MOTION-BASED OBJECT SEGMENTATION IN VIDEO SEQUENCES

Dissertation

to obtain the degree of Doctor at
Maastricht University,
on the authority of the Rector Magnificus,
Prof. dr. Rianne M. Letschert
in accordance with the decision of the Board of Deans,
to be defended in public
on Thursday 28 March, 2019 at 14:00 hours

by

Wei ZHAO

Supervisor:

Prof.dr.ir. R.L.M. Peeters

Co-supervisor:

Dr.ir.ing. N. Roos

Assessment Committee:

Prof.dr. G.B. Weiss (chair)
Dr. S. Asteriadis
Prof.dr. K. Coninx (Hasselt University)
Prof.dr. T. Gevers (University of Amsterdam)
Prof.dr.ir. J.C. Scholtes

The research reported in this thesis was financially supported by the China Scholarship Council (under the grant No. 2011607073), and the Department of Data Science and Knowledge Engineering, Maastricht University.

Printed by *Optima Grafische Communicatie, Rotterdam*
ISBN 978-94-6361-251-7

To my beloved parents.
致我亲爱的父母

CONTENTS

1

INTRODUCTION

How to make artificial intelligence see and interpret the real world like a human being, has been a fascinating task since the beginning of research in computer vision. Humans can perceive the external world by their visual system, which consists of the eyes, parts of the brain, and the pathways connecting them [2, 20, 122]. The eyes can capture the light entering the cornea and transform it to electric visual signals. The brain acts as a processing and interpretation unit of the visual signal from the eyes. Studies in computer vision try to mimic the process of human vision, where the camera is equivalent to the eyes and the computer fulfils the processing function of the human brain [90, 101, 176]. Robot vision is part of the broader field of computer vision, it mainly focus on the techniques that are applicable for robots. The ultimate goal of robot vision is to enable robots to perceive the external visual world and understand it in an intelligent way, so to help them perform various tasks by acting and interacting with their environment [117].

Although the study of computer vision is inspired by the mechanism of biological vision, the intrinsic processes of human vision and computer vision systems are still different because of hardware constraints of computers. Researchers have revealed that the human eye system has the capability of continuously receiving the equivalent of about 10 megabits of visual information per second, while much of the information is redundant [135]. The redundant data is compressed by the visual cortex and sent to the corresponding part of the brain. The human brain can interpret the compressed data and abstract the information in an effective way. Tasks that the human vision system can perform in seconds will take many hours even in the most advanced supercomputer. Moreover, the architectures are different between the human neuronal system and electronic computers so that some mechanisms of human vision are simply not applicable o electronic computers [273]. Although perceptual psychologists and neuroscientist have made some achievements in the field of cellular neuroscience, neural circuits, cognitive neuroscience, etc., the mystery of visual information processing in the human brain remains for a good deal unrevealed, especially on the high level of human vision system signals [163, 191]. By sending visual signals through nerves and between parts of the brain, the brain does all of the complex tasks of processing, analyzing and understanding the visual information [101].

The common goal of either computer vision or robot vision is to extract "information" from visual sources. Plenty of approaches and algorithms have been developed in these areas, based on studies and theories in the fields of psychology, artificial intelligence, statistics, geometry, physics and mathematics [113, 227, 263]. Research in industrial robot vision has made progress because the visual environment of such applications is predictable and controllable. Techniques for applications such as automatic assembly, components inspection, surveillance, etc., have been widely used in modern society. Meanwhile, researchers focus on developing intelligent systems that can deal with dynamic environments, in a robust way which provides a basis for autonomous robots operating in natural environments.

1.1. OBJECT DETECTION AND RECOGNITION

Recognizing the constituent objects in observed scenes is one of the primary tasks of any vision system. The terminology "object recognition" in psychology is the ability to perceive an object's physical properties (such as shape, colour and texture) and then apply to semantic attributes to the object, which includes the understanding of its use, previous experience with the object and how it relates to others [81]. The human vision system has the ability of carrying out such tasks effortlessly and very quickly, despite the fact that these objects may vary somewhat in form, colour, texture, etc. For example, a soccer player is able to perceive the ball and discriminate the team-mates from the counterparts in front at a glance, which enables him to make decisions about the next move. During a very short moment, this player carries out a series of recognition tasks:

1. Processing the visual signals perceived by the eyes to extract visual features such as colours, shapes, motion characteristics, etc.
2. Segmenting out the things of interest.
3. Labelling the segmented things to their categories based on empirical knowledge.

