

In Fig. 4.3, one can see an example of the submap reconstruction step. Each submap in this case is a set of consecutive keyframes which may contain an overlap with another submap. The camera trajectory for each submap is reconstructed by solving Eqs. (4.1) and (4.2). The dashed lines highlight the reconstructed trajectory for the frames that belong to the overlap of two submaps, therefore should represent the same trajectory. However, each reconstruction uses its own referential, so these trajectories must be rotated, scaled and translated with respect to each other. The next steps cope with the alignment of different referentials.

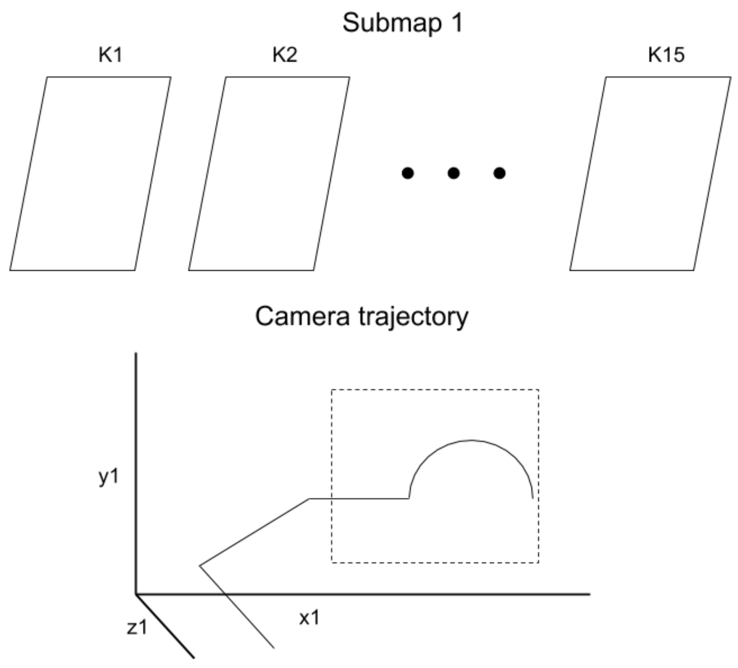
4.1.3 Pairwise Similarity Estimation

After the previous step, for each submap a camera trajectory and a cloud with triangulated points were estimated. However, the reconstruction for each submap was made according to a different coordinate system. In order to align all submaps, a three-dimensional similarity between pairs of submaps must be calculated, which can be seen as matrices that belong to the Lie group $Sim(3)$:

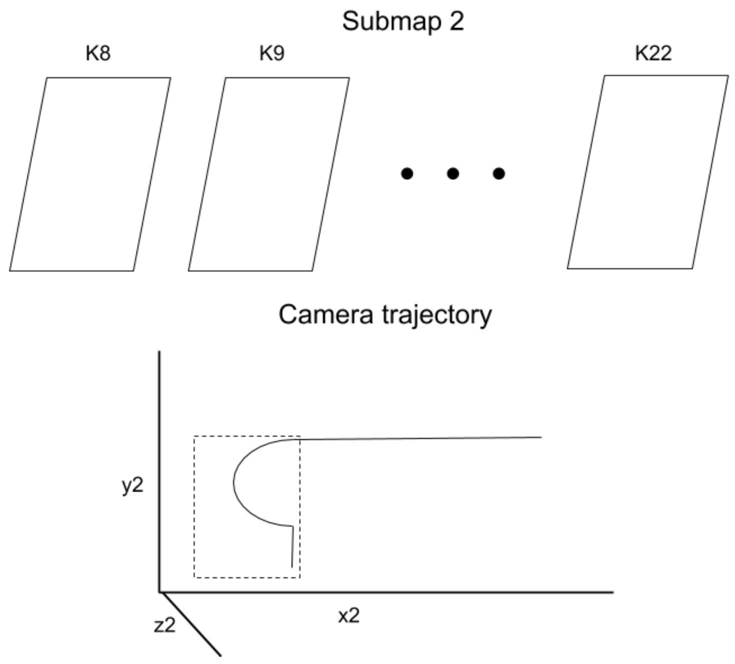
$$Sim(3) = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}, \quad (4.3)$$

with $R \in SO(3)$, $\mathbf{t} = [t_1, t_2, t_3]^T$ and $s \in \mathbb{R}^+$.

To reduce the number of similarities to compute, a bag-of-words [100] approach is applied to three-dimensional SURF descriptors of all submaps to find a unique descriptor for the whole submap. A similarity is determined for consecutive submaps and also between each submap and its 10 nearest neighbors using the bag-of-words descriptor as a metric of distance.



(a)



(b)

Figure 4.3: Example of the submap reconstruction step. In this example, each submap is composed of a sequence of 15 consecutive frames with eight frames of overlap with the previous and next submaps. For each submap, a reconstruction of the camera trajectory is computed using Eqs. 4.1 and 4.2. The dashed lines highlight the reconstructed trajectory for the frames in the overlap of two consecutive submaps. (a) Submap 1. (b) Submap 2.

In order to estimate a similarity between two submaps, SURF descriptors for each three-dimensional point are obtained by averaging the SURF descriptors of the image points that generated this triangulated point, and the descriptors are matched between submaps. A three-point algorithm [101] combined with the RANSAC is applied to obtain a three-dimensional similarity, which is refined by minimizing the sum of the position errors weighted by the covariance of each triangulated point.

Finally, considering that this similarity can be modeled as a concentrated Gaussian distribution on the group $Sim(3)$, a covariance for each similarity is also found. In the end of this step, the algorithm has computed similarities $Z_{ij} \in Sim(3)$ between the coordinate system of the submap i and the submap j along with a covariance Σ_{ij} for these estimates.

Fig. 4.4 exemplifies the pairwise similarity estimation. The frames inside each submap are used for the triangulation of three-dimensional points. By tracking triangulated points across different submaps, it is possible to compute a pairwise similarity transformation between two submaps, which is composed of rotation, translation and scaling, that aligns the axis for both reconstructions to a same common axis. The next step takes all relative similarities and maps all axes to a global reference.

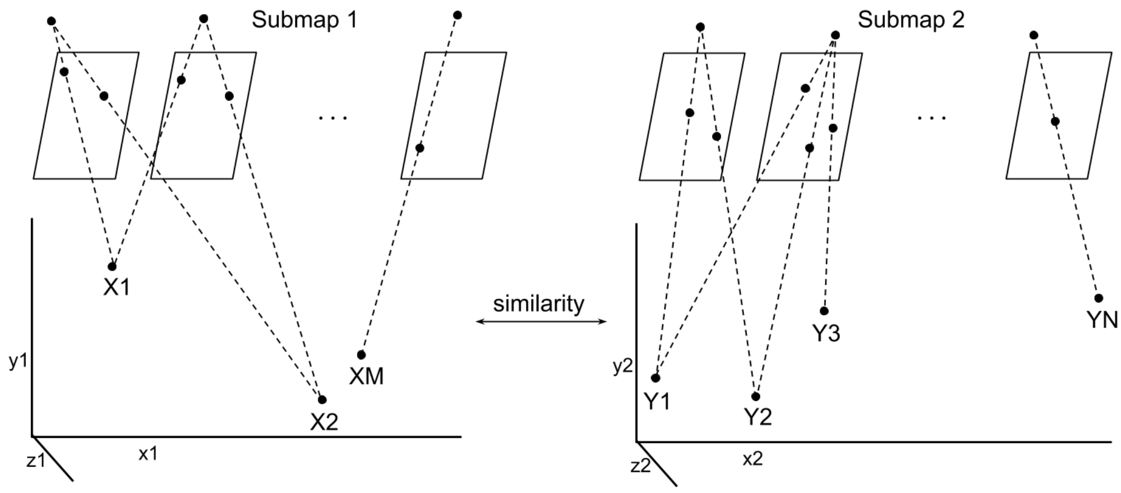


Figure 4.4: Pairwise similarity estimation step. For each submap, keypoints inside the frames are tracked and triangulated to three-dimensional points (points X1,...,XM for submap 1 and Y1,...,YM for submap 2). Using corresponding three-dimensional points from two submaps, a similarity transformation is computed that transforms a point represented according to the axis x1,y1,z1 to a point represented according to the axis x2,y2,z2

4.1.4 Relative Similarity Averaging

From the results obtained in the previous subsection, relative similarities Z_{ij} that align the submaps i and j were computed, along with a covariance matrix Σ_{ij}. In this step, we need to estimate the global similarities (X_{iS}, X_{jS}), that is, the three-dimensional similarities between a global reference frame S and each submap. Given the submaps i and j, one can consider that the similarity Z_{ij} that takes from the submap i to the submap j should be equivalent to going from the submap i to the reference frame S (using the global similarity X_{iS}), and then going from the reference frame S to the submap j (using X_{jS}^{-1}). Considering the existence of noise in the measurements, represented by the covariance matrix Σ_{ij}, the following model is obtained:

$$Z_{ij} = \exp^{\wedge}(b_{ij}^{i}) X_{iS} X_{jS}^{-1}, \tag{4.4}$$

where $b_{ij}^{i} \sim \mathcal{N}_{\mathbb{R}^p}(\mathbf{0}_{p\times 1}, \Sigma_{ij})$ is a white Gaussian noise.

Considering that the measurements Z_{ij} are outlier-free, an estimate of the global similarities X_{iS} and X_{jS} can be obtained by the relative similarity averaging problem, which minimizes the following cost function:

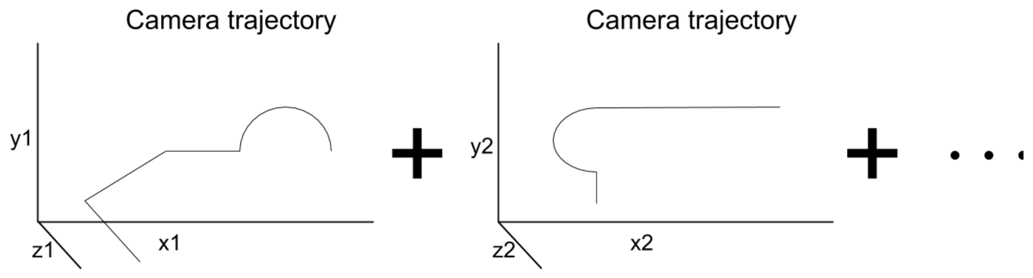
$$\underset{\{X_{iS}\}_{i\in\mathcal{V}}}{\operatorname{argmin}} \sum_{i,j\in\mathcal{E}} \left\| \log^{\vee} Z_{ij} X_{jS} X_{iS}^{-1} \right\|_{\Sigma_{ij}}^{2}, \tag{4.5}$$

with $\|\|_{\Sigma}^{2}$ representing the Mahalanobis distance. This equation is similar to a generalized least squares problem, where one estimates the distance between a model ($X_{jS}X_{iS}^{-1}$) and the estimate (Z_{ij}), pondering by the covariance of the error (Σ_{ij}). One can also note the function $\log^{\vee}$, which maps the similarities to the Lie algebra, where the optimization is performed.

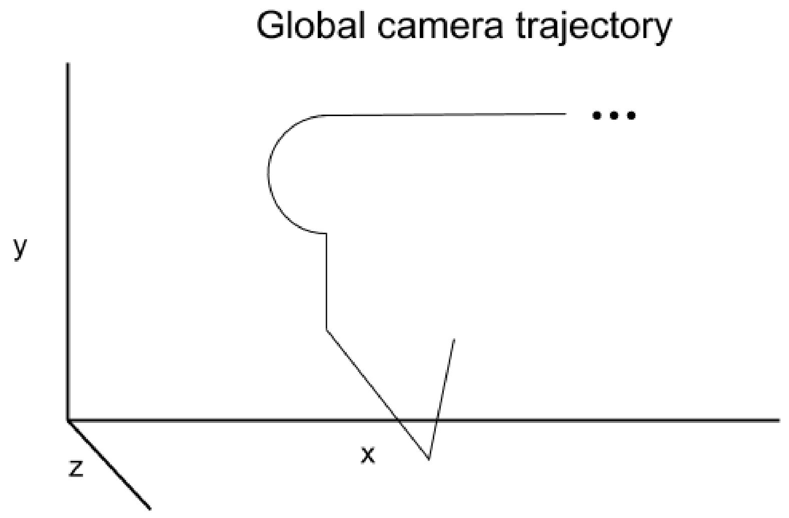
If the procedure of selecting pairs of submaps has chosen submaps that do not have common regions in the scene, a relative similarity computed can represent an outlier. In this case, the minimization problem represented by Eq. (4.5) can fail to find the correct global similarities. Thus, an outlier removal algorithm is necessary to solve the relative similarity averaging problem.

The problem given by Eq. (4.5) can also be seen as the inference problem in a factor graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. In this context, each vertex $\mathcal{V}_i$ corresponds to a global similarity measurement X_{iS} and each pairwise factor $\mathcal{E}_{ij}$ corresponds to a relative measurement Z_{ij} that links the vertices $\mathcal{V}_i$ and $\mathcal{V}_j$. The following subsections describe an outlier removal algorithm and a relative similarity averaging algorithm that uses notions of graph optimization.

In Fig. 4.5, one can see an example of the relative similarity averaging step. Using the relative similarities computed in the previous step, for each submap is computed a similarity transformation that maps its axis to a global referential. The trajectories found for each submap, that can now described with respect to the same referential, are merged to define the camera trajectory for the whole input video.



(a)



(b)

Figure 4.5: Example of the relative similarity averaging step. The camera trajectory is estimated for each submap according to its own referential and contains only a part of the total trajectory. After computing relative similarities between pairs of submaps, all referential are mapped to a global one and the parts of the trajectory are merged to compose the total trajectory of the camera along the whole video. (a) Camera trajectories for each submap that are combined to form a single one. (b) Global camera trajectory for the whole video that is a composition of the trajectories computed for each submap.

4.1.5 Outlier Removal Algorithm

To remove outlier measurements in the SLAM algorithm, it is assumed that every relative similarities between consecutive submaps are inliers. For the other relative similarities, the error inside a cycle, which should be small for a measurement to be considered as an inlier, is tested:

$$\epsilon^T P^{-1} \epsilon < t_{\chi^2}, \tag{4.6}$$

where ϵ is the cycle error, P is the covariance associated with this cycle and t_{χ^2} is a value based on the χ^2.

A naive algorithm to test a relative similarity Z_{kl} could be to test the cycle $Z_{kl}Z_{l(l-1)}Z_{(l-1)(l-2)}...Z_{(k-1)k}$, which contains the similarity between the k-th and l-th submaps (Z_{kl}), and all consecutive similarities from the l-th to k-th submaps ($Z_{l(l-1)}...Z_{(k-1)k}$). However, this approach can fail for larger cycles, since it accumulates any small errors in each similarity. Instead of using consecutive measurements in the cycle, an algorithm proposed in [5] searches for the shortest cycles (in the sense of minimum number of connections) that contain only inliers. This algorithm is described in Alg. 8.

Algorithm 8 Algorithm to remove outlier similarity measurements.

```
    Input: Relative similarities $Z_{ij}$, covariance matrices $\Sigma_{ij}$, value of $t_{\chi^2}$.
    Output: Graph containing only inlier relative similarities.
 1: Initialize an empty graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$
 2: Add the vertex $X_{1S}$ to $\mathcal{V}$
 3: for $k \in \{1, ..., N\}$ do
 4:     Add the vertex $X_{kS}$ to $\mathcal{V}$
 5:     Add the factor $\left\{Z_{(k-1)k}, \Sigma_{(k-1)k}\right\}$ to $\mathcal{E}$
 6:     for $l \in \{1, ..., k\}$ do
 7:         Find the shortest path from $X_{kS}$ to $X_{lS}$ in $\mathcal{G}$
 8:         Compute the cycle error $\epsilon$ and covariance $P$
 9:         if $\epsilon^T P^{-1} \epsilon < t_{\chi^2}$ then
10:             Add the factor $\{Z_{lk}, \Sigma_{lk}\}$ to $\mathcal{E}$
11:         end if
12:     end for
13: end for
```

4.1.6 Large-Scale Relative Similarity Averaging

An efficient algorithm to estimate a large number of similarities was also proposed in [5]. This method splits the original graph into N_S subgraphs of maximum size n and creates a supergraph that links all subgraphs. For each iteration, it alternates between solving the problem locally for each subgraph and computing messages sent from other subgraphs by building and solving a supergraph. A block diagram of this method is shown in Fig. 4.6.

Graph partitioning

In this step, the graph is split into N_S subgraphs with a maximum size n. This partition considers the temporal order of the vertices, so that the subgraphs are always composed of n consecutive subgraphs $X_{(k+1)S \cdots (k+n)S}$. The removed measurements are given the names of inter-measurements and in their locations are placed factors

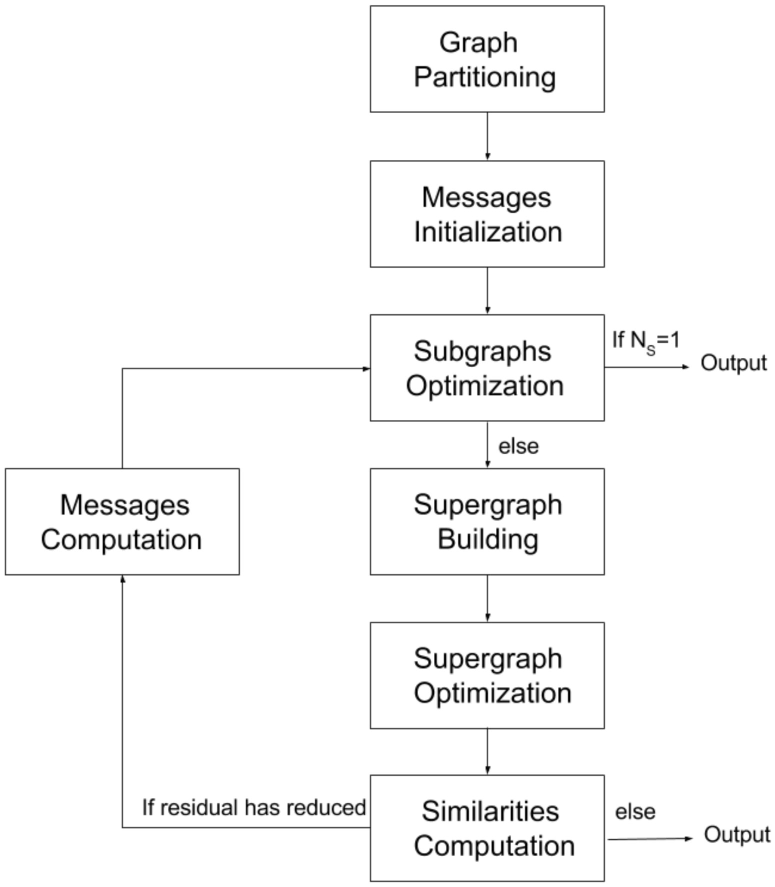


Figure 4.6: Block diagram of the large scale relative similarity averaging algorithm.

Z_{iR_k}, which are considered a message that connects this subgraph with the other subgraphs. Note that since each subgraph is processed independently, each one has its own reference R_k. The messages Z_{iR_k} will be responsible for linking all references to a common global reference. Fig. 4.7 shows an example of an original graph that is partitioned into 3 subgraphs of maximum size 3.

Message initialization

Each message Z_{iR_k} is initialized with the identity matrix, which represents the identity element of the group of similarities $Sim(3)$, and its covariance matrix $\Sigma^i_{iR_k}$ is initialized with infinite covariance, to indicate that there is no certainty in the current value. Through the course of the algorithm, these values are iteratively updated.

Subgraphs optimization

For each subgraph $\mathcal{G}^k = \{\mathcal{V}^k, \mathcal{E}^k\}$ an estimative of the global similarities $\{X_{iR_k}\}_{i\in\mathcal{V}^k}$ that relate each submap i to the subgraph reference R_k is found by using a Gauss-Newton algorithm [102]:

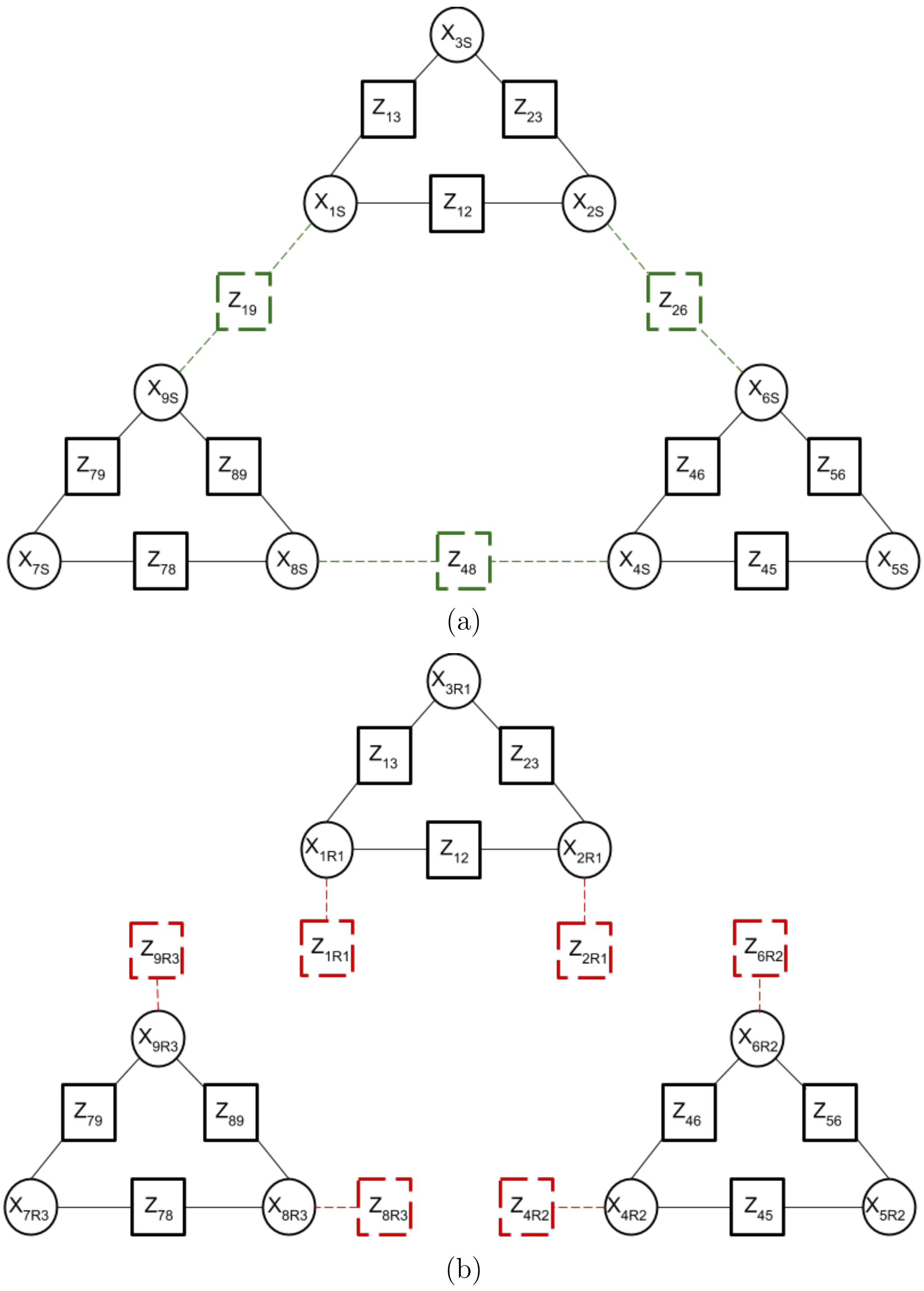


Figure 4.7: Example of the graph partitioning. (a) Original graph. The inter-measurements are marked with green dashed lines. (b) Subgraphs of maximum size 3. The inter-measurements are replaced by the messages marked with red dashed lines.

$$\underset{\{X_{iR_k}\}_{i\in\mathcal{V}^k}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{E}^k} \left\|log^{\vee}(Z_{ij}X_{jR_k}X_{iR_k}^{-1})\right\|^2_{\Sigma^i_{ij}} + \sum_{(i,j)\in\mathcal{V}^k} \left\|log^{\vee}(Z_{iR_k}X_{iR_k}^{-1})\right\|^2_{\Sigma^i_{iR_k}} . \tag{4.7}$$

In this optimization, the first term was derived from Eq. (4.5). It enforces that the similarity that takes the submap i to the reference R_k, combined with the similarity that takes the reference R_k to the submap j, should be similar to the similarity that relates directly the submaps i and j. The second term uses the current estimate of the messages Z_{iR_k}. Since this message is an estimate of the transformation between the submap i and the reference R_k, which is exactly the transformation the algorithm wants to estimate for X_{iR_k}, this term enforces that Z_{iR_k} and X_{iR_k} should be similar. The covariances $\{P^i_{iR_k}\}_{i\in\mathcal{V}^k}$ associated with the global similarities $\{X_{iR_k}\}_{i\in\mathcal{V}^k}$ are estimated by a Laplace approximation.

Supergraph building

After each subgraph has been solved, the algorithm builds a supergraph $\mathcal{G}^{\text{SUPER}} = \{\mathcal{V}^{\text{SUPER}}, \mathcal{E}^{\text{SUPER}}\}$ using the inter-measurements and the global similarities $\{X_{iR_k}\}_{i\in\mathcal{V}^k}$. This supergraph is used to estimate transformations that take each reference frame of each subgraph to a global reference frame, which are called super-measurements. Each inter-measurements Z_{ij} with covariance Σ^i_{ij} defines a super-measurement by the following equations:

$$Z_{R_kR_l} = X^{-1}_{iR_k} Z_{ij} X_{jR_l}, \tag{4.8}$$

where it is considered that the transformation from R_k to R_l is equivalent to the combination of the transformations from R_k to i ($X^{-1}_{iR_k}$), from i to j (Z_{ij}), and then from j to R_l (X_{jR_l}), and

$$\Sigma^{R_k}_{R_kR_l} = Ad_G(X^{-1}_{iR_k})(P^i_{iR_K} + \Sigma^i_{ij} + Ad_G(Z_{ij})P^j_{jR_l}Ad_G(Z_{ij})^T)Ad_G(X^{-1}_{iR_k})^T \tag{4.9}$$

is the associated covariance.

If multiple inter-measurements Z_{ij} connect the same nodes in each pair of subgraphs, an average of the results is used as the super-measurement. Fig. 4.8 shows an example of the supergraph built using the subgraphs of Fig. 4.7.

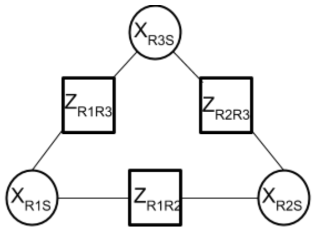


Figure 4.8: Example of a supergraph that connects the subgraphs in Fig. 4.7.

Supergraph optimization

To solve the supergraph, this algorithm is applied recursively, considering that the supergraph is a new graph input. The algorithm should be recursively called using smaller supergraphs as inputs until it can find only one subgraph, when it returns the results to a previous iteration.

Similarities computation

The quantities of interest($\{X_{iS}\}_{i\in\mathcal{V}}$) with covariance $\{P^i_{iS}\}_{i\in\mathcal{V}}$ are the global similarities between the nodes of the original graph and a global reference S. At this point, it was estimated how to represent the submap i with respect to the reference R_k (using X_{iR_k}), and how to map the reference R_k to the global reference S (using X_{R_kS}). Therefore, the global similarities $\{X_{iS}\}_{i\in\mathcal{V}}$ can be obtained by:

$$X_{iS} = X_{iR_k}X_{R_kS}, \tag{4.10}$$

with covariance

$$P^i_{iS} = P^i_{iR_k} + Ad_G(X_{iR_k})P^{R_k}_{R_kS}Ad_G(X_{iR_k})^T. \tag{4.11}$$

Once all quantities are computed, one can estimate the cost function given by Eq. (4.5). This cost function is used as a stop condition to the algorithm: If the cost is higher than the one found in a previous iteration, the algorithm exits, otherwise, the messages Z_{iR_k}, which in the first steps were initialized with $Z_{iR_k} = Id$ and $\Sigma^i_{iR_k} = \infty$, are updated and the algorithm returns to the subgraph optimization step.

Messages computation

If the algorithm decides that the error is still decreasing, the messages Z_{iR_k} are updated and the algorithm continues the loop. For each inter-measurement Z_{ij} the message Z_{iR_k} is updated using the previously estimated similarities, using the path from i to j, from j to R_l, from R_l to S, and then from S to R_k:

$$Z_{iR_k} = Z_{ij}X_{jR_l}X_{R_lS}X^{-1}_{R_kS}, \tag{4.12}$$

associated to the covariance

$$\begin{aligned}\Sigma^i_{iR_k} = Ad_G(Z_{ij})\Big[P^j_{jR_l} + Ad_G(X_{jR_l})\Big\{P^{R_l}_{R_lS} \\ +Ad_G(X_{R_lS}X^{-1})P^{R_k}_{R_kS}Ad_G(X_{R_lS}X^{-1}_{R_kS})^T\Big\}. \\ Ad_G(X_{jR_l})^T\Big]Ad_G(Z_{ij})^T + \Sigma^i_{ij}\end{aligned} \tag{4.13}$$

4.2 Experimental Results

To test the SLAM algorithm, several databases were used in order to reproduce the results seen in [5]. An initial experiment was performed in a demo video that employed a camera moving in a circular trajectory. Some frames from this video can be seen in Fig. 4.9. After applying the SLAM algorithm, it is expected that the method can estimate a circular-shaped trajectory for the camera, but without recovering correct scale of the scene. In Fig. 4.10, one can see the results of the SLAM algorithm for this video, which shows that the algorithm can correctly estimate a circular trajectory for the camera.

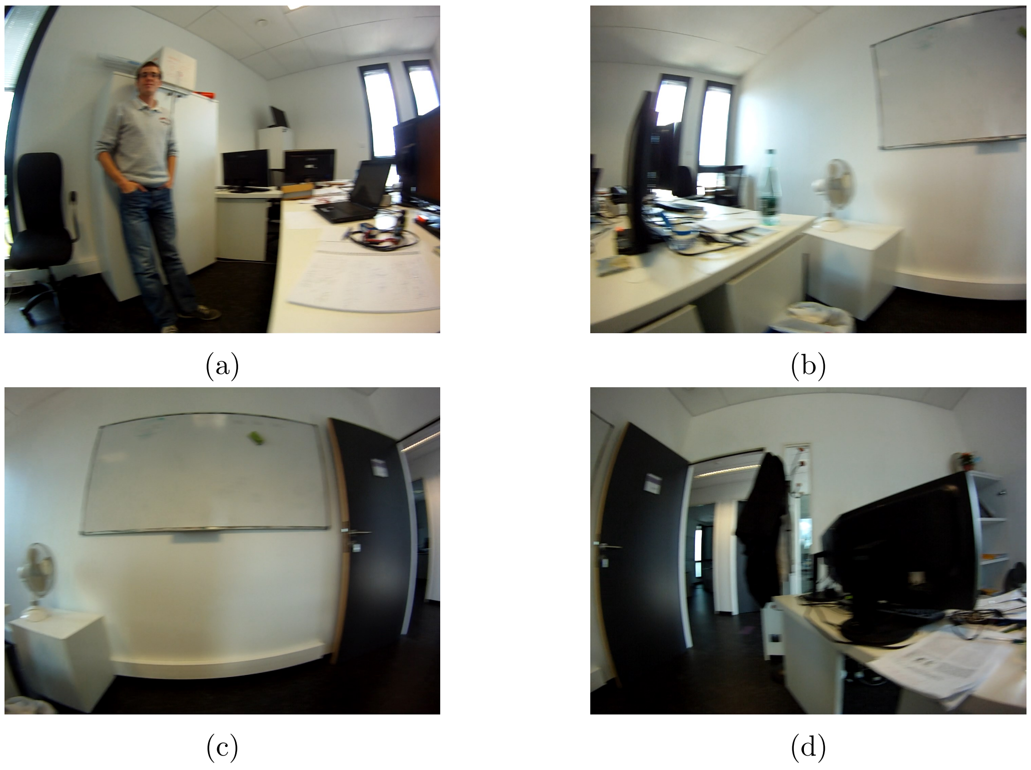


Figure 4.9: Example of a video with a circular camera trajectory. The figures from (a) to (d) represent the progression of the video.

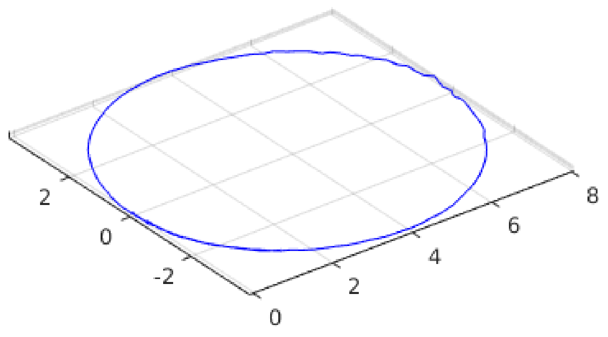


Figure 4.10: Circular trajectory estimated by the SLAM algorithm.

Another experiment was performed using videos from the KITTI database [103], which is composed of videos acquired using an autonomous driving car moving along a road. This database also employs a laser scanner and a GPS to provide an accurate ground truth for the camera position. Some examples of frames from this database are shown in Fig. 4.11. The SLAM algorithm was tested in some videos from this database, and the results were compared to the camera positions provided by the ground truth. Figs. 4.12 and 4.13 show the camera trajectory estimated for two different videos by the SLAM algorithm and the ground truth. For these videos, even though the camera moves along a trajectory more complex than the last experiment, with several curves for both sides, the algorithm can correctly estimate a trajectory for the camera that has the same shape as the ground truth, but with different scale, rotation and position (therefore there is a similarity transformation that aligns the two trajectories and makes them similar).

4.2.1 Tests with DORIS Videos

Another experiment was performed using the SLAM algorithm in the DORIS videos described in Section 1.2.1, containing a complete round on the rail. The parameters were the same used in the previous test. Several problems intrinsic to this application made the SLAM algorithm unable to execute all steps and estimate a trajectory.

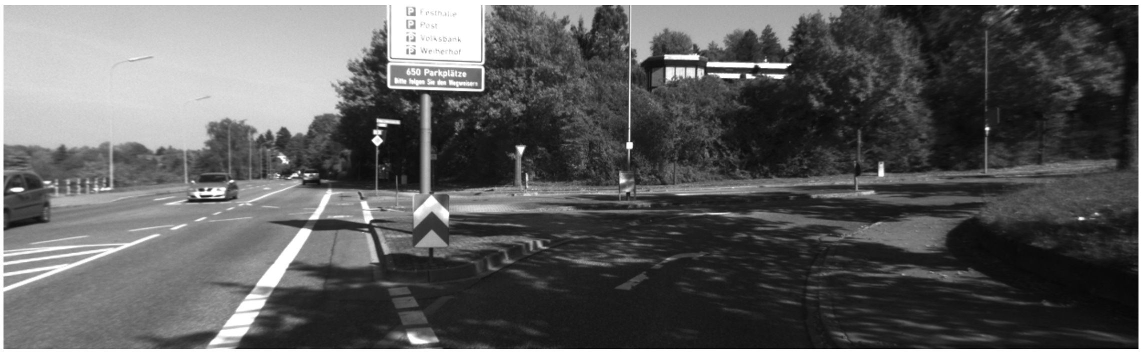
After analyzing the videos and the partial results, it was discovered that for certain regions of the videos, when the camera passes near a pillar, the algorithm consistently loses track of the trajectory. The causes for this issue are twofold.

As can be seen from the example shown in Fig. 4.14, these frames have a flat surface that occupies most of the frame. For the regions near the pillar, the SURF algorithm is not able to detect a sufficient number of keypoints, and the descriptor for each keypoint is not distinctive since the image does not have a diversified content. In addition, for these regions, other characteristics such as the lighting condition, the compression noise, or even a small disturbance of the camera become prominent and may lead to an erroneous displacement estimation [104, 105].

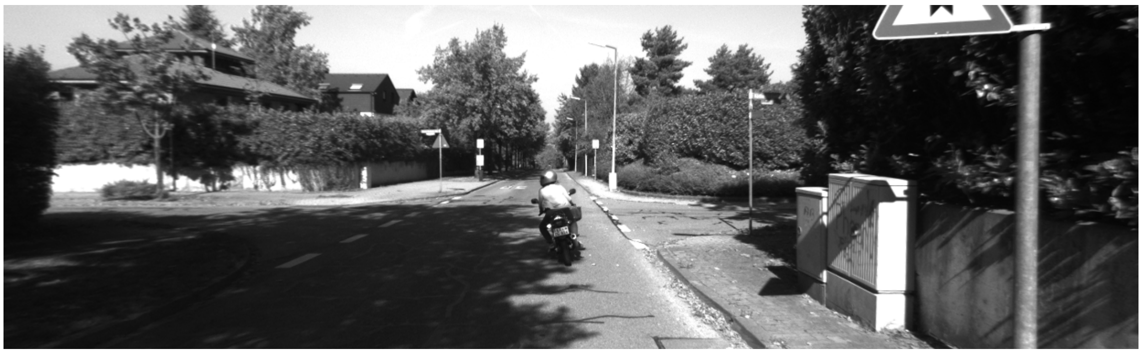
Hence, even if the algorithm finds a sufficient number of keypoints, the detected keypoints are not representative to describe the scene content. Consequently the procedure of finding corresponding points, computing the epipolar geometry and estimating the camera displacements (see Section 4.1.2) becomes unreliable. Also, some assumptions, such as the one that states that consecutive relative measurements are always inliers (see Section 4.1.5), can not be satisfied.

When trying to estimate the trajectory for the complete round, it is also not possible to ignore the regions with pillars and bypass the computation of the camera trajectory for these frames, for example, interpolating the displacement obtained

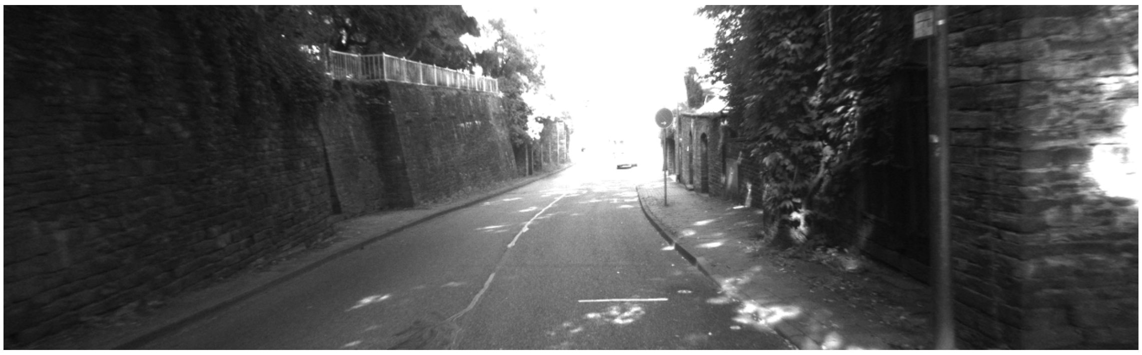
(a)



(b)

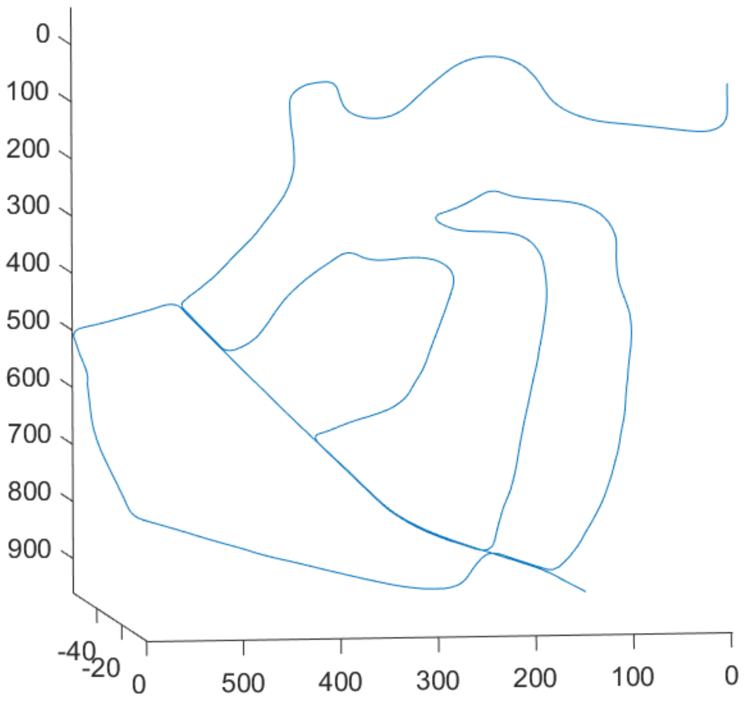


(c)

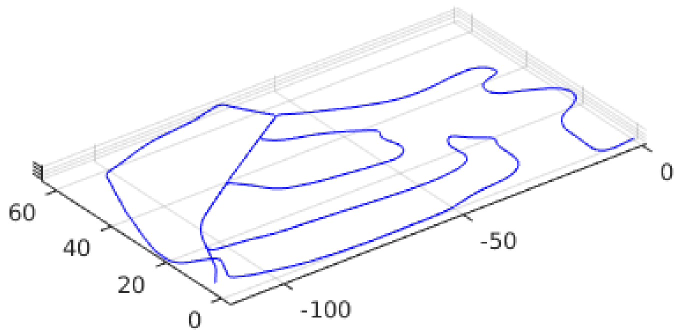


(d)

Figure 4.11: Frames of the video 2 in the KITTI odometry dataset. (a) Frame 30. (b) Frame 50. (c) Frame 1000. (d) Frame 4660.

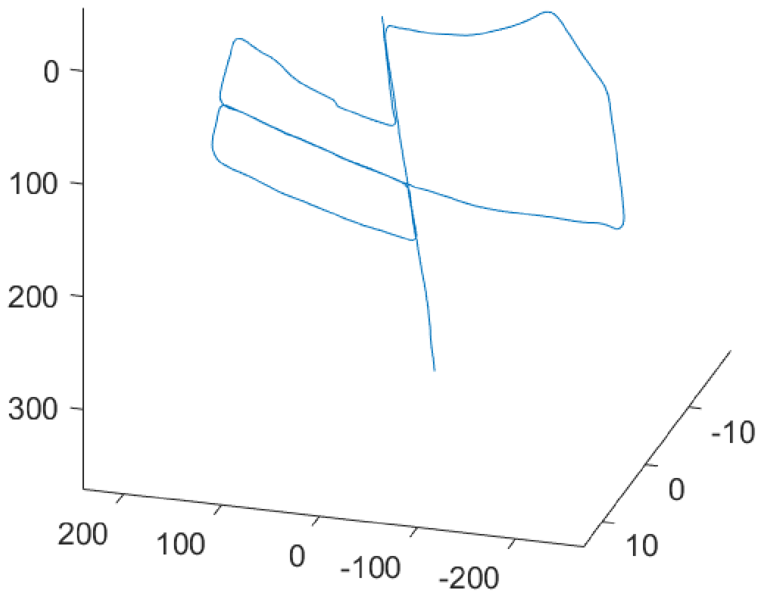


(a)

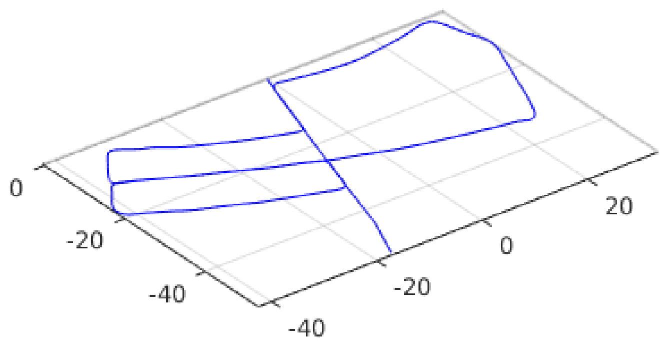


(b)

Figure 4.12: Camera trajectory for the video 2 in the KITTI odometry dataset. (a) Ground truth. (b) Result of the SLAM algorithm.

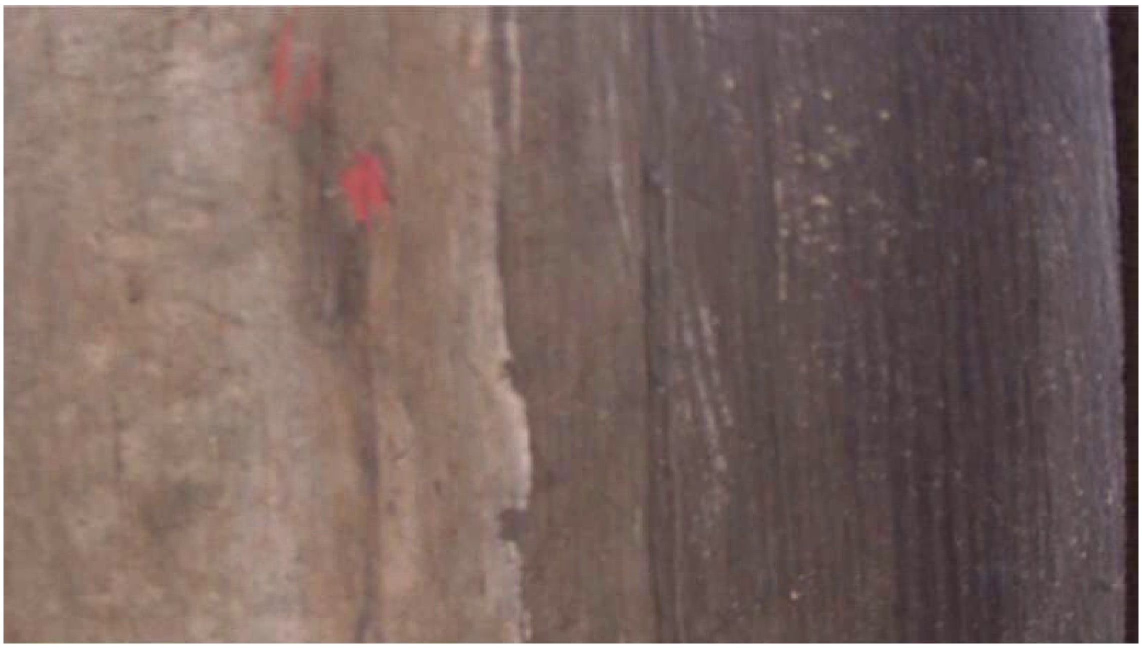


(a)

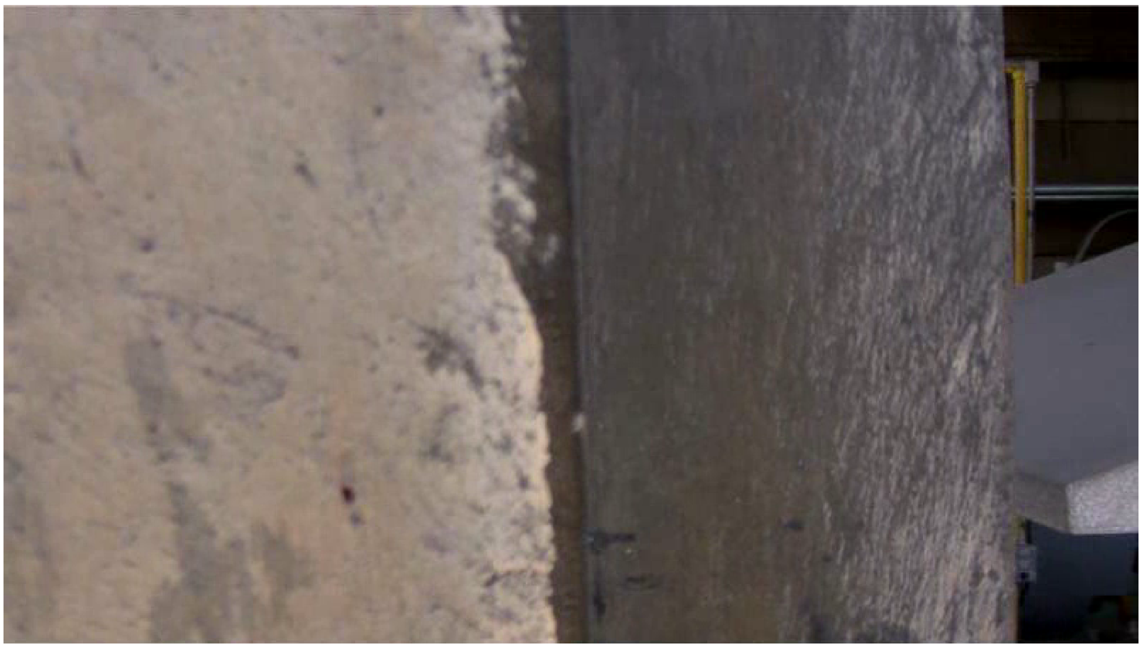


(b)

Figure 4.13: Camera trajectory for the video 5 in the KITTI odometry dataset. (a) Ground truth. (b) Result of the SLAM algorithm.



(a)



(b)

Figure 4.14: Example of frames from the DORIS videos with a flat surface occupying a significant portion of the frame. (a) Frame 7400. (b) Frame 12570.

using a frame before and a frame after the pillar. As can be seen from Fig. 4.15, the pillars may be so wide that there is almost no overlap in the scene before and after it, which makes the estimation of the camera trajectory unfeasible [106].