The tasks are the fundamental research issues of recognition in computer vision, which is one of the most challenging topics in this area and attracting extensive research. However recognition of common objects in computer vision is quite beyond the capability of artificial systems proposed so far. Why is recognition so difficult for a computer or a robot? That is because of variation in the world. Objects presented in natural scenes show variation within their categories, but also between categories. Moreover, the appearance of one object can vary due to changes in pose, illumination, texture, deformation, and under occlusion. These variations make it unlikely that a common method for recognition under different conditions in computer vision can be developed at present. Researchers are mostly dedicated to develop schemes and algorithms to meet specific requirements and constraints of different applications. Substantial success has been achieved in detection and recognition of specific objects, such as handwritten digits, fingerprints, faces, and road signs [124, 160, 213, 235, 269]. Meanwhile, significant development has also been made for developing schemes of object recognition in more generic situations [174, 234, 246].

1.1.1. DEFINITIONS

Visual object recognition in computer vision depends on many aspects, such as processing input images or videos, learning object representations, analysing perceived scenes,

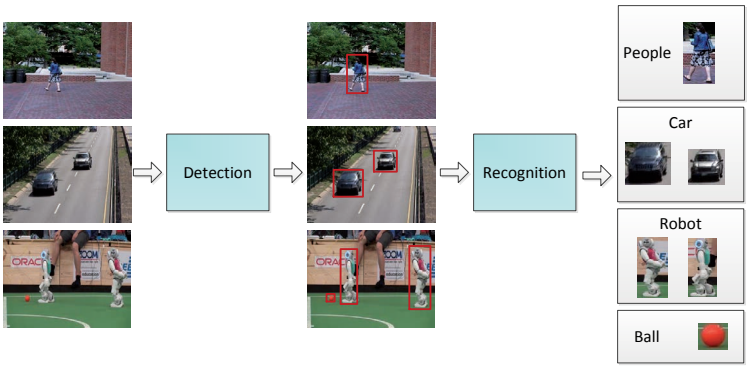


Figure 1.1: The generic pipeline of object detection and recognition

locating the objects, classification of objects, etc. [11, 89, 189, 227]. The recognition problem in computer vision is broken down into many sub-fields, and the terminologies varies subtly in different applications [11]. In this thesis, we use a fairly general definition of "object detection", which refers to determining the location and scale of all object instances, if any, that are present in an image [227, 246]. The objects are to be detected no matter the viewpoint, against a cluttered background, or with partial occlusions [227]. "Object recognition" in this thesis means object class recognition, which includes determining the classes of object instances present in the scene [227]. Figure 1.1 illustrates the general conceptions of object detection and recognition used in this thesis.

1.1.2. APPLICATIONS

Techniques and algorithms used for object detection and recognition differs, due to the requirements in applications. Successes have been achieved in specific domains. We will present some representative applications in this section.

Facial Recognition Systems

Facial recognition is a common instance of specific object detection. It aims at identifying or verifying a person from a digital image or a video frame from a video source. For given images or videos, the face recognition system aims to detect the regions of faces and recognize the faces using a stored database of faces [275].

A face is an important biometric, which shows distinctive and measurable characteristics that are used to label and describe individuals. Face perception is a routine task for humans, thus the recognition of human faces is an inevitable topic for developing machine vision systems. Given a static image from a sequence of images, a generic face recognition system is able to detect the face region (see Figure 1.2a) and identify the face automatically. Face detection is usually related to the topic of image processing and segmentation, while the identification of a face is generally a recognition problem. Inspired by the human vision system, face recognition techniques generally contain solutions for the following three tasks [227]:

(a) (b)

Figure 1.2: Face detection based on facial features [169]: (a) Face detection in different scale. (b) Principal facial features (in white +) located.

- Extracting the local features of the face image.
- Representing the face with a model of the extracted features.
- Performing classification on the representations.

In early research in this field, the problem of face recognition was treated as a 2D pattern recognition problem. In such approaches, explicit local facial features, such as eyes, nose, and mouth are first located (see Figure 1.2b), and the face image is represented based on the geometry of local features, such as distances and positions [31, 86, 131, 264]. Techniques such as graph matching and principal component analysis are used in the development of geometry based methods. The geometric feature-based methods are robust to the changes in illumination [95]. However, the performance of such approaches strongly relies on the accuracy of registration of local facial features, which becomes the bottleneck of these approaches.

Many recent approaches are developed based on the *eigenfaces* [170, 241], which takes a holistic approach to face recognition. Such approaches represent the facial image as a projection on lower-dimensional subspaces called eigenfaces [227]. Statistic techniques provide powerful solutions for this problem, such as kernel principal component analysis (kernel-PCA), kernel linear discriminant analysis (kernel-LDA), discrete cosine transform (DCT), hidden Markov model (HMM), Fourier transforms (FT) and Support Vector Machines (SVMs) [42]. Artificial Neural Networks are another successful tool for patter recognition problems. Techniques as Deep Neural Networks (DNNs), Multilayer Perceptrons (MLPs) and Self-Organizing Maps (SOMs) are widely used in modern face recognition approaches [227]. There are also many hybrid approaches which use both statistical pattern face recognition techniques and neural networks [187, 203].

3D face recognition emerged recently, with the purpose of building a 3-dimensional geometry of the human face for recognition. It has been shown that the approaches using 3D face models can achieve higher accuracy in recognition than their 2D counterparts [227], because 3D methods are better at capturing and explaining the full variability of facial appearance in a wide range of viewing angles. The performance of 3D face recognition is strongly limited by the technique of acquiring the 3D information accurately. Some approaches use 3D sensors to capture the 3D points on the face [31]. Other approaches use multiple cameras to recover the 3D points based on multiple view geometry [164]. Moreover, multiple images from different angles from a common camera can also to be used to

create the 3D model with significant-processing [45]. Nowadays 3D face recognition is an active research topic in the field of face recognition.

Great progress has been made in the field of face recognition since the pioneers started do research on recognizing human faces using computers in the 1960's [59]. During the last 50 years, thousands of techniques and algorithms have been developed to address different requirements of applications. Nowadays, face recognition is not only a research topic in laboratories, but it can be found in a variety of commercial applications, such as consumer-level photo applications, human–computer interaction (HCI) systems, identity verification, desktop login, parental controls, and patient monitoring as well as security [227]. Although great improvements have been made in face recognition, building a computer system comparable to the human vision system is still an on-going research area. More descriptions and discussions about this topic can be found in a number of surveys and books [1, 101, 227].

Human Activity Recognition

The goal of human activity recognition is to automatically analyse ongoing activities from an unknown video (i.e. a sequence of image frames) [6]. The terminology "activity" refers to a sequence of human body movements, which are performed by one or more agents who could be interacting with each other in a constrained manner [240]. In some vision literature, the terms "action" and "activity" are used interchangeably. In this work, we use the concepts proposed by Aggarwal and Ryoo [6], who categorize the general "activity" into four levels:

1. *Gestures* are elementary movements of a person's body part, such as "shaking an arm" or "raising a leg".
2. *Actions* are activities of a single person, which are composed of a series of gestures, such as "walking" and "running"
3. *Interactions* refer to the human activities that involve two or more persons and/or objects, such as "two persons fighting" or "one person bouncing a ball".
4. *Group activities* are those performed by conceptual groups composed of multiple persons and/or objects, such as "a group of persons marching".

Figure 1.3 shows some examples of different human activities.

The task of activity recognition is challenging mainly due to variations in motion performance, recording settings and inter-personal differences. Human activities are complex and highly diverse. At low-level, gesture recognition needs to address the tasks of detecting human parts (faces, hands, arms, etc.) and analysing their motions [5]. For high-level activity recognition, i.e. actions, interactions and group activities, there are intra-class variations and inter-class variations. For instance, a simple walking movement can differ in speed and stride length. The inter-class variations are caused by the anthropometric differences between individuals, such as on the boundary between "running" and "walking". Besides, changes of the environment, occlusion, illumination, viewpoint and camera settings are issues which influence the appearance of activities, thus affect the activity recognition. Since activities are segmented in time, the temporal variations also become a challenge of activity recognition problem. Moreover, the acquisition of "ground-truth"