In a second experiment, each video sequence was split into several smaller sequences, removing the parts of the videos that are near a pillar. It was also decided to use only sequences where the camera performs simpler movements. Two types of video excerpt were tested: pieces of the video with curves in the rail, but the camera movement is entirely contained in the horizontal plane, and pieces of the video in which the camera moves in a straight section of the rail.

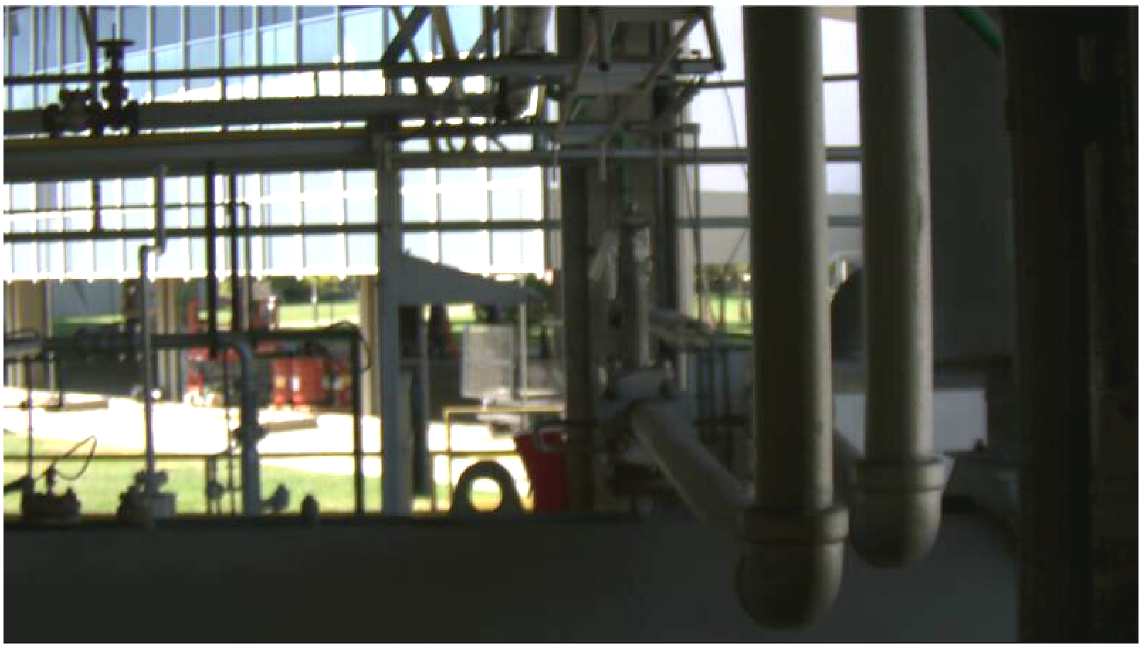
For this experiment, the algorithm is able to execute all its steps but still provides a result that does not match the expected trajectory, as shown in Fig. 4.17. Several causes were identified as being responsible for this problem.

Even with the removal of the regions with pillars, several other textureless objects may occupy a large portion of the frames, due to the presence of large machinery in the industrial environment, which, as previously mentioned, deteriorates the results. Examples of the textureless objects contained in the scene can be seen in Fig. 4.16.

Another characteristic from the DORIS videos is that they often have objects closer to the cameras when compared to the videos from the KITTI dataset. These objects create larger regions with occlusion of the background, which increases the number of outliers in the correspondences between keypoints.

A further analysis revealed another problem on the DORIS videos that can make the SLAM algorithm fail, which is related to the type of movement performed by the camera. In these videos, the camera moves along a direction that is perpendicular to the orientation of the camera, contrary, for example, to the videos in the KITTI dataset, which have the camera pointed to the front of a car. In this case, the viewpoints disappear much faster in the videos, which reduces the field of view in common between several views. Consequently, it becomes harder to create connections between frames, the algorithm identifies fewer loop closures and provides less robust results.

For the sake of comparison, it was discussed in Section 4.1 that the frames must have a minimum camera displacement between them, and it was defined a step to select keyframes that present a displacement of around 5% of the image width. Therefore, one could expect that, if an object enters the camera field of view in the left of the image, and moves to the right until it vanishes (or the other way around), it should appear in around 20 keyframes. For the KITTY videos, during a curve, the objects that are closest to the camera and go through the entire camera field of view appear in around 40 frames, or 4 s of the video. In the DORIS videos, however, the turns are much faster. An object close to the camera appears in at most 20 frames, or 2 s of the video.


(a)

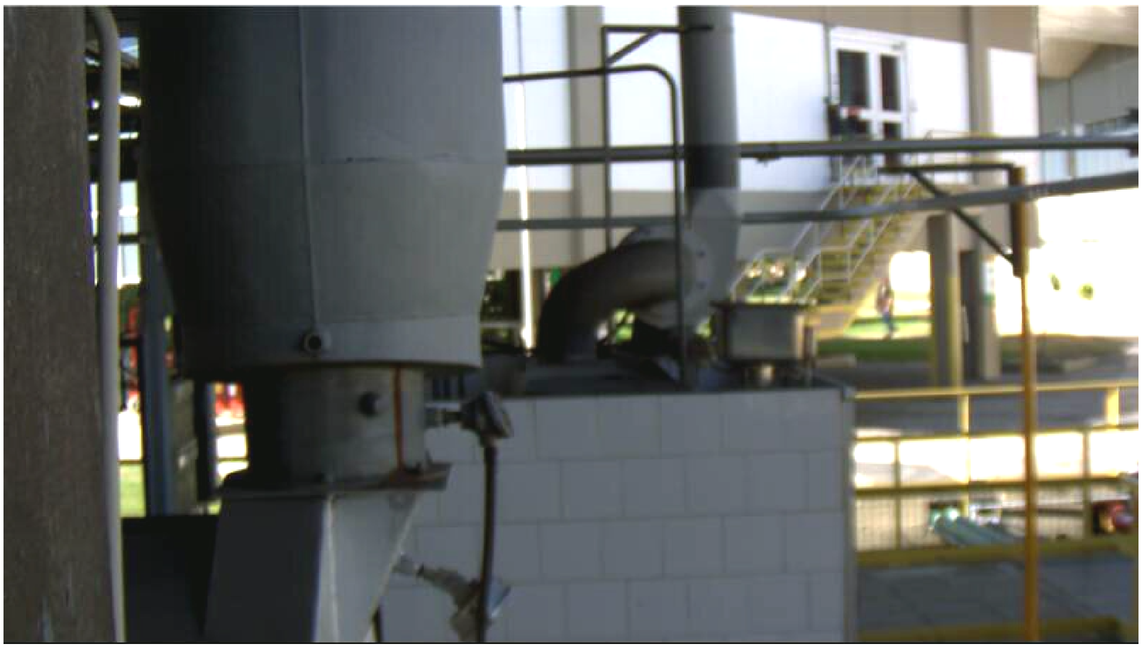
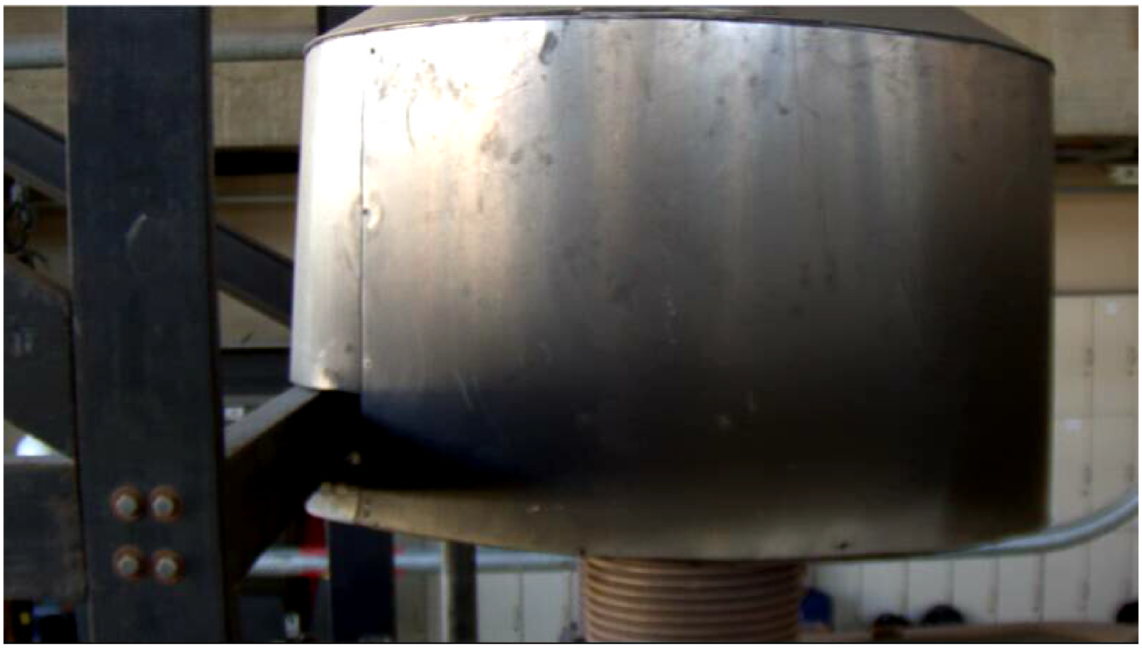
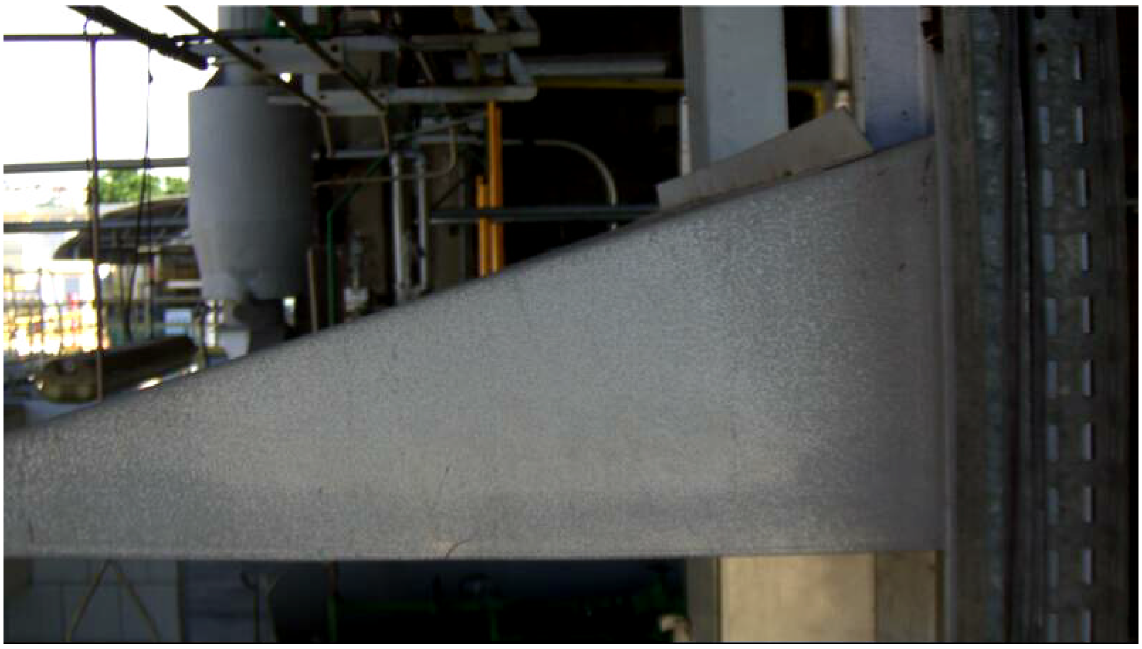

(b)

Figure 4.15: Example of frames from the DORIS videos showing the lack of overlap in the scenes before and after a pillar. (a) Frame obtained to the left of the pillar shown in Fig. 4.14(a). (b) Frame obtained to the right of the pillar shown in Fig. 4.14(a).



(a)



(b)

Figure 4.16: Example of frames from the DORIS videos containing large textureless objects. (a) Object in a sequence with curves in the rail. (b) Object in a sequence on a straight section of the rail.

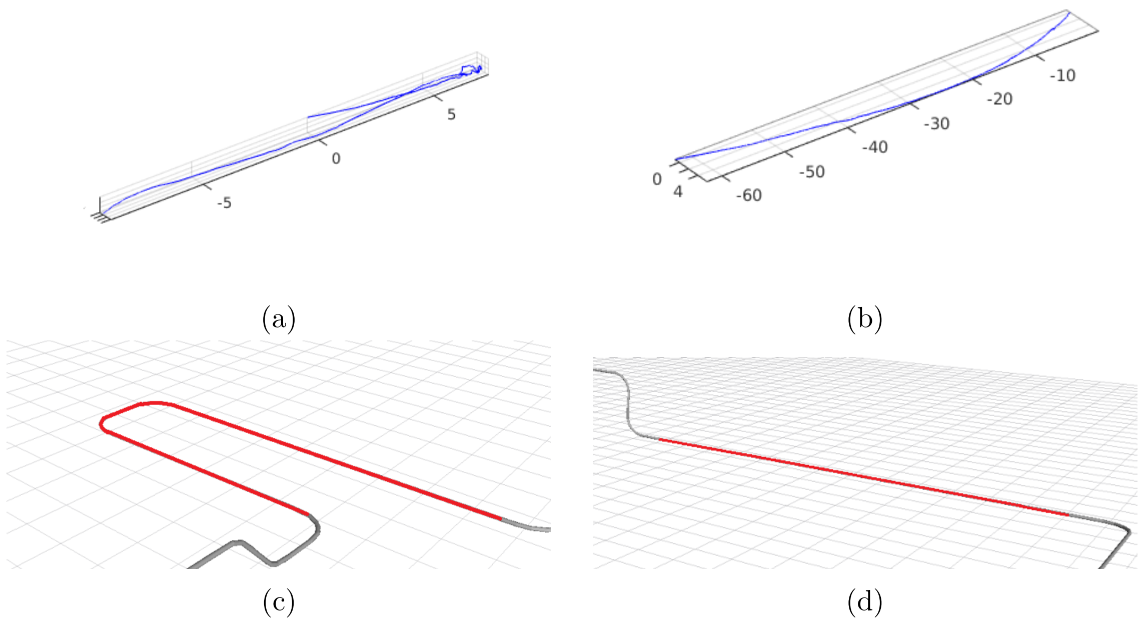


Figure 4.17: Camera trajectory for some excerpts of the DORIS videos. (a) Result for the u-shaped region of the rail. (b) Result for the straight section of the rail. (c) Ground truth of the U-shaped region of the rail (marked in red). (d) Ground truth of the straight section of the rail (marked in red).

4.3 Summary

This chapter described an approach to perform the estimation of the camera trajectory and mapping of an environment for videos acquired with a moving camera. The algorithm was tested in several databases with results similar to the ground truth.

However, when testing with the DORIS videos, is was observed that the algorithm was not able to operate properly. A discussion was made about several conditions that provide a challenge for the SLAM algorithms and how they appear in the DORIS videos. It should be concluded that the DORIS videos present a new challenging scenario with several restrictions for the SLAM algorithm, and it should be adopted as another benchmark to foster the development of more robust SLAM algorithms.

The next chapter describes some advances in the computation of the optical flow, and applies the optical flow algorithm to perform a spatial alignment of two frames from the DORIS system. Some concepts of the operation on the matrix space performed by this algorithm are also carried out to the next chapter.

Chapter 5

Video Spatial Alignment Using Optical Flow

The previous chapters dealt with the temporal alignment or synchronization of video sequences. However, due to differences in the sampling rate or difficulties in the acquisition, the synchronized frames can still be significantly different from each other. In some applications, it is also necessary to include a step to perform the spatial alignment of the frames. An approach that detects video anomalies [1] performs this step by computing a simple homography transformation [13] between the frames, which is a transformation that maps one plane into another plane. However, if the scene contains a wide range of depths it cannot be approximated as being contained on a single plane, therefore this transformation is likely to fail. In order to perform this alignment, one can consider instead the computation of the optical flow, which estimates the apparent motion field between the images and represents a generic translation computed independently for each pixel or region.

The study of motion fields in image sequences is a relevant topic in computer vision since the movement of objects provides important characteristics which allow a different level of data analysis. The applications that can benefit from this study are various: for surveillance it makes possible the detection of moving objects in a steady background, besides providing information on the motion that may be used on the objects tracking; in meteorology it allows the interpretation of clouds and detection of climatic phenomena; for video compression it gives means to reduce data storage and transmission of frames by estimating it using its optical flow and a close frame. However, it is important to bear in mind that the real motion field of the objects, in general, is unknown and must be estimated from projections of the objects in the images.

Optical flow consists of the calculation of the apparent motion of the image pixels. Using two images, the goal is to calculate a field of two-dimensional vectors that register the movement of points from one image to the other. For such calculation, an algorithm of optical flow must be able to overcome a few obstacles that can arise in real videos, such as outliers from the discontinuities or occlusions, different lighting conditions and regions large motions between images.

Among the optical flow methods, an approach less developed in the literature is the representation of images where the content for each pixel is a feature instead of a luminance value, like those defined inside the context of a Riemannian manifold. This kind of representation allows a better generalization of the image space, enabling it to be represented with a non-Euclidian geometry. An example for the use of images inserted in a Riemannian manifold is given by the eigenfaces, in which the Euclidian distance between pixels from an image is not the best way to represent the distance between two faces. Thus, if one maps the image space to some specific feature space, it is possible that an algorithm is able to better represent the image content.

This chapter studies several data structures and image descriptors in the context of the optical flow estimation. These required the development of an algorithm that handles matrix data. The optical flow algorithm is extended to operate on images where each pixel contains a tensor that belongs to a Riemannian manifold. The algorithms were tested in a benchmark for evaluation of optical flow. Finally, tests using videos from the DORIS system depicted in Section 1.2.1 and the VDAO database described in Section 1.2.2 were also performed.

This chapter is organized as follows. Section 5.1 details the basics of the optical flow computation. Section 5.2 depicts several techniques that extract complex properties from the set of pixels, defining a tensor or a descriptor for each pixel, which can provide more information than just the gray level of the pixels, and can be used in the optical flow estimation. Since some studied techniques map the original images to tensor-valued images, Section 5.3 shows an extension of the optical flow algorithms able to operate such spaces. Section 5.4 shows several results for the developed algorithms for an optical flow database, and also some results for videos from the surveillance system application described in Section 1.2.1 and the database described in Section 1.2.2.

5.1 Optical Flow Estimation

Given two images $\mathbf{I_1}$ and $\mathbf{I_2}$ of size $M \times N$, the optical flow searches for the vector fields $\mathbf{u}$ and $\mathbf{v}$ that makes one image match the other, that is, for the discrete case:

$$I_1(i,j) = I_2(i + u(i,j), j + v(i,j)), \tag{5.1}$$

where $u(i,j)$ and $v(i,j)$ are respectively elements of $\mathbf{u}$ and $\mathbf{v}$. In the following sections, two optical flow algorithms are derived. First, a modern version of the Horn and Schunk method [49], using quadratic functions, is defined. Then, several algorithms that were based on the Horn and Schunk method are discussed and a generic algorithm is shown.

5.1.1 Horn and Schunck Method

The first successful optical flow method [49] consists in the estimation of the motion field based on the intensity values of the image pixels. The estimation of $\mathbf{u}$ and $\mathbf{v}$ can be made with the following optimization procedure:

$$\begin{aligned}(\hat{\mathbf{u}}, \hat{\mathbf{v}}) &= \arg\min_{\mathbf{u},\mathbf{v}} \sum_{i,j} E_D(i,j) + \lambda E_S(i,j) \\ &= \arg\min_{\mathbf{u},\mathbf{v}} \sum_{i,j} \rho_D(I_1(i,j) - I_2(i + u(i,j), j + v(i,j))) + \\ &\qquad \lambda[\rho_S(u(i,j) - u(i+1,j)) + \rho_S(u(i,j) - u(i,j+1)) + \\ &\qquad \rho_S(v(i,j) - v(i+1,j)) + \rho_S(v(i,j) - v(i,j+1))]. \end{aligned} \tag{5.2}$$

In Eq. (5.2), the term $E_D(i,j)$ represents a data term and $\rho_D(x)$ is a penalty function. This term is responsible for finding similar regions in the images related to the vector field. that is, to enforce the validity of the brightness constancy given by Eq. (5.1). The term $E_S(i,j)$, where $\rho_S(x)$ is also a penalty function, weighted by a constant λ, represents the spatial or regularization term which is responsible for finding vectors with spatial consistency, reducing outliers. For the basic optical flow algorithm proposed in [49], both penalty functions are of the form $\rho(x) = x^2$. This corresponds to making a Gaussian assumption on the noise, so this method cannot deal with motion boundaries and occlusions, and is not robust in these cases.

One main issue of Eq. (5.2) is the non-linearity of the image intensities in the values of $u(i,j)$ and $v(i,j)$. Assuming these displacement values to be small, one can perform a first order Taylor approximation to linearize the data term:

$$I_2(i + u(i,j), j + v(i,j)) \approx I_2(i,j) + I_{2x}(i,j)u(i,j) + I_{2y}(i,j)v(i,j), \tag{5.3}$$

where I_{2x} and I_{2y} represent, respectively, the horizontal and vertical image gradients.

Since a linearization is necessary, the algorithm may fail when the displacement is too large. A common practice in modern algorithms is to include a warping strategy, in combination with a course-to-fine estimation. For each iteration, a current estimate of the motion field is used to create a warped version of the image $\mathbf{I_2}$, and only the flow increment is computed. For this case, the following linearization is performed:

$$\begin{aligned} I_2(i + u_k(i,j) + du(i,j), j + v_k(i,j) + dv(i,j)) \approx I_2(i + u_k(i,j), j + v_k(i,j)) \\ + I_{2x}(i + u_k(i,j), j + v_k(i,j))du(i,j) \\ + I_{2y}(i + u_k(i,j), j + v_k(i,j))dv(i,j), \end{aligned} \tag{5.4}$$

where $u_k(i,j)$ and $v_k(i,j)$ are the current estimates of the flow and $du(i,j)$ and $dv(i,j)$ are their increments. The term $I_2(i + u_k(i,j), j + v_k(i,j))$ represents the warped version of the image I_2, where the pixels (i,j) are displaced by the current estimates of the flow.

To deal with large motions, the optical flow computation can be performed in an incremental way by a multiscale approach that uses subsampled versions of each frame [51, 54]. If one subsamples the two images, the motion field is numerically smaller, so the linearizations should hold, and they provide a rough estimate to the true displacement, therefore the flow increment that must be estimated should also be small. The algorithm creates a pyramid with several levels of subsampling. During the optimization, the optical flow obtained in a previous iteration, where the input is a smaller version of the image, is used as an initialization for the algorithm using a higher resolution.

Since the regularization term compares the flow for each position with the flow on its neighbors, one can create a linear system to simultaneously optimize this term for all pixels. For the data term, the following expression is found:

$$\begin{aligned} \sum_{i,j} E_D(i,j) = \sum_{i,j} (I_{2x}(i + u_k(i,j), j + v_k(i,j))du(i,j) \\ + I_{2y}(i + du(i,j), j + v_k(i,j))dv(i,j) + I_t(i,j))^2, \end{aligned} \tag{5.5}$$

where $I_t(i,j) = I_1(i,j) - I_2(i + u_k(i,j), j + v_k(i,j))$.

For simplicity in the notations, we define $I_{2w}(i,j) = I_2(i + u_k(i,j), j + v_k(i,j))$, which is equivalent to assuming that the pixels (i,j) are displaced by the current flow estimate $(u_k(i,j), v_k(i,j))$ creating a warped version (I_{2w}) of the image I_2.

$$\begin{aligned} \sum_{i,j} E_D(i,j) &= \sum_{i,j} (I_{2x}(i,j)du(i,j) + I_{2y}(i,j)dv(i,j) + I_t(i,j))^2 \\ &= \sum_{i,j} \left(I_t(i,j) + \begin{bmatrix} I_{2x}(i,j) & I_{2y}(i,j) \end{bmatrix} \begin{bmatrix} du(i,j) \\ dv(i,j) \end{bmatrix} \right)^2, \end{aligned} \tag{5.6}$$

where $I_{2x}(i,j)$, $I_{2y}(i,j)$ and $I_t(i,j)$ are computed using the warped image I_{2w}.

In Eq. (5.6), the sum of squares can be transformed into the L_2 norm of a vector.

Stacking all pixels (i,j) in a single vector, one finds the following equation:

$$\sum_{i,j} E_D(i,j) = \left\| \mathbf{b} + \mathbf{J} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|_2^2, \tag{5.7}$$

where:

$$\mathbf{du} = \begin{bmatrix} du(1,1) \\ \vdots \\ du(M,N) \end{bmatrix}_{T\times 1}, \tag{5.8}$$

$$\mathbf{dv} = \begin{bmatrix} dv(1,1) \\ \vdots \\ dv(M,N) \end{bmatrix}_{T\times 1}, \tag{5.9}$$

$$\mathbf{b} = \begin{bmatrix} I_t(1,1) \\ \vdots \\ I_t(M,N) \end{bmatrix}_{T\times 1}, \tag{5.10}$$

and

$$\mathbf{J} = \begin{bmatrix} I_{2x}(1,1) & & 0 & I_{2y}(1,1) & & 0 \\ & \ddots & & & \ddots & \\ 0 & & I_{2x}(M,N) & 0 & & I_{2y}(M,N) \end{bmatrix}_{T\times 2T}, \tag{5.11}$$

with $T = MN$.

For the regularization term, a similar derivation can be made:

$$\begin{aligned} \sum_{i,j} E_S(i,j) = \sum_{i,j} & (u_k(i,j) + du(i,j) - u_k(i+1,j) + du(i+1,j))^2 \\ & +(u_k(i,j) + du(i,j) - u_k(i+1,j) + du(i,j+1))^2 \\ & +(v_k(i,j) + dv(i,j) - v_k(i+1,j) + dv(i+1,j))^2 \\ & (v_k(i,j) + dv(i,j) - v_k(i+1,j) + dv(i,j+1))^2. \end{aligned} \tag{5.12}$$

Given the matrices $\mathbf{F_y}$ and $\mathbf{F_x}$ such that:

$$\mathbf{F_y x} = \begin{bmatrix} 1 & -1 & 0 & & \cdots & & 0 \\ 0 & 1 & -1 & 0 & & \cdots & 0 \\ \vdots & & \vdots & & & \vdots & \end{bmatrix} \begin{bmatrix} x(1,1) \\ \vdots \\ x(M,N) \end{bmatrix} = \begin{bmatrix} x(1,1) - x(2,1) \\ \vdots \\ \end{bmatrix} \tag{5.13}$$

and

$$\mathbf{F_x x} = \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & \\ 0 & 1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & & \end{bmatrix} \begin{bmatrix} x(1,1) \\ \vdots \\ x(M,N) \end{bmatrix} = \begin{bmatrix} x(1,1) - x(1,2) \\ \vdots \\ \end{bmatrix}. \tag{5.14}$$

Stacking all pixels (i,j) in a single vector, the comparison between different neighbor displacements seen in Eq. (5.12) can be expressed through a product with the matrices $\mathbf{F_y}$, $\mathbf{F_x}$ and the stacked vector. Therefore:

$$\begin{aligned} \sum_{i,j} E_S(i,j) = \|\mathbf{F_y u} + \mathbf{F_y du}\|_2^2 + \|\mathbf{F_x u} + \mathbf{F_x du}\|_2^2 \\ + \|\mathbf{F_y v} + \mathbf{F_y dv}\|_2^2 + \|\mathbf{F_x v} + \mathbf{F_x dv}\|_2^2 \,, \end{aligned} \tag{5.15}$$

where:

$$\mathbf{u} = \begin{bmatrix} u_k(1,1) \\ \vdots \\ u_k(M,N) \end{bmatrix}_{T\times 1} , \tag{5.16}$$

$$\mathbf{v} = \begin{bmatrix} v_k(1,1) \\ \vdots \\ v_k(M,N) \end{bmatrix}_{T\times 1} , \tag{5.17}$$

and **du** and **dv** are the same as before.

If one stacks **u** and **v** in a single vector, a more concise form can be found:

$$\sum_{i,j} E_s(i,j) = \left\| \mathbf{r} + \mathbf{F} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|_2^2 , \tag{5.18}$$

where

$$\mathbf{r} = \begin{bmatrix} \mathbf{F_y u} \\ \mathbf{F_x u} \\ \mathbf{F_y v} \\ \mathbf{F_x v} \end{bmatrix}_{4T\times 1} , \tag{5.19}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{F_y} & \mathbf{0} \\ \mathbf{F_x} & \mathbf{0} \\ \mathbf{0} & \mathbf{F_y} \\ \mathbf{0} & \mathbf{F_x} \end{bmatrix}_{4T\times 2T} . \tag{5.20}$$

Replacing Eqs. (5.18) and (5.7) in Eq. (5.2), one can consider the data and regularization terms in a single equation:

$$
\begin{aligned}
(\mathbf{d\hat{u}}, \mathbf{d\hat{v}}) &= \arg\min_{\mathbf{du},\mathbf{dv}} \|\mathbf{b} + \mathbf{J}\begin{bmatrix}\mathbf{du}\\ \mathbf{dv}\end{bmatrix}\|_2^2 + \lambda\|\mathbf{r} + \mathbf{F}\begin{bmatrix}\mathbf{du}\\ \mathbf{dv}\end{bmatrix}\|_2^2 \\
&= \arg\min_{\mathbf{du},\mathbf{dv}} \|\mathbf{g} + \mathbf{H}\begin{bmatrix}\mathbf{du}\\ \mathbf{dv}\end{bmatrix}\|_2^2, \qquad (5.21)
\end{aligned}
$$

$$
\mathbf{g} = \begin{bmatrix}\mathbf{b}\\ \lambda\mathbf{r}\end{bmatrix}_{5T\times 1}, \qquad (5.22)
$$

$$
\mathbf{H} = \begin{bmatrix}\mathbf{J}\\ \lambda\mathbf{F}\end{bmatrix}_{5T\times 2T}. \qquad (5.23)
$$

Finally, after each intermediate iteration, a median filter is applied to the flow computed, which helps in the removal of outliers [53]. The full method is described in Alg. 9 and a block diagram is shown in Fig. 5.1.

Algorithm 9 Baseline coarse-to-fine optical flow algorithm.

```
    Input: Image pair $\mathbf{oI_1}$ and $\mathbf{oI_2}$ of size $M \times N$, regularization parameter $\lambda$,
    pyramid levels $L$, pyramid factor $\eta$.
    Output: Vector fields $\mathbf{u_k}$ and $\mathbf{v_k}$.
 1: Initialize $\mathbf{u_k} = \mathbf{0}_{M/L\eta \times N/L\eta}$ and $\mathbf{v_k} = \mathbf{0}_{M/L\eta \times N/L\eta}$
 2: for $l \in \{L-1, ..., 0\}$ do {Loop for the multiscale approach}
 3:    Downsample the original images $\mathbf{oI_1}$ and $\mathbf{oI_2}$ by $l\eta$ to create $\mathbf{I_1}$ and $\mathbf{I_2}$
 4:    Upsample the previous flow $\mathbf{u_k}$ and $\mathbf{v_k}$ by $\eta$
 5:    Assign $\mathbf{u_k} = \eta\mathbf{u_k}$ and $\mathbf{v_k} = \eta\mathbf{v_k}$
 6:    for $n \in \{1, ..., N_{iter}\}$ do {Inner loop}
 7:       Warp $\mathbf{I_2}$: $I_2(i,j) = I_2(i + u_k(i,j), j + v_k(i,j))$
 8:       Compute $\mathbf{I_2x}$, $\mathbf{I_2y}$ and $\mathbf{I_t}$
 9:       Compute $\mathbf{g}$ and $\mathbf{H}$ from Eqs. (5.22) and (5.23)
10:       Use a solver to find the solution of Eq. (5.21)
11:       Update the flow $\mathbf{u_k} = \mathbf{u_k} + \mathbf{du}$ and $\mathbf{v_k} = \mathbf{v_k} + \mathbf{dv}$
12:       if $l \neq 0$ then
13:          Apply a median filter to $\mathbf{u_k}$ and $\mathbf{v_k}$
14:       end if
15:    end for
16: end for
```

5.1.2 General Optical Flow Algorithm

Several approaches propose variations to the method in [49], described in Section 5.1 above. Some works propose different penalty functions, like $\rho(x) = \sqrt{(x^2 + \sigma)}$ [55], $\rho(x) = \log\left(1 + \frac{x^2}{2\sigma}\right)$ [54] and $\rho(x) = (x^2 + \epsilon^2)^a$ [50]. For this case, one can approximate Eq. (5.2) to a quadratic penalty around the current motion, introducing

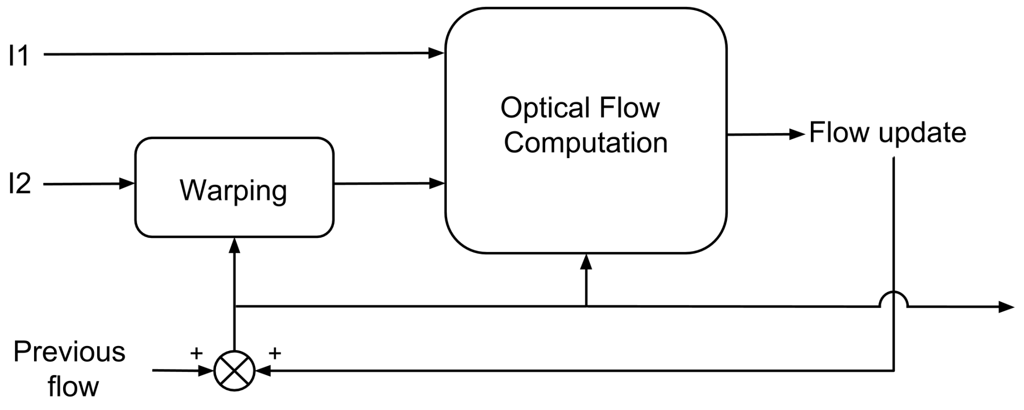


Figure 5.1: Block diagram of the iterative optical flow algorithm.

a weight $w(x)$ that depends on the selected robust function and the current motion, which is analogous to M-estimator [107] methods:

$$w(x) = \frac{\frac{\partial \rho(x)}{\partial x}}{x}. \tag{5.24}$$

Note that for a quadratic function, $w(x) = 2$. Therefore, the weight is constant and the problem becomes the one described in Section 5.1.

For the data term, this leads to the following equation:

$$\begin{aligned}\sum_{i,j} E_D(i,j) &= \sum_{i,j} w_D(I_t(i,j))(I_{2x}(i,j)du(i,j) + I_{2y}(i,j)dv(i,j) + I_t(i,j))^2 \\ &= \sum_{i,j} w_D(I_t(i,j))(I_t(i,j) + \begin{bmatrix} I_{2x}(i,j) & I_{2y}(i,j) \end{bmatrix} \begin{bmatrix} du(i,j) \\ dv(i,j) \end{bmatrix})^2, \end{aligned} \tag{5.25}$$

where w_D is computed for each iteration using Eq. (5.24) with the desired robust function ρ_D.

Performing the same development seen in the previous section, one finds that

$$\sum_{i,j} E_D(i,j) = \left\| \mathbf{b} + \mathbf{J} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|^2_{\mathbf{\Omega_D}}, \tag{5.26}$$

where

$$\|X\|^2_\Omega = X^T \Omega X \tag{5.27}$$

is a Mahalanobis norm and $\mathbf{\Omega_D} = diag(w_D(\mathbf{b}))$.

For the regularization, one finds:

$$\sum_{i,j} E_s(i,j) = \left\| \mathbf{r} + \mathbf{F} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|^2_{\mathbf{\Omega_S}}, \tag{5.28}$$

with $\mathbf{\Omega_S} = diag(w_S(\mathbf{r}))$ and w_S is obtained using Eq. (5.24) for the desired ρ_S (which is often the same as ρ_D).

Replacing Eqs. (5.26) and (5.28) in Eq. (5.28), the optimization problem becomes:

$$(\mathbf{d\hat{u}}, \mathbf{d\hat{v}}) = \arg \min_{\mathbf{du},\mathbf{dv}} \left\| \mathbf{g} + \mathbf{H} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|^2_{\mathbf{\Omega}}, \tag{5.29}$$

with

$$\mathbf{\Omega} = \begin{bmatrix} \mathbf{\Omega_D} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Omega_S} \end{bmatrix}. \tag{5.30}$$

In order to optimize non-convex functions, a graduated non-convexity scheme [56] is applied. First, the algorithm estimates the optical flow by using a convex penalty function. In any following step, the results from the previous step are used as an initial condition, and the penalty function to be used is a combination of convex and non-convex functions. In the last step, the algorithm optimizes the optical flow using only the desired non-convex function. This procedure is equivalent to minimizing a variant of Eq. (5.29):

$$(\mathbf{d\hat{u}}, \mathbf{d\hat{v}}) = \arg \min_{\mathbf{du},\mathbf{dv}} \alpha \left\| \mathbf{g_Q} + \mathbf{H_Q} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|^2_{\mathbf{\Omega_Q}} + (1-\alpha) \left\| \mathbf{g} + \mathbf{H} \begin{bmatrix} \mathbf{du} \\ \mathbf{dv} \end{bmatrix} \right\|^2_{\mathbf{\Omega}}, \tag{5.31}$$

where $\mathbf{g_Q}$, $\mathbf{H_Q}$ and $\mathbf{\Omega_Q}$ were obtained using Eqs. (5.22), (5.23) and (5.30) considering a quadratic robust function. The parameter α controls the importance given to the quadratic approximation and the desired robust function, and is changed along the iterations.

Among the more advanced algorithms, one can cite [50], which reviews improvements proposed in decades of algorithmic development and compares them with the original algorithm. For this approach, the use of median filters [53] can reduce the number of outliers and is proved to be one of the main responsible for the performance gain in the estimation. A different algorithm was proposed, which estimates $\mathbf{u}$ and $\mathbf{v}$ and replaces the regularization term $E_S(u(i,j), v(i,j))$ in Eq. (5.2) by:

$$E_S(u(i,j), v(i,j)) = \sum_{(i',j') \in N_{i,j}} z_{i,j}^{i',j'} (\rho_S(\hat{u}(i,j) - \hat{u}(i',j')) + \rho_S(\hat{v}(i,j) - \hat{v}(i',j'))), \tag{5.32}$$

where $N_{i,j}$ represents a neighborhood around (i,j) and $z_{i,j}^{i',j'}$ is a weight that control how much the displacement in the pixel (i',j') influences the estimation of the displacement in the pixel (i,j), and is defined as:

$$z_{i,j}^{i',j'} \propto \exp\left(-\frac{|i-i'|^2+|j-j'|^2}{2\sigma_1^2} - \frac{|I_1(i,j)-I_1(i',j')|^2}{2\sigma_2^2 n_c}\right)\frac{o(i,j)}{o(i',j')}, \tag{5.33}$$

for some constants σ_1, σ_2, and n_c, and $o(i,j)$ represents an occlusion estimation [108]. This equation means that the regularization gives more weight to pixels close to each other, to pixels whose value are similar, and to locations where no occlusion is detected.

It was also shown that performing this regularization can be approximated as replacing the median filtering step in the optical flow algorithm (see Alg. 9) by a weighted median, with weights given by Eq. (5.33). The complete algorithm is described in Alg. 10.

Algorithm 10 Optical flow algorithm with a non-quadratic robust function and a weighted non-local regularization.

```
    Input: Image pair $\mathbf{oI_1}$ and $\mathbf{oI_2}$ of size $M \times N$, regularization parameter $\lambda$, pyramid levels $L$, pyramid factor $\eta$.
    Output: Vector fields $\mathbf{u_k}$ and $\mathbf{v_k}$.
 1: Initialize $\mathbf{u_k} = \mathbf{0}_{M/L\eta \times N/L\eta}$ and $\mathbf{v_k} = \mathbf{0}_{M/L\eta \times N/L\eta}$
 2: for $\alpha \in \{1, 0.5, 0\}$ do {Loop for the graduated non-convexity approach}
 3:   for $l \in \{L-1, ..., 0\}$ do {Loop for the multiscale approach}
 4:     Downsample the original images $\mathbf{oI_1}$ and $\mathbf{oI_2}$ by $l\eta$ to create $\mathbf{I_1}$ and $\mathbf{I_2}$
 5:     Upsample the previous flow $\mathbf{u_k}$ and $\mathbf{v_k}$ by $\eta$
 6:     Assign $\mathbf{u_k} = \eta\mathbf{u_k}$ and $\mathbf{v_k} = \eta\mathbf{v_k}$
 7:     for $n \in \{1, ..., N_{iter}\}$ do {Inner loop}
 8:       Warp $\mathbf{I_2}$: $I_2(i,j) = I_2(i + u_k(i,j), j + v_k(i,j))$
 9:       Compute $\mathbf{I_2x}$, $\mathbf{I_2y}$ and $\mathbf{I_t}$
10:       Compute $\mathbf{g}$, $\mathbf{H}$ and $\mathbf{\Omega}$ from Eqs. (5.22), (5.23) and (5.30)
11:       Compute $\mathbf{g_Q}$, $\mathbf{H_Q}$ and $\mathbf{\Omega_Q}$ from Eq. (5.31)
12:       Use a solver to find the solution of Eq. (5.31)
13:       Update the flow $\mathbf{u_k} = \mathbf{u_k} + \mathbf{du}$ and $\mathbf{v_k} = \mathbf{v_k} + \mathbf{dv}$
14:       if $l \neq 0$ then
15:         Compute the weights $z_{i,j}^{i',j'}$ defined in Eq. (5.33)
16:         Apply a weighted median filter to $\mathbf{u_k}$ and $\mathbf{v_k}$ using the weights $z_{i,j}^{i',j'}$
17:       end if
18:     end for
19:   end for
20: end for
```

5.2 Advanced Data Terms

Several data representation techniques were considered for this study. In [109], it is proposed a tensorial representation of images in the HSL color space in order to enable the use of tensor theory in color dissimilarity computation. Another case studied is the structure tensor [110], which is a widely used matrix that summarizes local structure information derived from the image gradients. Finally, an image descriptor similar to the one proposed in [62] was also tested, since it is robust to illumination changes and is defined by a set of oriented gradients that are more general than the ones used for the structure tensor computation.

5.2.1 Color Tensor

The work of Rittner *et al.* [109] proposes a two-dimensional second-order tensor derived from the hue, saturation and luminance (HSL)color space. More specifically, if the tensor is symmetric, the eigenvalues are real and the eigenvectors are perpendicular [111]. Due to this property, this tensor can be univocally described by an ellipse. The eigenvectors of the tensor can be associated to the orientation of the ellipse, and the eigenvalues can be combined to define eccentricity and trace (also called sum) of the ellipse.

Those three properties (orientation, eccentricity, and trace) can intuitively be associated to colors in the HSL space. The hue, which has orientation information, defines the ellipse orientation. The saturation, which keeps information of the ratio of colors to determine the purity of a given color, can be associated with the eccentricity. The lightness, which defines the amount of brightness or energy in the color, has an association to the trace.

Alg. 11 defines the maps between RGB and HSL color spaces. Based on the relation between the HSL colors and the desired properties of the tensor, Alg. 12 defines the map from an RGB value to the color tensor.

Examples

For grayscale images, the 3 RGB channels are equal, so the pixel value is $\begin{bmatrix} C & C & C \end{bmatrix}$. This color is represented in the HSL space as $\begin{bmatrix} 0 & 0 & C \end{bmatrix}$ and is mapped to the tensorial space as:

$$\begin{cases} \theta = 0 \\ \lambda_1 = \lambda_2 = C/2 \\ \mathbf{R} = \begin{bmatrix} C/2 & 0 \\ 0 & C/2 \end{bmatrix} \end{cases}. \tag{5.34}$$

A visualization of the equivalent tensors for several grayscale values can be seen in Fig 5.2.

Algorithm 11 Color map from RGB to HSL.

```
    **Input**: RGB value ($R, G, B \in [0, 1]$).
    **Output**: HSL value ($H, S, L \in [0, 1]$).
1: Compute $C_{max} = max(R, G, B)$
2: Compute $C_{min} = min(R, G, B)$
3: Compute $\Delta = C_{max} - C_{min}$
4: Hue calculation:
       $$H = \begin{cases} 0, & \Delta = 0 \\ \frac{1}{6}(\frac{G-B}{\Delta}), & C_{max} = R, G > B \\ \frac{1}{6}(\frac{B-R}{\Delta} + 2), & C_{max} = G \\ \frac{1}{6}(\frac{R-G}{\Delta} + 4), & C_{max} = B \\ \frac{1}{6}(\frac{G-B}{\Delta} + 6), & C_{max} = R, G < B \end{cases}$$
5: Lightness calculation:
       $$L = (C_{max} + C_{min})/2$$
6: Saturation calculation:
       $$S = \begin{cases} 0, & \Delta = 0 \\ \frac{\Delta}{1-|2L-1|}, & \Delta \neq 0 \end{cases}$$
```

Algorithm 12 Map from the RGB color space to the color tensor of [109].

```
    **Input**: RGB value ($R, G, B \in [0, 1]$).
    **Output**: Color tensor R ($R \in \mathbb{R}^{2\times 2}$).
1: Convert to the HSL space using Alg. 11
2: Compute $\theta = \pi H$
3: Compute $\lambda_1 = \frac{L}{2-S}$
4: Compute $\lambda_2 = \frac{L(1-S)}{2-S}$
5: Compute:
       $$\mathbf{V} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$
6: Compute:
       $$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$
7: Compute the color tensor:
       $$\mathbf{R} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$
```

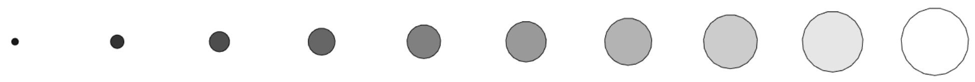


Figure 5.2: Visualization of the tensorial representation [109] for several grayscale values. The tensor for each pixel is represented as an ellipse whose axes correspond to the two eigenvectors and the radius are given by the two eigenvalues.

An image that contains values for the red, blue, and green colors covering the range $0 \cdots 255$ is seen in Fig. 5.3. The corresponding tensorial representation is given by Fig. 5.4.

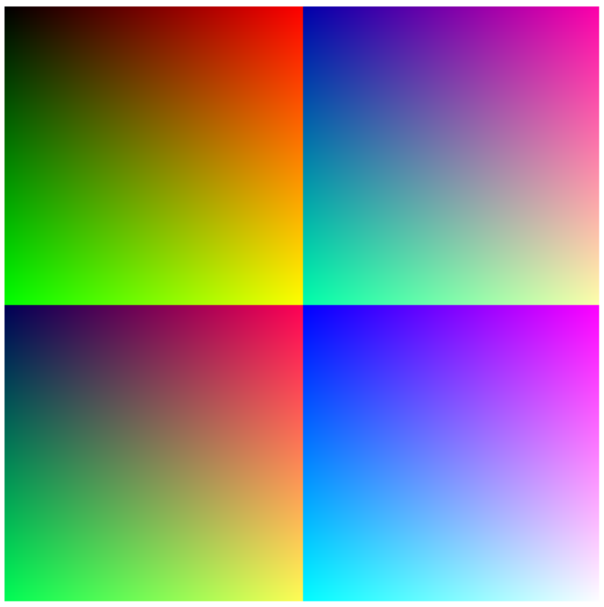


Figure 5.3: Example of color image obtained with the concatenation of four linear images. For each quadrant, R and G varies linearly in the range $0 \cdots 255$. The image in the top left has $B = 0$, the one in the top right has $B = 170$, the bottom left image has $B = 85$ and the bottom right image has $B = 255$

Based on Figs. 5.2 and 5.3, one can conclude that for values close to grayscale, the tensor has a structure that is almost isotropic (the ellipses are close to circles). The tensor for each pixel is represented as an ellipse whose axes correspond to the two eigenvectors and the radius are given by the two eigenvalues. Since for this range of values the hue component, which represents the dominant color, is not well defined, this isotropic behavior is desired since it means that for any orientation, which is defined by the hue, the resulting ellipse is almost the same.