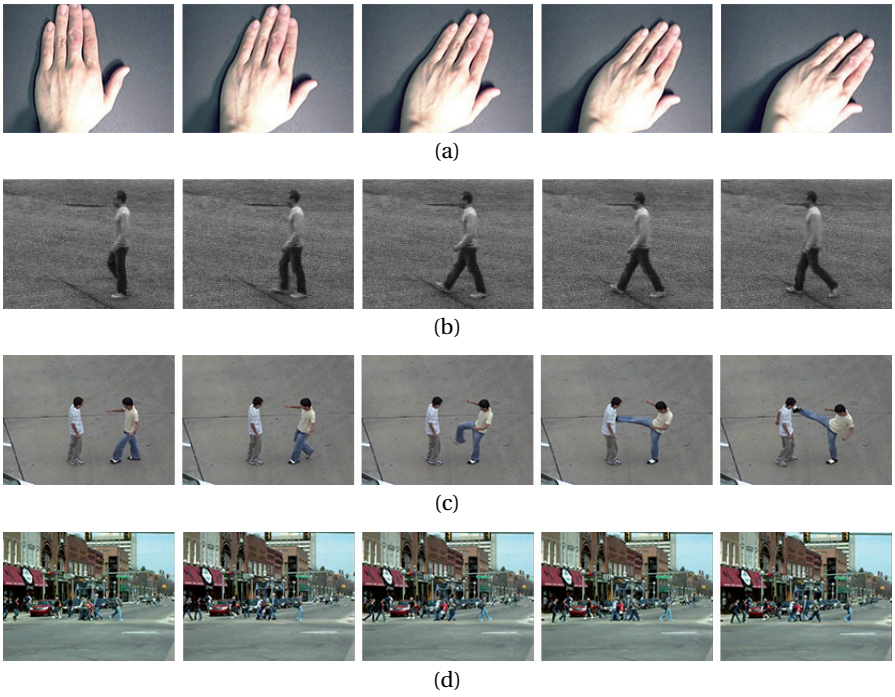


Figure 1.3: Examples of different level of activities: (a) Gesture: a hand waving [161]; (b) Action: one person walking [210]; (c) Interaction: kicking [206]; (d) Group activities: crossing the road [63].

training data is challenging, because the collection and the labelling of training data are laborious tasks.

Activity recognition is closely related to other research topics, of which two are the most significant: vision based human motion estimation and human/pedestrian detection. Human motion analysis focuses on the recovery of human poses and motions from image sequences. Human activity recognition is generally addressed a classification problem. Human or pedestrian detection is a related area to human activity recognition. Since a broad range of research and techniques are related to this topic, variations of taxonomies of human activity recognition methodologies are proposed according to different criteria in existing survey papers. Some researchers use a taxonomy based on the hierarchy of "activity" categories, which correspond with different levels of vision tasks. Some use a functional taxonomy with the subsequent phases. More information about this work can be found in a number of surveys [6, 62, 190, 240].

Although strategies and techniques vary in different human activity recognition system, the two parts: image representation and activity classification, are present in every vision-based activity recognition approach. The general design of a human activity recognition system follows a bottom-up construction, which involves a process of interpreting the images from low-level visual features to high-level semantic interpretations. Vision techniques from different levels of processing are related to different parts of the

human activity recognition system [5, 6, 32]. At the lower levels there are modules such as background–foreground segmentation, feature tracking, object detection. Techniques and algorithms of optical flow, feature extraction, tracking, background subtraction, spatio-temporal filters, etc., are developed as solutions for these modules [120]. At the mid-level there are gesture and action recognition modules, where the motions can be represented as trained templates or with certain parametric models. Approaches such as hidden Markov models (HMMs), Linear Dynamical Systems (LDSs) and Nonlinear Dynamical Systems, etc., are used for modelling the actions (or gestures) [240]. At a high level, the reasoning engines encode the semantic activities (interactions and group activities) based on lower level action primitives [6]. In order to model complex activities, researchers successfully developed tools based on graphical Models, such as Belief Networks, Petri Nets, etc. Syntactic and logic-based approaches are other active branches in this field.

Solutions to activity recognition are applicable to domains of visual surveillance, video retrieval, human–computer interaction, motion sensing games, and so on. One of the most commercially viable applications is retrieval of consumer content such as sports videos. Effective recognition of human behaviour is quite helpful in creating computers that can better interact with humans, of which the Kinect for Xbox is a successful example.

Vision based Mobile Robot Navigation

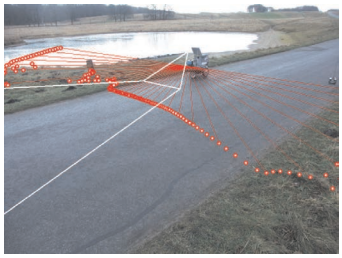
For mobile robots, the ability to navigate in the environment is crucial. Navigation can roughly be described as the process of determining a suitable and safe path between a starting and a goal point for a robot travelling between them [34]. Thus a navigation system needs to address two fundamental issues: determining the path to the goal point, and the avoidance of obstacles, for which the acquisition of environment information is an essential issue. A variety of sensors are used for acquiring environment information, such as sonar, position sensing device (PSD), laser, radar, and cameras.