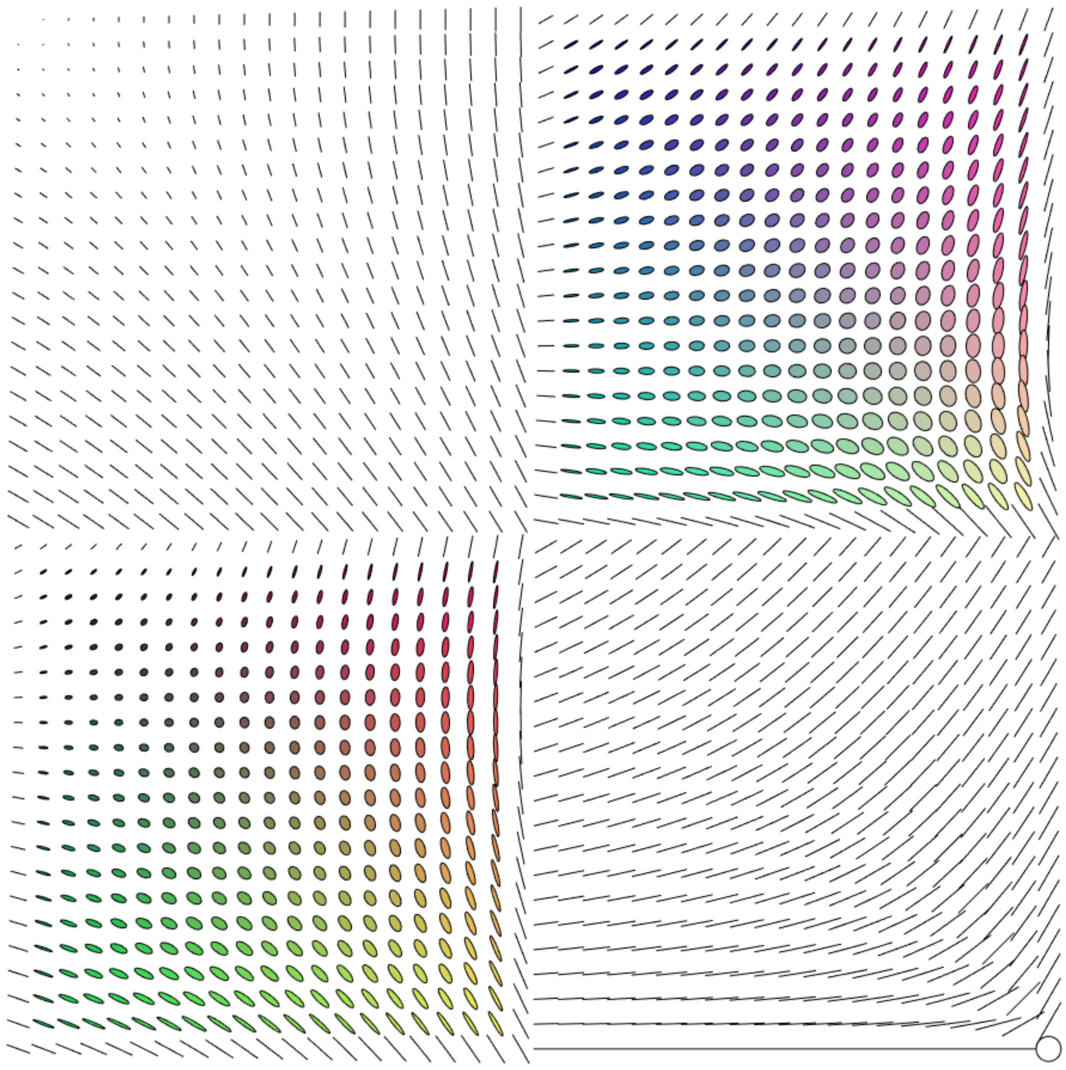


Figure 5.4: Visualization of the tensorial representation [109] corresponding to Fig. 5.3. The tensor for each pixel is represented as an ellipse whose axes correspond to the two eigenvectors and the radius are given by the two eigenvalues.

However, this representation has the disadvantage that it can create tensors with null eigenvalues (one null eigenvalue if the saturation S is 1 or two null eigenvalues if the lightness L is 0), which makes it unfit to be used with any operation of matrix inversion. In addition, one can also see a fast transition in the tensor space when a color has a saturation close to one, which can be seen in the borders of the images. This behavior may create discontinuities, which makes the optical flow algorithm more prone to errors due to the linearizations and estimation of derivatives performed in the algorithm.

5.2.2 Proposed Color tensor

In order to prevent the equivalent color tensor from ever having null eigenvalues, the previous map was altered. First, a small offset ε was included to prevent a null eigenvalue in the cases of a lightness L close to 0. Then, we also define the map using a correction value M in place of the saturation, to handle some discontinuities in the extremum values (saturation close to 0 and 1). Finally, a constant K was included to prevent the cases where a value of M creates a null eigenvalue. The proposed map is defined in Alg. 13.

Algorithm 13 Map from the RGB color space to the proposed color tensor.

```
    Input: RGB value ($R, G, B \in [0, 1]$).
    Output: Color tensor R ($R \in \mathbb{R}^{2\times 2}$).
 1: Convert to the HSL space using Alg. 11
 2: Compute $M = KL(1-L)S, \quad 0 < K < 4$
 3: Compute $\theta = \pi H$
 4: Compute $\lambda_1 = \frac{L+\varepsilon}{2-M}$
 5: Compute $\lambda_2 = \frac{(L+\varepsilon)(1-M)}{2-M}$
 6: Compute:
        $$\mathbf{V} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$
 7: Compute:
        $$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$
 8: Compute the color tensor:
        $$\mathbf{R} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$
```

Examples

Examples for the tensorial representation of grayscale and color images can be seen, respectively, in Figs. 5.5 and 5.6. The proposed map keeps the same advantages of the previous one, but also circumvents the cases where a tensor with a null eigenvalue could be created.

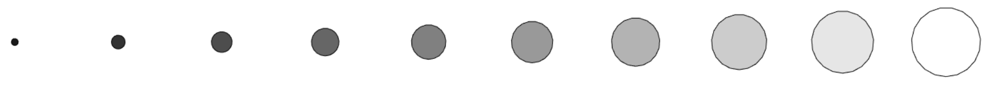


Figure 5.5: Visualization of the proposed tensorial representation for several grayscale values. The tensor for each pixel is represented as an ellipse whose axes correspond to the two eigenvectors and the radius are given by the two eigenvalues.

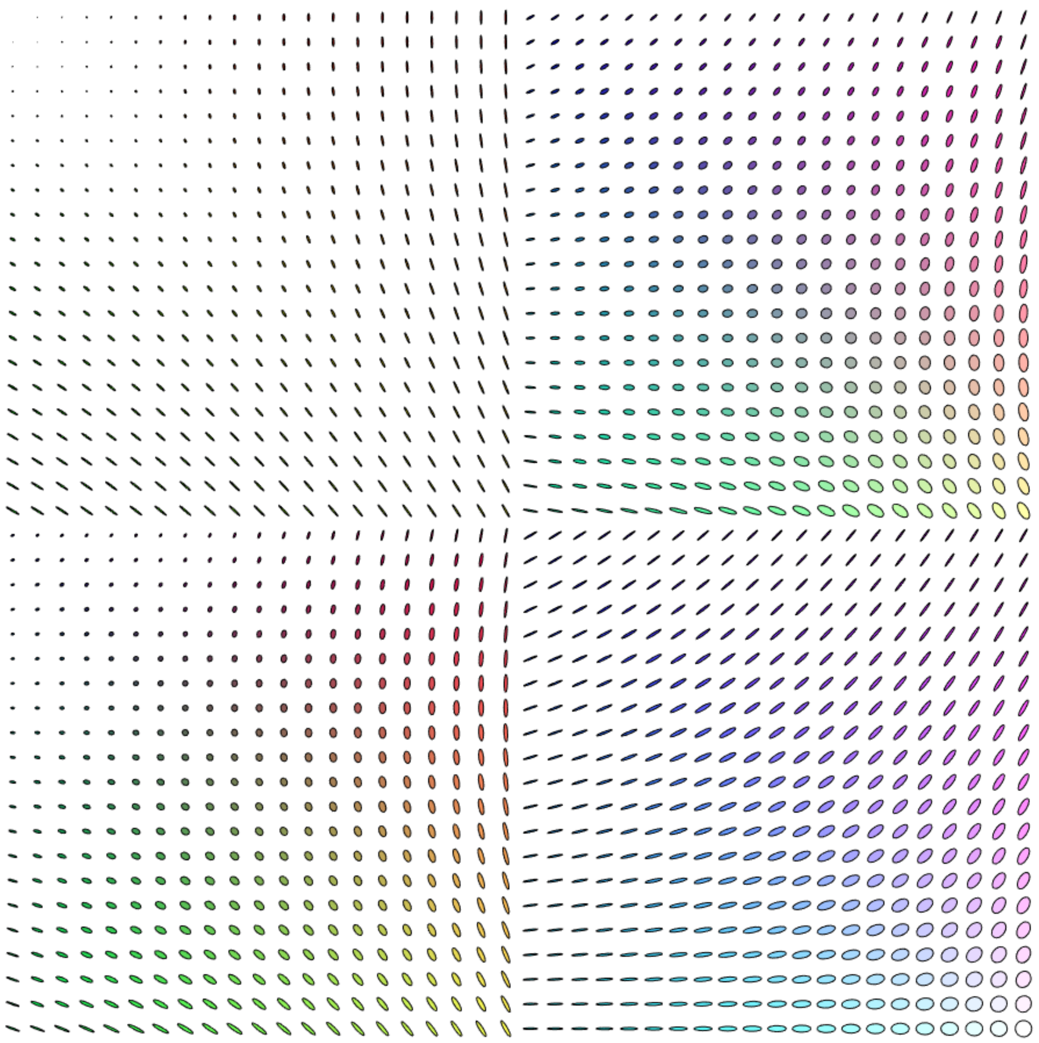


Figure 5.6: Visualization of the proposed tensorial representation corresponding to Fig. 5.3. The tensor for each pixel is represented as an ellipse whose axes correspond to the two eigenvectors and the radius are given by the two eigenvalues.

5.2.3 Structure Tensor

Image structure tensor is a matrix that contains information about the local statistics of the first-order intensity distribution. It is defined as:

$$\mathbf{S}(i,j) = \begin{bmatrix} [W * I_x^2](i,j) & [W * I_x I_y](i,j) \\ [W * I_x I_y](i,j) & [W * I_y^2](i,j) \end{bmatrix}, \tag{5.35}$$

where W is a smoothing window and $*$ is a two-dimensional convolution.

The advantage of this representation is that it is able to identify simple geometrical structures, based on the eigenvalues λ_1 and λ_2, which also provides an interpretation of the matrix as an ellipse. Edges and dominant orientations are denoted by the cases where $\lambda_1 \gg \lambda_2$, which have a physical interpretation as ellipses with high eccentricity, and the dominant orientation is associated to the eigenvector corresponding to λ_1. Corners are identified by the regions with $\lambda_1, \lambda_2 \gg 0$ and $\lambda_1 \approx \lambda_2$, that is, when the ellipses are close to isotropic and sufficiently large. Smooth or untextured regions appear when $\lambda_1, \lambda_2 \approx 0$.

In Fig. 5.7, a patch of the *Venus* image contained in the Middlebury [112] training dataset (see Section 5.4.1) is highlighted. It contains flat regions and edges in different orientations. Fig. 5.8 shows the corresponding image where each pixel is replaced by its structure tensor. Each tensor is represented as an ellipse whose axes correspond to the two eigenvectors and the radii are given by the two eigenvalues. One should notice that the ellipses tend to be thinner in the borders, where the gradients have a similar orientation.

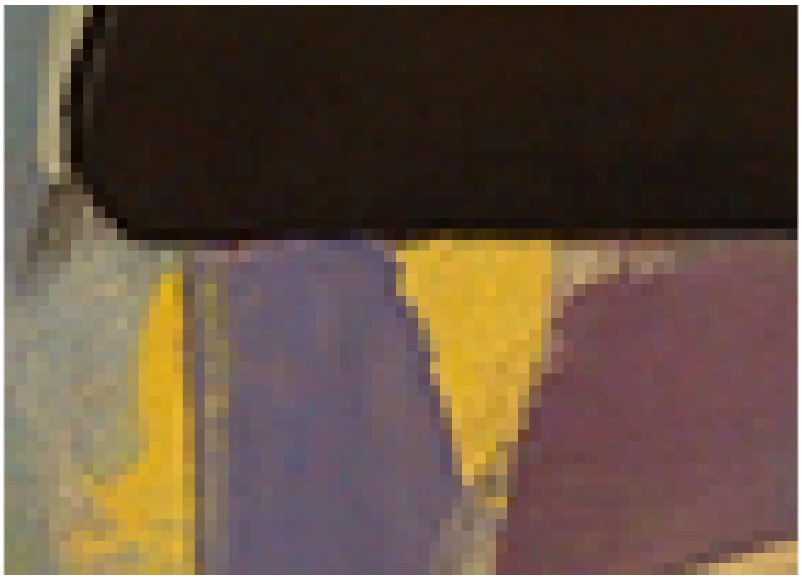


Figure 5.7: Patch from the image *Venus* in the Middlebury [112] training dataset (see Section 5.4.1).

5.2.4 ROF-NND Image Descriptor

An image descriptor that presents several illumination invariances was described in [62]. It divides the image in patches and defines a set of displacements based on the patch size. Each component of the k-th image descriptor for each pixel $\mathbf{x}$ is defined as:

$$C_{\mathcal{P}_k}(i,j) = \sum_{i',j' \in \mathcal{W}} (I(i',j') - I(i'+i_k, j'+j_k))^2, \tag{5.36}$$

which represents a comparison between the patch $\mathcal{W}$ centered on (i,j), the current pixel, and the patch $\mathcal{P}_k$ centered on $(i+i_k, j+j_k)$. The different patches $\mathcal{P}_k$ represent all the patches centered on a pixel contained on $\mathcal{W}$ other than (i,j). Therefore, the number of possible displacements (i_k, j_k), and consequently the number of compo-

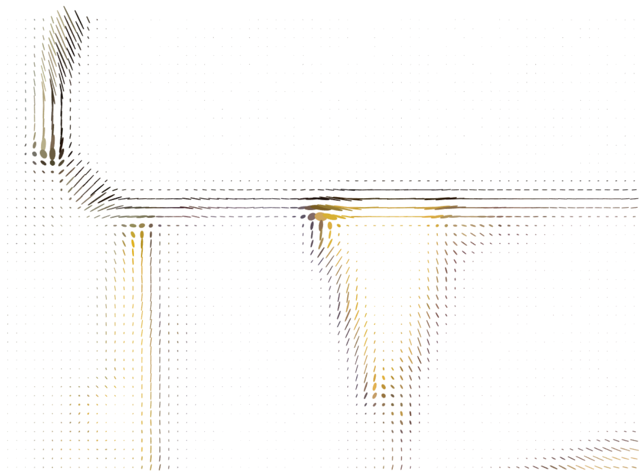


Figure 5.8: Visualization of the structure tensor image corresponding to Fig. 5.7. The tensor for each pixel is represented as an ellipse whose axes correspond to the two eigenvectors and the radii are given by the two eigenvalues.

nents for this descriptor, is given by the amount of pixels inside $\mathcal{W}$ minus one (for example, for 3×3 patches a descriptor with eight components is created, with displacements $(i_k, j_k) \in [(-1,1),(0,1),(1,1),(-1,0),(1,0),(-1,-1),(0,-1),(1,-1)]$). This procedure is exemplified in Fig. 5.9.

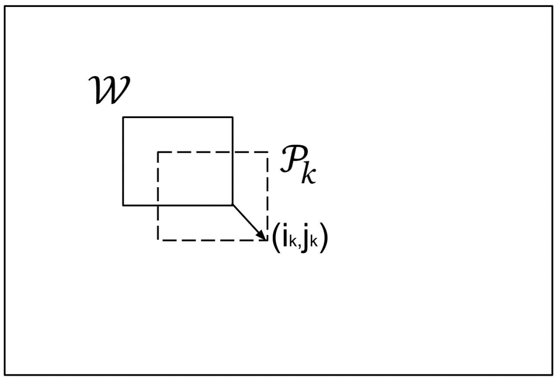


Figure 5.9: Image descriptor as described in [62]. A patch $\mathcal{W}$ is centered in each pixel, and compared to the patches $\mathcal{P}_k$.

One should notice that each descriptor represents information analogous to a directional gradient. The comparison tends to yield a higher value for $C_{\mathcal{P}_k}$ if, in the direction pointed by (i_k, j_k), there is a higher intensity variation. If the patches are similar, there is almost no variation in the direction of (i_k, j_k) and $C_{\mathcal{P}_k}$ has a lower value.

In order to make this descriptor robust to illumination changes, a normalization is applied and the descriptor is mapped as

$$C'_{\mathcal{P}_k}(i,j) = \exp\left(\frac{-C_{\mathcal{P}_k}(i,j)}{\sigma_x^2(i,j)}\right), \tag{5.37}$$

where σ_x is a measure of the local variation defined as the mean of the descriptors computed with a displacement of one pixel in the horizontal and vertical directions, that is

$$\sigma_x^2(i,j) = \frac{1}{4}\sum_{k=1}^{4} C_{\mathcal{P}_k}(i,j) = \frac{1}{4}\sum_{k=1}^{4}\sum_{i',j'\in\mathcal{W}} (I(i',j') - I(i'+i_k, j'+j_k))^2. \tag{5.38}$$

In Fig. 5.10 two descriptors obtained from Fig. 5.7 are shown. One can see that each component keeps a different piece of information with respect to the image intensity variation in different directions.

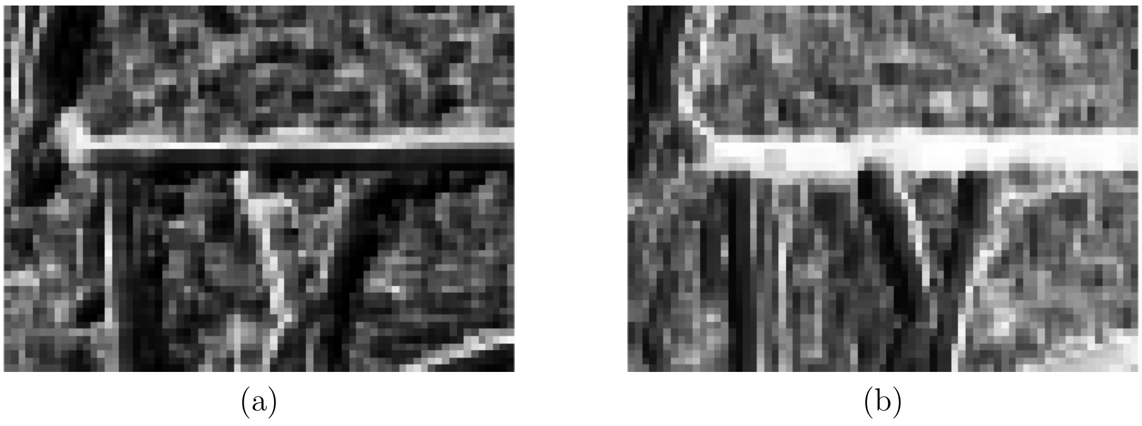


Figure 5.10: Visualization of two components of the descriptor image corresponding to Fig. 5.7. One can see that each component highlights different information in the image.

5.2.5 Proposed Image Descriptor

A few modifications were proposed in order to reduce the dimensionality, to adapt to the tensorial framework, and to improve its performance. From Fig. 5.9 and Eq. (5.37), one can see that there is a redundancy in the computation of the descriptors. The result of Eq. (5.37) for a patch $\mathcal{W}$ centered on (i,j) and a patch $C_{\mathcal{P}_k}$ obtained with a displacement of (i_k, j_k) is the same as the result obtained for a patch $\mathcal{W}$ centered on $(i+i_k, j+j_k)$ and a patch $C_{\mathcal{P}_w}$ obtained with a displacement of $(i_w, j_w) = (-i_k, -j_k)$. We reduce to half the number of descriptors by merging the redundant descriptors as

$$\hat{C}_{\mathcal{P}_k} = \frac{C_{\mathcal{P}_k} + C_{\mathcal{P}_{N/2+k}}}{2}. \tag{5.39}$$

We also extend this definition to define an SPD matrix similar to a structure tensor. One can see a similarity between Eq. (5.36) and Eq. (5.35), with the sum along the patch acting as the spatial integration given by the convolution with W, and the term $(I(i',j') - I(i'+i_k, j'+j_k))^2$ being similar to the elements I_x^2,I_y^2 in the structure tensor definition. For this reason, we define a structure tensor based on this descriptor information as

$$\mathbf{S}(i,j) = \begin{bmatrix} \hat{C}_{\mathcal{P}_1}(i,j) & \hat{C}_{1,2}(i,j) & \hat{C}_{1,3}(i,j) & \hat{C}_{1,4}(i,j) \\ \hat{C}_{1,2}(i,j) & \hat{C}_{\mathcal{P}_2}(i,j) & \hat{C}_{2,3}(i,j) & \hat{C}_{2,4}(i,j) \\ \hat{C}_{1,3}(i,j) & \hat{C}_{2,3}(i,j) & \hat{C}_{\mathcal{P}_3}(i,j) & \hat{C}_{3,4}(i,j) \\ \hat{C}_{1,4}(i,j) & \hat{C}_{2,4}(i,j) & \hat{C}_{3,4}(i,j) & \hat{C}_{\mathcal{P}_4}(i,j) \end{bmatrix}, \tag{5.40}$$

where $\hat{C}_{a,b}(i,j)$ is a generalization of Eq. (5.36) to generate the cross-term

$$\hat{C}_{a,b}(i,j) = \sum_{i',j' \in \mathcal{W}} (I(i',j') - I(i'+i_a, j'+j_a))(I(i',j') - I(i'+i_b, j'+j_b)). \tag{5.41}$$

For this tensor, we use a different normalization step. Instead of performing the normalization given by Eq. (5.37) for each component, we compute the structure tensor and normalize it using the trace such that

$$\hat{\mathbf{S}}(i,j) = \mathbf{S}(i,j)/\mathrm{trace}(\mathbf{S}(i,j)). \tag{5.42}$$

5.3 Optical Flow for Tensor-valued Images

In this section we extend the algorithms shown in Section 5.1.1 to operate on tensor-valued images, such as the case when the color tensor described in Section 5.2.1 or the structure tensor described in Section 5.2.3 are used for the optical flow computation. Considering the case of tensor-valued images $\mathbf{R_1}$ and $\mathbf{R_2}$, we define the data term as

$$E_D(i,j) = d^2(\mathbf{R_2}(i,j), \mathbf{R_2}(i + u_k(i,j) + du(i,j), j + v_k(i,j) + dv(i,j)), \tag{5.43}$$

where $d(\mathbf{A},\mathbf{B})$ is a metric to compute the similarity between two tensors (note that we are still not considering the use of a non-linear penalty function in the algorithm). Among the similarity metrics, one can cite the Frobenius norm proposed in [113, 114]:

$$d^2(i,j) = \|\mathbf{R_1}(i,j) - \mathbf{R_2}(i + u_k + du, j + v_k + dv)\|_F^2, \tag{5.44}$$

and the Riemannian distance proposed in [115, 116]:

$$d^2(i,j) = \left\|\log(\mathbf{R_1}^{-1}(i,j)\mathbf{R_2}(i+u_k+du, j+v_k+dv))\right\|_F^2. \tag{5.45}$$

In order to optimize this function, we employ a Gauss-Newton framework. Considering the function $d(i,j,u_k,v_k,du,dv)$, for each iteration we find the update du, dv that minimizes the cost function, which requires the following linearization for the function d:

$$\begin{aligned} d(i,j,u_k,v_k,du,dv) \approx & d(i,j,u_k,v_k,0,0) + \frac{\partial}{\partial du} d(i,j,u_k,v_k,du,dv)|_{du=0} du \\ & + \frac{\partial}{\partial du} d(i,j,u_k,v_k,du,dv)|_{dv=0} dv, \end{aligned} \tag{5.46}$$

where the term $d(i,j,u_k,v_k,0,0)$ is equivalent to the term $I_t(i,j) = I_1(i,j) - I_2(i+u_k(i,j), j+v_k(i,j))$ in the algorithm described in Section 5.1.1. The term $\frac{\partial}{\partial du} d(i,j,u_k,v_k,du,dv)$ is similar to the image gradient I_{2x}, and can be estimated with finite differences, for example, with the following equation (and analogous for dv and I_{2y}):

$$\begin{aligned} \frac{\partial}{\partial du} d(i,j, & u_k,v_k,du,dv)|_{du=0} = \\ & \frac{1}{2}(d(i+1,j,u_k,v_k,0,0) - d(i-1,j,u_k,v_k,0,0)). \end{aligned} \tag{5.47}$$

The remaining of the algorithm is similar to the ones developed in Section 5.1.1. If one performs the same development, Eqs. (5.48) and (5.49) are replaced by:

$$\mathbf{b} = \begin{bmatrix} R_t(1,1) \\ \vdots \\ R_t(M,N) \end{bmatrix}_{T\times 1}, \tag{5.48}$$

and

$$\mathbf{J} = \begin{bmatrix} R_x(1,1) & & 0 & R_y(1,1) & & 0 \\ & \ddots & & & \ddots & \\ 0 & & R_x(M,N) & 0 & & R_y(M,N) \end{bmatrix}_{T\times 2T}, \tag{5.49}$$

where

$$R_t(i,j) = d(i,j,u_k,v_k,0,0), \tag{5.50}$$

$$R_x(i,j) = \frac{\partial}{\partial du} d(i,j,u_k,v_k,du,dv)|_{du=0}, \tag{5.51}$$

and

$$R_y(i,j) = \frac{\partial}{\partial dv} d(i,j,u_k,v_k,du,dv)|_{dv=0}. \tag{5.52}$$

5.4 Experimental Results

The algorithms described in this chapter were tested. For this study, different data terms were compared. The results obtained with grayscale images, which are common in the literature, are compared to the ones obtained with the features defined in Section 5.2.

In addition, a common practice to gain robustness against illumination changes [52, 53] is to apply a Rudin–Osher–Fatemi (ROF) [117] structure texture decomposition method in the input images, which was also considered in the experiments that follow. For this decomposition, the structure component is obtained through an optimization procedure that estimates a denoised version of an image I:

$$\hat{S} = \arg\min_{S} \|\nabla S\|_1 + \lambda \|S - I\|_2^2, \tag{5.53}$$

where $\hat{S}$ is the estimated structure component. The texture part is estimated as the image with the structure removed:

$$\hat{T} = I - \hat{S}. \tag{5.54}$$

The pre-processed input image is defined as a linear combination of the texture and structure components:

$$\hat{I} = \frac{(k\hat{T} + \hat{S})}{k+1}, \tag{5.55}$$

where k is defined as 20 according to [53].

5.4.1 Middlebury Dataset

The Middlebury dataset [112] is a widely used dataset to benchmark optical flow algorithms. It is composed of real videos captured in a controlled environment containing either a non-rigid motion or sequences adapted from stereo matching and synthetic images with a realistic scenario. For the real sequences, a hidden fluorescent texture was painted on the objects, and a tracking of those features provide ground truth for the motion field. The sequences are divided into two sets, the training set, for which the ground truth is provided, and the test set, which is used to create a rank among the methods since the ground truth is hidden. Two consecutive frames for the sequences from the training set are shown in Figs. 5.11 and 5.12. For these sequences, the motion between consecutive frames is at most

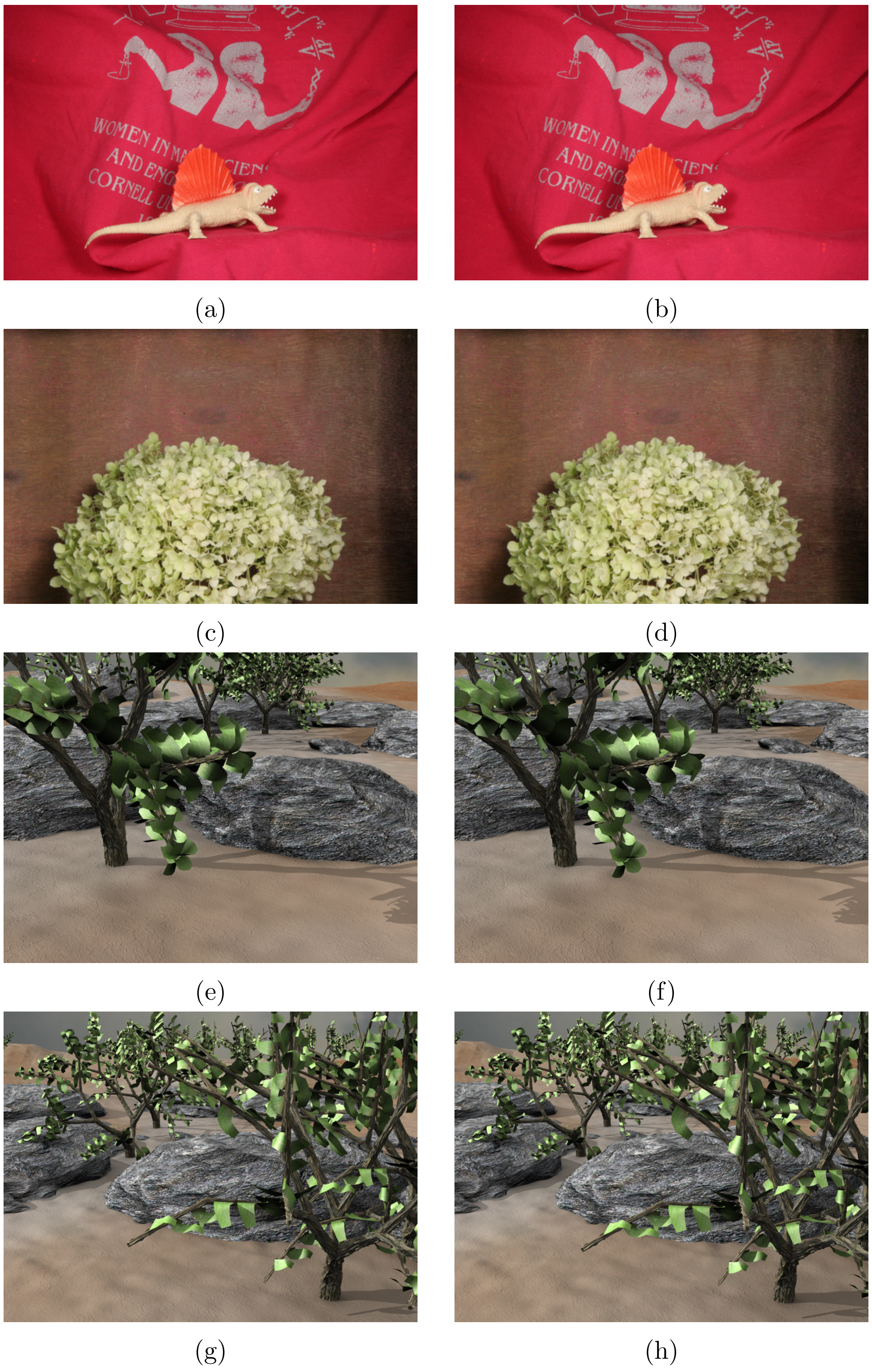


Figure 5.11: Sequences from the Middlebury training dataset - part 1. (a) and (b) Dimetrodon, frames 10 and 11. (c) and (d) Hydrangea, frames 10 and 11. (e) and (f) Grove2, frames 10 and 11. (g) and (h) Grove3, frames 10 and 11.

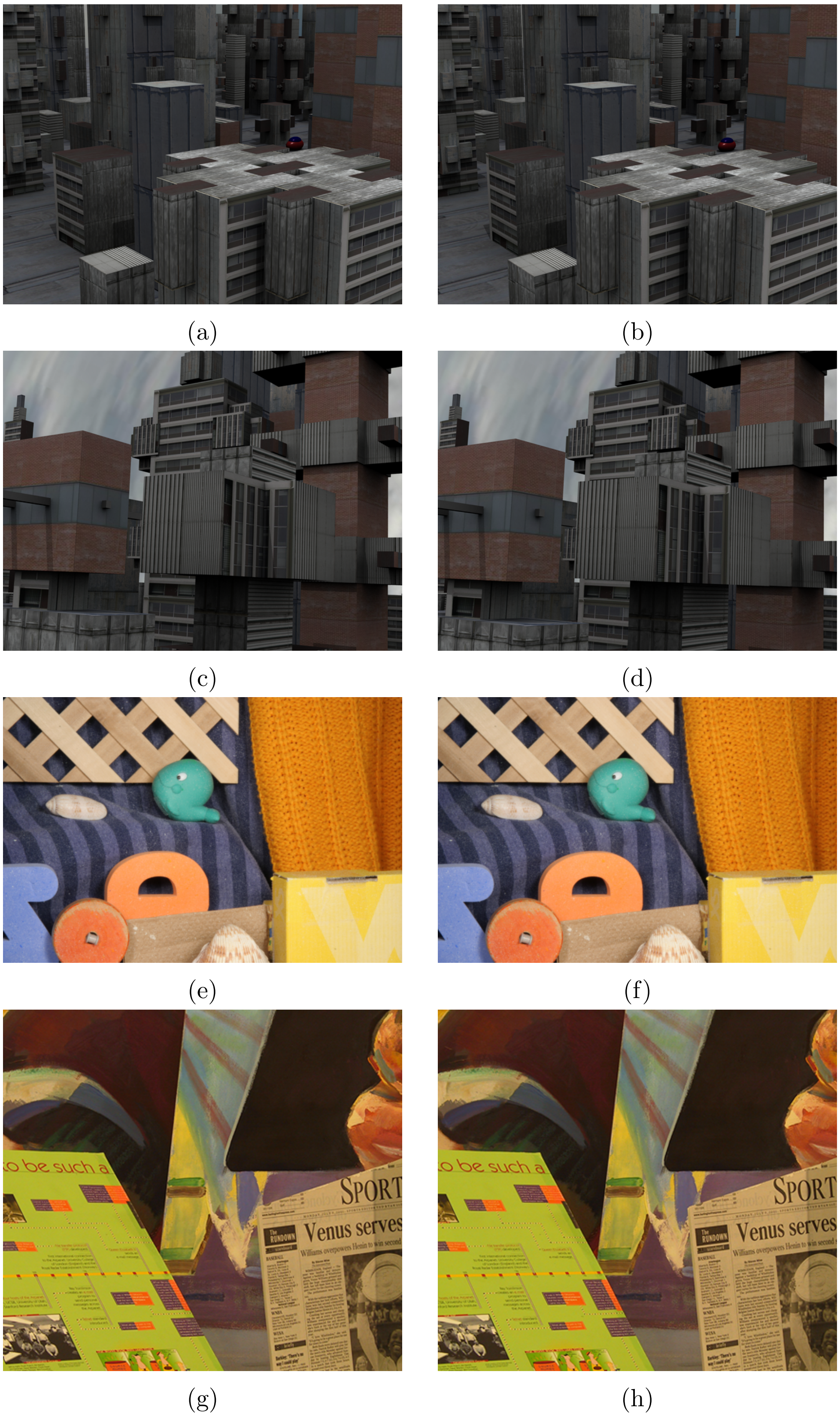


Figure 5.12: Sequences from the Middlebury training dataset - part 2. (a) and (b) Urban2, frames 10 and 11. (c) and (d) Urban3, frames 10 and 11. (e) and (f) RubberWhale, frames 10 and 11. (g) and (h) Venus, frames 10 and 11.

around 20 pixels, which is less than 10% of the image dimensions. However, in order for the linearization steps of the algorithms to still be valid, a multiscale approach may be necessary.

Several error metrics are evaluated, with the main ones being the average angular error (AAE) [118] and the average end-point error (EPE) [119]. The average angular error is computed by measuring for each pixel the angular distance between the ground truth and the estimated flow, while for the average endpoint error the Euclidian distance between ground truth and the estimated flow for each pixel is computed. In both cases, a mean value for the whole image is determined.

This database also defines a color-based coding for visualization of the flow fields. For each displacement vector, the orientation is assigned to a hue value and the magnitude is associated with a saturation value, and the corresponding RGB image is displayed. The ground truth for the motion between frames 10 and 11 of the sequences displayed in Figs. 5.11 and 5.12 is shown in Fig. 5.13.

5.4.2 Quadratic Formulation

The optical flow algorithm depicted in Alg. 9 was analyzed. It uses a quadratic function in the data term and regularization term, being closer to the original formulation of Horn and Schunk.

For each case, the algorithms were tested in the Middlebury training sequences shown in Figs. 5.11 and 5.12 with several values for the regularization parameter λ (see Eq. (5.2)), and the best result was selected.

In Tab. 5.1, one can see the results for the EPE between the flow obtained with each method from Sections 5.1, 5.2, 5.3, and the ground truth. The flow obtained for the sequence Grove2 using the various methods is displayed in Fig. 5.14. The methods based on brightness information, in addition to being simple, provide slightly better results than the other methods, which could be explained by the fact that they provide a significant amount of information without introducing further non-linearities.

5.4.3 Robust Function

In this section, the experiments from Section 5.4.2 were performed replacing the quadratic term by a non-quadratic function in the optical flow algorithm, in this case, the Charbonier function [55], which is one of the most used penalty functions in the optical flow computation [50]. For this experiment, Alg. 10 was used (see Section 5.1.2), however, the weighted median filter (steps 14-17) was removed and replaced with the simple median filter used in the steps 12-14 of Alg. 9. For this

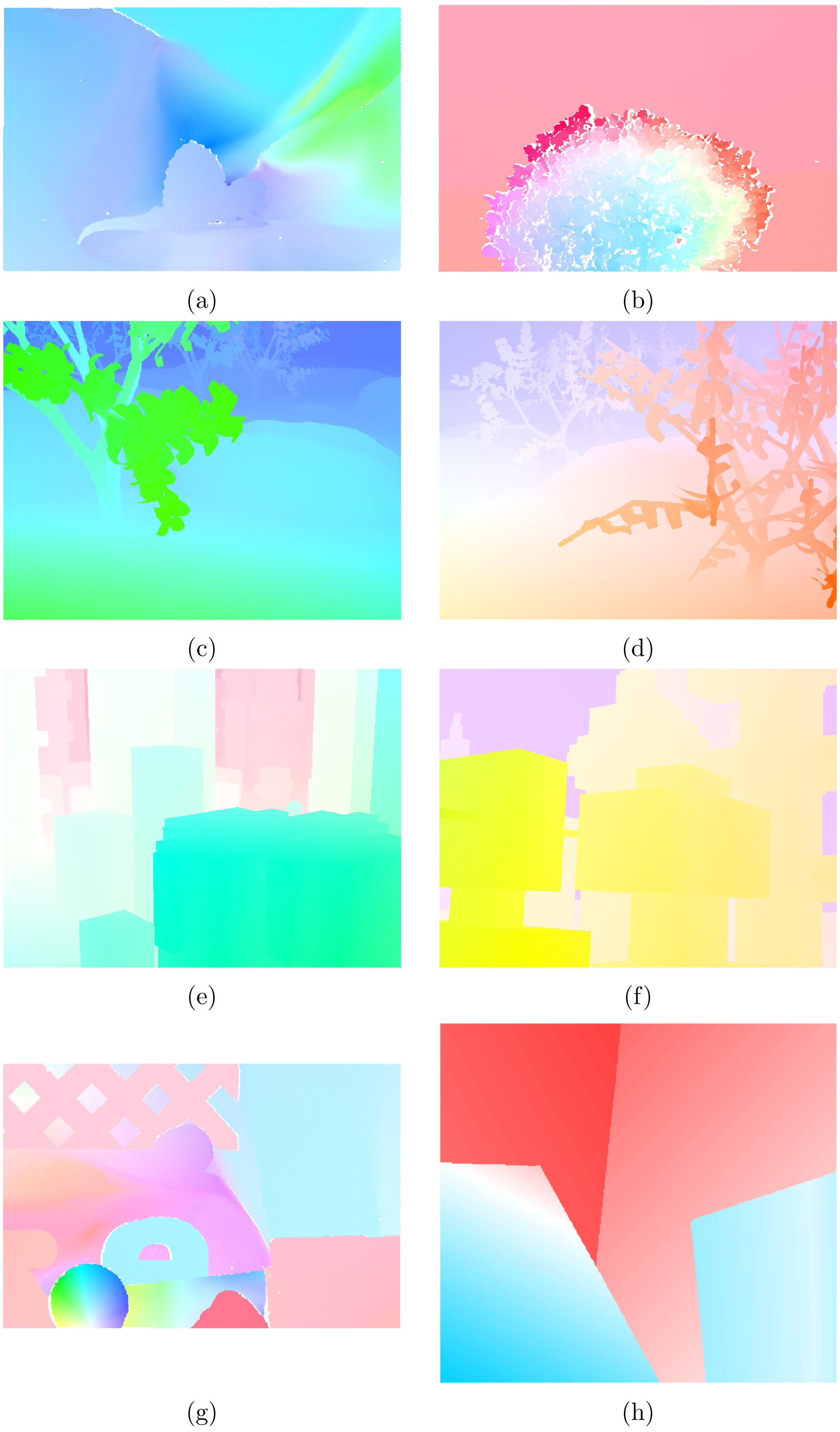


Figure 5.13: Ground truth for the sequences from the Middlebury training dataset. The motion field is coded into an RGB image by mapping, for each pixel, the orientation of the motion vector for that pixel to the hue of the color, and the magnitude of the motion vector to the saturation of the color, according to [112]. (a) Dimetrodon. (b) Hydrangea. (c) Grove2. (d) Grove3. (e) Urban2. (f) Urban3. (g) RubberWhale. (h) Venus.

Table 5.1: Average end-point error (EPE) on the Middlebury training dataset for different data terms and a quadratic penalty function. 1- Dimetrodon. 2- Grove2. 3- Grove3. 4- Hydrangea. 5- RubberWhale. 6- Urban2. 7- Urban3. 8- Venus.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Average**
Brightness (Section 5.1)	0.24	0.19	0.63	0.26	0.16	0.54	0.73	0.34	**0.38**
Brightness with ROF decomposition (Eqs. (5.53),(5.54) and (5.55))	0.23	0.20	0.68	0.19	0.12	0.48	0.84	0.34	**0.39**
Color tensor (Section 5.2.1)	0.17	0.22	0.70	0.22	0.18	0.66	0.83	0.55	**0.44**
Proposed color tensor (Section 5.2.2)	0.16	0.19	0.64	0.24	0.17	0.44	0.75	0.49	**0.39**
Structure tensor (Section 5.2.3)	0.30	0.39	0.94	0.31	0.19	1.27	1.34	0.48	**0.65**
Image descriptor (Section 5.2.4)	0.28	0.32	0.81	0.21	0.15	0.60	1.10	0.37	**0.48**
Proposed image descriptor (Section 5.2.5)	0.30	0.30	0.80	0.21	0.16	0.59	0.95	0.40	**0.46**
Structure Tensor derived from image descriptor (Section 5.2.5)	0.31	0.31	0.82	0.22	0.17	0.58	0.90	0.42	**0.46**

algorithm, it was also required to configure the parameter λ. To this end, several values were tested and the best result for each case was selected.

Tab. 5.2 shows the results obtained with the version of the algorithm using the Charbonnier penalty and the flow obtained for the *Grove2* sequence is shown in Fig. 5.15. One can notice that all methods benefit from having the Charbonnier penalty, and the results are consistently better than the ones seen in Tab. 5.1. For this case, the methods based on brightness information yield once again the best results.

5.4.4 Improved Regularization

This section shows results for the same set of experiments performed in Section 5.4.3, replacing the median filter used in the steps 12-14 of Alg. 9 by the weighted median filter (Eq. 5.32), therefore using the improved algorithm defined in Alg. 10. This should in principle generate an optical flow algorithm capable of handling occlusions and outliers, due to the robust function, and a regularisation able to simultaneously adapt the smoothing effect according to the image content, due to the non-local weighted term, and the flow contrast, due to the robust function.

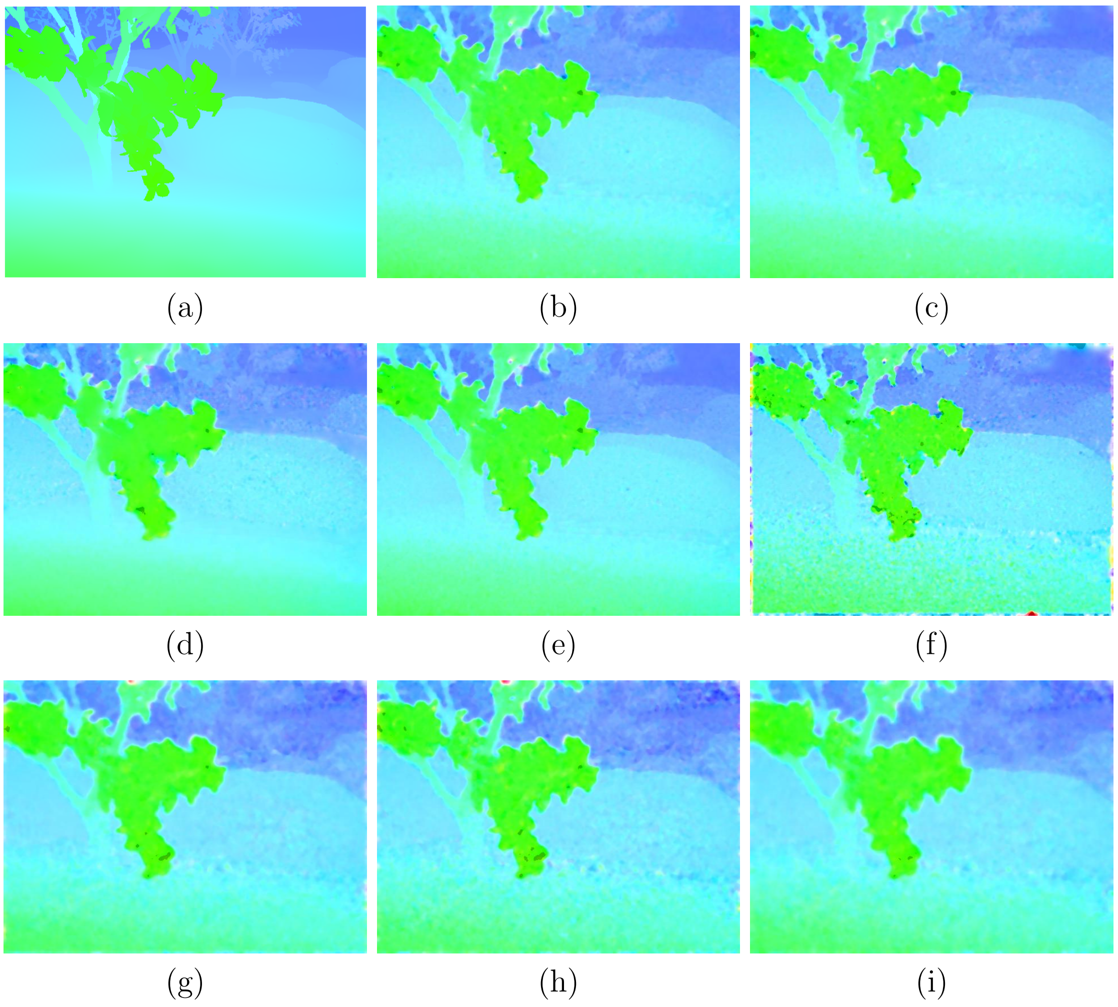


Figure 5.14: Example of the optical flow computation for the image *Grove2* in the Middlebury database using a quadratic penalty function. (a) Ground truth. Results obtained with them methods based on: (b) brightness (c) brightness with ROF decomposition, (d) color tensor, (e) proposed color tensor, (f) structure tensor, (g) image descriptor, (h) proposed image descriptor, (i) structure tensor derived from the image descriptor.

In Tab. 5.3 one can see the results obtained for each method, and an example of the flow obtained for the sequence *Grove2* is shown in Fig. 5.16. For this case, the methods based on the brightness information are once again better than the other methods, but now with less intensity. However, they have the drawback of not being robust to some forms of illumination changes, which will be demonstrated through in the experiments in the following subsections.

It is also important to notice from the summary in Tab. 5.4 that the proposed modifications consistently yield slight improvements in relation to the original ones. For the one based on the color tensor, the proposed modifications reduce the non-linearities of the color maps, which tends to reduce the errors due to the performed linearizations. For the one based on image descriptors, the proposed modifications

Table 5.2: Average end-point error (EPE) on the Middlebury training dataset for different data terms using the Charbonnier penalty [55]. 1- Dimetrodon. 2- Grove2. 3- Grove3. 4- Hydrangea. 5- RubberWhale. 6- Urban2. 7- Urban3. 8- Venus.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Average**
Brightness (Section 5.1)	0.17	0.15	0.58	0.21	0.14	0.36	0.43	0.26	**0.29**
Brightness with ROF decomposition (Eqs. (5.53),(5.54) and (5.55))	0.15	0.16	0.64	0.16	0.09	0.34	0.55	0.28	**0.30**
Color tensor (Section 5.2.1)	0.15	0.17	0.61	0.18	0.13	0.51	0.55	0.45	**0.34**
Proposed color tensor (Section 5.2.2)	0.13	0.15	0.57	0.22	0.14	0.39	0.51	0.36	**0.31**
Structure tensor (Section 5.2.3)	0.17	0.28	0.86	0.24	0.14	1.05	1.10	0.40	**0.53**
Image descriptor (Section 5.2.4)	0.20	0.27	0.78	0.18	0.12	0.45	0.83	0.28	**0.39**
Proposed image descriptor (Section 5.2.5)	0.21	0.23	0.71	0.17	0.10	0.44	0.74	0.29	**0.36**
Structure Tensor derived from image descriptor (Section 5.2.5)	0.19	0.25	0.78	0.17	0.12	0.44	0.62	0.29	**0.36**

reduce the redundancy in the components of the descriptor and provide them with more information, which tends to improve its results.

5.4.5 Illumination

In order to evaluate the influence of illumination changes, we simulate illumination changes as proposed in [61, 120]. For each image pair, the intensities from the second image are transformed as follows:

$$\hat{I}_2 = 255m\left(\frac{I_2}{255}\right)^{\gamma} + a, \tag{5.56}$$

where the parameter m represents a multiplicative term, the parameter a represents an additive term, and the parameter γ represents a gamma correction. Fig. 5.17 shows the impact of each of these three perturbations in the final image.