In particular, many researchers focus on vision based navigation using visual sensors, such as photometric cameras and laser-based range finders. Cameras are the most popular visual sensors in applications because they have the advantage of providing extensive visual information while having low weights, small sizes, and low costs [80]. Solutions of vision-based navigation are applied to a wide range of areas, such as autonomous ground vehicles (AGVs), unmanned aerial vehicles (UAVs/drones), assistant robots, robots for exploration, etc. Figure 1.4a and 1.4b show examples of indoor and outdoor robots respectively.

Vision based navigation of mobile robots aims at building a map or some other model of the environment around the robots by analysing the images or videos from visual sensors. Typically, navigation involves three tasks: environment interpretation, localization and path planning. The interpretation of the environment includes the process of extracting information from the surrounding environment. The environment is perceived in the form of geometric information, such as landmarks, object models or environment maps, in two or three dimensions. Localization finds the agent's position within the map or the model. Path planning involves the determination of a sequence of actions aimed at accomplishing some goal.

The biggest challenge of vision based navigation is in handling the variations of the environment, including illumination changes, weather conditions, time or season changes

(a) Route of an indoor cleaning robot [197]



(b) An outdoor robot [10]

Figure 1.4: Examples of mobile robots operated in different environment

and dynamic obstacles. Therefore, the representation of the environment significantly affects the performance of localization and path planning.

Schemes of vision-based navigation vary according to the requirements and characteristics of applications they are designed for. For instance, autonomous underwater vehicles have to cope with the special characteristics of light propagation undersea which the autonomous ground vehicles do not meet. De Souza and Kak [73] structured vision-based navigation in two main categories: indoor navigation and outdoor navigation. Bonin-Font et al. [34] divided the vision navigation system in those that need prior knowledge of the whole environment and those that perceive the environment while they navigate through it.

Map-Based Navigation consists of providing the robot with a model of the environment. For some navigation systems, a complete map of the environment is required before the navigation starts (map-using systems). Other systems are able to explore the environment and automatically build a map of it (map-building systems) [34]. Map-based navigation can improve the precision in localization that at the cost of consuming more computational resources, time and storage. Thus they are mostly used in indoor environments, which are usually smaller and simpler than outdoor environments. The environment models may contain different degrees of detail, varying from a complete CAD model of the environment to a simple graph of interconnections or interrelationships between the elements in the environment. Research on 3D metric environment representations has grown fast in recent decades. Localization of map-based navigation is achieved by tracking and matching of some known features (landmarks), where techniques of feature detection and matching, low-level image processing, etc., are involved in this topic.

For mapless navigation systems, knowledge of the environment is acquired by processing the images or videos obtained by visual sensors in real-time. During the last decades new vision techniques have been used to vision-based navigation systems. Optical flow estimation provides an efficient way to analyse the apparent motion features of image pixels. Researchers have built numerous mapless navigation systems by combining optical flow with other techniques, which has been categorized as a big group by Bonin-Font et al. [34]. Some approaches represent the environment objects by appearance-based models, where the localization is achieved by image segmentation and matching algorithms and object recognition techniques. New navigation strategies arise with the development of new techniques for tracking moving elements (corners, lines, object outlines or specific

regions) in a video sequence. More details about these categories of vision-based navigation are available in a number of surveys and books on this topic [34, 73, 237]

1.1.3. CHALLENGES

Although progress has been made, object detection and recognition remains an open problem today. There are real applications on the consumer market that have contributed to our daily life, as well advanced applications that are presented in laboratory nowadays. In general, researchers are pursuing a robust, accurate and high performance approach, which remains a great challenge today. The difficulty of the object detection and recognition problem comes from the challenges on the different levels of the process. The typical challenges in an object detection and recognition system are as follows:

Appearance Changes of Objects

The appearance of an object in the image varies due to the following reasons:

1. *Illumination changes*
   In natural scenes, light always changes due to the dynamic environment we live in. The sunlight varies during the day in outdoor scenes, while illuminations can also changes in indoor scenes. Illumination strongly affects intensities of pixels in the images or videos, which lead to changes in the appearance of objects. Shadows and their changes caused by illumination make the problem even more complicated.