In this experiment, we vary the values of m, a and γ, and test the performance of the flow estimation. It was noticed that only the information based on the image descriptor are robust to all three types of illumination changes, and the algorithms based on the structure tensor and the brightness using ROF decomposition present

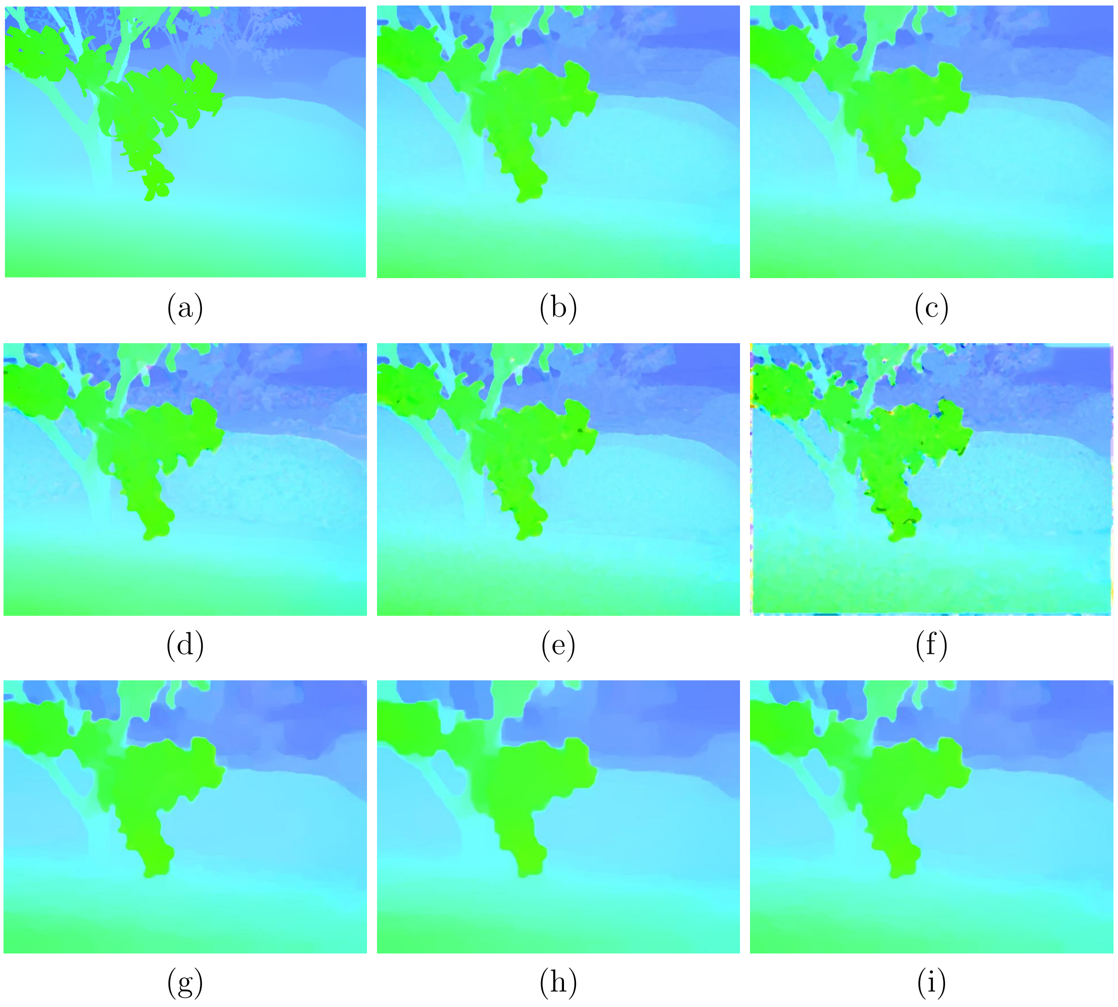

(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 5.15: Example of the optical flow computation for the image *Grove2* in the Middlebury database using the Charbonnier penalty [55]. (a) Ground truth. Results obtained with the methods based on: (b) brightness (c) brightness with ROF decomposition, (d) color tensor, (e) proposed color tensor, (f) structure tensor, (g) image descriptor, (h) proposed image descriptor, (i) structure tensor derived from the image descriptor.

robustness to some of the changes. In Figs. 5.18 to 5.20, we omit the results for the algorithm using brightness information and the color tensor, since they do not show any robustness to illumination changes.

In Fig. 5.18, one can see the EPE obtained when including an additive term in one of the images. The methods have shown strong robustness to it, since it is easier to compensate the difference in the mean values of the images. The method based on the ROF decomposition starts to fail from a given brightness threshold. As can be seen from Eqs. (5.53), (5.54) and (5.55), this method decomposes the image into a component of texture, which only contains high frequency content and should be immune to the additive term, and a component of structure, which is similar to a denoised version of the image, and therefore contains the additive term. The

Table 5.3: Average end-point error (EPE) on the Middlebury training dataset for different data terms using the Charbonnier penalty [55] and the improved regularization of [50]. 1- Dimetrodon. 2- Grove2. 3- Grove3. 4- Hydrangea. 5- RubberWhale. 6- Urban2. 7- Urban3. 8- Venus.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Average**
Brightness (Section 5.1)	0.16	0.10	0.40	0.21	0.12	0.23	0.30	0.22	**0.22**
Brightness with ROF decomposition (Eqs. (5.53),(5.54) and (5.55))	0.13	0.10	0.47	0.15	0.07	0.22	0.38	0.24	**0.22**
Color tensor (Section 5.2.1)	0.14	0.12	0.47	0.18	0.12	0.40	0.45	0.47	**0.29**
Proposed color tensor (Section 5.2.2)	0.14	0.13	0.42	0.22	0.14	0.31	0.42	0.43	**0.28**
Structure tensor (Section 5.2.3)	0.16	0.24	0.78	0.21	0.13	0.56	0.95	0.35	**0.42**
Image descriptor (Section 5.2.4)	0.16	0.20	0.67	0.18	0.10	0.32	0.74	0.24	**0.33**
Proposed image descriptor (Section 5.2.5)	0.16	0.18	0.62	0.17	0.09	0.33	0.58	0.24	**0.30**
Structure Tensor derived from image descriptor (Section 5.2.5)	0.16	0.18	0.61	0.16	0.10	0.32	0.50	0.24	**0.28**

image used in the algorithm is a combination of the structure and texture images, therefore it is still influenced by the additive term, which is the reason why the algorithm starts to fail for a high value of a. The other methods use information based on image gradients in the algorithms, therefore it is expected that they should be immune to this type of illumination changes. The fluctuation in the curves are caused by the gradients of the image borders.

Fig. 5.19 shows the results when a multiplicative term is applied to one of the images. For this case, only the descriptor-based methods show EPEs that are robust to this parameter. This behavior is due to the fact that they include a normalization step (Eqs. (5.37) and (5.42)), which is responsible for removing the multiplicative term. The curves for the ROF decomposition and the structure tensor show that they do not present any robustness to this effect, since the error increases drastically for any value of m different than 1 (which represents the result shown in Tab. 5.3).

The results obtained with the inclusion of a gamma correction in one of the images is depicted in Fig. 5.20. Once again, the descriptor-based methods are the only ones with some level of robustness to this effect. The EPE for the methods based on the ROF decomposition and the structure tensor increases for values of γ

Table 5.4: Average end-point error (EPE) on the Middlebury training dataset summarizing the results from Tabs.5.1, 5.2, and 5.3.

	Quadratic function	**Robust function**	**Improved regularization**
Brightness (Section 5.1)	0.38	0.29	0.22
Brightness with ROF decomposition (Eqs. (5.53),(5.54) and (5.55))	0.39	0.30	0.22
Color tensor (Section 5.2.1)	0.44	0.34	0.29
Proposed color tensor (Section 5.2.2)	0.39	0.31	0.28
Structure tensor (Section 5.2.3)	0.65	0.53	0.42
Image descriptor (Section 5.2.4)	0.48	0.39	0.33
Proposed image descriptor (Section 5.2.5)	0.46	0.36	0.30
Structure Tensor derived from image descriptor (Section 5.2.5)	0.46	0.36	0.28

different than 1 (where the result for $\gamma = 1$ is the same as the one shown in Tab. 5.3).

5.4.6 Results on VDAO Database and DORIS Videos

The performance of the studied methods was analyzed in a real application. In the object detection framework such as the one embedded in the DORIS system (see Section 1.2.1), it is often necessary to perform a spatial alignment, whether to align frames from synchronized videos, whether to align frames from the same video.

In this section, we analyze the performance of the optical flow algorithm in the alignment of synchronized videos. Two possible problems can appear, as described in Section 3.4.3. The camera can shake between different recordings, creating differences in the camera field of view, shown in Fig. 3.4. Another common problem is the recording at different times of the day, which creates a difference in the illumination, as seen in Fig. 3.5. This shows the importance of defining illumination-robust methods.

We assess the optical flow algorithms by using them to align the pairs of images given by Figs. 3.4 and 3.5. We report results only for the methods that provided the

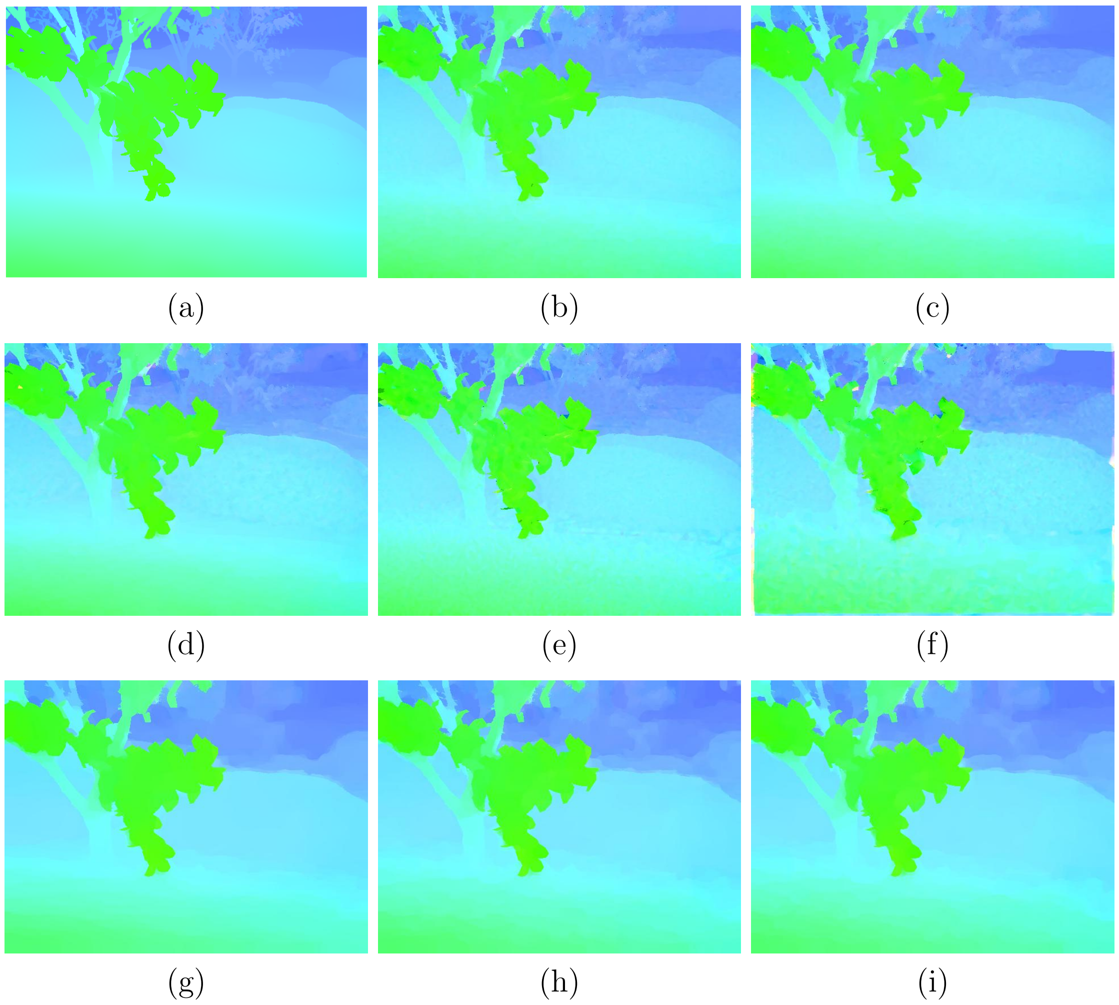


Figure 5.16: Example of the optical flow computation for the image *Grove2* in the Middlebury database using the Charbonnier penalty [55] and the improved regularization of [50]. (a) Ground truth. Results obtained with the methods based on: (b) brightness (c) brightness with ROF decomposition, (d) color tensor, (e) proposed color tensor, (f) structure tensor, (g) image descriptor, (h) proposed image descriptor, (i) structure tensor derived from the image descriptor.

best results for each approach, that is, we omit the ones based on the original color tensor and the original image descriptor. However, for these sequences, since there is no ground truth available, we can only measure the alignment error. After the flow computation, we warp the second image using the motion field, which compensates the displacement for each pixel and makes the resulting image similar to the first image. We perform a qualitative evaluation of the obtained motion field and the warped image, and them compute the mean square error (MSE) and structural similarity (SSIM) between the first image and the warped second image, removing the image borders. These objective metrics are not ideal, since they do not take into account the influence of occlusions, interpolation errors or difference of illumination, but can give a hint about the algorithm behavior.

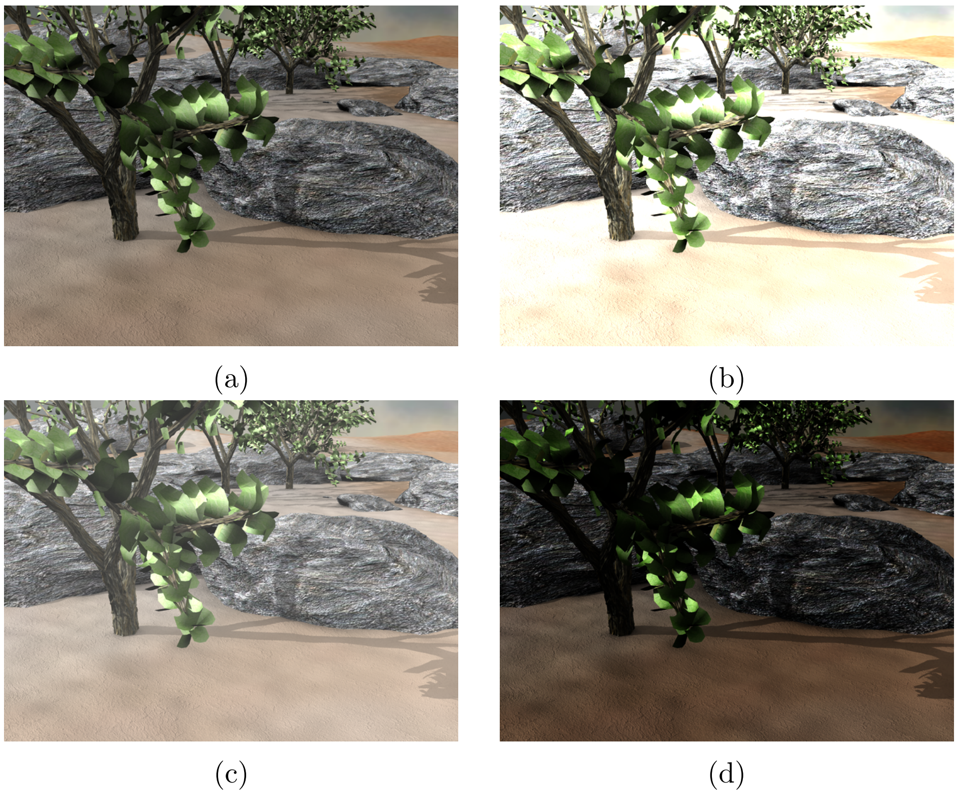


Figure 5.17: Example of illumination changes applied in the second image of the Grove2 pair. (a) Original image. (b) Multiplicative term. (c) Additive term. (d) Gamma correction.

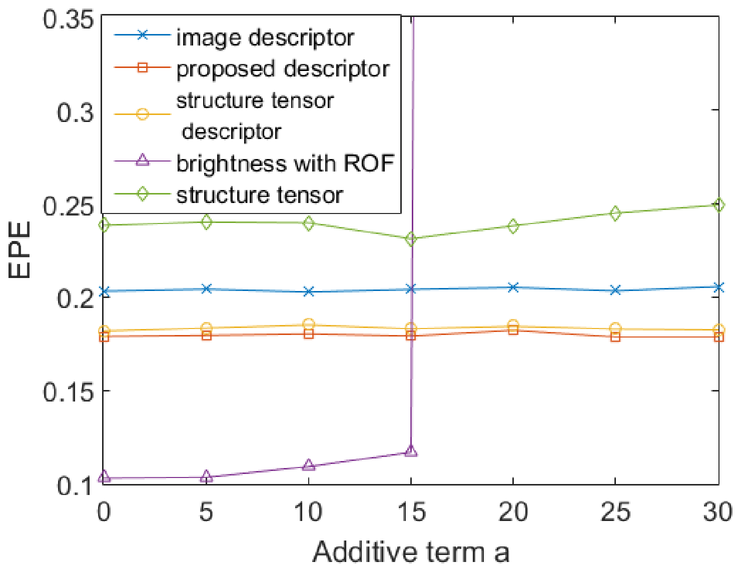


Figure 5.18: EPE for the optical flow algorithms with the inclusion of an additive term in one of the images.

Fig. 5.21 shows the estimated motion field for the images from the DORIS system with illumination changes. The flow is coded using the definition given in Subsection 5.4.1. For this pair of images, there is a small horizontal displacement, therefore

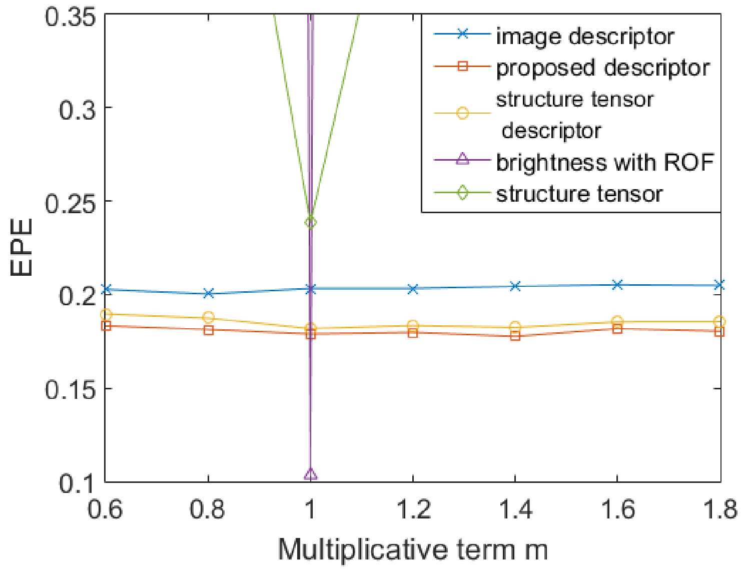


Figure 5.19: EPE for the optical flow algorithms with the inclusion of a multiplicative term in one of the images.

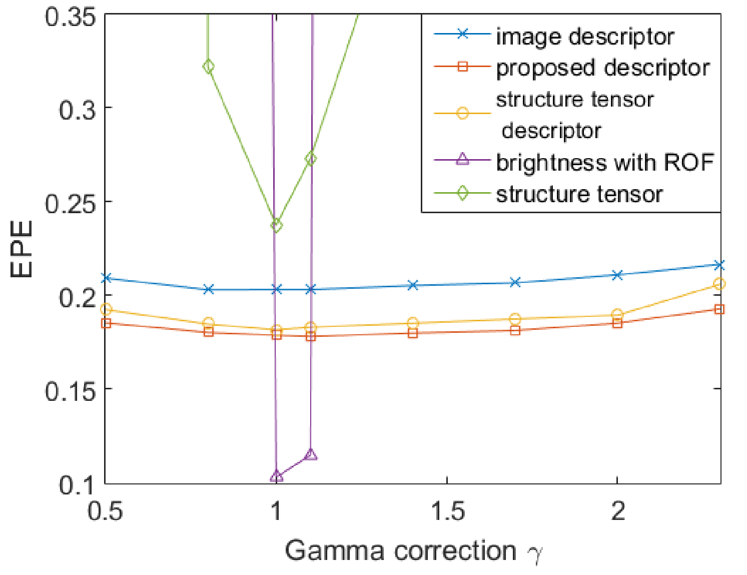


Figure 5.20: EPE for the optical flow algorithms with the inclusion of a gamma correction in one of the images.

it is expected that the flow for the whole image has the same horizontal orientation with a magnitude proportional to the scene depth, which translates to this color coding as a blue image with different levels of saturation.

One can notice that in Figs. 5.21(c)-(f) the estimated flow seems influenced by the difference of illumination in images. In the region where there is a difference of illumination in the lawn, the estimated flow has a different orientation (the color changes to purple or red) and a much higher magnitude (the color is highly saturated when compared to the pixels outside this region).

Comparing the results to the warped images shown in Fig. 5.22, one can conclude that the algorithm tries to compensate the motion of the shaded region, filling the lawn with illuminated pixels from outside the lawn. On the other hand, the results from Figs. 5.21(g) and (h), using the descriptors-based methods, show a motion that is more coherent, as expected, with little or no influence of the difference in illumination.

However, for the methods influenced by the illumination pattern, the error between the first image and the warped second image, shown in Tab. 5.5, is smaller. Even if they find a wrong displacement, they still create a warped image more similar to the first image in terms of magnitude.

Fig. 5.23 presents the motion field for the example of the VDAO database that shows a frame mismatch. In this example, since there is no evident difference in the lighting conditions, all methods have a similar performance, with the methods based on the brightness information apparently having a result that seems less corrupted by noise. The methods based on the structure tensor and the image descriptor, which use information related to the image gradients, seem to be more prone to errors in the borders, where the estimation of the image gradients is less precise.

However, one can notice that for the first image in Fig. 5.24(a) there is a small light source reflecting in pipes in the upper left section, which makes some methods fail to compute the flow in this region. Only the algorithms based on the image descriptor seems to be immune to this effect, and the method based on the structure tensor shows some reduction upon this effect. In Tab. 5.5, one can also see that the approach based on the structure tensor provides the lowest MSE, which is due to the fact that in this region with reflection in the upper right, the algorithm fills the pixels with a gray color that is closer to the light pattern.

Figs. 5.25 and 5.26 show an additional experiment containing frames from the VDAO database with a pipe partially occluding the scene. All methods completely fail in the occluded region, which is somewhat expected. The methods based on the image descriptor seem to be more influenced by the occlusion created by fast moving objects, since in Figs. 5.26(g) and (h) they fail to represent the moving pipe.

5.5 Summary

This chapter discussed the spatial alignment of images using the optical flow algorithm. The basic composition of the algorithm was presented and several variations were tested. The results showed that the traditional methods have a better performance with images without illumination changes, which is a problem that appears in the applications studied in this thesis. Modifications to some of the algorithms were proposed, and the results demonstrate that they provide a small increase in

Table 5.5: Alignment error for the examples given in Figs. 5.22, 5.24 and 5.26 for several methods. For each case, the motion field between two images is computed, the second image is warped to match the first and the MSE and SSIM metrics between the first image and the warped second image are computed, excluding the image borders.

	Light		Mismatch		Occlusion	
	MSE (10^3)	SSIM	MSE (10^3)	SSIM	MSE (10^3)	SSIM
Brightness (Section 5.1)	0.07	0.87	0.82	0.69	0.22	0.94
Brightness with ROF decomposition (Eqs. (5.53),(5.54) and (5.55))	0.07	0.87	0.72	0.66	0.39	0.93
Proposed color tensor (Section 5.2.2)	0.07	0.88	0.78	0.69	0.11	0.96
Structure tensor (Section 5.2.3)	0.04	0.87	1.01	0.57	0.09	0.95
Proposed image descriptor (Section 5.2.4)	0.07	0.86	2.81	0.61	0.70	0.91
Structure Tensor derived from image descriptor (Section 5.2.5)	0.12	0.84	2.93	0.58	0.55	0.91

performance.

Some qualitative tests were performed in frames from the DORIS system, which showed that despite not having the best overall performance, the approach based on the proposed image descriptor provides results with more robustness to the illumination changes that occur during a day, which are inevitable for the intended application.

The next chapter studies another application of computer vision techniques to estimate the position of dark fringes acquired using the experiment described in Section 1.2.3, which can be used to estimate properties of a given material.

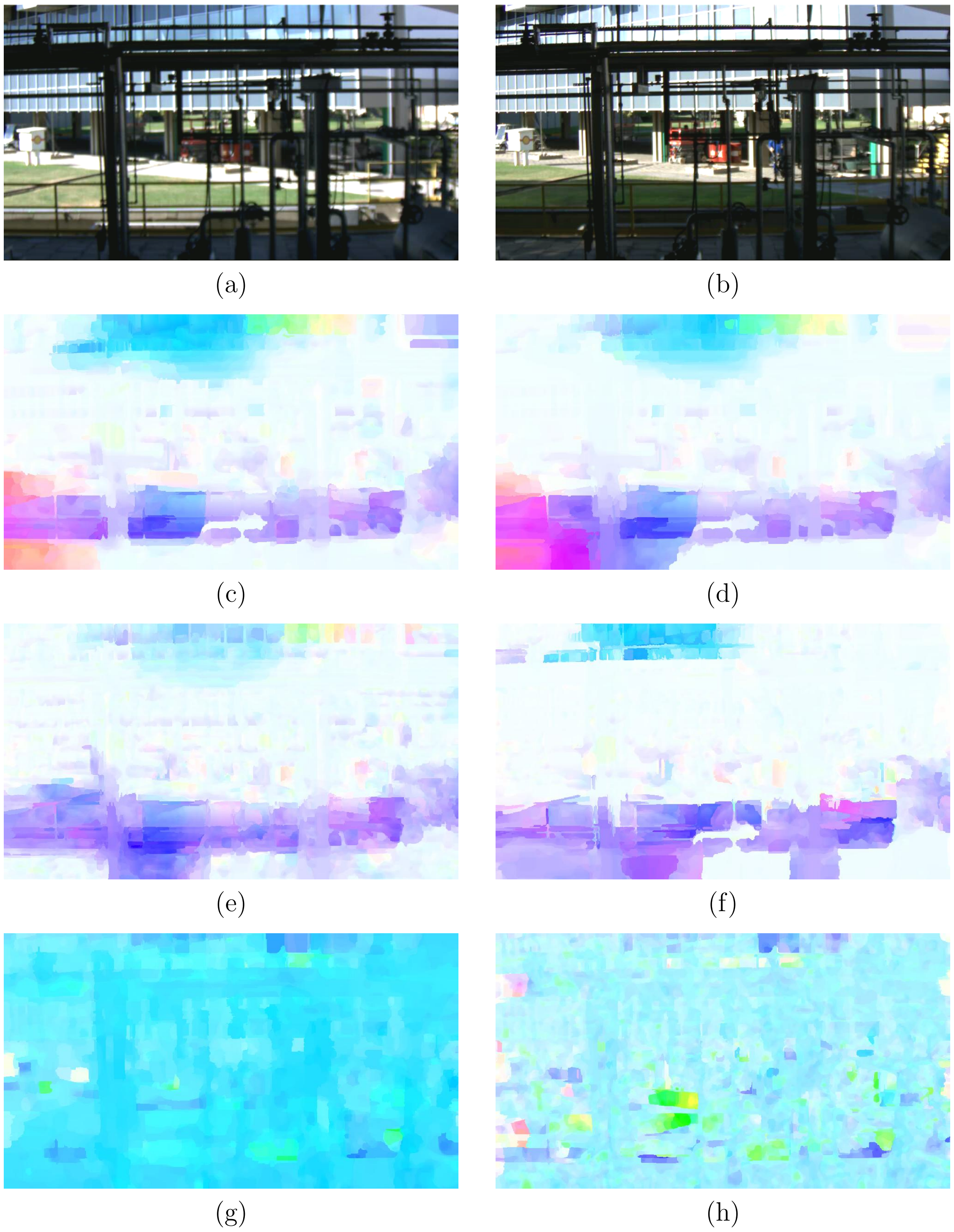


Figure 5.21: Example of the optical flow computation for two images with illumination changes in the DORIS system. In this part, there is a small displacement in the horizontal direction, so it is expected that the computed flow has the same orientation along the image, therefore should have be represented as an image with the same hue and different levels of saturation. (a) Original frame from the first sequence. (b) Original frame from the second sequence. Motion field obtained with: (c) brightness method, (d) brightness with ROF decomposition, (e) proposed color tensor, (f) structure tensor, (g) proposed image descriptor, (h) structure tensor derived from the image descriptor.

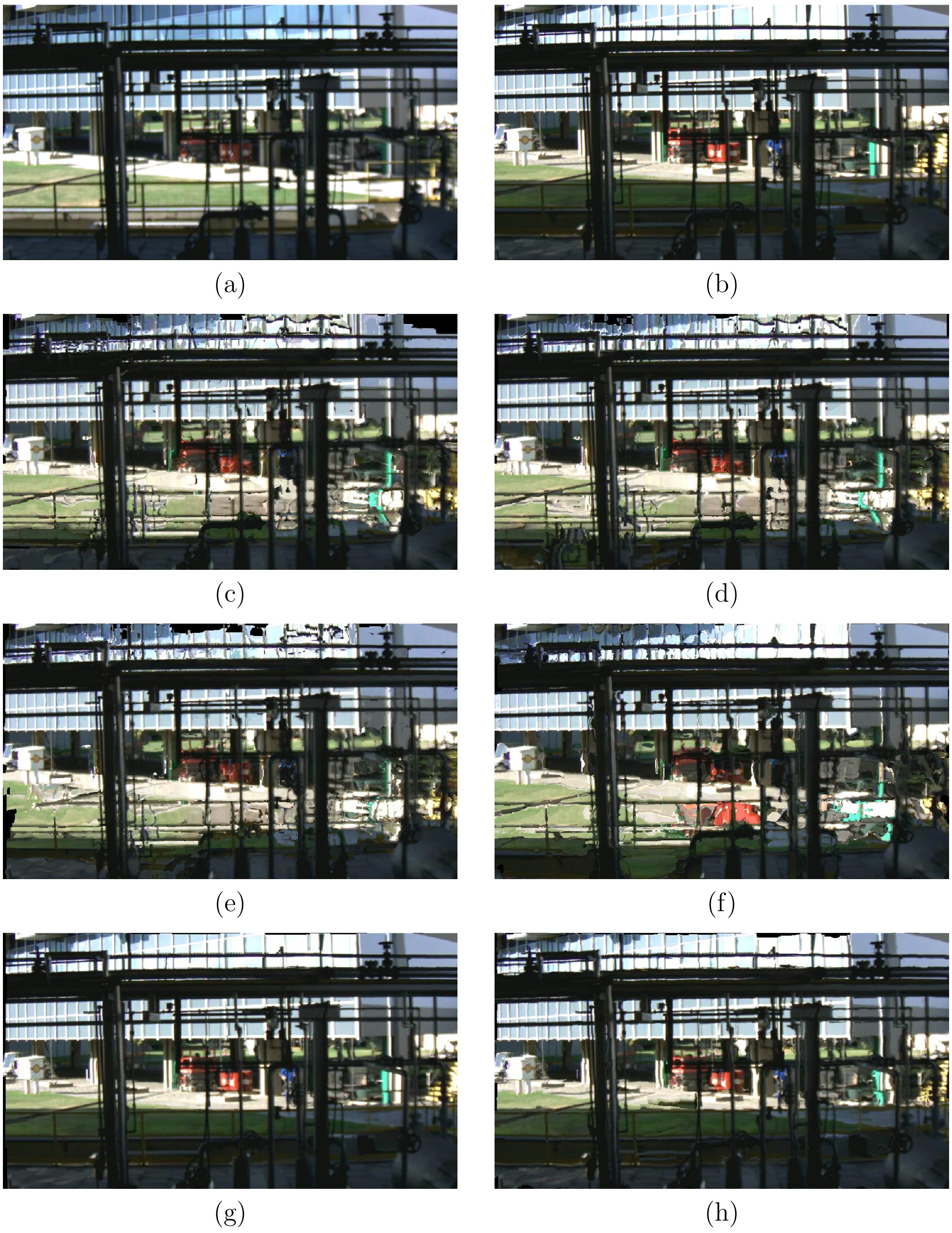


Figure 5.22: Example of the frame alignment for two images with illumination changes in the DORIS system. Using the estimated motion field shown in Fig. 5.21, one warps the image (b) to make it similar to the image (a). Warped images obtained with: (c) brightness method, (d) brightness with ROF decomposition, (e) proposed color tensor, (f) structure tensor, (g) proposed image descriptor, (h) structure tensor derived from the image descriptor.

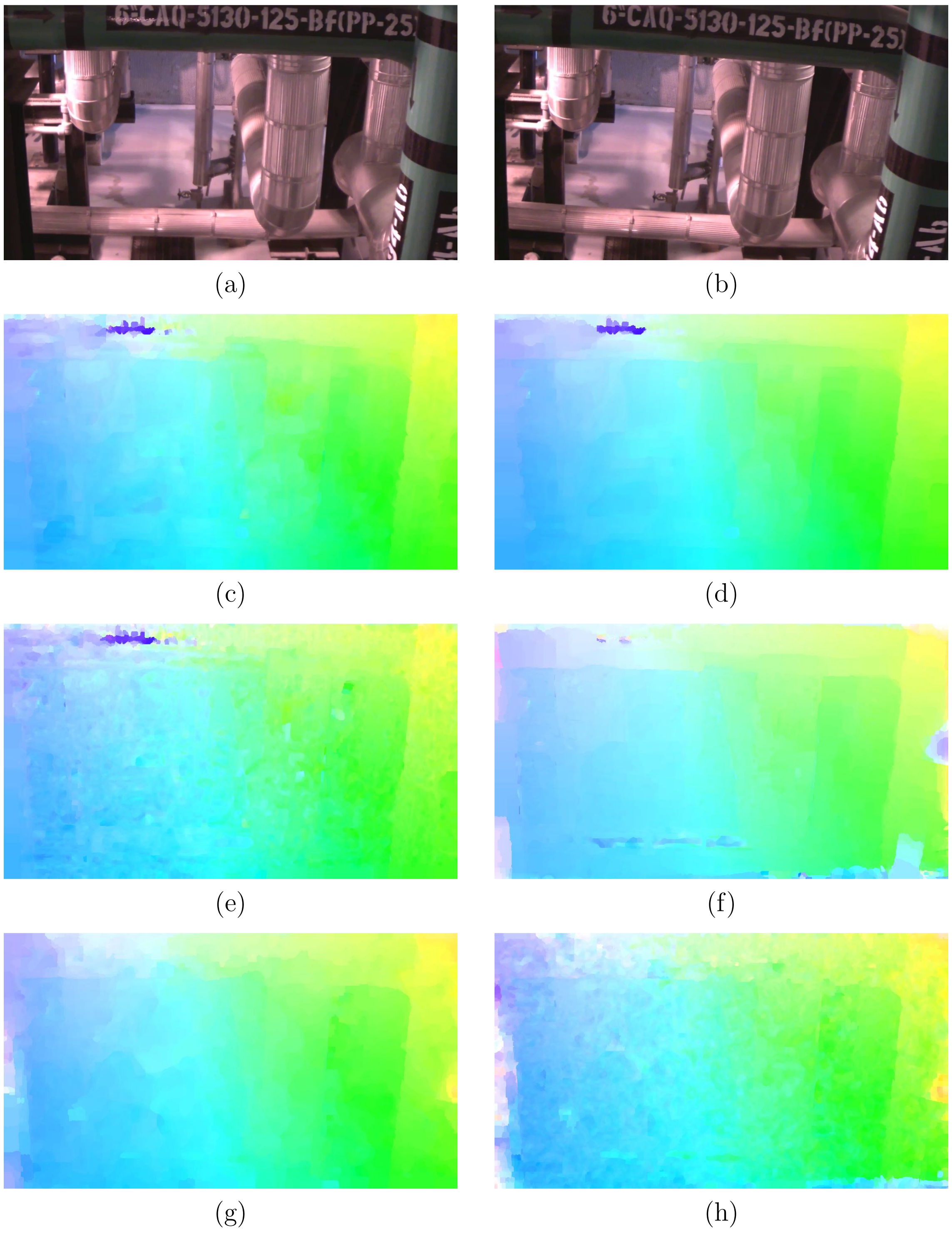


Figure 5.23: Example of the optical flow computation for two images with a frame mismatch in the VDAO database. For these sequences, there is a small displacement in the horizontal direction, so it is expected that the computed flow has the same orientation along the image. However, due to the frame mismatch, the images are also rotated with respect to each other. (a) Original frame from the first sequence. (b) Original frame from the second sequence. Motion field obtained with: (c) brightness method, (d) brightness with ROF decomposition, (e) proposed color tensor, (f) structure tensor, (g) proposed image descriptor, (h) structure tensor derived from the image descriptor.

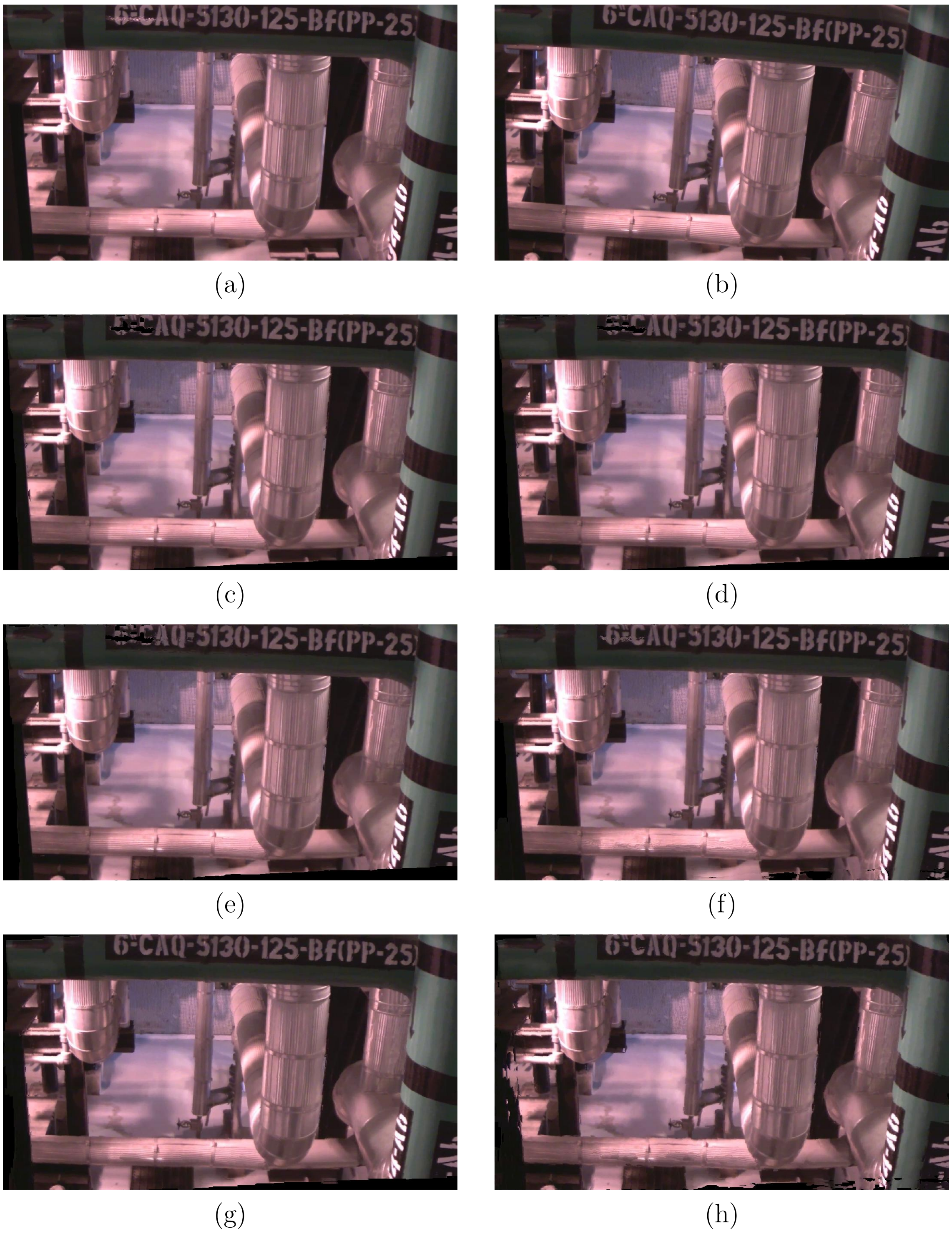


Figure 5.24: Example of the frame alignment for two images with a frame mismatch in the VDAO database. Using the estimated motion field shown in Fig. 5.23, one warps the image (b) to make it similar to the image (a). Warped images obtained with: (c) brightness method, (d) brightness with ROF decomposition, (e) proposed color tensor, (f) structure tensor, (g) proposed image descriptor, (h) structure tensor derived from the image descriptor.

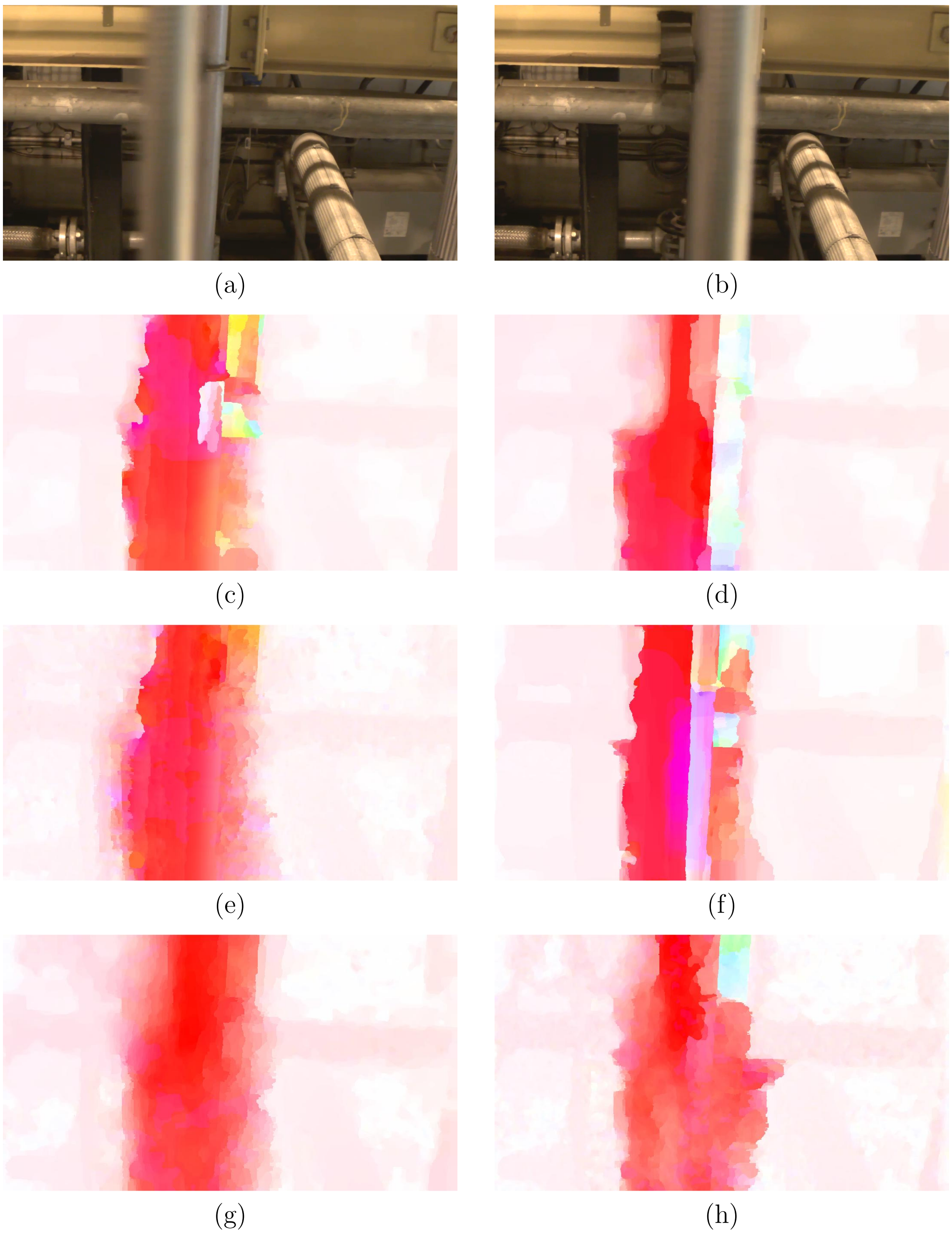


Figure 5.25: Example of the optical flow computation for two images with a frame occlusion in the VDAO database. For these sequences, there is a small displacement in the horizontal direction, so it is expected that the computed flow has the same orientation along the image. However, there is a pipe close to the image position creating an occlusion of the background. (a) Original frame from the first sequence. (b) Original frame from the second sequence. Motion field obtained with: (c) brightness method, (d) brightness with ROF decomposition, (e) proposed color tensor, (f) structure tensor, (g) proposed image descriptor, (h) structure tensor derived from the image descriptor.

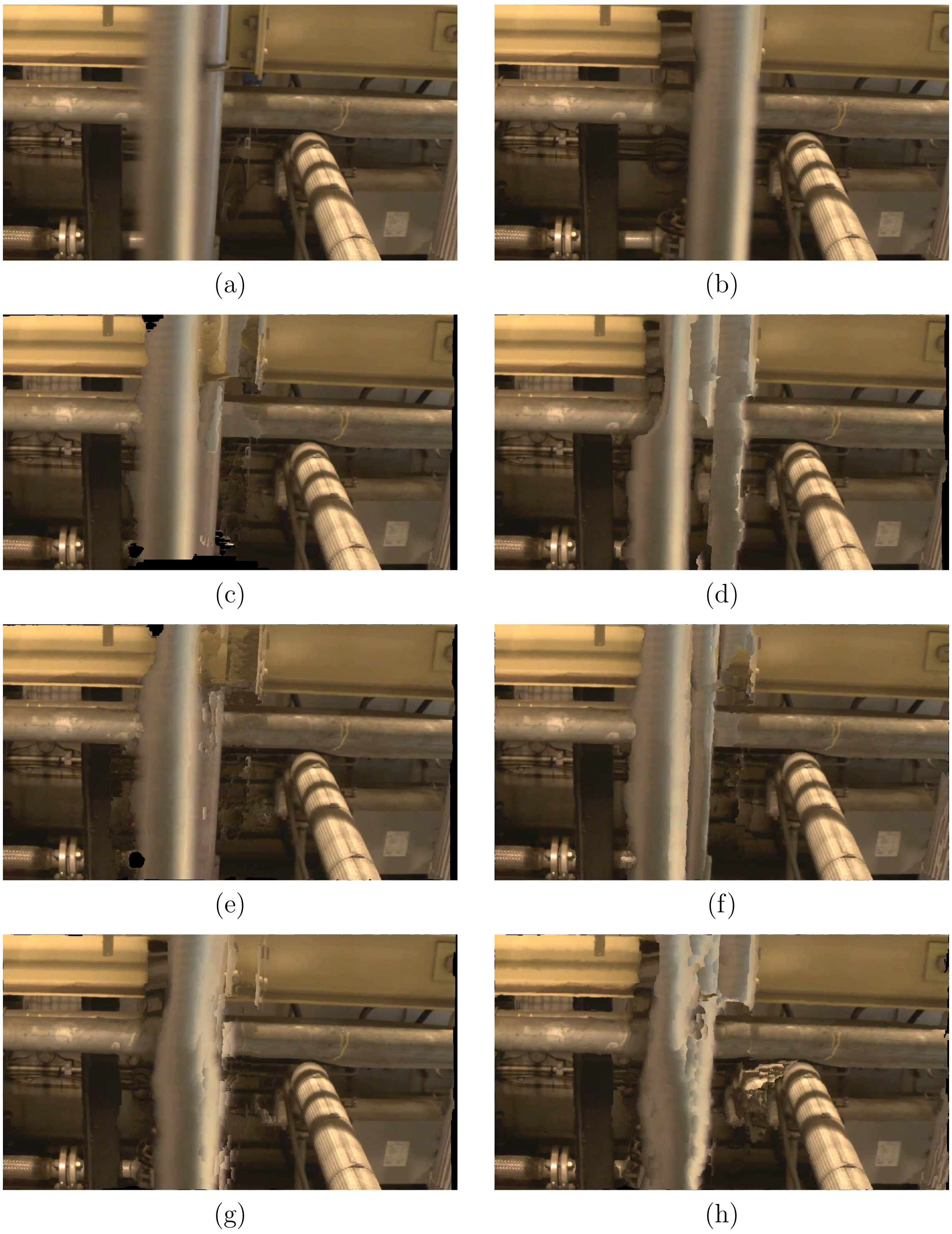


Figure 5.26: Example of the frame alignment for two images with a frame occlusion in the VDAO database.Using the estimated motion field shown in Fig. 5.23, one warps the image (b) to make it similar to the image (a). Warped images obtained with: (c) brightness method, (d) brightness with ROF decomposition, (e) proposed color tensor, (f) structure tensor, (g) proposed image descriptor, (h) structure tensor derived from the image descriptor.

Chapter 6

Polymer Characterization Using Mathematical Morphology

An interesting application to the detection of events is the experiment described in Section 1.2.3. In this application, a polymer flows inside a specific apparatus that induces a birefringence property on the fluid. With a camera placed at a proper location, it is possible to acquire images that show a pattern of light and dark fringes due to the birefringence property, and the understanding of this pattern provides information about the spatial evolution of the stress in the molten polymer. An example of an image acquired from this experiment can be seen in Fig. 6.1

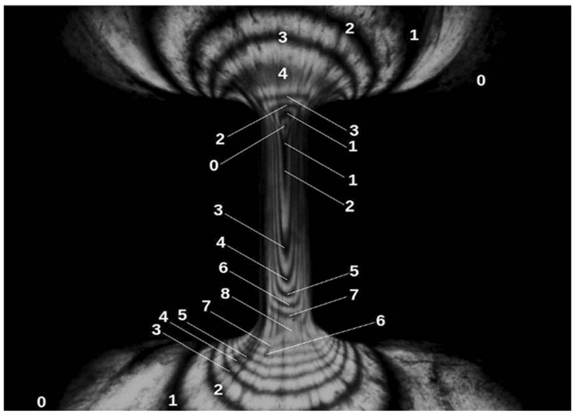


Figure 6.1: Example of image obtained in a flow-induced birefringence experiment. For each dark fringe is associated a fringe order k, according to [88], marked as the white numbers.

The patterns of bright and dark fringes can be converted into the principal stress difference (PSD) profile, which is an important property to characterize the polymer, using the semi-empirical stress optical rule [18, 19, 87, 88, 121, 122]. Along the flow centerline, the PSD is equal to the first normal stress difference, that is

$$|PSD| = |\sigma_{11} - \sigma_{22}| = |\tau_{11} - \tau_{22}| = \frac{k\lambda}{\mid C \mid d}, \tag{6.1}$$

where σ_{11} and σ_{22} are the first principal stress values, τ_{11} and τ_{22} are the first normal stress values, C is the stress optical coefficient, λ is the wavelength of the polarized light which hits the birefringent medium and d is the anisotropic medium length.

The variable $k = 0, 1, 2, \ldots$ is the fringe order and is represented as the white numbers in Fig. 6.1. Using Eq. (6.1) if the values of the parameters given by the experiment are known, the PSD profile along the centerline can be determined by knowing the relative retardation of the dark fringe order k, which is fixed for each experiment, and the position each fringe appears in the image, which must be estimated for each image. If one needs to assess the stress property on a large number of polymers under different conditions, it is imperative the development of methods capable of automatically identifying the occurrence of dark fringes in the images.

This work aims to present a novel approach using mathematical morphology techniques to identify the patterns of the dark fringes from the birefringence images presented in [88]. In addition, it also performs the automatic detection of the dark fringes centers, allowing the collection of measurements to be obtained in a faster and more accurate way. To introduce such methodology, the remaining of this paper is organized as follows: Section 6.1 describes techniques of mathematical morphology that were used for the development of the proposed algorithm. In Section 6.2, a step-by-step description of the fringe position detector based on mathematical morphology techniques is presented. Section 6.3 assesses the proposed methodology by comparing its results with the ones obtained with the GIMP software and with other methods adapted to the context of birefringence images [15, 18, 19, 88, 123].

6.1 Mathematical Morphology

Mathematical morphology is the study of the shape of spatial structures, often used to remove imperfections or identify patterns [124]. It provides tools to analyze the relationship between the pixel values of an image, considering their spatial ordering and a small template called structuring element. These interactions between an image and a structuring element are explored using two basic operations, namely erosion and dilation [125].

The erosion of an image $\mathbf{F}$ using an structuring element $\mathbf{B}$ can be defined as follows. One starts assuming that $\mathbf{F}$ and $\mathbf{B}$ exist in the same two-dimensional Euclidian space, where $\mathbf{F}$ is a grayscale image and $\mathbf{B}$ is a structuring element that is defined as a binary set.

The eroded value of $\mathbf{F}$ for each pixel $\mathbf{x}$ is the minimum intensity value inside the window delimited by the structuring element $\mathbf{B}$. If $\mathbf{b}$ is a translation belonging to the structuring element, the eroded image $\varepsilon_{\mathbf{B}}(\mathbf{F})$ is equal to:

$$\textit{Erosion:}\,[\varepsilon_{\mathbf{B}}(\mathbf{F})](\mathbf{x}) = \min_{\mathbf{b}\in\mathbf{B}} \mathbf{F}(\mathbf{x}+\mathbf{b}). \tag{6.2}$$

Similarly, the dilation operation of **F** by **B**, which is the dual operation of the erosion, is defined for each pixel **x** as the maximum value of the image intensity value in the window defined by the structuring element **B**, leading to:

$$\textit{Dilation:}\,[\delta_{\mathbf{B}}(\mathbf{F})](\mathbf{x}) = \max_{\mathbf{b}\in\mathbf{B}} \mathbf{F}(\mathbf{x}+\mathbf{b}). \tag{6.3}$$

Examples of the erosion and dilation operations are despicted, respectively, in Figs. 6.2(c) and 6.2(d), where it can be observed that the erosion shrinks the objects while the dilation enlarges them.

The erosion and dilation operations can be used to define more complex morphological operations, among which we can mention the opening and closing operations, widely used in morphological processing [125]. The opening consists of an erosion of **F** by the structuring element **B**, followed by a dilation using the same structuring element. The following equation describes it:

$$\textit{Opening:}\,\gamma_{\mathbf{B}}(\mathbf{F}) = \delta_{\mathbf{B}}[\varepsilon_{\mathbf{B}}(\mathbf{F})]. \tag{6.4}$$

The closing operation is the dual to the opening operation, and is performed by applying a dilation of **F** by the structuring element **B** followed by a erosion using the same structuring element. It is equivalent to applying the opening to the complement of the original image, and then taking the complement of the result, as shown in the following equation:

$$\textit{Closing:}\,\phi_{\mathbf{B}}(\mathbf{F}) = \varepsilon_{\mathbf{B}}[\delta_{\mathbf{B}}(\mathbf{F})]. \tag{6.5}$$

Figs. 6.2(e) and 6.2(f) show, respectively, examples for the opening and closing operation. One can see that the opening removes the image tips while the dilation fills the image holes (which is equivalent to removing the tips on the complement of the image).

6.1.1 Geodesic Morphology

The basic morphology operations consider only an input image and compute its relation with a structuring element. Geodesic transformations [125] consider instead two input images: the original input image, called marker, and a second image which acts as a mask. A morphological transformation is applied to the marker image, using a simple structuring element. However, if for a given position the result is bigger or smaller than the mask, depending on the operation, it is set as the value of the mask.

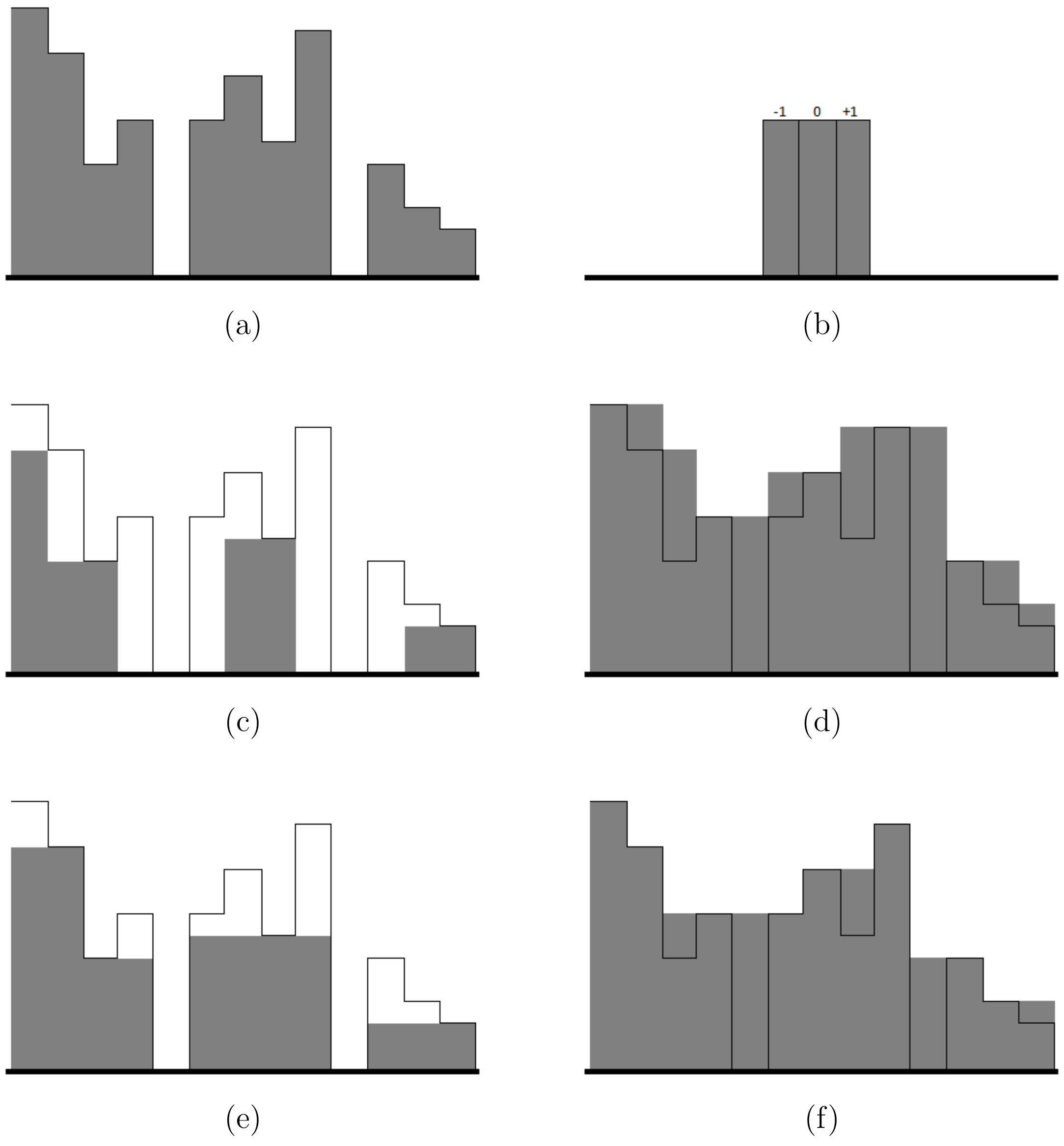


Figure 6.2: Erosion, dilation, opening and closing operations. (a) Original image. (b) Structuring element defining a window for the maximum and minimum operations, consisting of one pixel to the left, the current pixel and one pixel to the right. (c) Erosion: The object is reduced. (d) Dilation: The object is enlarged in the borders. (e) Opening: The object tips are removed, making it smaller. (f) Closing: The object holes are filled, making it bigger. In (c)-(f), the solid lines represent the values of the original image before the morphological operations and the gray rectangles show the result after each operation.

A geodesic erosion of size 1 consists of a morphological erosion of a marker image **F** using an elementary isotropic structuring element **B**, such as the one given by Fig. 6.2(b), followed by a pointwise maximum between the result and the mask **G**, as follows:

$$\textit{Geodesic erosion:}\, \varepsilon_G^{(1)}(\mathbf{F}) = \varepsilon^{(1)}(\mathbf{F}) \vee \mathbf{G}, \tag{6.6}$$

where $a \vee b$ represents the maximum value between a and b. This operation is despicted in Figs. 6.3(d) and 6.3(e), which show that the geodesic erosion uses the mask to limit the shrinkage effect caused by the erosion (see Fig. 6.2(c)).

A geodesic dilation of size 1 consists of a morphological dilation of a marker image **F** using an elementary structuring element **B**, followed by a pointwise minimum between the result and the mask **G**, as follows:

$$\textit{Geodesic dilation:}\, \delta_G^{(1)}(\mathbf{F}) = \delta^{(1)}(\mathbf{F}) \wedge \mathbf{G}, \tag{6.7}$$

where $a \wedge b$ represents the minimum value between a and b. An example of this operation is shown in Figs. 6.3(b) and 6.3(c), where one can see that, when compared to the dilation seen in Fig. 6.2(d), the geodesic dilation forces the dilation process to propagate the image only to regions delimited by the mask.

6.2 Processing of Birefringence Images Using Mathematical Morphology Techniques

In this chapter, we propose a method to automatically detect the dark fringes in the birefringence patterns, which is an improvement to the four-step approach that was introduced in [86]. This approach differs from other, more traditional, methods as it works strictly on the geometrical domain, avoiding the use of more complex tools that can also be more error-prone, such as frequency-domain techniques and transformations [74, 76, 84]. A block diagram summarizing the workflow of the proposed method is presented in Fig. 6.4. The first step consists of image enhancement that removes some of the image noise, providing a sharper version of the input image that allows a more precise detection in the subsequent steps. In the second step, the minima contours are detected through the application of the watershed [126] method, further detailed. A skeleton [127] of the dark fringes is created from the previous-step output and the center of the image, which is the region of interest for the fringe position detection, is selected in the third step. Finally, in step four, post-processing is performed to increase the precision on the dark-fringe final detection. A detailed explanation of each of those steps is given in the subsections that follow.

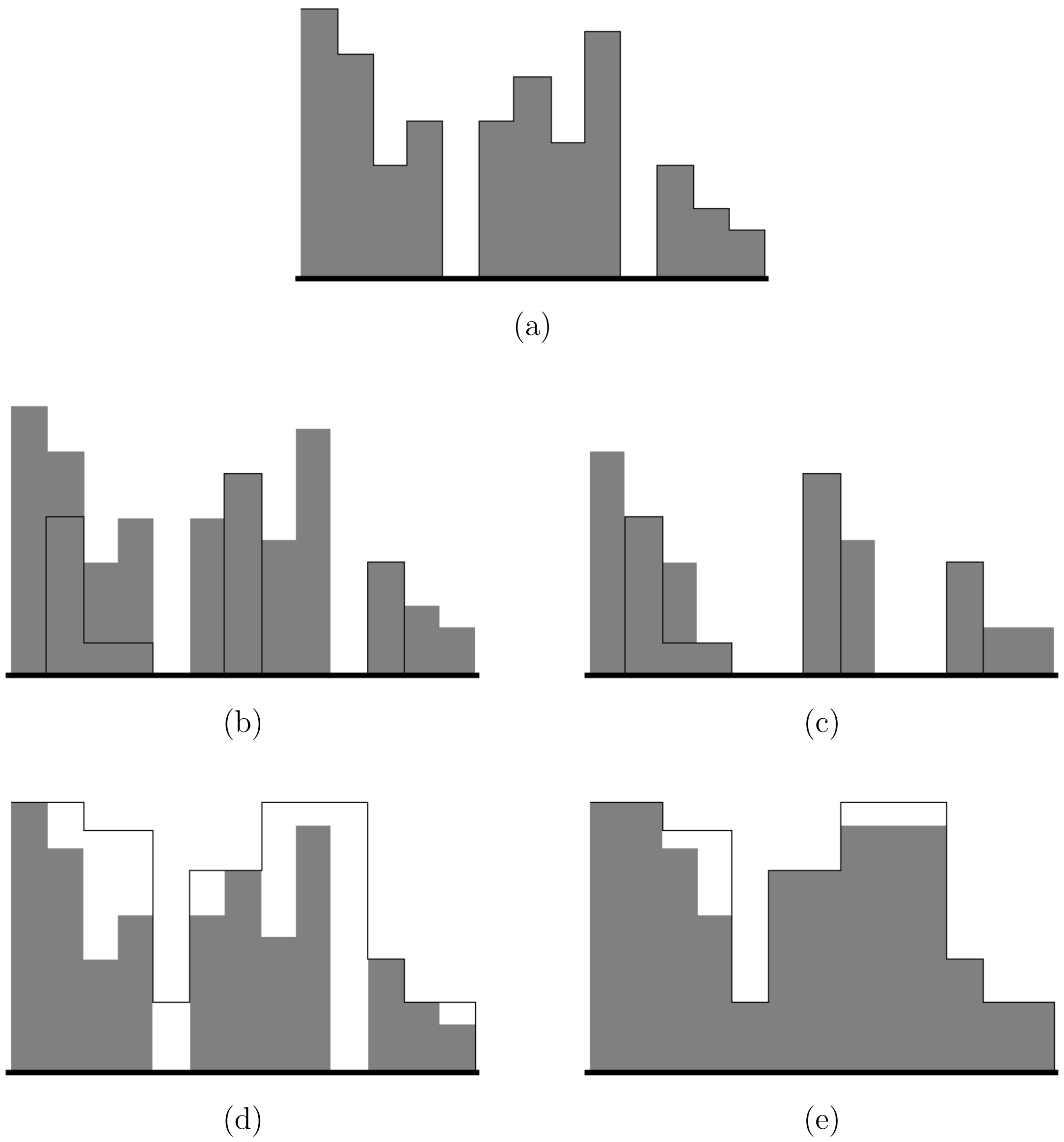


Figure 6.3: Geodesic erosion and dilation operations. (a) Original image - marker image. (b) Mask image used in the geodesic erosion, whose values are given by the solid black lines. The gray rectangles show the marker image, for comparison. (c) Geodesic erosion: Erosion of the marker image. The mask prevents the decrease in the marker caused by the erosion. (d) Mask image used in the geodesic dilation, whose values are given by the solid black lines. The gray rectangles show the marker image, for comparison. (e) Geodesic dilation: Dilation of the marker image. The mask prevents the growth in the marker caused by the dilation.

6.2.1 Input Image Enhancement

To obtain an improved version of the image, thus improving the detection algorithm results, this section describes an image enhancement method that is applied in the proposed algorithm as a preprocessing step.

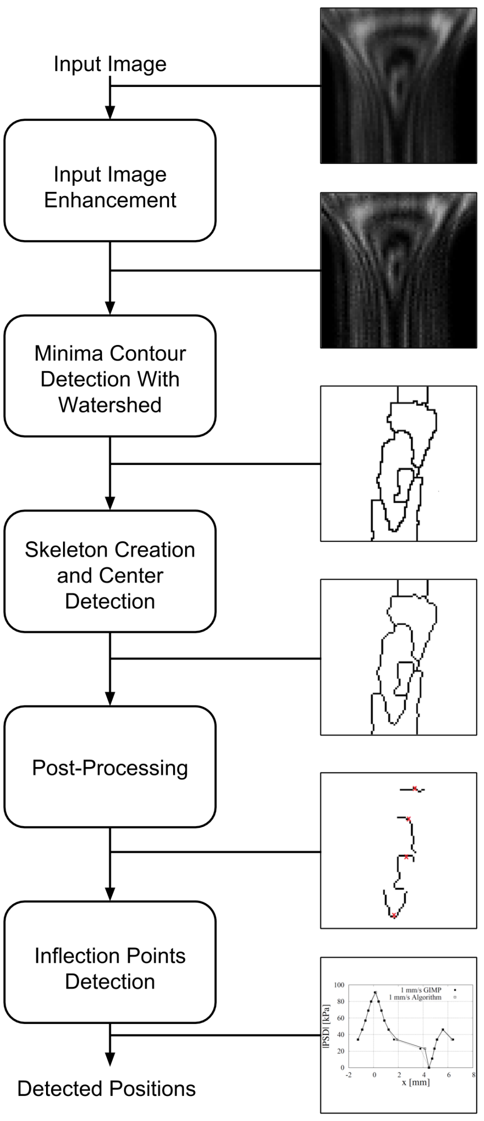

Input Image
Input Image Enhancement
Minima Contour Detection With Watershed
Skeleton Creation and Center Detection
Post-Processing
Inflection Points Detection
Detected Positions

Figure 6.4: Block diagram of the proposed method.

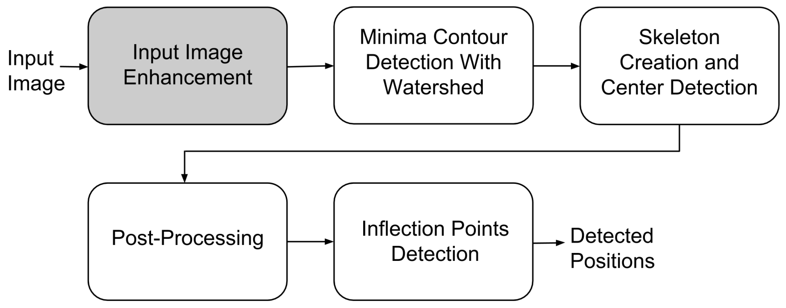

Input Image
Input Image Enhancement
Minima Contour Detection With Watershed
Skeleton Creation and Center Detection
Post-Processing
Inflection Points Detection
Detected Positions

Figure 6.5: Input image enhancement.

Due to a noisy acquisition process, the birefringence images often have a cracked appearance, as can be seen in Fig. 6.6. There it can be observed that neither their white nor their dark fringes are well defined, having many gaps that can connect consecutive dark fringes. They make the task of correctly identifying the fringes positions very difficult for the image processing algorithms. One way to approach this problem is to apply some morphological operations on the input images as a pre-processing step (Fig. 6.5) that aims to highlight the fringe silhouettes, thus making the algorithms less prone to errors.

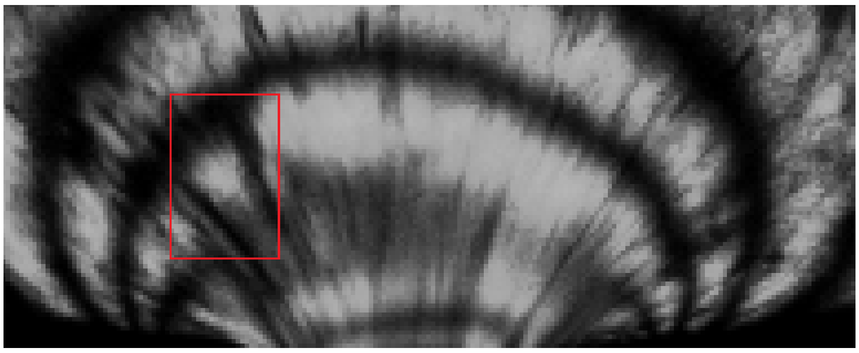


Figure 6.6: Highlight of the gaps of the fringes connecting consecutive dark and bright fringes. The dark regions have lower pixel values whereas the bright regions have higher pixel values.

In this work we propose to use two morphological operations in succession to improve the image quality, namely *white top-hat* and *black top-hat*. The white top-hat operation [125] aims to obtain the peak grayscale values of the image by applying morphological operations over it. In order to do so a modified version of the image, obtained via the morphological opening operation defined in Section 6.1 is subtracted from the original image, as described by:

$$\textit{White top-hat:} WTH(\mathbf{F}) = \mathbf{F} - \gamma(\mathbf{F}), \tag{6.8}$$

where $\gamma()$ represents the opening operation.

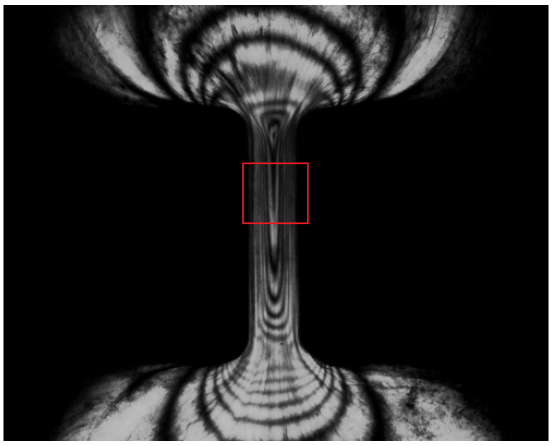
The idea behind this operation is that, as shown in Fig. 6.2(e), the opening removes the peaks in the shape of the objects, keeping the parts of the object that do not contain peaks. The white top-hat operation then takes the original image and subtracts from it a version of the image with the peaks removed, so after the subtraction only the peaks remain.

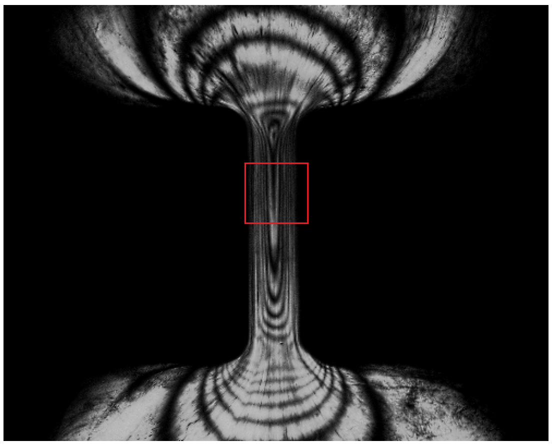
In the opposite way, the black top-hat operation [125] is responsible for extracting the valleys of the grey-level images. To perform this operation, the original image is subtracted from a modified version of the image, after going through a closing operation, as given by:

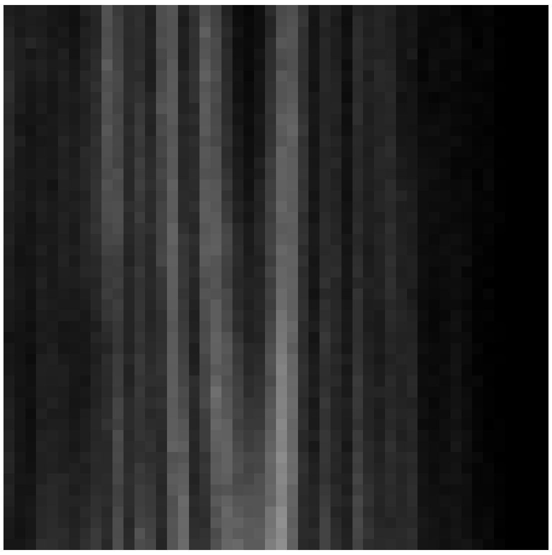
$$\textit{Black top-hat:} BTH(\mathbf{F}) = \phi(\mathbf{F}) - \mathbf{F}, \tag{6.9}$$

where $\phi()$ represents the closing operation.

This equation means that initially the closing operation is applied to create a version of the image with all valleys filled, such as the one seen in Fig. 6.2(f).


(a)


(b)


(c)

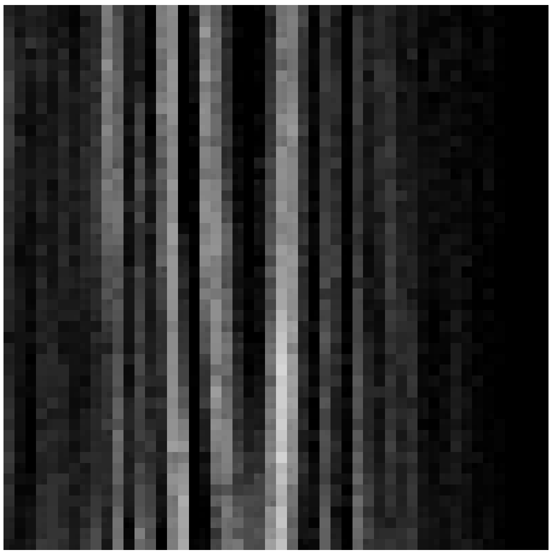

(d)

Figure 6.7: Example of birefringence images before and after the enhancement step based on the *white top-hat* and *black top-hat* operations. Its effect is best observed at the consecutive bright and dark fringes in the modified image. (a) Original image. (b) Enhanced image with increased contrast. (c) Detail of the original image. (d) Detail of the enhanced image.

Afterwards, by subtracting this result from the original image, only the filled valleys remain, so the result of the black top-hat operation is an image containing only the valleys of the original image.

The resulting images after the described procedures, that are the white top-hat image containing the peaks and the black top-hat image containing the valleys, are added to the original image, emphasizing the peaks and valleys of the grayscale image, therefore enhancing its contrast. The use of these techniques yields a better distinction between the bright and dark fringes, as illustrated in Fig. 6.7.

6.2.2 Minima Contour Detection with Watershed

Since the current state of the image is still not easy to process, in order to locate the minima contours of the pre-processed image we apply the watershed operation as described in this section.

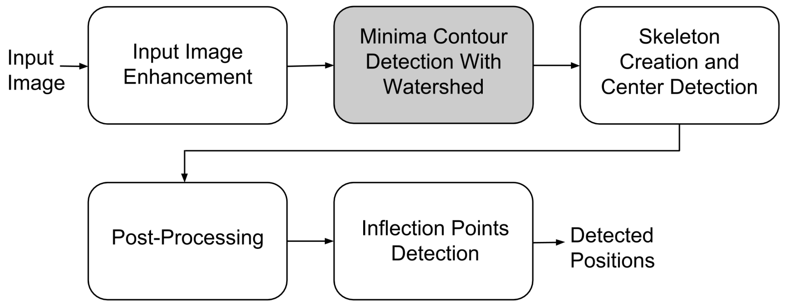

Input Image
Input Image Enhancement
Minima Contour Detection With Watershed
Skeleton Creation and Center Detection
Post-Processing
Inflection Points Detection
Detected Positions

Figure 6.8: Minima contour detection with watershed.

In order to detect the center of each dark fringe (Fig. 6.8), a different mathematical morphology method was used, namely the watershed [126] algorithm. This method aims to detect local maxima positions in a given grayscale image. The intuition behind the method is the following: the method considers the image topology to be a succession of peaks and valleys; in the bottom of the valleys it is assumed to exist holes. If a liquid would enter the valleys through these holes, at the same time for holes at the same depth, there would be an instant when the liquids in two consecutive valleys would merge. The position where those liquids merge is considered to be a local maximum, as depicted in Fig. 6.9.

The loci of the maxima can be used to detect the center of each one of the dark fringes. To do so one initially needs to obtain the complement of the image, therefore making each peak a valley and each valley a peak. After that, one should apply the watershed method to the inverted image. An example of the result obtained with this procedure can be seen in Fig. 6.10(a), where the resulting analysis still presents some detection artifacts.

Although this result seems to be initially encouraging, one can readily see that it is not enough to solve the problem of detecting the position of the dark fringes, as this simple application of the watershed method creates a result that has at the same time regions with under-segmentation, showing missing boundaries, and over-segmentation, displaying undesired boundaries. This can be improved by feeding the algorithm with some prior knowledge, like the approximate positions of the maxima (or minima). In our proposed approach a human user must input the location of every bright fringe (the minima points in an inverted image). The imposition of those minima locations [125] creates a mask **Fm** whose values are $t_{\max}$ (the maximum value allowed to the pixel) except where the minima were imposed (where the mask

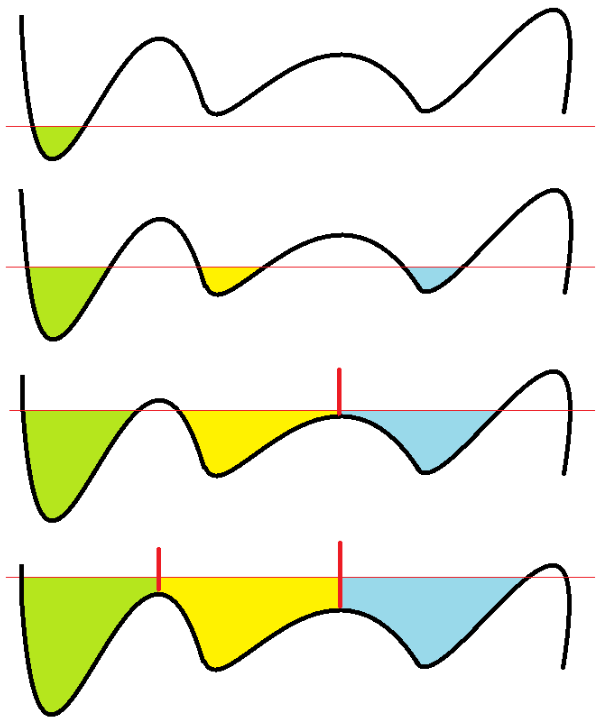


Figure 6.9: Watershed peak detection method. The inner peaks are detected and marked as the red vertical lines.

value is set to 0), as given by

$$\mathbf{Fm} = \begin{cases} 0, & \text{if a minimum was imposed} \\ t_{\max}, & \text{otherwise} \end{cases}, \tag{6.10}$$

where $t_{\max}$ is the largest possible pixel value in the grayscale image. This process is detailed in Fig. 6.11.

We then take the minimum between each pixel value and the value of the mask in that position, whose result is called $(\mathbf{F}+1) \wedge \mathbf{Fm}$ in Fig. 6.11(b). After that, the signal $\mathbf{Fm}$ is considered the marker and the signal $(\mathbf{F}+1) \wedge \mathbf{Fm}$ the mask of a geodesic erosion, as defined in Subsection 6.1.1. Successive geodesic erosions are performed (Figs. 6.11(c)-(g)) until the image becomes unchanged by further erosions (Fig. 6.11(h)). Note that, when you perform the minima imposition, the imposed minima will be the only local minima in the resulting image, lending to it a smooth appearance.

After the minima imposition, we then apply the previously described watershed method to the resulting image to obtain the maxima contours, avoiding in this case under- and over-segmentation issues. An example of the results after the minima imposition can be seen in Fig. 6.10b, which improves upon the results shown in Fig. 6.10a. One can see in the region highlighted by a blue ellipse that the minima imposition technique reduces some over-segmentation artifacts while, as seen in the

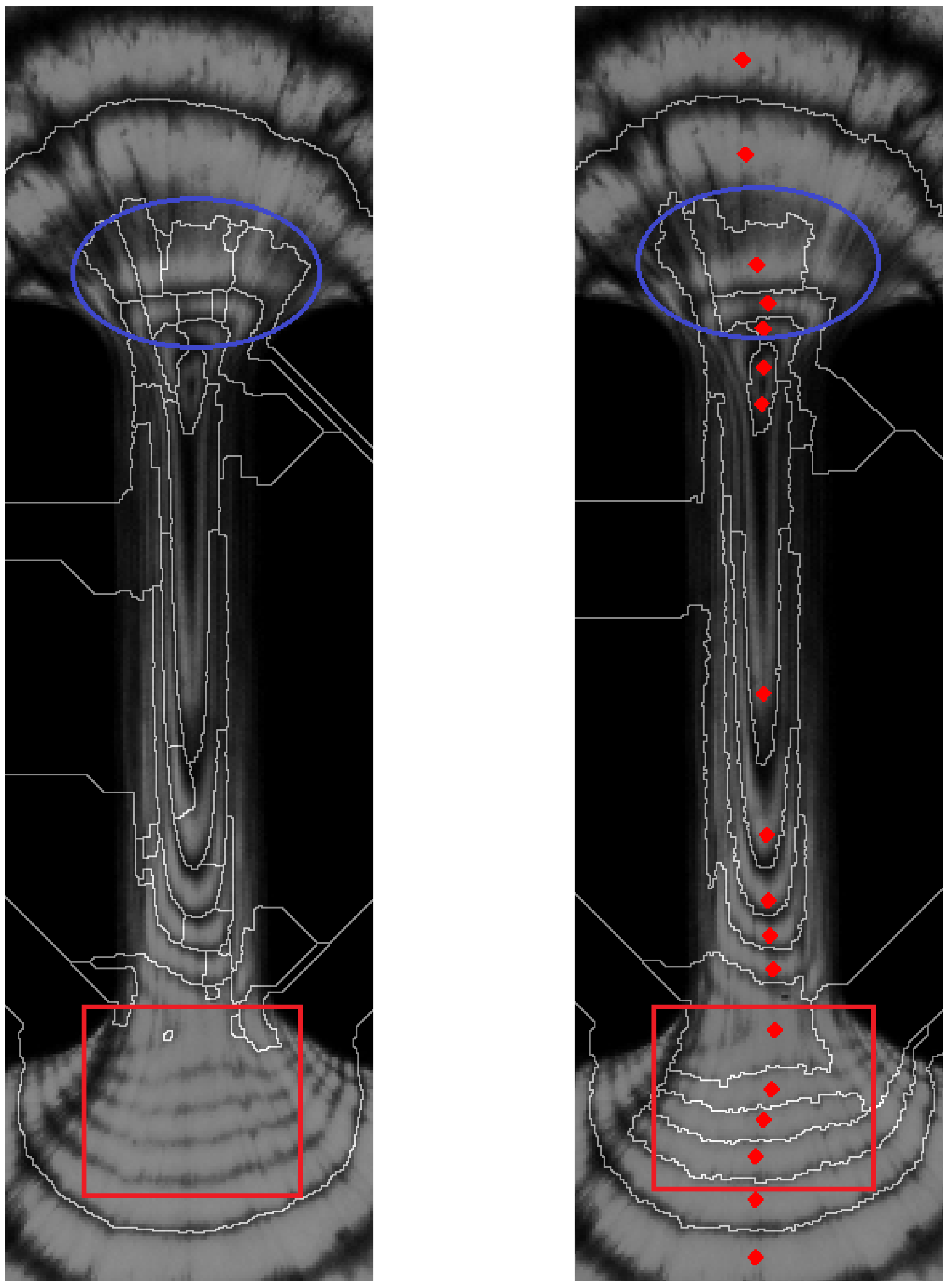
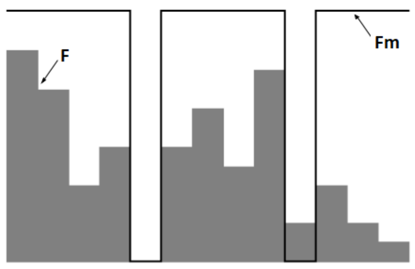
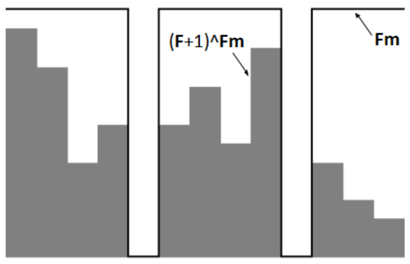

(a) (b)

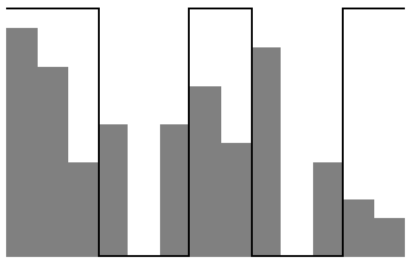
Figure 6.10: Contours of the dark fringes obtained with the use of the watershed method. (a) without minima imposition. (b) with initial minima imposition (marked as a red diamond). The blue ellipses and the red rectangles show, respectively, regions where there is over-segmentation and undetected fringes in the version without minima imposition.



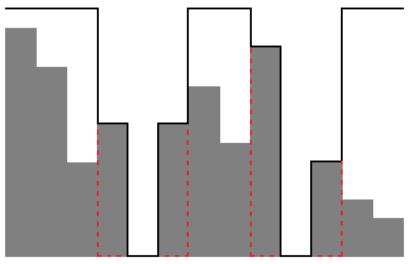
(a)



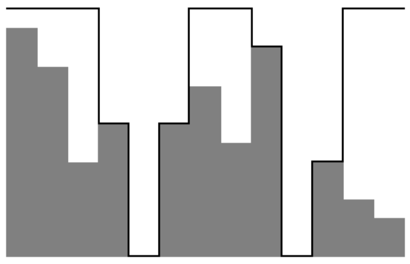
(b)



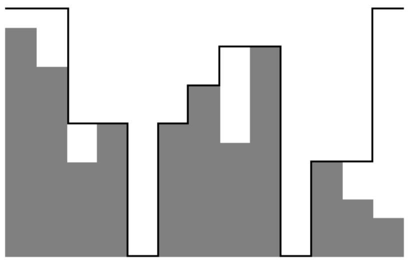
(c)



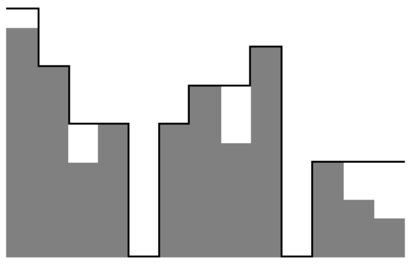
(d)



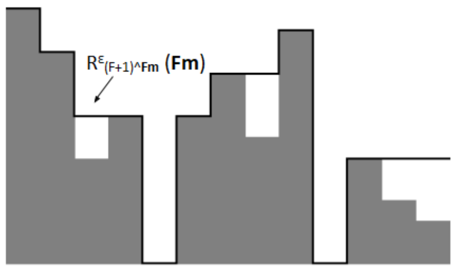
(e)



(f)



(g)



(h)

Figure 6.11: Minima imposition method. (a) Original figure **F** and mask **Fm**. (b) Pixelwise minimum between **Fm** and (**F** + 1), which is defined as (**F** + 1) ∧ **Fm**. (c) Morphological erosion of **Fm**. (d) Maximum between (**F** + 1) ∧ **Fm** and the result of the morphological erosion of **Fm**. The red dashed line shows the values that were changed in the maximum operation. (e) Final result of the first geodesic erosion of **Fm**. (f) Second geodesic erosion of **Fm**. (g) Third geodesic erosion of **Fm**. (h) Fourth geodesic erosion of **Fm**. The signal remains unchanged after any other geodesic erosion, so it is considered the reconstruction via successive erosions of (**F** + 1) ∧ **Fm** from the mask **Fm**.

region delimited by the red rectangle, and also increases the amount of successfully detected fringe contours.

6.2.3 Skeleton Creation and Center Detection

After we obtain the minima contour with the watershed operation we need to further process it in order to produce 1 pixel-wide contours. To do so we employ the skeletonisation technique described in this section.

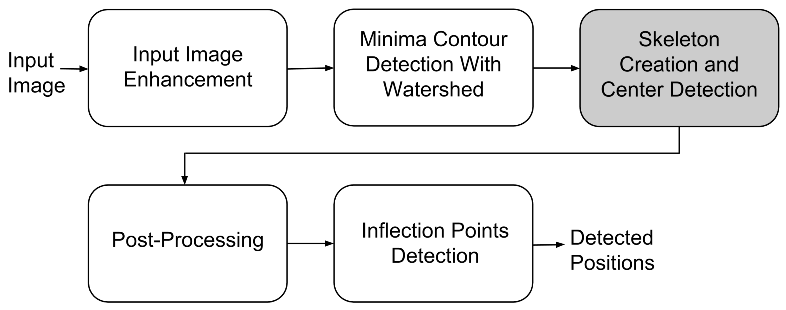

Input Image
Input Image Enhancement
Minima Contour Detection With Watershed
Skeleton Creation and Center Detection
Post-Processing
Inflection Points Detection
Detected Positions

Figure 6.12: Skeleton creation and center detection.

The result of the watershed method is a binary image that indicates boundaries that separate each bright region, so it should represent a centerline for the dark fringes. In order to select the exact position of the fringe centers it is necessary that the contour lines are exactly one-pixel wide. Sometimes, after using the watershed method the contour lines are thicker than one pixel, thus lowering the accuracy of the method. To deal with this problem we propose (Fig. 6.12) the use of a skeletonisation method [127]. A morphological skeleton of a binary image such as the contours in Fig. 6.10 is defined as the locus of the center of the maximal disks completely contained in the image. Fig. 6.13 shows the step-by-step procedure to obtain the skeleton of an image. This procedure creates thinner segments with a thickness of one pixel, which is mandatory for the next steps.

6.2.4 Post-processing

After obtaining the pixel-thin contours we proceed to the estimation of the fringe center position, as well as, removing the false detections. This section describes the post-processing methods employed in this task.

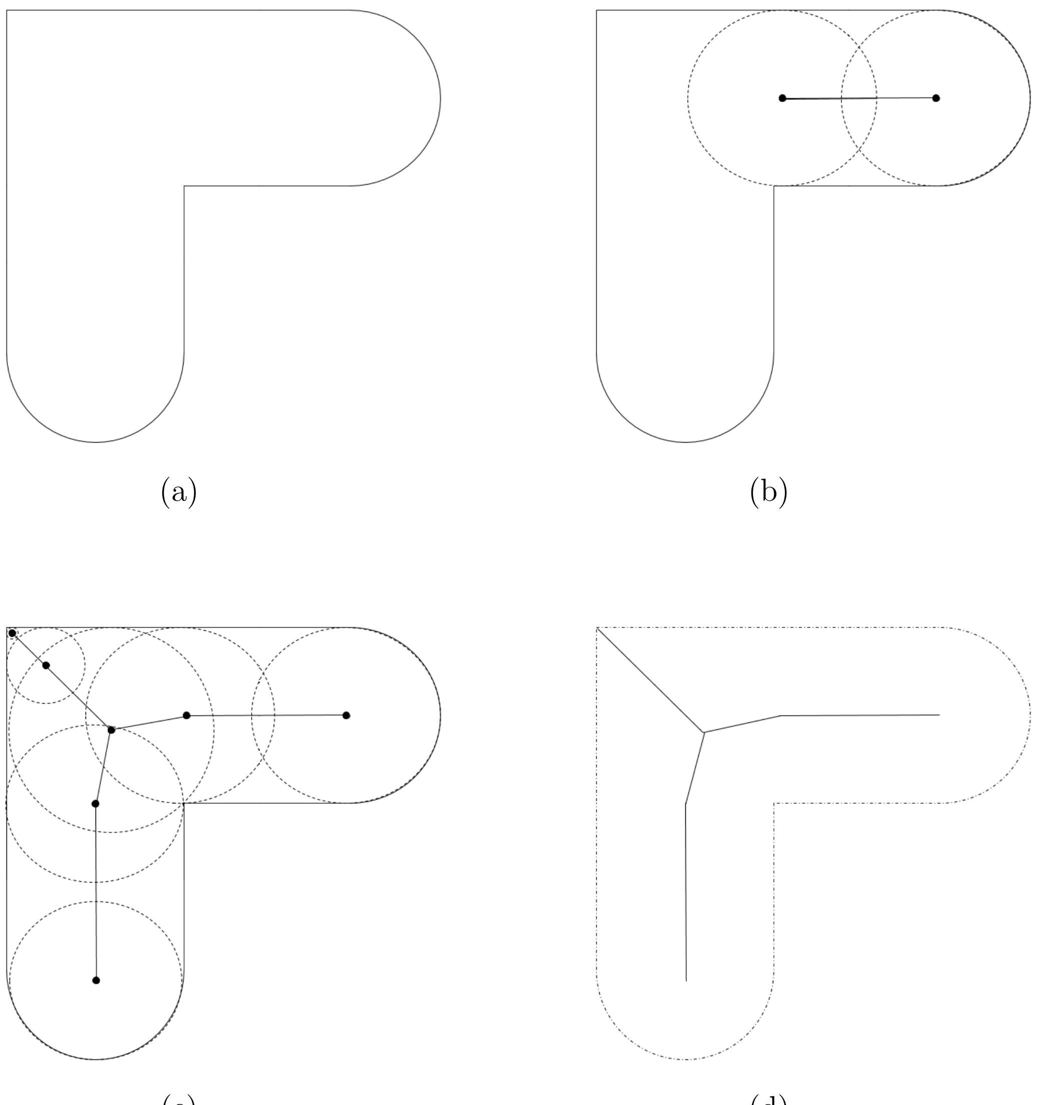


Figure 6.13: Skeletonisation using maximal disks. The union of all maximal-disk centers defines the skeleton. (a) Original image. (b) Some maximal disks and the initial construction of the skeleton. (c) Maximal disks covering the whole image. (d) Complete skeleton.

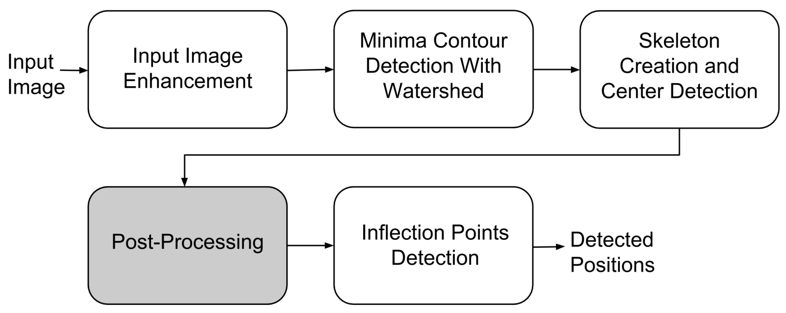

Input Image
Input Image Enhancement
Minima Contour Detection With Watershed
Skeleton Creation and Center Detection
Post-Processing
Inflection Points Detection
Detected Positions

Figure 6.14: Post-processing.

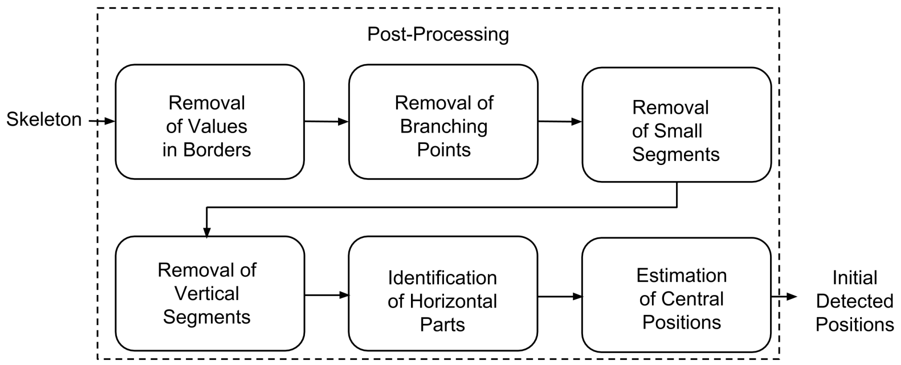

Post-Processing
Skeleton
Removal of Values in Borders
Removal of Branching Points
Removal of Small Segments
Removal of Vertical Segments
Identification of Horizontal Parts
Estimation of Central Positions
Initial Detected Positions

Figure 6.15: Detailed block diagram of the post-processing.