2. *View point changes*
   The appearance of objects varies between different view points. Even humans can suffer the vision paradox problem in some situations. Handling the change of objects between different view points is an essential issue in object detection and recognition.

3. *Pose variation*
   The pose variation of an object causes the appearance to change, and sometimes leads to self-occlusion.

4. *Deformation*
   Deformation is a change of the shape or size of an object due to some factors, such as an applied tensile (pulling) force or temperature changes. Usually the deformation is negligible for a rigid object, but it can be substantial for non-rigid objects.

5. *Occlusion*
   Objects can be occluded dynamically due to location changes and movements of objects in the scene. It is a challenge to discriminate the objects that are occluded and to recognize their locations.

6. *Clutter*
   The presence of background clutter increases the difficulty to segment out the objects. It is even more challenging to detect an object in camouflage.

Noise
Images and videos are recorded by cameras, which are usually superimposed with noise due to the intrinsic properties of a camera system. For example, the sensor noise and compression artifacts are two typical types of noise. Noise contaminate the visual signal, which may affect the performance of a vision system.

Motions
Natural scenes always contain movements. Motions present in a scene affect the video processing and object detection in the following ways:

- The camera equipped by a mobile robot could be unstable. The movement of a camera could affect the videos and introduce noise.
- Objects present in a scene can also move in different ways. The motion characteristics (speed, orientation, etc.) of the objects play an important role in detection.
- Some movements, like waving trees and moving clouds, are not welcome in detecting foreground objects (such as in a vehicle detection system). Background clutter therefore can increase the difficulty of detecting foreground objects.

Scale, Orientation and Shape Changes
Some objects vary in scale, orientation and shape during time (e.g. non-rigid objects).

In recent years, the several of the above mentioned challenges have been successfully addressed by deep neural networks for specific applications areas of computer vision [12, 138, 145, 265]. There still remain challenges and application areas where more research is required [188, 276]. One of these is motion-based image segmentation [26, 158].

1.2. PROBLEM STATEMENT AND RESEARCH QUESTIONS

The objective of this thesis is to segment out moving objects using motion information available in a video. With robot soccer as an application domain in mind, we mostly deal with rigid body motions and videos recorded by a monocular camera. The challenge is that we do not assume knowledge of the number of objects present, the appearances of these objects and their motion models. The background could be cluttered, and the camera can either be static or undergo a general 3D rotation and translation.

We choose not to use some black-box approach such as (deep) neural networks to address our research objective since we wish to gain a clear understanding of how motion information can be exploited. In other words, we prefer a glass-box or explainable AI approach. This implies that we will not focus on learning based approaches despite the recent successes of deep neural networks in computer vision. This also avoids the need for a large set of annotated motion-based training data. The results of our research may be beneficial to choose a learning-based approach in future research. There exist different neural networks architectures, which may encode specific algorithms; e.g., a convolution algorithm. Knowledge about how motion information can be used for segmentation, may help to choose the appropriate neural network architecture.

We aim at segmenting out the moving objects based on the motions occurring in a video, as shown in Figure 1.5. This is a process of interpreting the low-level visual informa-

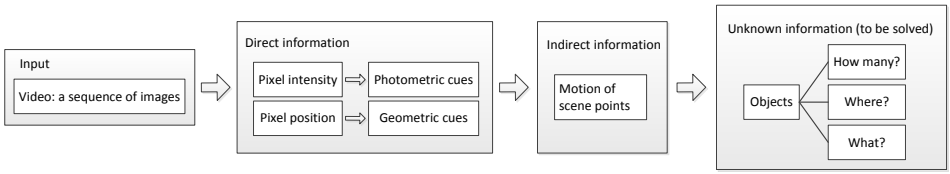


Figure 1.5: The objective of our research is to interpret the low-level information (pixel properties) to high-level information (object descriptions), by recovering the implicit information (motion).

tion, which contains redundancy and in this form is meaningless for scene understanding, to high-level descriptions of the scene contents. Since motion information is implicit in a video, our research focuses on the acquirement, analysis and utilization of motion data. Multiple tasks are involved in accomplishing this research objective. These tasks are addressed by the following research questions:

Research Question 1 How to design a framework for motion based video object segmentation?