At this point, the contour lines, which we expect to be present only in the dark fringes, may also erroneously appear in a bright fringe. As can be seen from Fig. 6.10, the contours have some segments that do not belong to the dark fringes and should not be used to estimate their position. This figure also shows that those segments have a shape similar to a vertical line, being perpendicular to the fringe variation. The algorithm uses this fact to detect and remove flaws in the contours by separating it into small segments and removing those that have a vertical orientation. In addition, one is not interested in identifying the whole dark fringe but only the position of its center, therefore the contours can be further processed in order to extract only the required information. The post-processing pipeline is depicted in Figs. 6.14 and 6.15 and is described in the next sections.

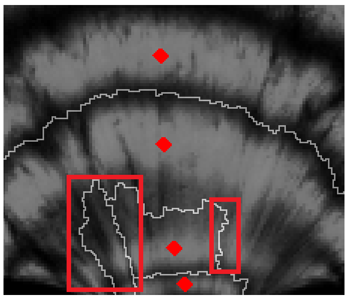


Figure 6.16: Detail of the contours of the dark fringes obtained with the use of the watershed method. One can notice some flaws in the detection, with parts of the contour highlighted by a red square appearing in the bright fringe.

Removal of values in borders:

In the first step, only the center columns of the image are kept, which contain the region of interest for the detection of the fringe positions. We remove any contour outside the center columns of the image, delimited by the thin region that contains a fringe pattern (see Fig. 6.7).

Removal of branching points:

Afterwards, the branching points, which are points connected to three or more points, are removed from the remaining skeleton, that after this operation will be just a set of unconnected segments, as seen in Fig. 6.17.

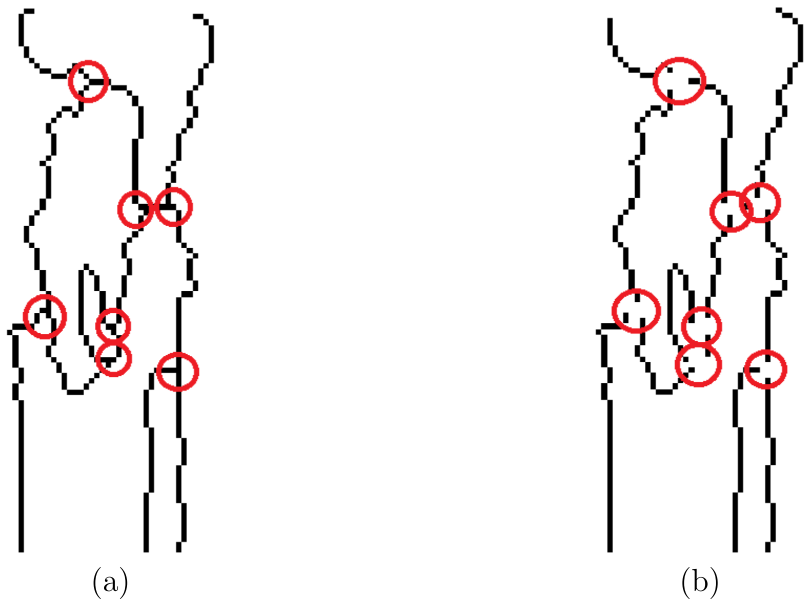


Figure 6.17: Detail of the branching points removal method, where the circles show the points to be removed. The branching points are considered as the points connected to three or more points. (a) Before branching point removal; (b) after branching point removal.

Removal of small segments:

After this step, several spurious segments may still remain. We count for each segment the number of points contained in it, and the segments that have less than a given number of points are removed.

Removal of vertical segments:

We also analyze each segment and measure its horizontal length, which is done by counting the number of columns that it occupies, and remove those that and have a horizontal length smaller than a threshold (in other words, are ”too vertical”), since they most likely do not correspond to the desired center position of the fringes. An example of segments that remain after these cleaning steps can be seen in Fig. 6.18.

Identification of horizontal parts:

In this step, we want to determine the vertical position for the center of each dark fringe. Comparing Figs. 6.10 and 6.18, one can see that due to the shape of the dark

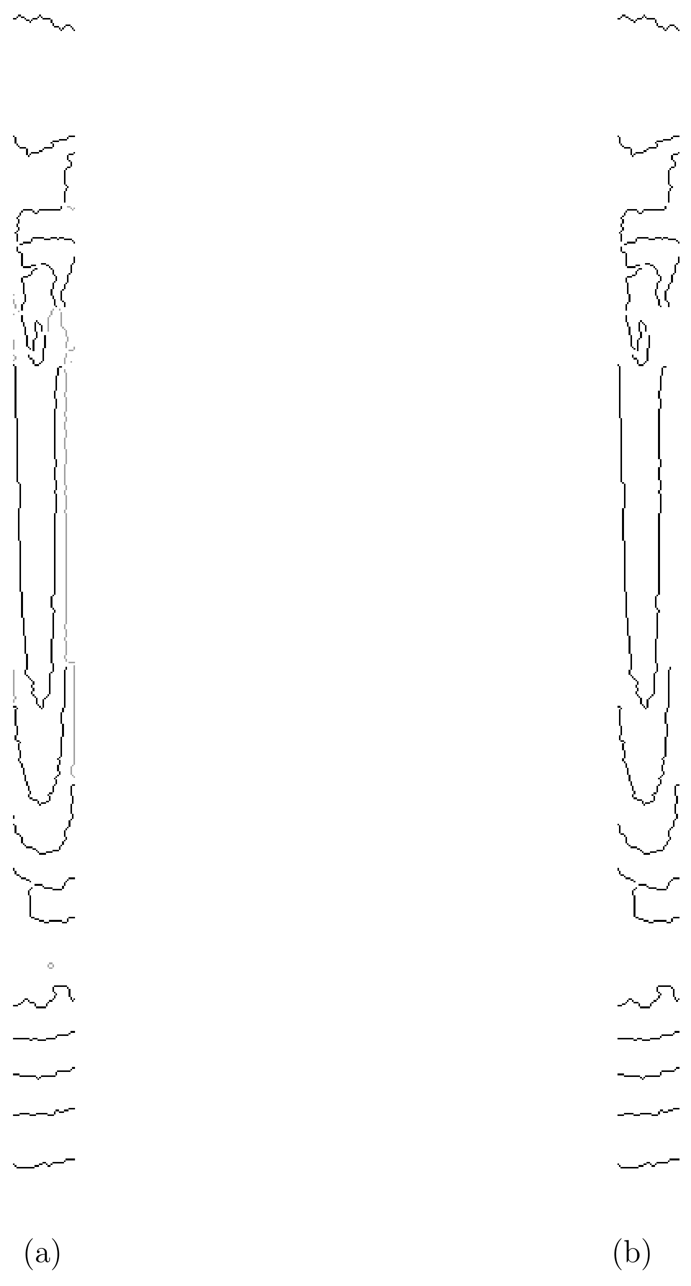


Figure 6.18: Final selection of the segments after the spurious segments are removed. (a) Before segment removal; (b) after segment removal. The removed spurious segments are marked as light gray in the figure.

fringes the remaining segments have a parabolic shape with a vertical orientation, whose peak indicates the position of the center of the fringe. Therefore, in order to identify this center, we detect the peaks in the segments by using a horizontal window to select for each segment the part that has the largest number of horizontal points. As illustrated in Fig. 6.19, when sliding the horizontal window along the vertical direction, the position that yields the largest segment corresponds to the location of the peak (within a certain accuracy given by the window width).

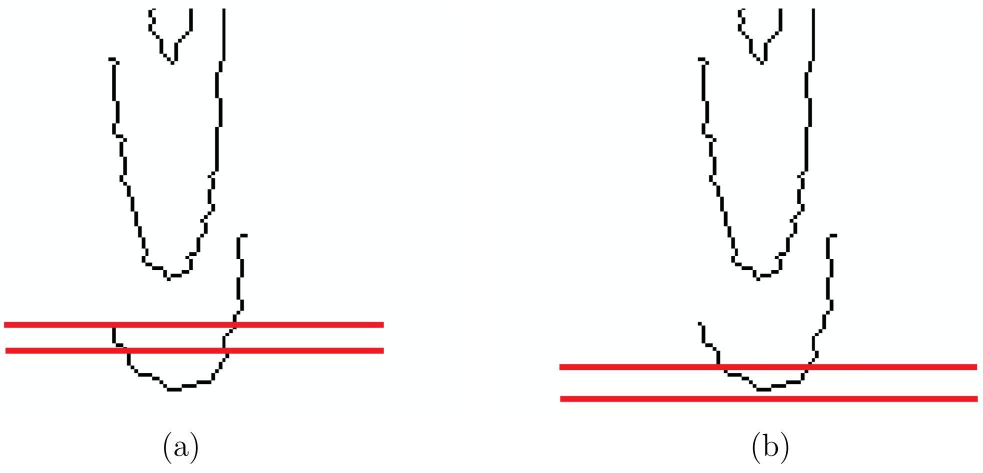


Figure 6.19: Selection of the horizontal part of the segments. (a) Wrong region - few points. (b) Correct region - largest number of points.

Estimation of central positions:

Finally, the center position of each fringe is selected as the mean of the vertical coordinates of each skeleton pixel within the horizontal window. If after this step more than one segment is detected between two consecutive maximal points manually input, only the one with the greater number of points is kept. This step is illustrated in Fig. 6.20(a) and the final result is shown in Fig. 6.20(b).

6.2.5 Inflection Points Detection

After selecting the correct segments to represent each fringe, we proceed to the detection of the inflection point of the image. This section describes the employed procedure.

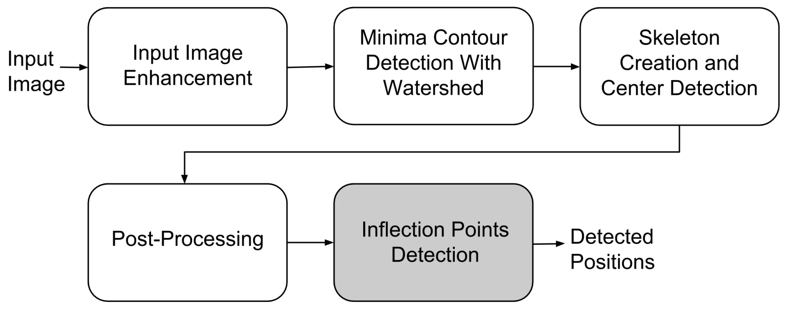
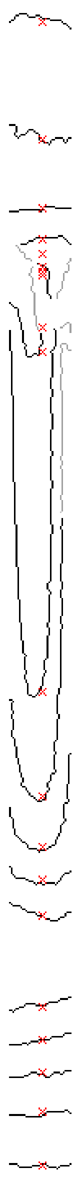

Input Image
Input Image Enhancement
Minima Contour Detection With Watershed
Skeleton Creation and Center Detection
Post-Processing
Inflection Points Detection
Detected Positions

Figure 6.21: Inflection points detection.

A final important point for the characterization of the flow is an inflection point. As reported by [88], this point helps to characterize the geometry of the experiment


(a)

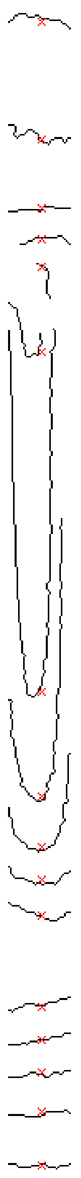

(b)

Figure 6.20: Detected positions for each segment. The red X marks in the center column the vertical coordinates for the segment that has the largest number of horizontal points, which represents the center for each dark fringe according to the procedure described in Section 6.2.4. (a) Result before the elimination of double detections. (b) Final result, after the elimination of double detections, when there is more than one segment to represent a dark fringe. The ambiguous segments, marked as light gray, were removed.

and is defined with the maximum retardation order k in the flow, despite not being associated to the horizontal part of any dark or bright fringe in the images, as one can see in Fig. 6.22. In order to estimate this point, the algorithm described in

Subsections 6.2.1-6.2.4 is applied in the transposed images to detect the two vertical dark fringes around the inflection point (Fig. 6.21), and the position ofo the inflection point is defined as the average of the center of the two vertical fringes. An example of this detection is show in Fig. 6.23.

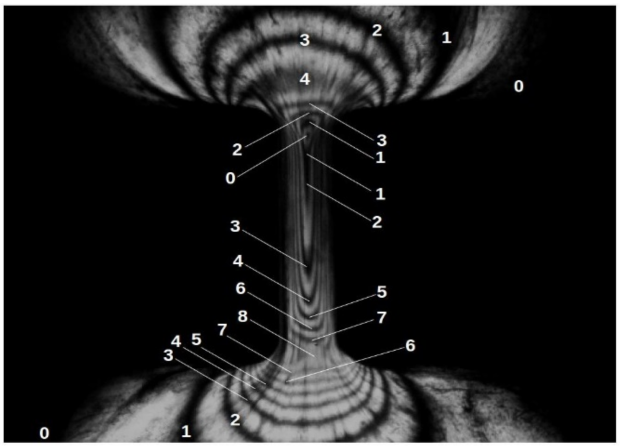


Figure 6.22: Fringe retardation order (k) of sample GPPS1 at 1 mm/s. The inflection point is reported in [88] as the one with a retardation order $k = 8$ and is related to the geometry of the experiment.

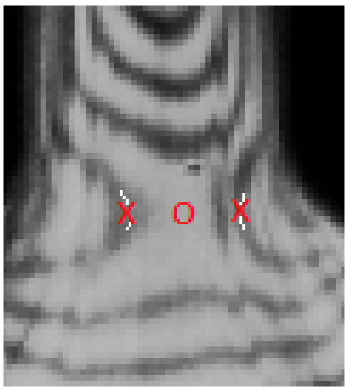


Figure 6.23: Example of the detection of the inflection point. The center of the white contours associated to each vertical fringe is marked as a cross and the estimated inflection point is marked as a circle.

6.3 Experimental Results

In this section we will analyze the algorithm performance, starting from a description of the experimental framework, followed by a performance assessment to calibrate the algorithm. The results of the proposed algorithm are first compared to other methods found in the literature and then compared to the reference method.

6.3.1 Experimental Framework

The algorithm described in Section 6.2 was developed in MATLAB© [89] environment using morphological function implementations present on the Image Processing Toolbox. The results were obtained using a computer with a Xeon E3-1270 v5 3.6GHz processor and 32GB of RAM.

The flow-induced birefringence images acquired during a steady state for two polystyrene samples were presented in [88] as was also the identification of fringe orders k (based on the knowledge of the physical properties of the experiment). It is important to note that the algorithm proposed here is not entirely automatic because it requires that, for each different experiment, an operator inputs a few minima to be imposed to one of the frames, using the procedure described in Section 6.2.2, which are used in all images from the same experiment.

6.3.2 Performance Assessment

In this section, the proposed algorithm is thoroughly analyzed. In the first experiment, the pre-processing step described in Subsection 6.2.1, which includes image enhancement and minima imposition, is enabled and disabled to show its effectiveness. Then, two parameters present in Subsection 6.2.4 have their value adjusted, namely the threshold that controls the removal of spurious elements (see Fig. 6.18) and the width of the window that selects the horizontal part for each segment, which represents the center of each dark fringes (see Fig. 6.19).

In order to define an objective evaluation, the measurements obtained in [88] were considered as a ground truth, and for each algorithm setup the mean square error between the obtained fringe positions and the ground truth was computed. For this evaluation, only the experiment obtained with the GPPS1 flow at 1 mm/s was used, since it shows a pattern where the fringes are neither too few nor too thin. From this experiment, 15 images were obtained and used for this study.

Determination of the fringe center:

As described in Section 6.2.4, the detected contours have a shape similar to a noisy parabola positioned inside the dark fringes, and the center of each dark fringe is identified by the vertex of this parabola. A simple way that has been devised to find the position of the vertex is to use a moving horizontal window, analyzing for each position the part of the contour that fits entirely inside this window and identifying the one with the largest length. In order to so, we must define the optimal width of the horizontal window. A window that is too thin may lead to a false detection due to noise. If it is too wide, the algorithm is able compensate the noise but it loses precision. In this experiment we analyze different window widths and determine the one that provides the lowest average error for the algorithm.

Fig. 6.24 shows an evolution of the mean squared error obtained by the algorithm for several values of the window width. In the figure, the multiple curves represent the results for a fixed value of the length that separates spurious and accepted elements. In Fig. 6.24(a), the pre-processing is disabled while in Fig. 6.24(b) it is enabled. It is important to notice the range of the error values, since in the curves with the pre-processing disabled, the minimum observed error is higher than the maximum error with the pre-processing enabled. Comparing only the best results for each case, the pre-processing step reduces by up to 20 times the error in the proposed algorithm.

In Fig. 6.24(b), the curves reveal the two cases where the error tends to be higher: if the window is too restrict, so it is more likely to contain small peaks in the data, and too big, so the window contains larger segments that may not represent the fringe centers. There seems to be a region for some intermediate values where the error reaches its minimum and is less sensitive to the variation of this parameter. To determine a compromise in the algorithm configuration, the window was defined to have a width of 5 pixels.

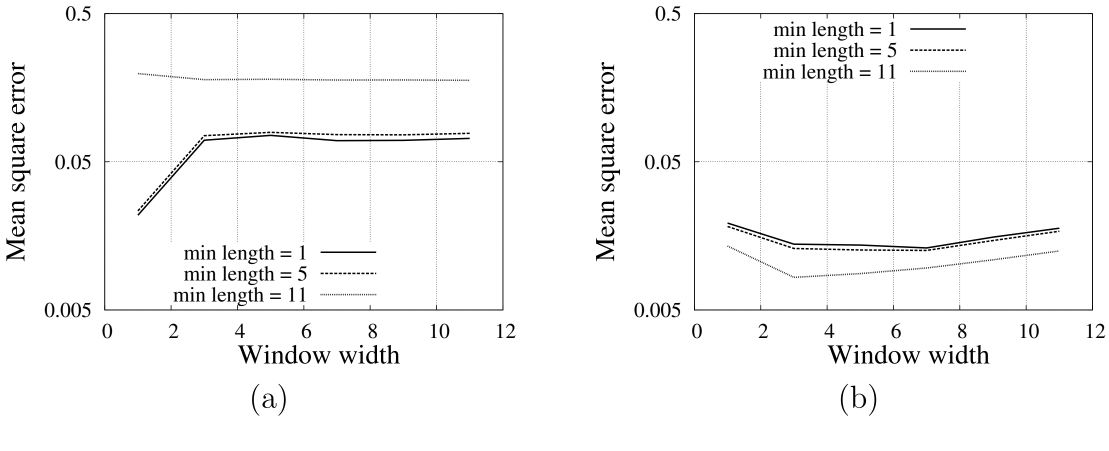


Figure 6.24: Mean square error (mm^2) for several values of the window width. (a) Without pre-processing. (b) With pre-processing.

Removal of spurious elements:

In the detected contours, part of the noise is due to impurities in the experiments, that create undesirable contours inside the bright fringes. As described in Section 6.2.4, a post-processing step separates the contours in segments and removes from the final detection the ones with a length smaller than a threshold. There is an optimum value for this threshold. If it is too big, it is possible that all segments associated to a fringe are removed and no position estimate can be provided. In this experiment, we evaluate the impact of the threshold that removes undesirable segments in the detection of the fringe centers.

In Fig. 6.25 one can see the relation between the mean square error and the minimum segment length for a few fixed values of the window width. Another important aspect in this experiment is the detection rate, that is, the amount of fringes where there is at least one detected segment and the algorithm can estimate a fringe center. This characteristic is exemplified in Fig. 6.26.

Comparing Figs. 6.25(a) and 6.26(a), one can see that the error increases with a bigger threshold. This behavior can be explained by the fact that without the pre-processing step, the algorithm creates spurious segments that are indistinguishable from the true segments (the ones that follow the fringe pattern), so any attempt to remove wrong segments also removes the true ones, and the detection rate drastically falls when the threshold is increased. With the inclusion of the pre-processing step, a different behavior can be seen, as shown in Figs. 6.25(b) and 6.26(b). With bigger values for the acceptable segment length, more spurious segments are removed while still keeping an amount of accepted segments, since the detection rate shows a smaller decrease, and the mean square error is in fact reduced. In order to provide an algorithm able to detect the fringe positions for at least 95% of the cases with the smallest error, the threshold value was defined as 5 pixels.

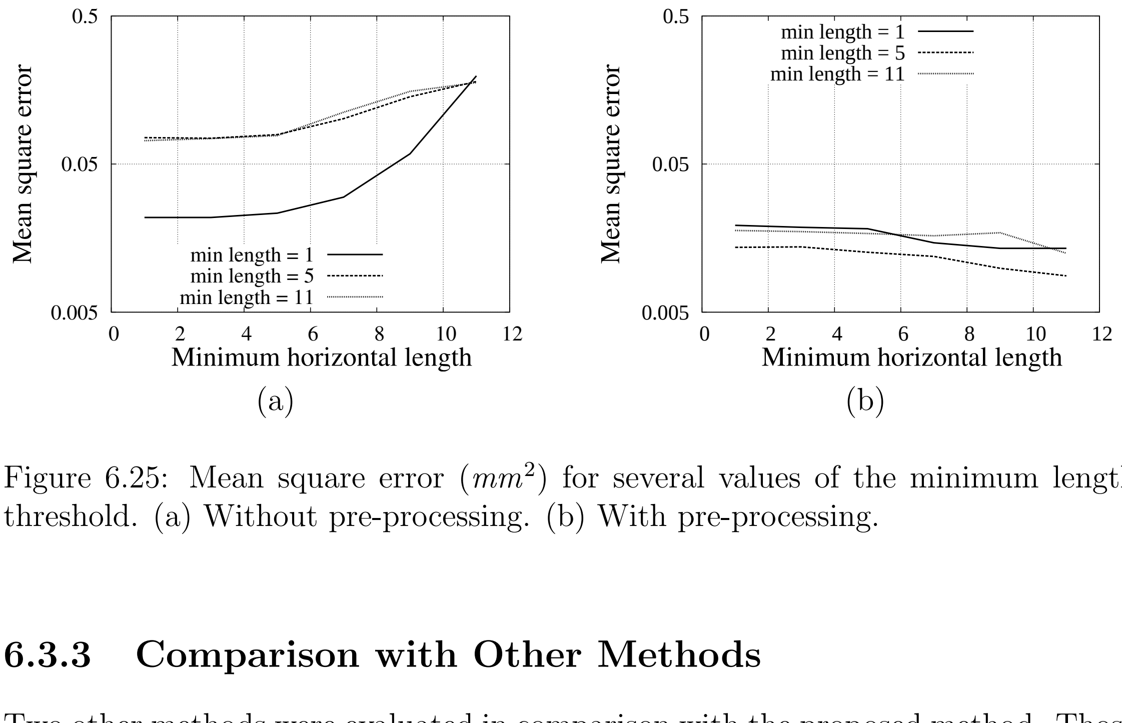
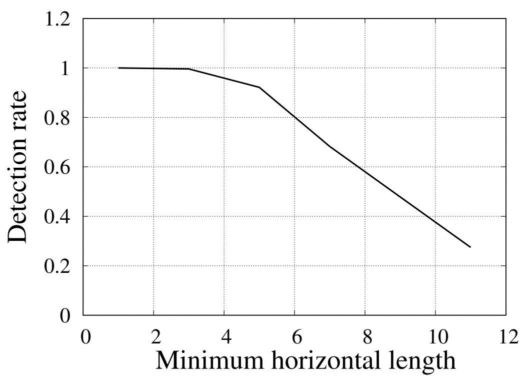


Figure 6.25: Mean square error (mm^2) for several values of the minimum length threshold. (a) Without pre-processing. (b) With pre-processing.

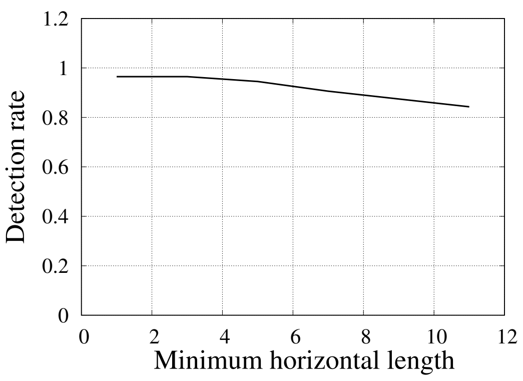
6.3.3 Comparison with Other Methods

Two other methods were evaluated in comparison with the proposed method. Those methods were developed to detect fringes on images originated in different experiments and were adapted to the context of birefringence images.

The method [74] separates the bright and dark fringes by binarizing the images using a threshold that can be constant or adaptive, and then applies a skeletonisation



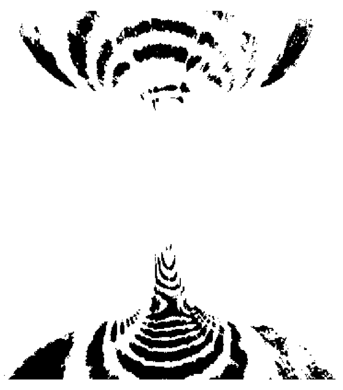
(a)



(b)

Figure 6.26: Detection rate for several values of the minimum length threshold. (a) Without pre-processing. (b) With pre-processing.

technique to find the central position for each fringe. Due to impurities in the experiments, the image has a cracked appearance. A simple constant threshold is not able to distinguish every bright and dark fringe in the images, as can be seen in Fig. 6.27(a). However, by manually selecting two levels of threshold to be used in different parts of the image, a good compromise between algorithm performance and model overfitting can be found, as shown in Fig. 6.27(b).


(a)

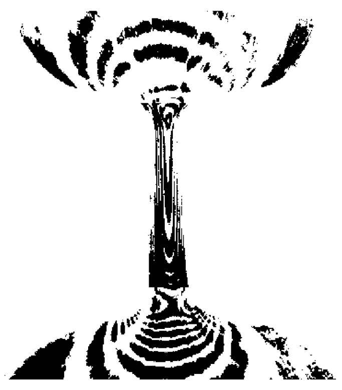

(b)

Figure 6.27: Birefringence image binarization. (a) With a constant threshold. (b) With two levels of threshold.

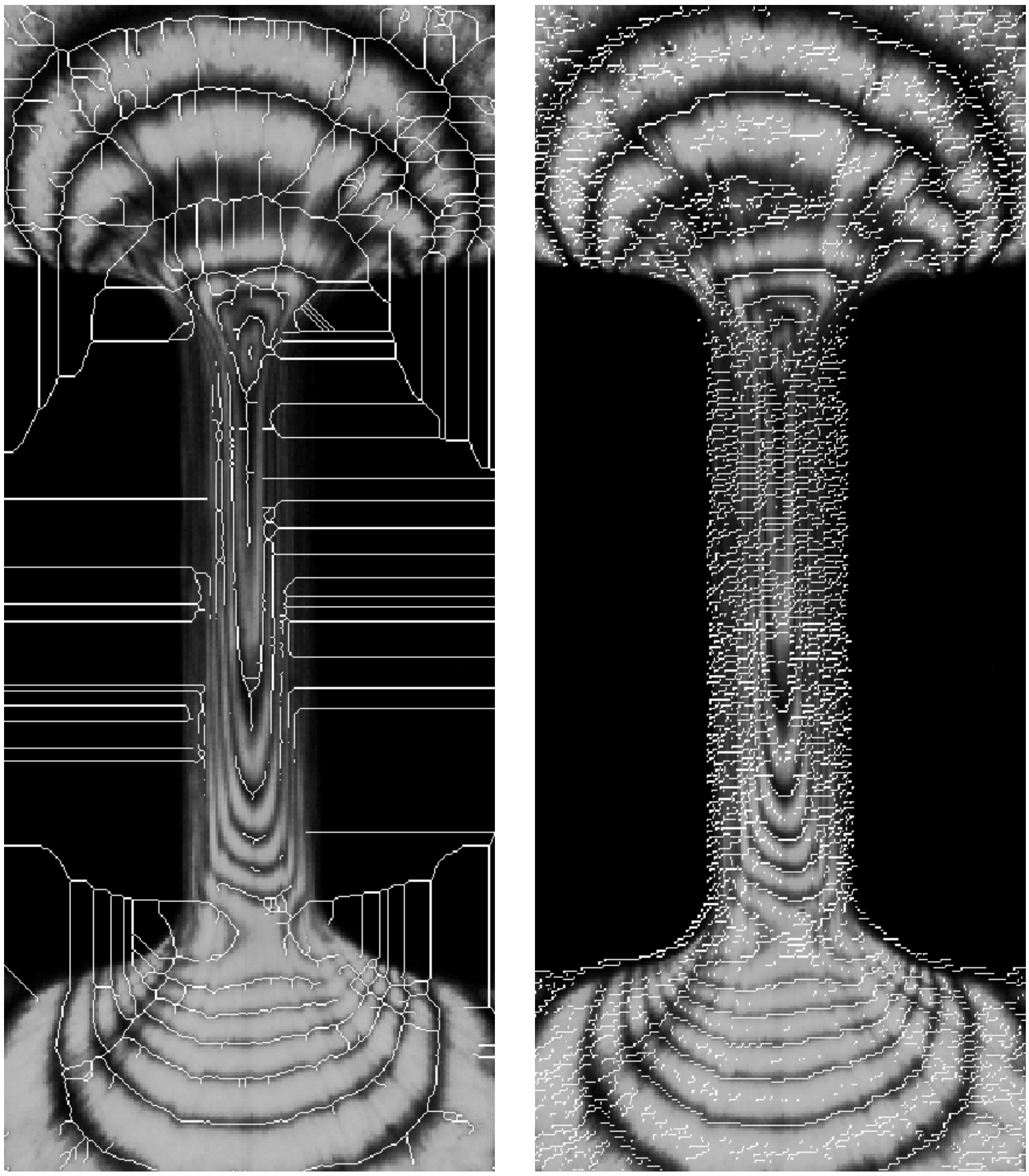
In [76, 84] a technique to detect fringes in light interferometric experiments is described. First, it estimates a local orientation map for each pixel [128], and pre-processes the images by applying a directional median filtering that uses the estimated orientation in order to remove the noise in the direction tangent to fringe orientations while also keeping sharp transitions perpendicular to the fringe orientations. Afterwards, the method computes the directional derivatives of the pre-processed images, also using the orientation map. Then it binarizes the directional derivatives and considers the fringe centers as the edges of this binarized image, that is, the positions where there is a fast transition between a positive and a negative gradient.

However, also due to the characteristics of the birefringence images, the estimation of an orientation map is likely to produce a noisy result. We implement this algorithm using only a horizontal median filtering, since it corresponds to the expected tangent of orientation of the fringes in the center of the images, which is the region-of-interest for the detection. For this reason,, we also compute only the vertical derivative to be used in the binarization and detection step. An example of the implementation of the methods in [74] and [76, 84] applied to the birefringence images can be seen in Fig. 6.28.

The same objective evaluation used to set up and validate the proposed algorithm is employed in the comparison with the aforementioned methods. For each method, we compute the mean square error between the obtained contours and the manual marking, and count the amount of fringes that were not detected to estimate the detection rate. Since those methods only provide a basic skeleton of the fringe image, no precise central position is extracted. In order to properly compare with the proposed method, the post-processing step described in Subsection 6.2.4 and optimized in Subsection 6.3.2 is also applied.

In Tab. 6.1, one can see a comparison of the mean squared error obtained with each method. For the method [74], no pre-processing step is defined, so, in order to provide a fair comparison, the proposed pre-processing(which includes the input enhancement and minima imposition) was also tested with this method. The works [76, 84] already include a median filtering to enhance the image, and for this reason two cases were considered. In the first one, the algorithm was tested using only the median filter, which is labelled as no pre-processing in Tab. 6.1. In the second one, the algorithm was tested with the median filter in addition to the presently proposed pre-processing, which is labelled as with pre-processing in Tab. 6.1.

Tab. 6.2 shows the detection rate for the same methods and configuration. Comparing the performance of the mean square error and the detection rate, one can see that the proposed method, with the correct configuration, produces a result with the lowest error while providing the second-best detection rate, being better than the binarization method in both metrics In addition, while it produces a detection rate lower than the one of the derivative method, its mean squared error is twice as small. In addition, one can also see from the results in Tab. 6.1 that the works [76, 84] have a loss of performance in the presence of the pre-processing step. The reason for this behavior is that, as a byproduct of the minima imposition technique, the resulting image loses its original local minima, presenting only the imposed minima, as discussed in Subsection 6.2.2.



(a) (b)

Figure 6.28: Contours of the dark fringes obtained with: (a) [74] and (b) [76, 84].

Table 6.1: Mean square error (mm^2) obtained with the proposed method, the method based on the image derivative [76, 84] and the method based on image binarization [74].

Pre-processing	Proposed	Derivative	Binarization
no	0.086	0.029	0.019
yes	0.013	0.043	0.019

Table 6.2: Detection rate obtained with the proposed method, the method based on the image derivative [76, 84] and the method based on image binarization [74].

Pre-processing	Proposed	Derivative	Binarization
no	0.85	1.00	0.93
yes	0.95	1.00	0.93

6.3.4 Comparative Evaluation with Reference Method (GIMP)

The results obtained using the approach presented in Subsection 6.2 after processing 15 images for each experiment were compared with those disclosed in [88], in which a methodology based on GIMP software was proposed. For this purpose, we consider the flow direction through the slit at three different velocities (0.5, 1 and 2 mm/s in the experiments using GPPS1 sample and 0.2, 0.5 and 1 mm/s in the experiments using GPPS2 sample). Note that the flow-induced birefringence images acquired during a steady state for two polystyrene samples were presented in [88] together with the determination of fringe order (k).

Fig. 6.29 shows a comparison for GPPS1 and GPPS2 samples of the measurements obtained with GIMP software and also those acquired with the proposed morphological approach. The values for the principal stress difference module along the flow centerline were calculated using Eq. (6.1).

As mentioned in [87, 88], the flow in the slit-die is characterized by two regions: (i) the inlet in which PSD values reach the maximum and (ii) the exit region in which the PSD value is zero. These regions correspond, respectively to the maximum value of k next to fringe position 0 mm, which is the spatial region where molten polymer reaches the geometry channel, and to the value of $k = 0$ that corresponds to channel length (fringe position = 5 mm). According to the figures, it is possible to observe that the results obtained with the proposed methodology presented good agreement with the results obtained with the GIMP software and the proposed methodology was able to determine the points with maximum retardation order, which are related to the fringe position equal to 0 mm, for all the experiments, with the exception of the experiment for the sample GPPS2 carried out at 0.2 mm/s (Fig. 6.29(d)). In this case, the proposed methodology was not able to properly detect the fringe $k = 8$, due to the overlap of fringes $k = 7$ and $k = 8$ in the image.

The standard deviation (S) of the fringe positions for both techniques is presented in Figs. 6.30 and 6.31. It is possible to observe in Figs. 6.30(b) and 6.31(b), which represent the proposed approach via mathematical morphology, a more predictable behavior of the standard deviation. For these cases, the highest values of standard deviation are between the middle and the exit of the slit, that is, ap-

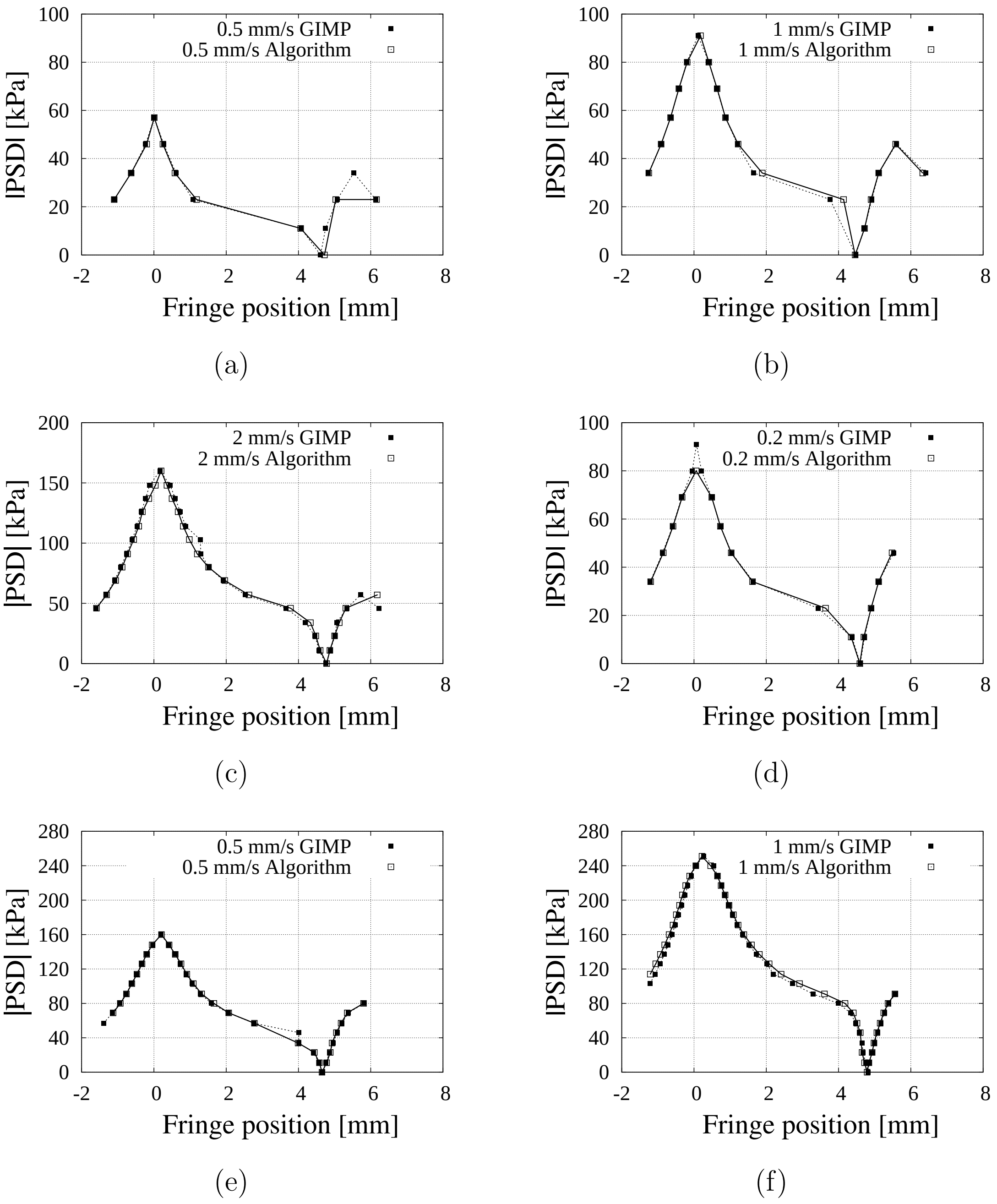
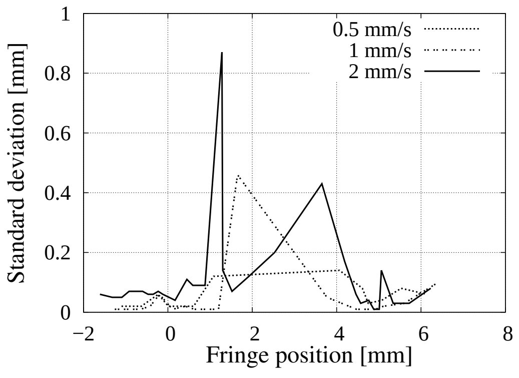
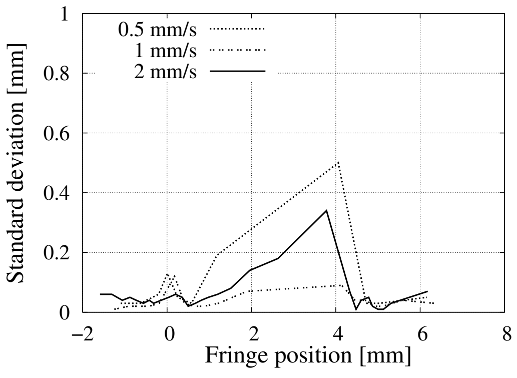


Figure 6.29: |PSD| as a function of distance along centerline. (a) GPPS1 flow at 0.5 mm/s. (b) GPPS1 flow at 1 mm/s. (c) GPPS1 flow at 2 mm/s. (d) GPPS2 flow at 0.2 mm/s. (e) GPPS2 flow at 0.5 mm/s. (f) GPPS2 flow at 1 mm/s.

proximately between fringe position = 2 and 4 mm. Also, experiments with lower velocities, where the fringes are larger and the images display a lower number of fringes, show, in general, larger values of standard deviation. On the other hand, for the GIMP approach, there is no simple pattern for the standard deviation since it also depends on subjective factors. These observations suggest that the morphological approach helps to increase the reproducibility of the results. In addition, the full processing of 15 images, together with the statistical treatment of the results, takes around 30 s. With the GIMP software, the processing must be performed for one image at a time, taking around 8 min for determination of the dark fringe centers for the 15 images tested.

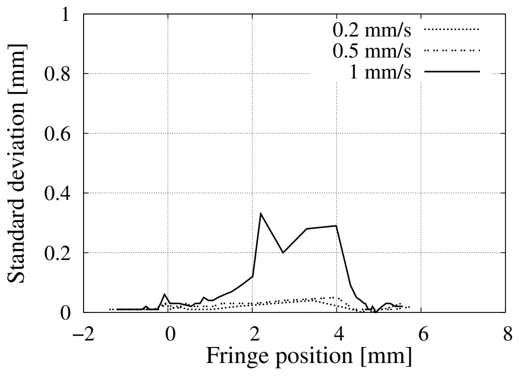


(a)

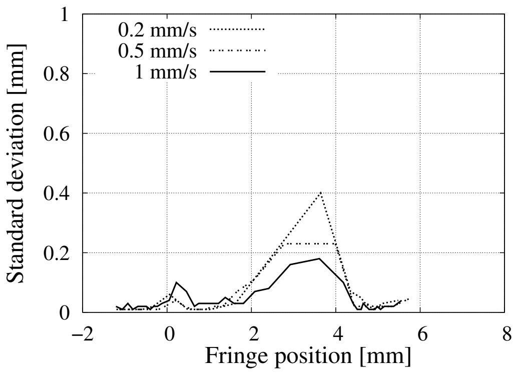


(b)

Figure 6.30: Standard deviation S as a function of distance along centerline of GPPS1. (a) Manual method (GIMP). (b) Proposed morphological approach.



(a)



(b)

Figure 6.31: Standard deviation S as a function of distance along centerline of GPPS2. (a) Manual method (GIMP). (b) Proposed morphological approach.

6.4 Summary

In this chapter, a novel approach to the problem of detecting the position of dark fringes in flow-induced birefringence images obtained during the melt flow in the multipass rheometer was presented and applied for a sample of polystyrene. The proposed methodology allows fast and accurate measurements of the dark fringe centers. In addition, it is an automatic method, which provides the minimum human intervention in the image processing, in contrast to the semi-automatic methodology using GIMP software, presented in [88], in which the user must interfere directly in the measurements. The results obtained have been shown to be consistent with those of semi-automatic detection using GIMP software, with the advantage of a smaller variance of the measurements. Most of the deviation observed is due to difficulties in the image acquisition, which cause some fringes to have low contrast and also make consecutive fringes appear mixed in the recorded image. However, it is important to observe that this problem can be solved with improvements in the experiments, such as lens adjustments in which the passage of polarized light is changed in order to improve the quality of the images allowing clearer and more defined fringes. One should also bear in mind that the amount of time taken to correct those easily identifiable mistakes of the algorithm is still much smaller than the one it takes to perform all the measurements using the GIMP methodology. The obtained results indicate that the proposed method is a reliable alternative to the manual (and semi-automatic) methods currently in use, as it is able to perform the detection of the positions with high confidence and in a faster way than the alternative methods.

Chapter 7

Conclusions and Future Work

This chapter summarizes the results obtained in this thesis and highlights the main contributions. Some ideas for the continuation of the developed activities are also discussed.

7.1 Conclusions

This thesis investigated the use of signal processing and computer vision with a focus on the alignment of signals and identification of events. The study can be divided into three main topics: temporal alignment, spatial alignment and analysis of events.

In the first part, several techniques were studied in order to determine a synchronism between signals. Initially, this thesis showed a method to perform the alignment of two given sets of signals obtained in a surveillance system, considering that they show a periodic behavior and the periods for each set are similar. Therefore, they only required the estimation of a delay to properly align them, which we obtain by maximizing the similarity of the signals. The results are compared to an odometry system and they are shown to be similar to the odometry system results.

We also considered the case where the signals may have different lengths, in which the signal alignment requires a technique more complex than the estimation of a delay. We investigated a temporal alignment algorithm based on a dynamic time-warping approach, showing results on videos signals obtained with a surveillance system application, which allowed us to define the best cost function to be used on video signals. We also studied different signals recorded using this system and assessed the obtention of the alignment for these signals.

Additionally, we ponder the use of an algorithm to estimate the camera trajectory, which can also be used to perform the temporal alignment of two videos while also providing information about their positioning in the environment. We presented an algorithm and tested with several databases. However, we observed that the DORIS videos possess several characteristics that are challenging for the SLAM computation, which require the development of more robust algorithms.

For the second part, we dealt with the problem of the spatial alignment of two images. We discussed the basics of the optical flow algorithm, intending to use it on this problem, showed some improvements and studied other techniques that can be used in this context. We tested the algorithms on a traditional database, and the results showed that classical methods have a better performance. However, some tests on the DORIS system and VDAO database revealed that they have a significant number of imperfections, such as different lighting conditions and objects creating a large region with occluded objects. Based on a qualitative analysis, the approach using a proposed image descriptor was determined as being the one that provides results with more robustness to the illumination changes that occur during a day.

The third part of this thesis contemplated an industrial application that required an analysis through image processing techniques. We proposed an approach to estimate the positions of dark fringes in a given image, which were used to estimate a physical property of a material. The results were compared to other fringe detection methods and to a manual marking using the GIMP software, evidencing the superior performance of the proposed method, which we consider to be a reliable alternative to the manual method currently in use.

7.2 Future Work

In this section, we present the main ideas for future works that can be done in the areas addressed in this thesis.

7.2.1 Online Sequence Synchronization Based on Dynamic Time-Warping

The techniques reported in Chapter 3 have been applied to align videos from the VDAO database in the work proposed in [129], which are used in training of a deep learning classifier to detect video anomalies. During the development of the work [129], it was noticed that if one uses videos with a number of frames per second larger than the one originally employed in the results from this chapter, then the camera trepidation is more influent and can even create a situation where the displacement of the camera between two frames is contrary to the expected movement of the robot. This condition creates a situation where the best alignment between two sequences is not a smooth warping path such as the ones shown in this thesis, but instead a path that should allow a small fluctuation, up to some range that depends on the camera vibration. A possible continuation of this work is to develop a new algorithm able to compensate for the camera vibration.

7.2.2 Camera Trajectory Estimation

The method studied in Chapter 4 performs SLAM intended to operate on any input video. Instead of using a generic algorithm, one could, for instance, include information that is only valid for videos from the application considered in this thesis. Since in the DORIS system the robot moves in fixed rail, one can include the model of the rail in the algorithm, to estimate the camera poses and the similarities in the cases that the algorithm fails, or use the model to help the computation of the epipolar geometry (for example, using the model to guide the choice of the best solution for the essential matrix in the five-point algorithm).

7.2.3 Video Spatial Alignment Using Optical Flow

The tests performed in Chapter 5 used a subset of the Middlebury database which contains the ground truth, since the results for the remainder of the database can only be obtained upon a submission to the developers, which must follow some protocol and may take a certain amount of time. Since in this thesis a more detailed analysis was performed, it was not practical to obtain results on these sequences. A future contribution is to formalize a proposed method and to submit results to this database.

Regarding the results from the VDAO database and DORIS videos, since there is no available ground truth, the only quantitative result was obtained by computing the error between aligned images. This metric is not suitable since the images naturally have differences in illumination and may even contain different objects, whose detection is exactly the purpose for which the databases were developed.

Different strategies could be used to measure a quantitative result. The alignment obtained with the optical flow methods could be compared to the alignment obtained after the computation of a homography, as in [1]. One can also replace the spatial alignment in the object detection framework and assess the detection results. Finally, one can also use the manual marking information provided in the VDAO database, which contains a bounding box for the abandoned objects in the video, to estimate a ground truth for the motion field in the region defined by the bounding box.

7.2.4 Polymer Characterization Using Mathematical Morphology

The method developed in Chapter 6 depends on a previous analysis of the experiment and identification of the pattern to be detected, which was made in [88]. Therefore, a continuation of this work is the inclusion of a pre-processing step responsible for the identification of possible candidates of fringes to be detected by the algorithm, to be subsequently validated by an operator.

Another future contribution is to reproduce the results on different experiments. The method described in this thesis was developed specifically for the application shown in Section 1.2.3. If one performs an experiment with different equipment, the fringes may have another disposition due to being obtained with a different geometry, such as the image observed in Fig 7.1. Since these images show characteristics different from the ones studied in this thesis, the developed methods should be adapted, incorporating other visual clues that are identified in this new experiment.