Our research attempts to develop an overall approach that can automatically segment out and track the moving objects from video sequences. This is a complex problem that involves several sub-tasks and cannot be solved straightforward. The problem solving process can be generally divided into three stages: the motion estimation, motion segmentation and object classification, where each stage depends on the stage before. Therefore, we need to consider the effectiveness of the solutions both from a local and a global perspective. For example, suppose solution A can achieve better performance than solution B in motion estimation, but the motion segmentation based on solution A is worse than that based on solution B. Then A to be abandoned because it is locally optimized but not optimized globally. In this thesis, we will set up an effective framework for the problem solving process.

Research Question 2 How to extract and represent motion data from a video sequence?

The input video sequence consists of consecutive digital images, also called frames. A digital image is a discretization of a projection of the 3D world space. The motions occurring in the 3D real world cause the changes in the 2D images, that are taken at different times, i.e. the 2D image motions. For the task of segmenting out moving objects, it is essential to obtain accurate image motions [41]. It is challenging to obtain accurate estimates of image motion because illumination changes, occlusion, etc., also lead to changes in image frames. To extract the motion data from a video sequence, the fundamental issue is how to represent the motion? We can describe the motions between images by the movements of pixels, as pixels are the smallest elements in an image. We can also track local features that are visually unique in a video sequence. Moreover, the image alignments (image derivatives, optical flow, point correspondences) largely affect the quality of obtained motion data [155]. We will investigate the types of motion data that can be extracted from a video sequence, and evaluate which type is beneficial in which way to motion segmentation.

Research Question 3 How can the objects be segmented out based on the motion information restricting to 2D consistency in the images?

Given the estimated motions in a video sequences, it is possible to divide the images into multiple components, each being a group of pixels (or feature points) that follow the "same" motion. This task is called motion segmentation in computer vision [41]. Motion segmentation is affected by the accuracy of estimated motions, the camera models used for projection, motion models used to describe the "same" motion [155]. Since we focus on the motion segmentation in video sequence, we will investigate the design of a segmentation algorithm based on the acquired 2D motion data extracted from a video sequence, by addressing the following questions?

- What are the effects of camera projection model?
- How to model the motions that are extracted from a sequence accurately?
- For the video sequences containing multiple objects under different motions, how to determine the number of objects?

To investigate the influences of motion estimation on this stage of the overall framework, we will further evaluate the segmentation algorithm with respect to different motion estimation approaches.

Research Question 4 How to retrieve 3D motion consistency based on the 2D motion data?

The estimated motions in the image frames of the sequence are 2D projections of the true motions in 3D scene. Segmentation based on the 2D motions ignores the effects caused by projection, such as depth discontinuities, perspective effects, etc. These effects can break a 3D motion into different 2D motions, and therefore the object associated with the 3D motion is sometimes segmented into more than one region. This is called over-segmentation [155]. To explore the usage of 3D motion models, we will investigate to retrieve the 3D motion consistency based on the 2D motion information and projection geometrics, and to improve the segmentation results.

Research Question 5 How to segment out objects from static frames using moving information from videos?

In a video, the objects may move in some frames and be static in others. Segmenting from motion information implies that these object can only be detected when they move. How can we segment out these objects when there is no motion detected? Those frames segmented by motion provide the hints. We can learn object features from the segmented frames and then use this knowledge to segment out static objects from new frames without motion information.

1.3. THESIS OUTLINE

This thesis is organized as follows.

- In Chapter 2, we provide a general overview of related work in object segmentation from videos.

- In Chapter 3, an introduction of the basic techniques and algorithms related to our research is provided.
- In Chapter 4, we address the first research question. We design a framework for unsupervised video object segmentation based on motion. This framework addresses the process of acquiring and analysing the motion information from a video sequence, including the estimation of motion data, segmentation based on motion, and learning from motion. The influences of intermediate steps of this process, such as the choice of motion data types, the techniques used for analysing the motion data, will be investigated and evaluated in Chapters 5 and 6.
- In Chapter 5, we investigate the approaches for extracting motion data from a video sequence, which answers the second research question. We generate three types of motion data from the input videos, i.e. the two-frame pixel movements, the sparse point trajectories and the dense point trajectories. The influences of three different data types are explored in Chapter 6, since they are used as the input data for the motion segmentation algorithms.
- In Chapter 6, we propose a motion segmentation algorithm, which can deal with all three types of obtained motion data. We investigate the influences of different motion data types on the motion segmentation results. We also discuss the evaluation methodology used.
- In Chapter 7, the fourth research question is addressed. We investigate the 3D motion consistency based on the obtained 2D motion data. We also discuss the utilization of 3D motion consistency for motion segmentation.
- In Chapter 8, we investigate the solution to the fifth research question. We investigate the segmentation of objects from static images, by learning from the motion segmentation results.
- In Chapter 9, we summarize our findings and the characteristics of our approach and discuss the directions of future work.