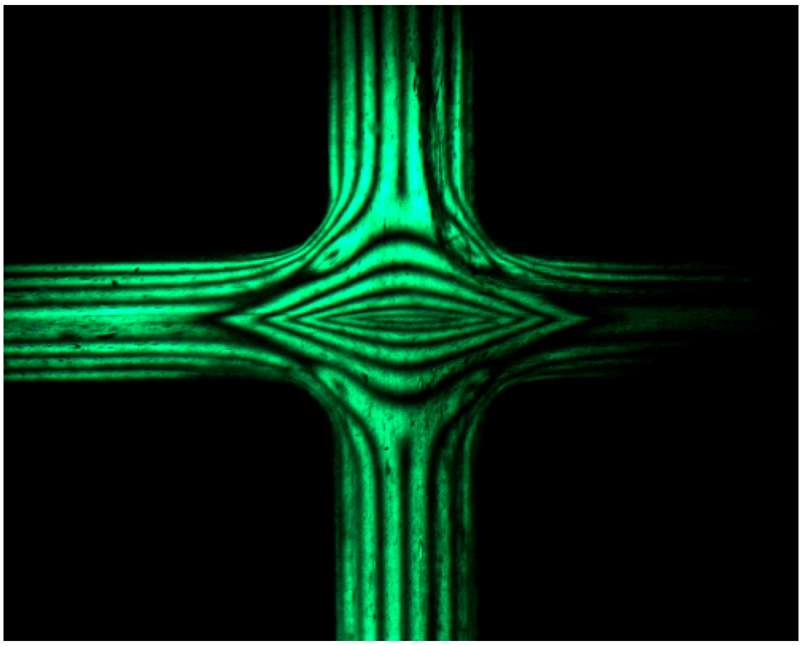


Figure 7.1: Example of birefringence image obtained with another experiment, which shows a different geometry than the one seen in Fig. 1.9.

Bibliography

[1] DE CARVALHO, G. H. F., THOMAZ, L. A., DA SILVA, A. F., et al. "Anomaly detection with a moving camera using multiscale video analysis", *Multidimensional Systems and Signal Processing*, v. 30, n. 1, pp. 311–342, Jan. 2019.

[2] THOMAZ, L. A., JARDIM, E., DA SILVA, A. F., et al. "Anomaly detection in moving-camera video sequences using principal subspace analysis", *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 65, n. 3, pp. 1003–1015, March 2018.

[3] NAKAHATA, M. T., THOMAZ, L. A., DA SILVA, A. F., et al. "Anomaly detection with a moving camera using spatio-temporal codebooks", *Multidimensional Systems and Signal Processing*, v. 29, n. 3, pp. 1025–1054, July 2018.

[4] TOMIOKA, Y., TAKARA, A., KITAZAWA, H. "Generation of an optimum patrol course for mobile surveillance camera". In: *IEEE Transactions on Circuits and Systems for Video Technology*, v. 22, pp. 216–224, Feb. 2012.

[5] BOURMAUD, G., MÉGRET, R. "Robust large scale monocular visual SLAM". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1638–1647, Boston, USA, June 2015.

[6] DE CARVALHO, G. P. S., FREITAS, G. M., COSTA, R. R., et al. "DORIS - monitoring robot for offshore facilities". In: *Offshore Technology Conference*, Rio de Janeiro, Brazil, Oct. 2013.

[7] GALASSI, M., RØYRØY, A., DE CARVALHO, G. P. S., et al. "DORIS - a mobile robot for inspection and monitoring of offshore facilities". In: *Anais do XX Congresso Brasileiro de Automática*, Belo Horizonte, Brazil, Sept. 2014.

[8] FREITAS, R. S., XAUD, M. F. S., MARCOVISTZ, I., et al. "The embedded electronics and software of DORIS offshore robot". In: *IFAC Workshop*

on Automatic Control in Offshore Oil and Gas Production, Florianópolis, Brazil, May 2015.

[9] PREGO, T. M., DE LIMA, A. A., NETTO, S. L., et al. "Audio anomaly detection on rotating machinery using image signal processing". In: *IEEE 7th Latin American Symposium on Circuits Systems (LASCAS)*, pp. 207–210, Feb. 2016.

[10] CARVALHO, G., DE OLIVEIRA, J. F. L., DA SILVA, E. A. B., et al. "Um sistema de monitoramento para detecção de objetos em tempo real empregando câmera em movimento". In: *XXXI Simpósio Brasileiro de Telecomunicações*, Fortaleza, Brazil, Sept. 2013.

[11] DE CARVALHO, G. H. F. *Automatic detection of abandoned objects with a moving camera using multiscale video analysis.* PHD Thesis, COPPE - Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2015.

[12] DE LIMA, A. A., PREGO, T. M., NETTO, S. L., et al. "On fault classification in rotating machines using fourier domain features and neural networks". In: *IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS)*, pp. 1–4, Feb. 2013.

[13] HARTLEY, R., ZISSERMAN, A. *Multiple view geometry in computer vision.* Cambridge University Press, 2004.

[14] MACKLEY, M., HASSELL, D. "The multipass rheometer: a review", *Journal of Non-Newtonian Fluid Mechanics*, v. 166, n. 9, pp. 421–456, 2011.

[15] LORD, T. D., SCELSI, L., HASSELL, D. G., et al. "The matching of 3D Rolie-Poly viscoelastic numerical simulations with experimental polymer melt flow within a slit and a cross-slot geometry", *Journal of Rheology*, v. 54, n. 2, pp. 355–373, 2010.

[16] HASSELL, D. G., MACKLEY, M. R. "An experimental evaluation of the behaviour of mono and polydisperse polystyrenes in Cross-Slot flow", *Rheologica Acta*, v. 48, n. 5, pp. 543–550, 2009.

[17] HASSELL, D. G., AUHL, D., MCLEISH, T. C. B., et al. "The effect of viscoelasticity on stress fields within polyethylene melt flow for a cross-slot and contraction–expansion slit geometry", *Rheologica Acta*, v. 47, n. 7, pp. 821–834, 2008.

[18] COLLIS, M., MACKLEY, M. "The melt processing of monodisperse and polydisperse polystyrene melts within a slit entry and exit flow", *Journal of Non-Newtonian Fluid Mechanics*, v. 128, n. 1, pp. 29–41, 2005.

[19] LEE, K., MACKLEY, M. "The application of the multi-pass rheometer for precise rheo-optic characterisation of polyethylene melts", *Chemical Engineering Science*, v. 56, n. 19, pp. 5653–5661, 2001.

[20] DOUZE, M., REVAUD, J., VERBEEK, J., et al. "Circulant temporal encoding for video retrieval and temporal alignment", *International Journal of Computer Vision*, v. 119, n. 3, pp. 291–306, 2016.

[21] DAI, C., ZHENG, Y., LI, X. "Accurate video alignment using phase correlation", *IEEE Signal Processing Letters*, v. 13, n. 12, pp. 737–740, Dec. 2006.

[22] CASPI, Y., IRANI, M. "Spatio-temporal alignment of sequences", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 11, pp. 1409–1424, Nov. 2002.

[23] DIEGO, F., PONSA, D., SERRAT, J., et al. "Video alignment for change detection", *IEEE Transactions on Image Processing*, v. 20, n. 7, pp. 1858–1869, July 2011.

[24] CHUPEAU, B., OISEL, L., JOUET, P. "Temporal video registration for watermark detection". In: *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, v. 2, pp. 157–160, May 2006.

[25] ITAKURA, F. "Minimum prediction residual principle applied to speech recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 23, n. 1, pp. 67–72, Feb. 1975.

[26] SAKOE, H., CHIBA, S. "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 26, n. 1, pp. 43–49, Feb. 1978.

[27] YUAN, Y., CHEN, Y.-P. P., NI, S., et al. "Development and application of a modified dynamic time warping algorithm (DTW-S) to analyses of primate brain expression time series", *BMC Bioinformatics*, v. 12, n. 1, 2011.

[28] KAPPELER, A., ILIADIS, M.AND WANG, H., KATSAGGELOS, A. K. "Block based video alignment with linear time and space complexity". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 3324–3328, Sept. 2016.

[29] RAKTHANMANON, T., CAMPANA, B., MUEEN, A., et al. "Searching and mining trillions of time series subsequences under dynamic time warping". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 262–270, Aug. 2012.

[30] SILVA, D. F., BATISTA, G. E. A. P. A. "Speeding up all-pairwise dynamic time warping matrix calculation". In: *SIAM International Conference on Data Mining*, pp. 837–845, Miami, USA, May 2016.

[31] SALVADOR, S., CHAN, P. "Toward accurate dynamic time warping in linear time and space", *Intelligent Data Analysis*, v. 11, n. 5, pp. 561–580, Oct. 2007.

[32] DIXON, S. "Live tracking of musical performances using on–line time warping". In: *8th International Conference on Digital Audio Effects (DAFx)*, pp. 92–97, Madrid, Spain, Sept. 2005.

[33] KYUTOKU, H., DEGUCHI, D., TAKAHASHI, T., et al. "Subtraction-based forward obstacle detection using illumination insensitive feature for driving-support". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 515–525, Berlin, Germany, 2012.

[34] MUKOJIMA, H., DEGUCHI, D., KAWANISHI, Y., et al. "Moving camera background-subtraction for obstacle detection on railway tracks". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 3967–3971, Phoenix, USA, Sept. 2016.

[35] KONG, H., AUDIBERT, J. Y., PONCE, J. "Detecting abandoned objects with a moving camera", *IEEE Transactions on Image Processing*, v. 19, n. 8, pp. 2201–2210, Aug. 2010.

[36] CASTELLANOS, J. A., MONTIEL, J. M. M., NEIRA, J., et al. "The SPmap: a probabilistic framework for simultaneous localization and map building", v. 15, pp. 948–953, 1999.

[37] DISSANAYAKE, M., NEWMAN, P., CLARK, S., et al. "Photographing long scenes with multi-viewpoint panoramas". In: *IEEE Transactions on Robotics and Automation*, v. 17, pp. 229–241, 2001.

[38] ENDRES, F., HESS, J., ENGELHARD, N., et al. "An evaluation of the RGB-D SLAM system". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1691 – 1696, Saint Paul, MN, 2012.

[39] STEDER, B., GRISETTI, G., STACHNISS, C., et al. "Visual SLAM for flying vehicles". In: *IEEE Transactions on Robotics*, v. 24, pp. 1088–1093, Oct. 2008.

[40] DAVISON, A. J. "Real-time simultaneous localisation and mapping with a single camera". In: *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV)*, pp. 1403–1411, Washington, DC, USA, 2003.

[41] DAVISON, A. J. "SLAM with a single camera". In: *SLAM/CML Workshop at ICRA*, 2002.

[42] SALAS-MORENO, R. F., NEWCOMBE, R. A., STRASDAT, H., et al. "SLAM++: simultaneous localisation and mapping at the level of objects". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1352–1359, Portland, USA, 2013.

[43] KONOLIGE, K., AGRAWAL, M. "FrameSLAM: from bundle adjustment to real-time visual mapping", *IEEE Transactions on Robotics*, v. 24, n. 5, pp. 1066–1077, Oct. 2008.

[44] CLEMENTE, L., DAVISON, A., REID, I., et al. "Mapping large loops with a single hand-held camera". Atlanta, USA, June 2007.

[45] TRIGGS, B., MCLAUCHLAN, P. F., HARTLEY, R. I., et al. "Bundle adjustment - a modern synthesis". In: *International Workshop on Vision Algorithms: Theory and Practice*, pp. 298–372, London, UK, 2000.

[46] ENGEL, J., SCHÖPS, T., CREMERS, D. "LSD-SLAM: large-scale direct monocular SLAM". In: *European Conference on Computer Vision (ECCV)*, pp. 834–849, Zurich, Switzerland, Sept. 2014.

[47] KLOPSCHITZ, M., ZACH, C., IRSCHARA, A., et al. "Generalized detection and merging of loop closures for video sequences". In: *International Symposium on 3D Data Processing, Visualization and Transmission*, Atlanta, USA, Sept. 2008.

[48] CHIRIKJIAN, G. S. *Stochastic models, information theory, and lie groups.* Springer-Verlag, 2012.

[49] HORN, B. K., SCHUNCK, B. G. *Determining optical flow.* Technical report, Cambridge, USA, 1980.

[50] SUN, D., ROTH, S., BLACK, M. J. "A quantitative analysis of current practices in optical flow estimation and the principles behind them", *International Journal of Computer Vision*, v. 106, n. 2, pp. 115–137, Jan. 2014.

[51] BROX, T., BRUHN, A., PAPENBERG, N., et al. "High accuracy optical flow estimation based on a theory for warping". In: *European Conference on Computer Vision (ECCV)*, v. 3024, pp. 25–36, Prague, Czech Republic, May 2004.

[52] WEDEL, A., POCK, T., BRAUN, J., et al. "Duality TV-L1 flow with fundamental matrix prior". In: *23rd International Conference Image and Vision Computing New Zealand*, pp. 1–6, Nov. 2008.

[53] WEDEL, A., POCK, T., ZACH, C., et al. "Statistical and geometrical approaches to visual motion analysis". chap. An improved algorithm for TV-L1 optical flow, pp. 23–45, Berlin, Heidelberg, 2009.

[54] BLACK, M. J., ANANDAN, P. "The robust estimation of multiple motions: parametric and piecewise-smooth flow fields", *Computer Vision and Image Understanding*, v. 63, n. 1, pp. 75–104, 1996.

[55] BRUHN, A., WEICKERT, J., SCHNÖRR, C. "Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods", *International Journal of Computer Vision*, v. 61, n. 3, pp. 211–231, Feb. 2005.

[56] BLAKE, A., ZISSERMAN, A. *Visual reconstruction.* MIT Press, 1987.

[57] ZIMMER, H., BRUHN, A., WEICKERT, J. "Optic flow in harmony", *International Journal of Computer Vision*, v. 93, n. 3, pp. 368–388, July 2011.

[58] PARK, S., KWAK, N. "Illumination robust optical flow estimation by illumination-chromaticity decoupling". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 1910–1914, Quebec City, Canada, Sept. 2015.

[59] MILEVA, Y., BRUHN, A., WEICKERT, J. "Illumination-robust variational optical flow with photometric invariants". In: *Proceedings of the 29th DAGM Conference on Pattern Recognition*, pp. 152–162, Berlin, Heidelberg, 2007.

[60] DRULEA, M., NEDEVSCHI, S. "Motion estimation using the correlation transform", *IEEE Transactions on Image Processing*, v. 22, n. 8, pp. 3260–3270, Aug. 2013.

[61] MOHAMED, M. A., RASHWAN, H. A., MERTSCHING, B., et al. "Illumination-robust optical flow using a local directional pattern", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 24, n. 9, pp. 1499–1508, Sept. 2014.

[62] ALI, S., DAUL, C., GALBRUN, E., et al. "Illumination invariant optical flow using neighborhood descriptors", *Computer Vision and Image Understanding*, v. 145, n. C, pp. 95–110, April 2016.

[63] WEINZAEPFEL, P., REVAUD, J., HARCHAOUI, Z., et al. "DeepFlow: large displacement optical flow with deep matching". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1385–1392, Washington, USA, 2013.

[64] REVAUD, J., WEINZAEPFEL, P., HARCHAOUI, Z., et al. "EpicFlow: edge-preserving interpolation of correspondences for optical flow". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1164–1172, Boston, USA, June 2015.

[65] FEI-FEI, L., PERONA, P. "A Bayesian hierarchical model for learning natural scene categories". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, v. 2, pp. 524–531, Washington, DC, USA, June 2005.

[66] BAILER, C., TAETZ, B., STRICKER, D. "Flow fields: dense correspondence fields for highly accurate large displacement optical flow estimation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 2017.

[67] DOSOVITSKIY, A., FISCHER, P., ILG, E., et al. "FlowNet: learning optical flow with convolutional networks". In: *IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, Dec. 2015.

[68] ILG, E., MAYER, N., SAIKIA, T., et al. "FlowNet 2.0: evolution of optical flow estimation with deep networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, July 2017.

[69] RANJAN, A., BLACK, M. J. "Optical flow estimation using a spatial pyramid network". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2720–2729, Honolulu, Hawaii, July 2017.

[70] HUI, T.-W., TANG, X., LOY, C. C. "LiteFlowNet: a lightweight convolutional neural network for optical flow estimation". In: *IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), pp. 8981–8989, Salt Lake City, USA, June 2018.

[71] SUN, D., YANG, X., LIU, M.-Y., et al. "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, Sept. 2018.

[72] HASSELL, D. G., LORD, T. D., SCELSI, L., et al. "The effect of boundary curvature on the stress response of linear and branched polyethylenes in a contraction–expansion flow", *Rheologica Acta*, v. 50, n. 7, pp. 675–689, Aug. 2011.

[73] REYNOLDS, C., THOMPSON, R., MCLEISH, T. "Pressure and shear rate dependence of the viscosity and stress relaxation of polymer melts", *Journal of Rheology*, v. 62, n. 2, pp. 631–642, 2018.

[74] FAROOQ, M., ASLAM, A., HUSSAIN, B., et al. "A comparison of image processing techniques for optical interference fringe analysis", *Photonic Sensors*, v. 5, n. 4, pp. 304–311, 2015.

[75] YE, G., WEI, L. "A method for interference fringe fast skeletonizing". In: *2nd International Conference on Computer Science and Network Technology*, pp. 1784–1786, Changchun, China, Dec. 2012.

[76] YU, Q., ANDRESEN, K. "Fringe-orientation maps and fringe skeleton extraction by the two-dimensional derivative-sign binary-fringe method", *Applied Optics*, v. 33, n. 29, pp. 6873–6878, 1994.

[77] EL-MORSY, M. "A new algorithm for automatic double bright fringe of multiple-beam fizeau fringe skeletonization using Fourier transform method of fringe pattern analysis", *Journal of Signal and Information Processing*, v. 3, n. 3, pp. 412–419, 2012.

[78] SOKKAR, T., DESSOUKY, H. E., SHAMS-ELDIN, M., et al. "Automatic fringe analysis of two-beam interference patterns for measurement of refractive index and birefringence profiles of fibres", *Optics and Lasers in Engineering*, v. 45, n. 3, pp. 431 – 441, 2007.

[79] POON, C. Y., KUJAWINSKA, M., RUIZ, C. "Automated fringe pattern analysis for Moiré interferometry", *Experimental Mechanics*, v. 33, n. 3, pp. 234–241, Sept. 1993.

[80] HUNTLEY, J. M. "Automated fringe pattern analysis in experimental mechanics: A review", *The Journal of Strain Analysis for Engineering Design*, v. 33, n. 2, pp. 105–125, 1998.

[81] FULONG, D., WANG, Z. "Automatic fringe patterns analysis using digital processing techniques: I. Fringe center method", *Acta Photonica Sinica*, v. 28, n. 8, pp. 700–6, 1999.

[82] WANG, Z., HAN, B. "Enhanced random phase shifting technique". In: *X SEM International Congress & Exposition on Experimental & Applied Mechanics*, Costa Mesa, USA, June 2004.

[83] MEYER, F. "Iterative image transformations for an automatic screening of cervical smears", *Journal of Histochemistry & Cytochemistry*, v. 27, n. 1, pp. 128–135, 1979.

[84] ZHANG, D., MA, M., AROLA, D. D. "Fringe skeletonizing using an improved derivative sign binary method", *Optics and Lasers in Engineering*, v. 37, n. 1, pp. 51–62, 2002.

[85] AGASSANT, J.-F., BAAIJENS, F., BASTIAN, H., et al. "The matching of experimental polymer processing flows to viscoelastic numerical simulation", *International Polymer Processing*, v. 17, n. 1, pp. 3–10, 2002.

[86] THOMAZ, L. A., DA SILVA, A. F., DA SILVA, E. A. B., et al. "A morphological approach to the automatic detection of dark fringes applied to birefringence images". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 739–743, Phoenix, USA, Sept. 2016.

[87] FARIAS, T. M., SECCHI, A. R., BUTLER, S., et al. "Utilização da técnica de birrefringência em reômetro multipasse para a diferenciação de grades de poliestireno cristal", *Polímeros*, v. 24, n. 5, pp. 596–603, 2014.

[88] CASTRO, A. M., PEREIRA, J. O., FARIAS, T. M., et al. "Application of the GIMP software in the analysis of birefringence images obtained in a multipass rheometer", *Rheologica Acta*, v. 57, n. 2, pp. 113–126, 2018.

[89] MATLAB. *version 8.0.0.783 (R2012b)*. Natick, Massachusetts, The MathWorks Inc., 2012.

[90] MATSUYAMA, T., OHYA, T., HABE, H. "Background subtraction for non-stationary scene". In: *Asian Conference on Computer Vision (ACCV)*, pp. 622–667, Taipei, Taiwan, Jan. 2000.

[91] WANG, Z., BOVIK, A. C., SHEIKH, H. R., et al. "Image quality assessment: from error visibility to structural similarity", *IEEE Transactions on Image Processing*, v. 13, n. 4, pp. 600–612, April 2004.

[92] DALAL, N., TRIGGS, B. "Histograms of oriented gradients for human detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893, Washington, USA, June 2005.

[93] JUNIOR, O. L., DELGADO, D., GONCALVES, V., et al. "Trainable classifier-fusion schemes: an application to pedestrian detection". In: *12th International IEEE Conference on Intelligent Transportation Systems*, pp. 1–6, Oct. 2009.

[94] DA SILVA, A. F., THOMAZ, L. A., CARVALHO, G., et al. "An annotated video database for abandoned-object detection in a cluttered environment". In: *International Telecommunications Symposium (ITS)*, pp. 1–5, São Paulo, Brazil, Aug. 2014.

[95] LUCAS, B. D., KANADE, T. "An iterative image registration technique with an application to stereo vision". In: *7th International Joint Conference on Artificial Intelligence*, Vancouver, Canada, 1981.

[96] HARRIS, C., STEPHENS, M. "A combined corner and edge detection". In: *Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[97] BAY, H., ESS, A., TUYTELAARS, T., et al. "Speeded-up robust features (SURF)", *Computer Vision and Image Understanding (CVIU)*, v. 110, n. 3, pp. 346–359, June 2008.

[98] NISTER, D. "An efficient solution to the five-point relative pose problem", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 6, pp. 756–770, June 2004.

[99] OLSSON, C., ERIKSSON, A., HARTLEY, R. "Outlier removal using duality". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1450–1457, San Francisco, USA, June 2010.

[100] SIVIC, J., ZISSERMAN, A. "Efficient visual search of videos cast as text retrieval", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 31, n. 4, pp. 591–606, April 2009.

[101] UMEYAMA, S. "Least-squares estimation of transformation parameters between two point patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 13, n. 4, pp. 376–380, April 1991.

[102] ABSIL, P.-A., MAHONY, R., SEPULCHRE, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[103] GEIGER, A., LENZ, P., URTASUN, R. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.

[104] GONZALEZ, R., RODRIGUEZ, F., GUZMAN, J. L., et al. "Combined visual odometry and visual compass for off-road mobile robots localization", *Robotica*, v. 30, n. 6, pp. 865–878, 2012.

[105] NOURANI-VATANI, N., BORGES, P. V. K. "Correlation-based visual odometry for ground vehicles", *Journal of Field Robotics*, v. 28, n. 5, pp. 742–768, 2011.

[106] SCARAMUZZA, D., FRAUNDORFER, F. "Visual odometry [Tutorial]", *IEEE Robotics Automation Magazine*, v. 18, n. 4, pp. 80–92, Dec. 2011.

[107] HUBER, P., WILEY, J., INTERSCIENCE, W. *Robust statistics*. Wiley New York, 1981.

[108] XIAO, J., CHENG, H., SAWHNEY, H., et al. "Bilateral filtering-based optical flow estimation with occlusion detection". In: *European Conference on Computer Vision (ECCV)*, pp. 211–224, Berlin, Heidelberg, 2006.

[109] RITTNER, L., FLORES, F. C., LOTUFO, R. A. "A tensorial framework for color images", *Pattern Recognition Letters*, v. 31, n. 4, pp. 277–296, March 2010.

[110] BIGUN, J., GRANLUND, G. H. "Optimal orientation detection of linear symmetry". In: *IEEE International Conference on Computer Vision (ICCV)*, London, UK, June 1987.

[111] BISHOP, R. L., GOLDBERG, S. I. *Tensor analysis on manifolds*. Dover Publications, 1980.

[112] BAKER, S., SCHARSTEIN, D., LEWIS, J. P., et al. "A database and evaluation methodology for optical flow", *International Journal of Computer Vision*, v. 92, n. 1, pp. 1–31, 2011.

[113] R WIEGELL, M., TUCH, D., B W LARSSON, H., et al. "Automatic segmentation of thalamic nuclei from diffusion tensor magnetic resonance imaging", *NeuroImage*, v. 19, pp. 391–401, July 2003.

[114] ZIYAN, U., TUCH, D., WESTIN, C.-F. "Segmentation of thalamic nuclei from DTI using spectral clustering". In: *9th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, v. 9, pp. 807–14, Copenhagen, Denmark, Feb. 2006.

[115] PENNEC, X., FILLARD, P., AYACHE, N. "A Riemannian framework for tensor computing", *International Journal of Computer Vision*, v. 66, n. 1, pp. 41–66, Jan. 2006.

[116] FILLARD, P., ARSIGNY, V., AYACHE, N., et al. "A Riemannian framework for the processing of tensor-valued images". In: *Deep Structure, Singularities, and Computer Vision*, pp. 112–123, Berlin, Heidelberg, 2005.

[117] RUDIN, L. I., OSHER, S., FATEMI, E. "Nonlinear total variation based noise removal algorithms", *Physica D*, v. 60, n. 1-4, pp. 259–268, Nov. 1992.

[118] BARRON, J. L., FLEET, D. J., BEAUCHEMIN, S. S. "Performance of optical flow techniques", *International Journal of Computer Vision*, v. 12, n. 1, pp. 43–77, Feb. 1994.

[119] OTTE, M., NAGEL, H. H. "Optical flow estimation: advances and comparisons". In: Eklundh, J.-O. (Ed.), *European Conference on Computer Vision (ECCV)*, pp. 49–60, Berlin, Heidelberg, 1994.

[120] HAFNER, D., DEMETZ, O., WEICKERT, J. "Why is the census transform good for robust optic flow computation?" In: *Scale space and variational methods in computer vision*, pp. 210–221, Berlin, Heidelberg, 2013.

[121] AHMED, R., LIANG, R., MACKLEY, M. "The experimental observation and numerical prediction of planar entry flow and die swell for molten polyethylenes", *Journal of Non-Newtonian Fluid Mechanics*, v. 59, n. 2, pp. 129–153, 1995.

[122] MACOSKO, C. W., LARSON, R. G. *Rheology: principles, measurements, and applications.* 1 ed. New York, Wiley, 1994.

[123] FULLER, G. G. *Optical rheometry of complex fluids.* New York, Oxford University Press, Inc, 1995.

[124] SERRA, J. *Image analysis and mathematical morphology.* Orlando, USA, Academic Press, Inc., 1983.

[125] SOILLE, P. *Morphological image analysis: principles and applications.* 2 ed. Secaucus, NJ, USA, Springer-Verlag New York, Inc., 2003.

[126] BEUCHER, S., LANTUÉJ, C. "Use of watersheds in contour detection". In: *International workshop on image processing, real-time edge and motion detection*, 1979.

[127] HARALICK, R. M., SHAPIRO, L. G. *Computer and robot vision.* 1st ed. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 1992.

[128] YU, Q., LIU, X., ANDRESEN, K. "New spin filters for interferometric fringe patterns and grating patterns", *Applied Optics*, v. 33, n. 17, pp. 3705–3711, 1994.

[129] AFONSO, B. M., CINELLI, L. P., THOMAZ, L. A., et al. "Moving-camera video surveillance in cluttered environments using deep features". In: *IEEE International Conference on Image Processing (ICIP)*, pp. 2296–2300, Athens, Greece, Oct. 2018.

[130] XU, G., ZHANG, Z. *Epipolar geometry in stereo, motion and object recognition.* Kluwer Academic Publishers, 1996.

[131] LOWE, D. G. "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, 2004.

[132] LEUTENEGGER, S., CHLI, M., SIEGWART, Y. "BRISK: binary robust invariant scalable keypoints". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2548–2555, 2011.

[133] ALAHI, A., ORTIZ, R., VANDERGHEYNST, P. "FREAK: fast retina keypoint". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 510 – 517.

[134] LONGUET-HIGGINS, H. C. "A computer algorithm for reconstructing a scene from two projections", *Nature*, v. 293, n. 5828, pp. 133–135, Sept. 1981.

[135] ZHANG, Z. "Determining the epipolar geometry and its uncertainty: a review", *International Journal of Computer Vision*, v. 27, n. 2, pp. 161–195, April 1998.

[136] LUONG, Q.-T., VIÉVILLE, T. "Canonical representations for the geometries of multiple projective views", *Computer Vision and Image Understanding*, v. 64, n. 2, pp. 193–229, Sept. 1996.

[137] FARAUT, J. *Analysis on Lie groups: an introduction.* Cambridge University Press, 2008.

[138] BOURMAUD, G. *Estimation de paramètres évoluant sur des groupes de Lie: Application à la cartographie et à la localisation d'une caméra monoculaire.* PHD Thesis, Université de Bordeaux, Bordeaux, France, 2015.

Appendix A

Camera Models and Projective Geometry

In this section, several concepts related to projection of the real world into a camera are depicted. This subject is of paramount importance to the understanding of the algorithms to recover the camera trajectory from a video sequence.

If several cameras record the same scene in different positions, there is a relationship between the positioning of the cameras and the images they create. As can be seen in Fig. A.1, image points in multiple views that represent the same three-dimensional point define an intrinsic geometrical property among the cameras.

In this example, a three-dimensional point $\mathbf{X}$ is projected in the left image onto the point $\mathbf{x}$, and the projection ray passes through the camera center $\mathbf{C}$. The point $\mathbf{x}'$ is also a projection of the $\mathbf{X}$ on the right image, which passes through the camera center $\mathbf{C}'$. Since the projection rays are assumed to be straight lines, the knowledge of an image point $\mathbf{x}$ in one image imposes a restriction on the position of the image point $\mathbf{x}'$ in the other image. The next sections define a model for the camera and for the geometry between multiple views, and show how to explore these geometrical properties to infer information about the camera positioning based on the image content.

A.1 Homogeneous Coordinates

The mathematical definition of the camera projection and the relationship between images can be simplified if one considers the use of homogeneous coordinates. A point in the space $\mathbb{R}^2$ is usually represented by a vector $(x, y)^T$. By extending the vector to include a third component, for example, $(x, y, 1)^T$ it is said that the vector is represented in homogeneous coordinates. This representation has the advantage of allowing a simplification of several operations.

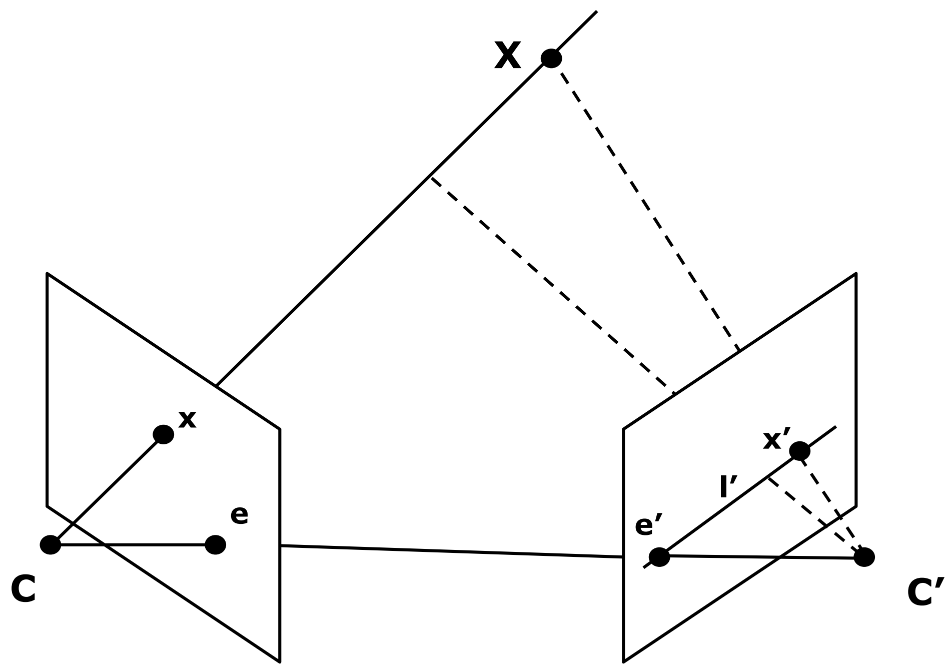


Figure A.1: Intrinsic geometry for two cameras representing the same scene. The three-dimensional point **X** is projected in the left image onto the point **x** and in the right image onto the point **x**′. The projection for each image is defined by the image plane and the camera center, **C** and **C**′ respectively for the left and right cameras.

For instance, a line in the space $\mathbb{R}^2$ is the set of points where the relation $ax + by + c = 0$ is valid. Based on this definition, a line can be represented as the vector $\mathbf{l} = (a, b, c)^T$. In order to test if the point $\mathbf{x} = (x, y, 1)^T$, in homogeneous coordinates, belongs to the line **l**, it is necessary and sufficient to compute the internal product between **x** and **l**, that is

$$\mathbf{x}^T\mathbf{l} = (x, y, 1)(a, b, c)^T = ax + by + c, \tag{A.1}$$

which is zero if **x** belongs to the line **l**. One should notice that if the vector $(x, y, 1)$ belongs to the line **l**, any vector of the form $(kx, ky, k)^T$ also belongs to this line, so that the set of vectors $(kx, ky, k)^T$ represent the same point $(x, y)^T$. Since the factor k can be arbitrary, it is often defined as $k = 1$.

Another simplification is the definition of a point at infinity $\mathbf{x}_\infty$. Consider two parallel lines $\mathbf{l_1} = (a, b, c)^T$ and $\mathbf{l_2} = (a, b, c')^T$. The point where the two lines intersect can be found by using the following expression [13]:

$$\mathbf{x}_\infty = \mathbf{l_1} \times \mathbf{l_2} = (c - c')(b, -a, 0)^T, \tag{A.2}$$

where one can see that it is mathematically impossible to normalize the coordinates such that the last coordinate is equal to one. However, it is known that $\mathbf{x}_\infty$ is obtained as the solution to the intersection of two parallel lines, therefore should represent a point at infinity. Thereby, any vector in homogeneous coordinates defined as $(x, y, 0)^T$ represents a point at infinity.

A.2 Camera Model

A camera is a device that captures still images by mapping points in the three-dimensional space onto a projection plane. A simple approach to understand the operation of a camera is the model of a pinhole camera. It considers that the camera is composed of a box with a tiny aperture and a projection surface, without any lens in the exterior, and every light ray passes through the aperture and projects an inverted image onto a surface in the opposite side of the camera.

For convention, to further simplify the mathematics, it is often defined a virtual projection plane placed in front of the camera. In this paradigm, the projection occurs when the light ray crosses the projection plane towards the aperture, which is called the camera center. Considering the camera center as the origin of a coordinate system, the point $\mathbf{X}$ in the three-dimensional space and the equivalent point $\mathbf{x}$ in the projected image space, and considering the projection plane to be perpendicular to the z axis passing through the point $Z = f$, the projection of the point $\mathbf{X}$ to the point $\mathbf{x}$ can be expressed as:

$$\mathbf{x} = \mathbf{PX} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} fX/Z \\ fY/Z \\ 1 \end{bmatrix}, \tag{A.3}$$

where the matrix $\mathbf{P}$ that transforms the point $\mathbf{X}$ to the point $\mathbf{x}$ is called the camera matrix and f is the focal length.

In Fig. A.2, the projection $\mathbf{p}$ of the camera center is considered as the origin of a coordinate system in the image. However, the origin of the coordinate system of an image is often defined as the top left or down left positions. If one wants to translate the coordinate system to a different location, the camera center must be compensated in Eq. (A.3), leading to the following equation:

$$\mathbf{x} = \begin{bmatrix} fX/Z + p_x \\ fY/Z + p_y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{A.4}$$

If the coordinate system of the three-dimensional space should also be rotated or translated, which is the case for instance if there are multiple cameras to be

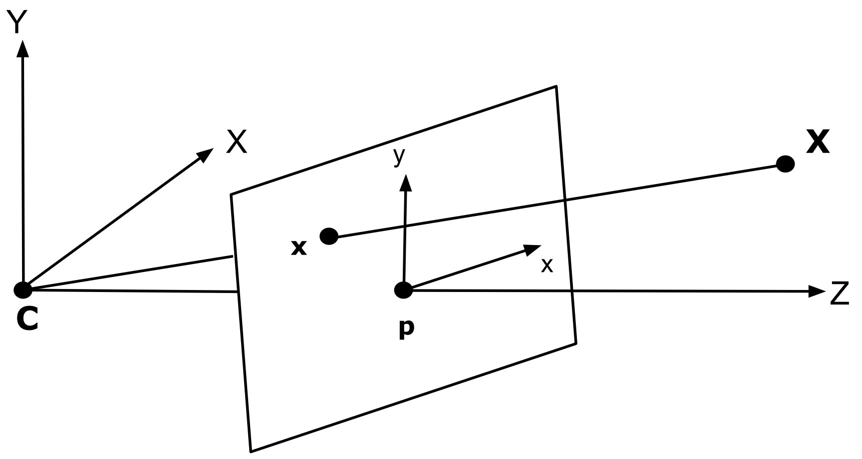


Figure A.2: Projection onto an image plane performed by a pinhole camera. The point **X** in the three-dimensional space is projected to the point **x** in the image plane through the light ray that crosses the camera center **C**.

modeled according to the same reference, the camera model has to be further refined to also cope with this change in the coordinates. The model is adapted to include a transformation that rotates and translates the coordinate system to coincide with the coordinate centered on the camera. The camera model becomes:

$$\mathbf{x} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{X} = \mathbf{K} \begin{bmatrix} \mathbf{R} & | & \mathbf{t} \end{bmatrix} \mathbf{X} = \mathbf{P}\mathbf{X}, \tag{A.5}$$

with

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.6}$$

The matrix **R** and the vector **t** represent, respectively, a rotation and a translation of the camera with respect to space coordinates, and are called the extrinsic parameters of the camera. The matrix **K** is called the calibration matrix and summarizes the intrinsic parameters of the camera, which are related to the projection of the three-dimensional points to generate the image points.

It is also possible to derive the expression of the camera center given the camera matrix **P**. Consider the points **A** and **C**, with **C** having the property that $\mathbf{PC} = \mathbf{0}$. Any point that belongs to the line connecting **A** and **C** can be generated using the following expression:

$$\mathbf{X} = \lambda\mathbf{A} + (1 - \lambda)\mathbf{C}, \tag{A.7}$$

where λ is a variable used to parameterize a point in the line. The projection of the point $\mathbf{X}$ in the image is:

$$\mathbf{x} = \mathbf{PX} = \lambda\mathbf{PA} + (1 - \lambda)\mathbf{PC} = \lambda\mathbf{PA}. \tag{A.8}$$

It should be noted that any point $\mathbf{X}$ defined by Eq. (A.7) possesses the same image point $\mathbf{x} = \lambda\mathbf{PA}$. One can conclude that this line represents a projection ray, and since there was no assumption about the point $\mathbf{A}$, the remaining point $\mathbf{C}$ such that $\mathbf{PC} = \mathbf{0}$ must be the camera center.

An even more generic model considers other effects. In CCD cameras, a pixel may not be square, which happens when the camera has different focal lengths in the horizontal (f_x) and vertical (f_y) directions. A camera may also have a shear distortion in the projected image, exemplified by the factor s, which occurs when the image axes x and y are not perpendicular. The camera model considering the aforementioned distortions is:

$$\mathbf{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.9}$$

A.3 Fundamental Matrix

As seen in Fig. A.1, given an image point from one camera and a second camera recording the same scene, any point in the second camera corresponding to a point in the first camera necessarily must belong to a line that also contains the projection of the camera center of the first camera. The fundamental matrix is an object that summarizes the geometric relation between points from the two views.

The geometric relation between the image points can be explained as follows. Consider two cameras with known camera matrices $\mathbf{P}$ and $\mathbf{P}'$, and an image point $\mathbf{x}$ in the first image that is the projection of the three-dimensional point $\mathbf{X}$. The projection ray that creates the image point can be defined by two points: the camera center, where $\mathbf{PC} = 0$, and any point that respects the relation $\mathbf{x} = \mathbf{PX}$. According to [130], the second point can be obtained by computing the pseudoinverse of $\mathbf{P}$, as $\mathbf{X}^+ = \mathbf{P}^+\mathbf{x}$. The projection ray is a line defined as the set of points $\mathbf{X}(\lambda)$, for a parameter λ,such that:

$$\mathbf{X}(\lambda) = \mathbf{X}^+ + \lambda\mathbf{C}. \tag{A.10}$$

In the second view, the projection of any projection ray such as the one defined in Eq. (A.10) is called an epipolar line, and it passes through the projection of the two known points that were used to define it, $\mathbf{X}^+$ and $\mathbf{C}$. The projection of the camera center is represented by $\mathbf{e}'$ and is called the epipole. The image points that define this epipolar line are:

$$\mathbf{e}' = \mathbf{P}'\mathbf{C}, \tag{A.11}$$

and

$$\mathbf{x}^{+} = \mathbf{P}'\mathbf{X}^{+} = \mathbf{P}'\mathbf{P}^{+}\mathbf{x}. \tag{A.12}$$

Representing the epipolar line as a vector, one can write [13]:

$$\mathbf{l}' = \mathbf{e}' \times \mathbf{x}^{+} = [\mathbf{e}']_{\times}\mathbf{P}'\mathbf{P}^{+}\mathbf{x} = \mathbf{F}\mathbf{x}, \tag{A.13}$$

and

$$\mathbf{F} = [\mathbf{e}']_{\times}\mathbf{P}'\mathbf{P}^{+}. \tag{A.14}$$

where $[\mathbf{e}']_{\times}$ is an antisymmetric matrix created from the components of $\mathbf{e}' = (e'_1, e'_2, e'_3)^T$ in order to transform a vectorial product into a scalar product, using the following map:

$$[\mathbf{e}']_{\times} = \begin{bmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{bmatrix}. \tag{A.15}$$

The matrix $\mathbf{F}$ is called a fundamental matrix and establishes a relationship between two views from the same scene: an image point $\mathbf{x}$ from the first image defines a projection ray, and the projection of this line in the second image defines the epipolar line $\mathbf{l}'$. If one knows the point $\mathbf{x}$ from one image and the fundamental matrix, the corresponding point $\mathbf{x}'$ in the second image must belong to the epipolar line $\mathbf{l}'$. Thus, one can find the following equation relating the corresponding points $\mathbf{x}$ and $\mathbf{x}'$ in the two views:

$$0 = \mathbf{x}'^{T}\mathbf{l}' = \mathbf{x}'^{T}\mathbf{F}\mathbf{x}. \tag{A.16}$$

A.4 Essential Matrix

The essential matrix can be interpreted as a particular case of the fundamental matrix when the calibration matrix is known. Since a camera matrix is given by the expression $\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & | & \mathbf{t}\end{bmatrix}$, one can remove the effect of the calibration matrix, which is equivalent to using a normalized coordinate system in the image, with:

$$\hat{\mathbf{P}} = \mathbf{K}^{-1}\mathbf{P} = \begin{bmatrix}\mathbf{R} & | & \mathbf{t}\end{bmatrix}, \tag{A.17}$$

and then

$$\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{P}\mathbf{X} = \mathbf{K}^{-1}\mathbf{x}. \tag{A.18}$$

Given a pair of normalized cameras, $\hat{\mathbf{P}} = \begin{bmatrix}\mathbf{I} & | & \mathbf{0}\end{bmatrix}$ and $\hat{\mathbf{P}'} = \begin{bmatrix}\mathbf{R} & | & \mathbf{t}\end{bmatrix}$, the essential matrix that represents the relationship between the cameras is given by:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R} = \mathbf{R}[\mathbf{R}^\mathbf{T}\mathbf{t}]_\times, \tag{A.19}$$

which shows that the essential matrix depends only on the relative rotation and translation between both cameras.

The essential matrix also defines the relation between corresponding normalized points, similarly to Eq. (A.16):

$$\hat{\mathbf{x}'}^\mathbf{T}\mathbf{E}\hat{\mathbf{x}} = 0. \tag{A.20}$$

Replacing Eq. (A.18) in Eq. (A.16), one can find a relation between the fundamental and essential matrices, given the calibration matrices:

$$\mathbf{E} = \mathbf{K}'^\mathbf{T}\mathbf{F}\mathbf{K}. \tag{A.21}$$

A.5 Computation of the fundamental matrix

Eq. (A.14) shows how to retrieve the fundamental matrix that relates two images if the camera matrices are known. However, a common application is the case where only the images are known, and one wants to infer properties of the positioning of the cameras and the three-dimensional space. In this situation, the fundamental matrix must be computed directly from the image information.

Feature detector and descriptor algorithms are tools that can be used to estimate corresponding points in two images. Algorithms such as the scale-invariant feature transform (SIFT) [131], the speeded-up robust features (SURF) [97], the binary robust invariant scalable keypoints (BRISK) [132], or the fast retina keypoint (FREAK) [133] detect representative points in the images and create, for each point, a feature descriptor, often based on the local information. These descriptors can be used to estimate corresponding points between the images, by pairing points with similar descriptors.

The estimation of the fundamental matrix can be made using a set of corresponding points. Each pair of corresponding points provides an equation on the elements of the fundamental matrix given by Eq. (A.16). Using a sufficient number of points, one can create a system of equations to solve for the elements of the fundamental matrix.

Some of the classical algorithms to compute the fundamental matrix are the eight-point algorithm [134] , the seven-point algorithm [135] and the five-point algorithm [98]. The eight-point algorithm [134] requires at least eight pairs of corresponding points to compute the eight unknown elements of the matrix (which is of size 3×3), assuming that in homogeneous coordinates the scale can be disregarded. The seven-point algorithm [135] also includes the restriction that $\det(\mathbf{F}) = 0$, therefore only seven points are necessary. The five-point algorithm [98] is used in the case where the camera calibration is known, and also includes restrictions on the essential matrix.

A.6 Reconstruction from Two Views

The previous sections depicted the relation between the disposition of the cameras in a scene and the image they create. If there is no information about the three-dimensional space and only those images are known, it is possible to estimate information of the cameras disposition from the image contents.

Eq. (A.14) shows the relation between the fundamental matrix $\mathbf{F}$ and the underlying camera matrices $\mathbf{P}$ and $\mathbf{P}'$. Combining Eqs. (A.14) and (A.3), one deduces that:

$$0 = \mathbf{x}'^T\mathbf{F}\mathbf{x} = (\mathbf{P}'\mathbf{X})^T\mathbf{F}(\mathbf{P}\mathbf{X}) = \mathbf{X}^T(\mathbf{P}'^T\mathbf{F}\mathbf{P})\mathbf{X}, \tag{A.22}$$

which, in order to be true for any $\mathbf{X}$, implies that $\mathbf{P}'^T\mathbf{F}\mathbf{P}$ is skew-symmetric.

A possible pair of camera matrices that defines the fundamental matrix $\mathbf{F}$ is the following:

$$\mathbf{P} = \begin{bmatrix}\mathbf{I} & | & \mathbf{0}\end{bmatrix} \qquad \text{and} \qquad \mathbf{P}' = \begin{bmatrix}\mathbf{S}\mathbf{F} & | & \mathbf{e}'\end{bmatrix}, \tag{A.23}$$

since

$$\begin{bmatrix}\mathbf{S}\mathbf{F} & | & \mathbf{e}'\end{bmatrix}^T \mathbf{F} \begin{bmatrix}\mathbf{I} & | & \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{F}^T\mathbf{S}^T\mathbf{F} & \mathbf{0} \\ \mathbf{e}'^T\mathbf{F} & 0\end{bmatrix} = \begin{bmatrix}\mathbf{F}^T\mathbf{S}^T\mathbf{F} & \mathbf{0} \\ \mathbf{0} & 0\end{bmatrix} \tag{A.24}$$

is skew-symmetric if $\mathbf{S}$ is also skew-symmetric. In [136] it is proposed that $\mathbf{S} = [\mathbf{e}']_\times$.

In fact, there is a family of matrices similar to the ones shown in Eq. (A.23) that define the same fundamental matrix $\mathbf{F}$. They are of the form:

$$\mathbf{P} = \begin{bmatrix}\mathbf{I} & | & \mathbf{0}\end{bmatrix} \qquad \text{e} \qquad \mathbf{P}' = \begin{bmatrix}[\mathbf{e}']_\times\mathbf{F} + \mathbf{e}'\mathbf{v}^T & | & \lambda\mathbf{e}'\end{bmatrix}, \tag{A.25}$$

for any vector $\mathbf{v}$ and scalar λ. This indicates that there is an ambiguity in the reconstruction, which is further discussed in the subsection A.8.

If the calibration matrices for both cameras are known beforehand, the cameras can be obtained from the essential matrix in a simpler way. As shown in Eq. (A.19), the essential matrix is defined by a rotation matrix $\mathbf{R}$ and a translation vector

$\mathbf{t}$. Using an SVD decomposition of the essential matrix, it is possible to define a factorization of the form $\mathbf{E} = \mathbf{SR}$. An SVD of the essential matrix is:

$$\mathbf{E} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^T. \tag{A.26}$$

Using

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{A.27}$$

one finds that

$$\mathbf{S} = \mathbf{UZU^T} = [\mathbf{t}]_\times \qquad \text{and} \qquad \mathbf{R} = \mathbf{UWV^T} \text{ or } \mathbf{UW^TV^T}. \tag{A.28}$$

From Eq. (A.28), one can see that there are two possible values for the matrix $\mathbf{R}$, which is due to a symmetry in the position of the cameras. In addition, with this decomposition, the matrix $\mathbf{S}$ necessarily has a Frobenius norm equal to $\sqrt{2}$ [13] and the corresponding vector $\mathbf{t}$ is unitary, which shows that it is not possible to find the true displacement in the three-dimensional space, but only a normalized vector can be estimated, unless some clue about the original scene is known beforehand.

Since the matrix $\mathbf{E}$ is also in homogeneous coordinates, it is not possible to infer the correct of sign of the components, as the scale of the matrix is normalized, therefore a scale of -1 produces the same matrix. Combining the indefinition of the sign and the two solutions for the rotation, one concludes that this decomposition defines four possible candidates for the camera position, which have are symmetrical among them, as seen in Fig. A.3. In order to distinguish among these four cases, it is often defined that all three-dimensional points must be facing the cameras. Thus, using a triangulation technique, one can obtain the solution that provides the largest number of points in front of the cameras.

In analogy to Eq. (A.25), which shows the pair of cameras when the calibration is unknown, the pair of cameras obtained using the essential matrix is:

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & | & \mathbf{0} \end{bmatrix} \qquad \text{and} \qquad \mathbf{P}' = \begin{bmatrix} \mathbf{R} & | & \lambda\mathbf{t} \end{bmatrix}, \tag{A.29}$$

for a scalar λ.

Once the camera matrices $\mathbf{P}$ and $\mathbf{P}'$ are known, the camera centers $\mathbf{C}$ and $\mathbf{C}'$ for this reconstruction can be found as the null space of the camera matrices, since $\mathbf{PC} = \mathbf{0}$ and $\mathbf{P}'\mathbf{C}' = \mathbf{0}$.

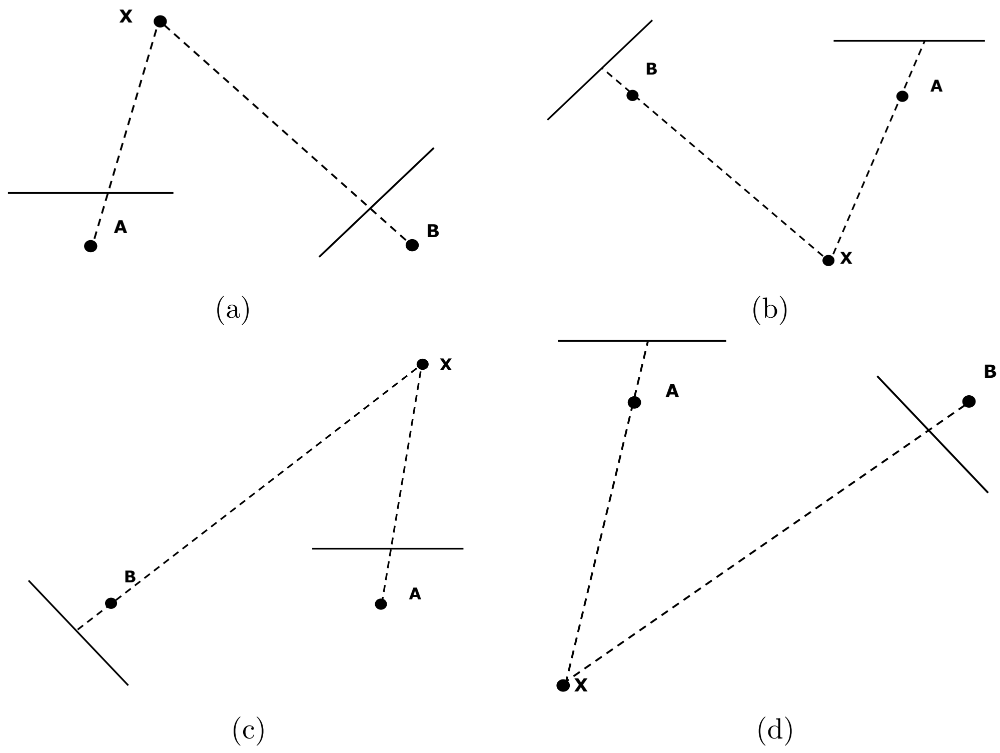


Figure A.3: Possible solutions for the decomposition of the essential matrix in a pair of cameras. The point **X** represents a three-dimensional point that is projected into two cameras with centers **A** and **B**. (a) Point in front of both cameras. (b) Point behind both cameras. (c) Point in front of camera **A** and behind camera **B**. (d) Point in front of camera **B** and behind camera **A**.

A.7 Triangulation

With a pair of camera matrices $\mathbf{P}$ and $\mathbf{P}'$ and the corresponding image points $\mathbf{x}$ and $\mathbf{x}'$, it is possible compute the position of the three-dimensional point that was projected into the cameras. Since a point in the image plane and the corresponding camera matrix define a projection ray, two projection rays can be found. The intersection of those lines define the point $\mathbf{X}$ in the space, which is the point where the projections $\mathbf{x} = \mathbf{PX}$ are $\mathbf{x}' = \mathbf{P}'\mathbf{X}$ valid. Fig. A.4 shows an example of the triangulation.

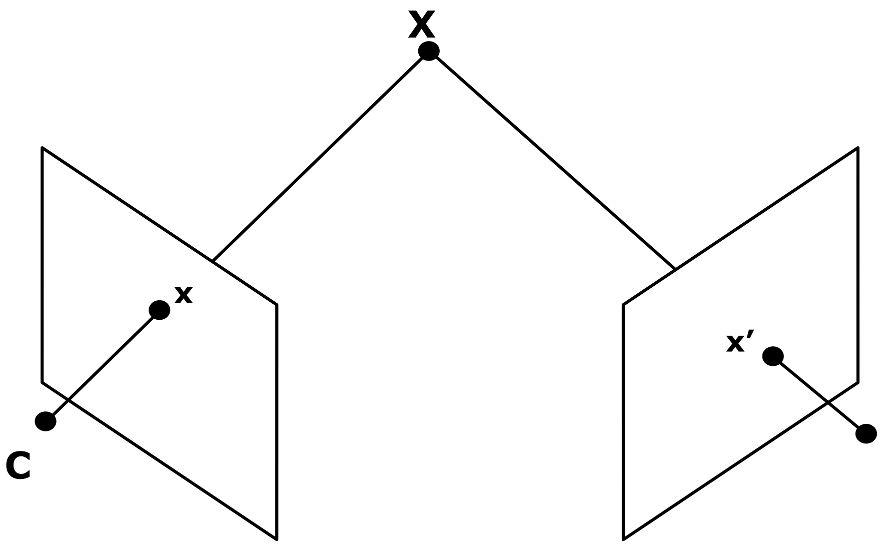


Figure A.4: Triangulation of a three-dimensional point without noise in the measurements. The point $\mathbf{X}$

Assuming that the estimation of the corresponding points can have noise, the restriction defined by Eq. (A.16) may not be satisfied, which potentially leads to an error in the computation of the camera matrices and the projection rays may not intersect. Therefore, it may not be possible to find a point $\mathbf{X}$ that is projected onto both images. In this case, one should either introduce some step to correct the measurements and define a criterion to select points that minimize a pre-defined error. Fig. A.5 shows an example of the triangulation when the measurements have noise.

According to [13], an optimum algorithm for the triangulation in the presence of noise minimizes a cost function that finds approximations of the corresponding points where the epipolar geometry given by Eq. (A.16) is valid. Considering the points $\mathbf{x}$ and $\mathbf{x}'$, one aims to obtain the points $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$, respectively, subject to the restriction that $\hat{\mathbf{x}}^{\mathbf{T}}\mathbf{F}\hat{\mathbf{x'}} = 0$, by minimizing the expression:

$$F = d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x'}})^2, \tag{A.30}$$

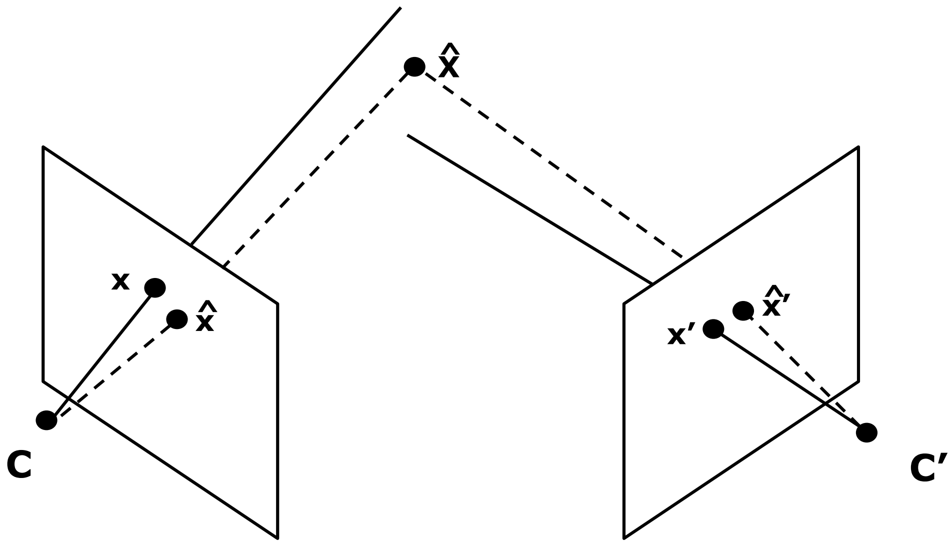


Figure A.5: Triangulation of a point with a noisy measurement of the corresponding points. The points $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ represent an approximation of the corresponding points $\mathbf{x}$ and $\mathbf{x}'$ for which the projection rays intersect, and the point $\hat{\mathbf{X}}$ is the resulting triangulation.

where the operator $d(\mathbf{x}, \mathbf{y})$ represents, for instance, the L_2 norm between $\mathbf{x}$ and $\mathbf{y}$.

Since for the corrected points $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ Eqs. (A.13) and (A.16) should be true, one can chose to correct the point $\mathbf{x}$ by minimizing in Eq. (A.30) the distance between the point $\mathbf{x}$ and some epipolar line $\mathbf{l}$, analogously for $\mathbf{x}'$ and $\mathbf{l}'$. In addition, since there is a relation between $\mathbf{l}$ and $\mathbf{l}'$, it is possible to parameterize both lines using the same variable t. In this scheme, the optimization searches for the value of t that minimizes:

$$F = d(\mathbf{x}, \mathbf{l}(t))^2 + d(\mathbf{x}', \mathbf{l}'(t))^2, \tag{A.31}$$

which can be found as the solution of a sixth-th order polynomium in t [13].

A.8 Ambiguity in the Reconstruction

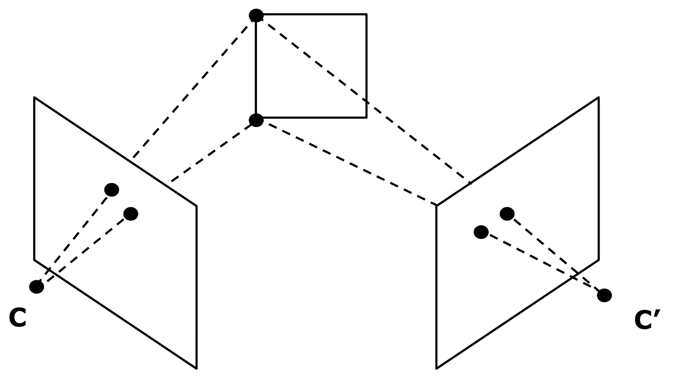
Section A.6 shows that even if the fundamental or essential matrix is known, it is not possible to define unequivocally the pair of cameras that generated the input images. In fact, if there is no information about the original coordinate system in the three-dimensional space, it is not possible to recover the exact location of the objects using only a projection of the space. Any valid solution and the true solution are related by a transformation. This section details this ambiguity in the reconstruction.

Using a set of corresponding points $\mathbf{x_i}$ and $\mathbf{x_i'}$, it is possible to obtain the reconstruction of the scene $\{\mathbf{P}, \mathbf{P}', \mathbf{X}_i\}$, with cameras $\mathbf{P}$ and $\mathbf{P}'$ and the triangulated points $\mathbf{X}_i$. However, for any projective transformation $\mathbf{H}$, one can find a new triangulated point $\bar{\mathbf{X}}_\mathbf{i} = \mathbf{H}\mathbf{X}_\mathbf{i}$ and a camera matrix $\bar{\mathbf{P}} = \mathbf{P}\mathbf{H}^{-1}$ that have the same projection in the images, since:

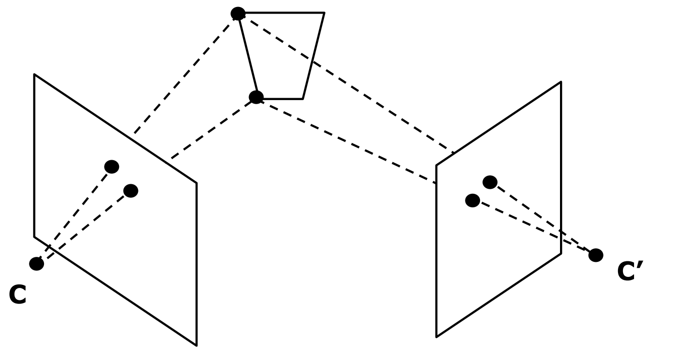
$$\bar{\mathbf{P}}\bar{\mathbf{X}}_\mathbf{i} = \mathbf{P}\mathbf{H}^{-1}\mathbf{H}\mathbf{X}_\mathbf{i} = \mathbf{P}\mathbf{X}_\mathbf{i} = \mathbf{x}, \tag{A.32}$$

analogously for the other camera.

One should notice that, if only the points $\mathbf{x}$ and $\mathbf{x}'$ are known, it is not possible to distinguish between the reconstructions $\{\mathbf{P}, \mathbf{P}', \mathbf{X}_i\}$ and $\{\bar{\mathbf{P}}, \bar{\mathbf{P}}', \bar{\mathbf{X}}_i\}$, since they map the image points and define the same epipolar geometry. In this case, it is said that any reconstruction differs from the true one by a projective transformation, as is exemplified in Fig. A.6. This conclusion can also be drawn from Eq. (A.25), which shows a decomposition of the fundamental matrix into two cameras with several degrees of freedom. In this case, the parametrization of the cameras mirror the degrees of freedom of the projective transformation that defines the ambiguity of the reconstruction.



(a)



(b)

Figure A.6: Projective reconstruction of a scene. The reconstruction (b) differs from the real reconstruction (a) by a projective transformation.

If the calibration matrices are known, only the extrinsic parameters of the cameras need to be estimated. Since the intrinsic parameters are fixed, it is possible to find a reconstruction of a scene such that for the set of valid solutions, the projection ray for the same point always forms the same angle with the image plane. In this case, the reconstruction is named a metric reconstruction, and any valid reconstruction is related to the true solution by a similarity transformation, as seen in Fig. A.7. This case is analogous to Eq. (A.29), which shows a pair of reconstructed cameras where the second camera is rotated with respect to the first one and it is not possible to find the true displacement between them.

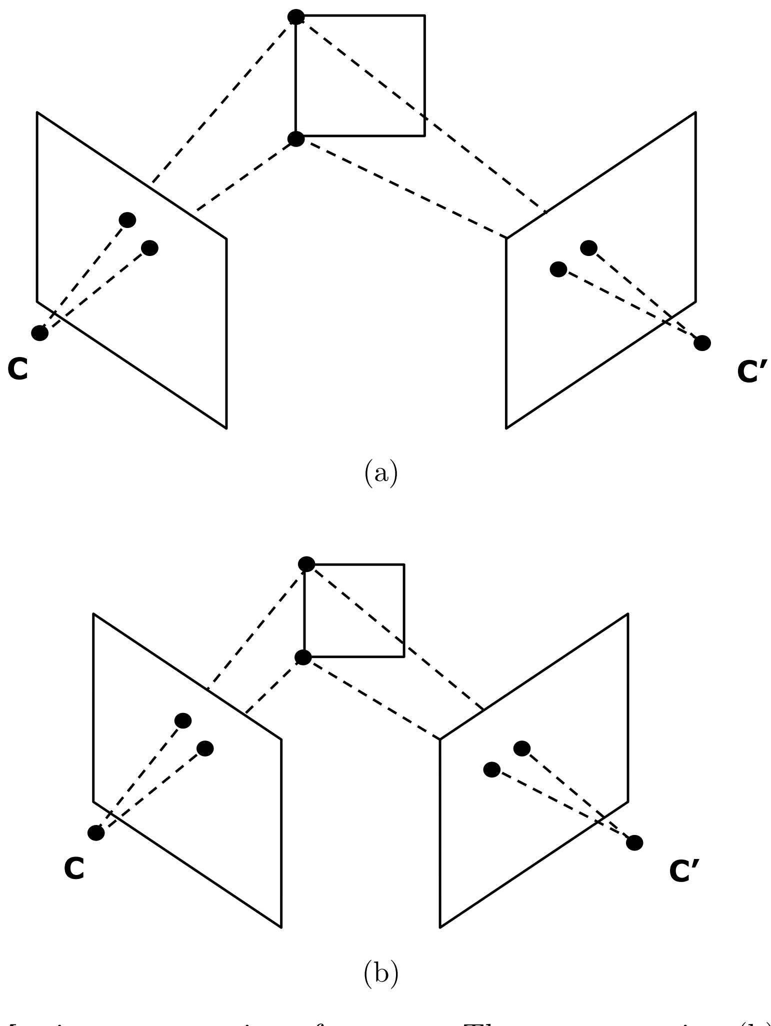


Figure A.7: Metric reconstruction of a scene. The reconstruction (b) differs from the real reconstruction (a) by a similarity transformation.

A.9 Reconstruction for Multiple Views

If three or more views are available, which is the case for instance if one uses a video with a moving camera, the mathematical development of the geometry that relates two views, described in the previous sections, can be extended. For three views, one can define a tensor [13] to relate the geometry of the views, in replacement of the fundamental matrix. Increasing the number of views, problems with an even greater dimensionality must be solved. For these cases, the solution often consists in splitting the views into pairs, reconstructing the scene for a pair of views and incrementally introduce the other views.

The structure from motion (SfM) algorithm is a method to reconstruct the scene for multiple views of a scene. This section describes a basic version of the SfM algorithm to incrementally solve the reconstruction for multiple views using the equations developed for two views.

Using an initial pair of images, one can find corresponding points, compute the fundamental matrix and find a valid pair of camera matrices and a set of triangulated points. Fig. A.8 shows this initial solution. In this figure, the cameras with centers $\mathbf{C_1}$ and $\mathbf{C_2}$ have correspondent of image points represented with the same color and some triangulated points $\mathbf{X_i}$ are found.

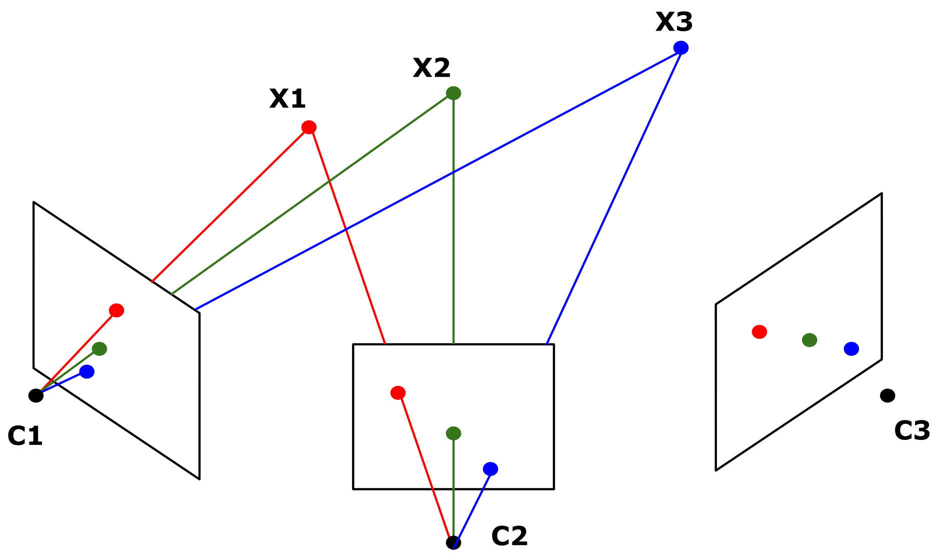


Figure A.8: Initial reconstruction of a scene.

In a subsequent step, the other views are incorporated onto the initial reconstruction. For each view, one finds corresponding points between the current view and any other view already included in the reconstruction. For those points that were already used in the reconstruction, the triangulated points are already known, therefore there is a relation between three-dimensional points $\mathbf{X_i}$ and image points $\mathbf{x_i}$ in the current image, as can be seen in Fig. A.9. With a sufficient number of points, the camera matrix for this view can be computed using Eq. (A.5). The corresponding points that were not used in the reconstruction are used to triangulate new points, increasing the point cloud, as depicted in Fig. A.10.

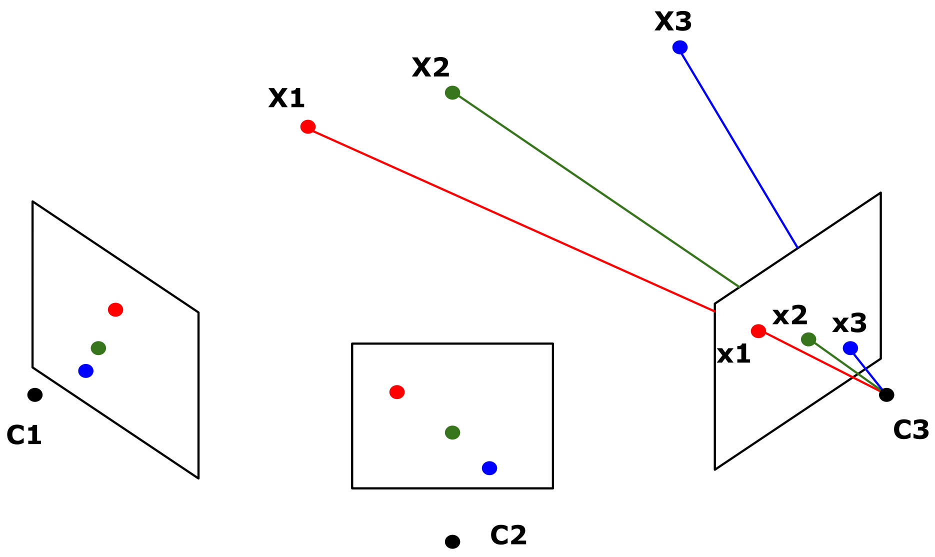


Figure A.9: Correspondences between the image points $\mathbf{x_i}$ from a new view and the cloud of triangulated points $\mathbf{X_i}$.

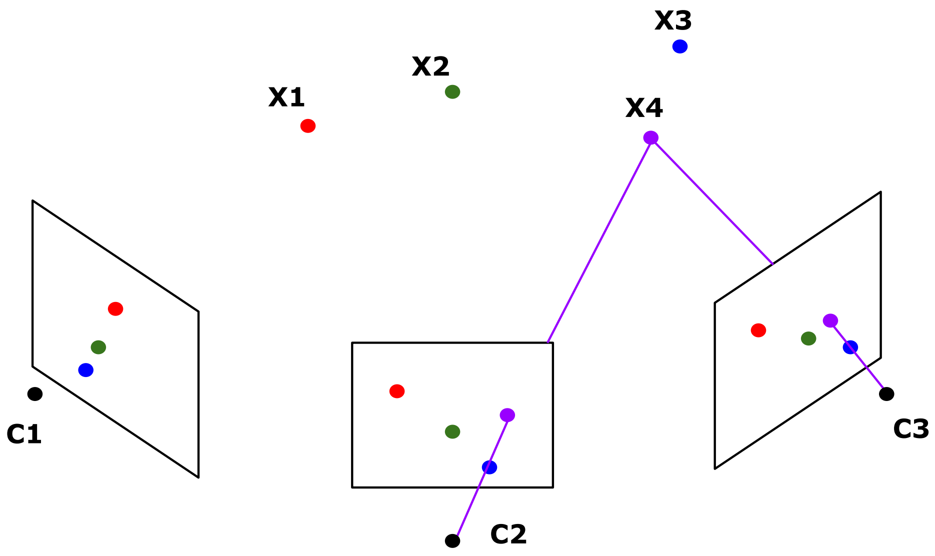
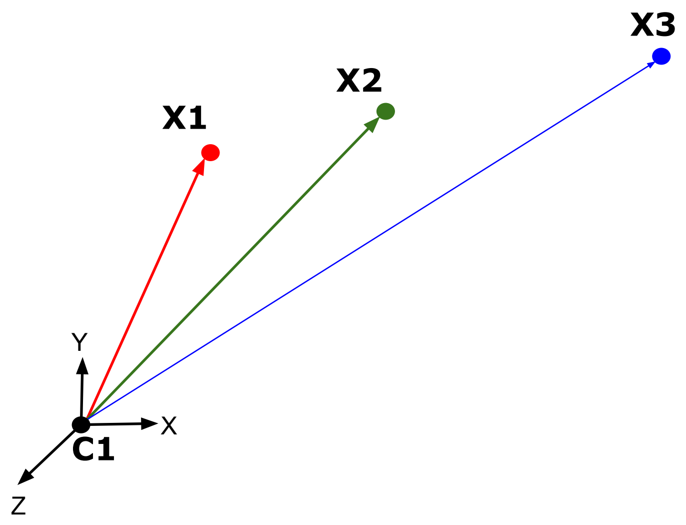


Figure A.10: Expansion of the number of triangulated points, with inclusion of the points triangulated from new correspondences.

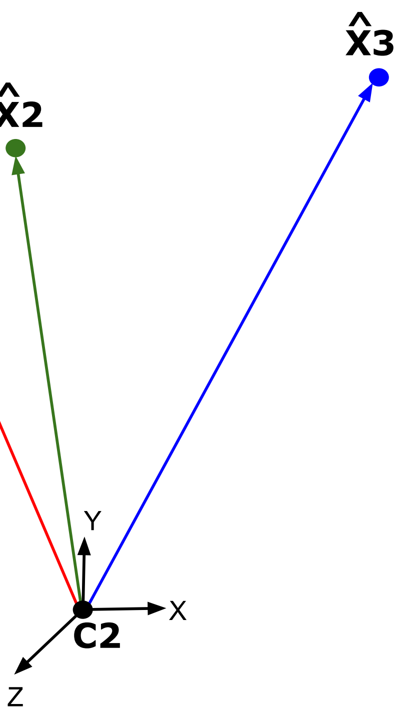
A different approach to solve the SfM involves the computation of an independent reconstruction for each pair of images, and then the normalization of every reconstruction to the same coordinate system. As discussed in Subsection A.8, given a pair of images, there is a family of possible solutions for the reconstruction of the scene. If one uses two pairs of images representing the same scene and finds the camera matrices independently for each pair, there is no guarantee that the reconstruction of the first pair and the reconstruction of the second pair are compatible, since due to the ambiguity, each one has its own coordinate system (for example, the three-dimensional space extrapolated from the images can have different scales, the origin of this space can be in different positions, etc.). However, among the family of solutions for the second reconstruction, one can assume that there is a solution in the same coordinate system of the first one, which is related to the solution previously found by a specific transformation, which depends on the type of reconstruction. The algorithm in this approach computes individual reconstructions and then finds the transformations that make them compatible.

Given a first image pair, one follows the procedure described in the previous sections to reconstruct the scene, finding camera matrices $\mathbf{P}_2$ and $\mathbf{P}_3$ and triangulated points, according to Fig. A.8, and those points in space are described using a coordinate system based on the two cameras. A new pair is formed using a new view and the last view, that creates another reconstruction with cameras $\hat{\mathbf{P}}_\mathbf{2}$ and $\hat{\mathbf{P}}_\mathbf{3}$ and a set of points $\hat{\mathbf{X}}_\mathbf{i}$. Using the correspondences between images, one can find points $\mathbf{X}_\mathbf{i}$ in a reconstruction of the scene that are equivalent to points $\hat{\mathbf{X}}_\mathbf{i}$ in another reconstruction. Fig. A.11 shows equivalent triangulated points, each one with its own coordinate system.

Since it is desired that every reconstruction is grouped in a global one, one estimates a transformation $\mathbf{X}_\mathbf{i} = \mathbf{H}\hat{\mathbf{X}}_\mathbf{i}$ that makes the second coordinate system coincide with the first one. In order for the projections in the image to remain the same, the new camera must be such that $\mathbf{P}_\mathbf{3} = \hat{\mathbf{P}}_\mathbf{3}\mathbf{H}$, which represents the camera matrix $\hat{\mathbf{P}}_\mathbf{3}$ found in the second reconstruction written with respect to the coordinate system used in the first one.



(a)



(b)

Figure A.11: Equivalent points in different coordinate systems. In (a), the points $\mathbf{X_i}$ are expressed in function of a coordinate system obtained from in the first reconstruction and centered in $\mathbf{C_1}$. In (b), the points $\hat{\mathbf{X}}_\mathbf{i}$ are expressed in function of a coordinate system obtained from the new reconstruction and centered in $\mathbf{C_2}$. Points with the same color in different reconstructions represent correspondences.

Appendix B

Lie Groups and Lie Algebra

Lie groups arise from several structures in nature that present a continuous symmetry. In order to properly introduce the concept of Lie groups, we define some basic algebraic definitions that will be useful for the remaining of this thesis. A few examples in the end of this section aid in the understanding of the concepts here presented.

B.1 Group

Consider that G is a set and $\circ$ is a binary operation, also called group operator, that takes any two elements of G and returns an element of G. The pair $(G, \circ)$ is called a groupoid:

$$\forall g_1, g_2 \in G : g_1 \circ g_2 \in G. \tag{B.1}$$

To be considered a group, a groupoid must respect the following properties:

- Associativity: $\forall g_1, g_2, g_3 \in G : \; g_1 \circ (g_2 \circ g_3) = (g_1 \circ g_2) \circ g_3$;
- Existence of identity element: $\exists e \in G | \; \forall g \in G : \; e \circ g = g \circ e = g$;
- Existence of inverse element: $\forall g \in G : \; \exists g^{-1} \in G | \; g^{-1} \circ g = g \circ g^{-1} = e$.

B.2 Field

A field is a set F together with two operations $+$ and $\cdot$ from $F \times F$ to F such that the following properties are true:

- Associativity: $\forall a, b, c \in F : \; (a + b) + c = a + (b + c)$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$;
- Commutativity: $\forall a, b \in F : \; a + b = b + a$ and $a \cdot b = b \cdot a$;

- Existence of identity element: $\exists e_a \in F|\ \forall a \in F:\ a + e_a = a$ and $\exists e_m \in F|\ \forall a \in F:\ a \cdot e_m = a$;
- Existence of inverse element: $\forall a \in F:\ \exists(-a) \in F|\ a + (-a) = e_a$ and $\forall a \in F, a \neq e_a:\ \exists(a^{-1}) \in F|\ a \cdot (a^{-1}) = e_m$;
- Distributivity of multiplication over addition: $\forall a, b, c \in F:\ a \cdot (b + c) = (a \cdot b) + (a \cdot c)$;

B.3 Vector Space

Given a field F and a set V, defining two operations $+$ from $V \times V$ to V and $\cdot$ from $F \times V$ to V, a vector space over F is the set V together with the operations $+$ and $\cdot$ with the following properties:

- Associativity of addition: $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V:\ (\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$;
- Commutativity of addition: $\forall \mathbf{u}, \mathbf{v} \in V:\ \mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$;
- Existence of additive identity element: $\exists \mathbf{e} \in V|\ \forall \mathbf{u} \in V:\ \mathbf{u} + \mathbf{e} = \mathbf{u}$;
- Existence of additive inverse element: $\forall \mathbf{u} \in V:\ \exists(-\mathbf{u}) \in V|\ \mathbf{u} + (-\mathbf{u}) = \mathbf{e}$;
- Associativity of scalar multiplication: $\forall \mathbf{u} \in V,\ a, b \in F:\ a \cdot (b \cdot \mathbf{u}) = (ab) \cdot \mathbf{u}$;
- Distributivity of scalar multiplication with respect to vector addition: $\forall \mathbf{u}, \mathbf{v} \in V,\ a \in F:\ a \cdot (\mathbf{u} + \mathbf{v}) = a \cdot \mathbf{u} + a \cdot \mathbf{v}$;
- Distributivity of scalar multiplication with respect to field addition: $\forall \mathbf{u} \in V,\ a, b \in F:\ (a + b) \cdot \mathbf{u} = a \cdot \mathbf{u} + b \cdot \mathbf{u}$;
- Existence of scalar multiplication identity element: $\exists e_m \in F|\ \forall \mathbf{u} \in V:\ e_m \cdot \mathbf{u} = \mathbf{u}$;

B.4 Algebra

Assuming a field F and a vector space A over F with an additional binary operation $\cdot$ from $A \times A$ to A, we say that A is an algebra over F if it has the following properties:

- Right distributivity: $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in A:\ (\mathbf{x} + \mathbf{y}) \cdot \mathbf{z} = \mathbf{x} \cdot \mathbf{z} + \mathbf{y} \cdot \mathbf{z}$;
- Left distributivity: $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in A:\ \mathbf{x} \cdot (\mathbf{y} + \mathbf{z}) = \mathbf{x} \cdot \mathbf{y} + \mathbf{x} \cdot \mathbf{z}$;
- Scaling: $\forall \mathbf{x}, \mathbf{y} \in A,\ \forall a, b \in F:\ (a\mathbf{x}) \cdot (b\mathbf{y}) = (ab)(\mathbf{x} \cdot \mathbf{y})$;

B.5 Lie Groups and Lie Algebra

A Lie group $(G, \circ)$ is a special kind of group that has a particular geometry for which the set G is a smooth manifold, such that the mappings $a(g_1, g_2) = g_1 \circ g_2$ and $b(g) = g^{-1}$ are both analytic, which means that the functions $a(g)$ and $b(g)$ are continuous, infinitely differentiable and can be expressed as a Taylor series that converge around any point in its domain.

In a matrix Lie group $(G, \circ)$, the elements are $\mathbf{g} \in G \subset \mathbb{R}^{N\times N}$ and the group operator $\circ$ is the matrix multiplication. Some important examples of matrix Lie group are:

- General linear group: $GL(N, \mathbb{R}) = \big\{\mathbf{A} \in \mathbb{R}^{N\times N} |\, \det(\mathbf{A}) \neq 0\big\}$;
- Orthogonal group: $O(N) = \big\{\mathbf{X} \in GL(N, \mathbb{R})|\ \mathbf{X}^T\mathbf{X} = \mathbf{I}\big\}$;
- Special orthogonal group: $SO(N) = \big\{\mathbf{X} \in GL(N, \mathbb{R})|\ \mathbf{X}^T\mathbf{X} = \mathbf{I}, \det(\mathbf{X}) = 1\big\}$;
- Rigid body motion: $SE(3) = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$, with $\mathbf{R} \in SO(3)$ and $\mathbf{t} = [t_1, t_2, t_3]^T$;
- 3D similarity: $Sim(3) = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$, with $\mathbf{R} \in SO(3)$, $\mathbf{t} = [t_1, t_2, t_3]^T$ and $s \in \mathbb{R}^+$.

Given a matrix Lie group, elements $\mathbf{g} \in G$ close to the identity element can be written as $\mathbf{g} = \exp(X)|\ X \in \mathcal{G}$, where $\mathcal{G}$ is an open neighborhood of $0^{N\times N}$ in the tangent space at the identity of G, and is called the Lie algebra $\mathcal{G}$ [48]. The matrix Lie algebra $\mathcal{G}$ associated to the Lie group G is the set of all matrices X such that the exponential of each X results in an element of the Lie group G. The opposite is also valid, and the matrix logarithm provides the inverse mapping between an element of the Lie algebra and an element of the Lie group:

$$\exp_G : \mathcal{G} \to G \tag{B.2}$$

$$\log_G : G \to \mathcal{G}. \tag{B.3}$$

The Lie algebra $\mathcal{G}$ associated to a p-dimensional Lie group G is a p-dimensional vector space, so there is also a mapping between $\mathcal{G}$ and $\mathbb{R}^p$ which is defined by the $\vee$ ("vee") operator:

$$[]_G^\vee : \mathcal{G} \to \mathbb{R}^p \tag{B.4}$$

$$[]_G^\wedge : \mathbb{R}^p \to \mathcal{G}. \tag{B.5}$$

In order to reduce the notation, it is also common to denote $\exp_G([]^\wedge_G)$ as $\exp^\wedge_G$ and $\log_G([]^\vee_G)$ as $\log^\vee_G$.

The theory of Lie groups provides a tool to define symmetries from a mathematical point of view. The Lie algebra represents the space tangent to the Lie group at the identity, having a one-to-one map between them. The importance of the Lie algebra is that, in general, it is easier to work on a linear space than the "curved" space defined by the Lie group.

B.6 Adjoint Representation

Since the Lie groups are usually non-commutative, we define a function Ad_G, called the adjoint representation of the Lie group G, to express the non-comutativity. For $\mathbf{X} \in G$ and $a \in \mathcal{G}$, we seek an element $\mathbf{b} \in \mathcal{G}$ in order to satisfy $\mathbf{X}\exp_G(\mathbf{a}) = \exp_G(\mathbf{b})X$. It can be proved that [48]:

$$\mathbf{b} = \mathbf{X}\mathbf{a}\mathbf{X}^{-1} = \mathrm{Ad}_G(\mathbf{X})\mathbf{a}. \tag{B.6}$$

Other measurement of commutativity is the function ad_G, called the adjoint representation of the Lie algebra $\mathcal{G}$. For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^p$, the function ad_G is defined as:

$$\mathrm{ad}_G(\mathbf{b})\mathbf{a} = \left[[\mathbf{b}]^\wedge_G [\mathbf{a}]^\wedge_G - [\mathbf{a}]^\wedge_G [\mathbf{b}]^\wedge_G\right]^\vee_G. \tag{B.7}$$

Recall that a product on the group represents the group operator $\circ$, which maps two elements of the group in another element of the group (for instance, a combination of two successive rotations define a new rotation). Therefore, the adjoint representation embodies the multiplicative structures of the group and the algebra.

B.7 Baker-Campbell-Hausdorff Formula

The BCH (Baker-Campbell-Hausdorff) formula [137] expresses the group product directly in $\mathbb{R}^p$. Given $X = \exp^\wedge(\mathbf{a})$ and $Y = \exp^\wedge(\mathbf{b})$, with $X, Y \in G$ and $\mathbf{a}, \mathbf{b}$, the following equation is valid:

$$\log^\vee_G(\exp^\wedge_G(\mathbf{a})\exp^\wedge_G(\mathbf{b})) = \mathbf{b} + \mathbf{J}_G(\mathbf{b})\mathbf{a} + O(\|\mathbf{a}\|^2), \tag{B.8}$$

where

$$\mathbf{J}_G(\mathbf{b}) = \sum_{n=0}^{\infty} \frac{B_n \mathrm{ad}_G(\mathbf{b})^n}{n!} = \mathbf{I} + \frac{1}{2}\mathrm{ad}_G(\mathbf{b}) + \cdots \tag{B.9}$$

is the left Jacobian of G and B_n are Bernoulli numbers. This equation defines a first-order Taylor linearization of the group product. One should also notice that this linearization is expressed with respect to the adjoint representation. If the Lie group is commutative, then

$$\log_G^\vee(\exp_G^\wedge(\mathbf{a})\exp_G^\wedge(\mathbf{b})) = \mathbf{b} + \mathbf{a}. \tag{B.10}$$

B.8 Concentrated Gaussian Distribution

The distribution of $\mathbf{X} \in G$ is called a (right) concentrated Gaussian distribution on G of mean μ and covariance $\mathbf{P}$, denoted $p(\mathbf{X}) = \mathcal{N}_G^R(\mu, \mathbf{P})$, if:

$$\mathbf{X} = \exp_G^\wedge(\epsilon)\mu, \tag{B.11}$$

where $p(\epsilon) = \mathcal{N}_{\mathbb{R}^p}(0, \mathbf{P})$ and $\mathbf{P} \subset \mathbb{R}^{p\times p}$ is a symmetric positive-semidefinite matrix.

If the maximum of the eigenvalues of $\mathbf{P}$ is small, the probability mass is concentrated around μ and we may approximate $p(\mathbf{X})$ as:

$$p(\mathbf{X}) \approx \frac{1}{(2\pi)^p \det(\mathbf{P})} e^{-\frac{1}{2}\left\|\log_G^\vee(\mu^{-1}\mathbf{X})\right\|_P^2}. \tag{B.12}$$

The next subsection shows an example of a concentrated Gaussian distribution.

B.9 Examples

B.9.1 Special Orthogonal Group $SO(2)$

The special orthogonal group $SO(2)$ represents the group of rotations in the two-dimensional plane, and is defined as:

$$SO(2) = \left\{\mathbf{R} \in \mathbb{R}^{2\times 2} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\right\}. \tag{B.13}$$

The associated Lie algebra is:

$$so(2) = \left\{ u = \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix} | \theta \in \mathbb{R} \right\}. \tag{B.14}$$

The adjoint representation $\mathrm{Ad}_{SO(2)}(\mathbf{R})$ is:

$$\mathrm{Ad}_{SO(2)}(\mathbf{R}) = \mathbf{I}. \tag{B.15}$$

It is important to mention that this group is commutative, since the combination of rotations with angles θ_1 and θ_2 is a rotation with angle $\theta_1 + \theta_2$. For this reason, the adjoint is the identity matrix, which validates the commutative property.

B.9.2 Special Orthogonal Group $SO(3)$

The special orthogonal group $SO(3)$ represents the group of rotations in the three-dimensional space, and is defined as:

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \right\}. \tag{B.16}$$

The associated Lie algebra is:

$$so(3) = \left\{ u = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix} | \theta_1, \theta_2, \theta_3 \in \mathbb{R} \right\}. \tag{B.17}$$

The adjoint representation $\mathrm{Ad}_{SO(3)}(\mathbf{R})$ is:

$$\mathrm{Ad}_{SO(3)}(\mathbf{R}) = \mathbf{R}. \tag{B.18}$$

Contrary to the group $SO(2)$, the group $SO(3)$ is noncommutative. For instance, if one rotates an object by 90 degrees in one axis, and after that rotates it by 90 degrees in another axis, the result is different from the one obtained if the order of rotations is the inverse. For this reason, the adjoint is not the identity matrix.

B.9.3 Special Euclidean Group $SE(2)$

The special Euclidean group $SE(2)$ represents rigid transformations in the two-dimensional space. The group has three dimensions, corresponding to translation and rotation in the plane, and can be defined as:

$$SE(2) = \left\{ \mathbf{X} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} | \mathbf{R} \in SO(2), \mathbf{t} = \begin{bmatrix} u \\ v \end{bmatrix} \in \mathbb{R}^2 \right\}. \tag{B.19}$$

The associated Lie algebra is:

$$se(2) = \left\{ u = \begin{bmatrix} 0 & -\theta & x \\ \theta & 0 & y \\ 0 & 0 & 0 \end{bmatrix} | \theta, x, y \in \mathbb{R} \right\}. \tag{B.20}$$

The adjoint representation $\mathrm{Ad}_{SE(2)}(\mathbf{X})$ is:

$$\mathrm{Ad}_{SE(2)}(\mathbf{X}) = \begin{bmatrix} \mathbf{R} & \mathbf{q} \\ \mathbf{0} & 1 \end{bmatrix} | \mathbf{R} \in SO(2), \mathbf{q} = \begin{bmatrix} v \\ -u \end{bmatrix} \in \mathbb{R}^2. \tag{B.21}$$

In Fig. B.1, an example of a concentrated Gaussian distribution for the $SE(2)$ group is presented. Under the assumption of the concentrated Gaussian distribution, the parameters of the Lie algebra $se(2)$ follow a normal distribution, as can be seen in

Fig B.1(a). After applying an exponential map $\exp^{\wedge}_{SE(2)}$ and moving the distribution around the average μ, the distribution of the parameters of the Lie group becomes the one seen in Fig. B.1(b). A way of interpreting the graph is to consider that the position and direction of the arrows correspond, respectively, to the position and orientation of an object. According to this interpretation, the concentrated Gaussian distribution represents a more complex modelling for a real application. It provides a tool to model the pose of the objects using a single structure.

B.9.4 Estimation of a Proper Three-Dimensional Rotation

In order to estimate a proper three-dimensional rotation, which has 3 degrees of freedom, one can write an expression of the matrix in function of the rotation angles. This is can be made by the use of the Euler angles parametrization:

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{B.22}$$

However, this parametrization, despite allowing the optimization using a single 3-parameter vector $[\alpha, \beta, \gamma]$, has a drawback. If, for instance, $\beta = \pi/2$, then

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{bmatrix}, \tag{B.23}$$

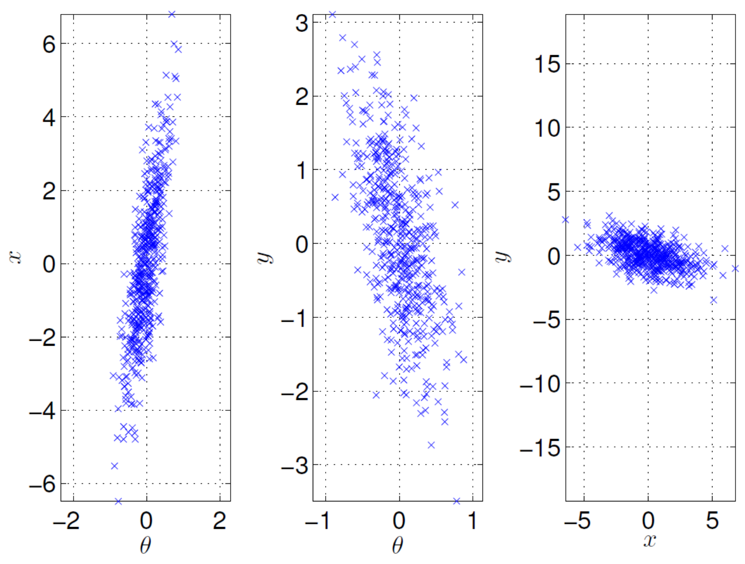
which means that the angles α and γ become coupled and changes in any of them produce the same result, a change in the angle $(\alpha + \gamma)$. This effect is called gimbal lock and produces a loss of a degree of freedom under certain conditions.

A second approach is to use the matrix space:

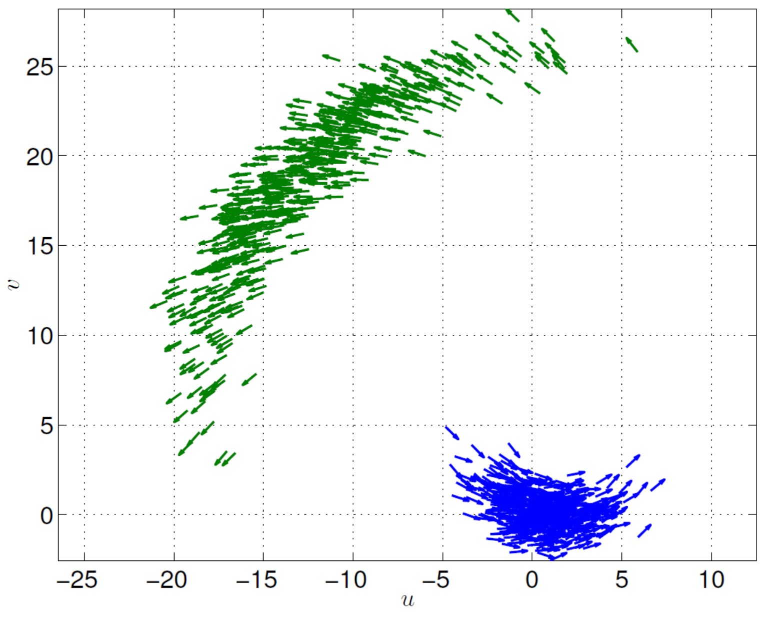
$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}. \tag{B.24}$$

For this approach, the optimization is performed in the 9-parameter vector $[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]$ with additional constraints $\mathbf{X}^T\mathbf{X} = \mathbf{I}$ and $\det(\mathbf{X}) = 1$. Thus, an algorithm has to estimate nine parameters, in contrast to the tree parameters used in the previous representation, solving a constrained optimization problem, which is more complex and more susceptible to ill-conditioning.

A third approach is to model the proper three-dimensional rotation as belonging to the Lie group $SO(3)$, which has an associated Lie algebra of the form:



(a)



(b)

Figure B.1: Example of a concentrated Gaussian distribution for the Lie group SE(2). (a) Samples of the parameters of the Lie algebra se(2) under a normal distribution. (b) Samples mapped to the concentrated Gaussian distribution in the Lie group SE(2). The samples are mapped using the $\exp^{\wedge}_{SE(2)}$ (blue arrows) and then moved around the average μ (green arrows) [138].

$$\mathbf{X} = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix} = \sum_{i=1}^{3} \theta_i \mathbf{E_i}, \tag{B.25}$$

for some bases

$$\mathbf{E_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{E_2} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{E_3} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{B.26}$$

Since the Lie algebra can be mapped to a three-dimensional Euclidean space, the optimization can be performed in a 3-parameter vector. In this case, the optimization operates in a space where every matrix has size 3×3 and intrinsically respects the constraints, which, besides providing a more elegant solution, also has better convergence properties. In addition, several operations such as composition, inversion, differentiation, and interpolation, can be addressed by the theory of Lie groups.