2

OVERVIEW OF VIDEO OBJECT SEGMENTATION

Remarkable success has been achieved in computer vision research owing to the development of computing devices and neuroscience. Numerous sophisticated techniques and algorithms have been proposed for object detection and recognition under various conditions and circumstances. Schemes and strategies vary in different object detection and recognition systems, for example, properties and features of interest are different for indoor and outdoor scenes. Nevertheless there exist common modules involved in any detection and recognition system. In this chapter, we will introduce the essential components of object detection and recognition systems and review the leading state-of-the-art techniques performed in each area.

2.1. SYSTEM COMPOSITION

In an object detection and recognition system, some basic activities are involved as shown in Figure 2.1 [9, 100]. The image processing refers to the low-level processing on pixels, such as noise removal, where the outputs are still images. This thesis focuses on the last three activities; i.e. feature extraction, segmentation and classification.

Feature extraction aims at extracting higher-level descriptors from images based on salient features rather than pixels [101]. Segmentation is the task of partitioning images into regions of interest [227]. Classification assigns input data to different categories [227]. Given processed input images, any object detection and recognition system could contain the three activities, or some subset of them [227]. The stages for implementing these activities are not fixed because they can be used to solve different problems under different schemes. For example, features can be extracted based on segmentation results [195, 238], while segmentation can be applied starting from the feature points [54, 98, 146, 154]. Moreover, classification algorithms can be used to recognize the detected objects [84, 138, 213], while object detection can also be achieved by classification algorithms in a top-down system [51, 141, 183, 212].

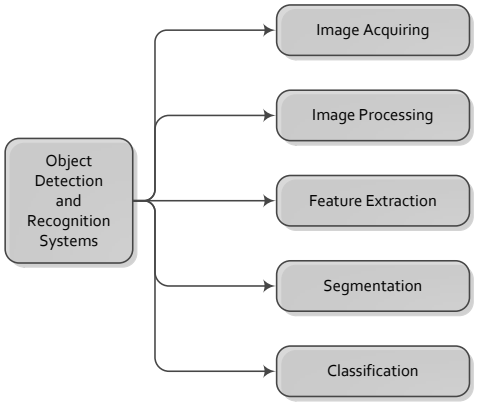


Figure 2.1: Basic components of an object detection and recognition system

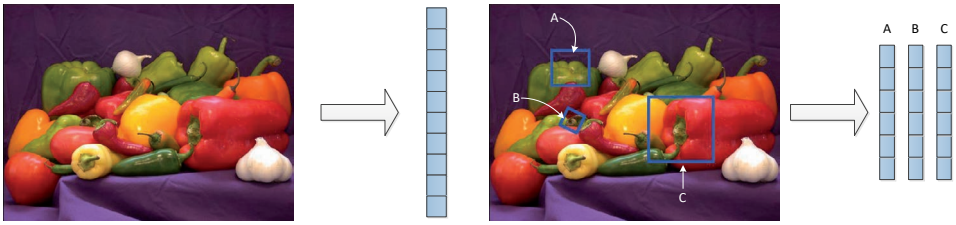
Generally, approaches of object detection and recognition from images follow three types of paradigms: bottom-up, top-down and hybrid [35]. In a bottom-up paradigm, the objects are detected by analyzing and segmenting the images from low-level image data. This scheme is usually used for unsupervised or semi-supervised tasks where additional information about objects is not provided. Top-down approaches analyze the images from a high (semantic) level, and are used in applications of detecting particular objects. In systems with more sophisticated requirements, a combination of the two paradigms often is more powerful [13, 276].

2.2. FEATURE EXTRACTION

Feature extraction addresses the problem of transforming "raw" pixels into a reduced set of features. It is an essential step in pattern recognition, machine learning, image processing and computer vision. When the input data is too large and redundant, feature extraction techniques aim to obtain the most relevant information from the input data and represent that information in a lower dimensional space [139]. In this section, we focus on feature extraction in computer vision, including the major types of features, techniques of feature extraction, and some examples of applications.

A common problem in computer vision applications is to discover the semantic concepts from input images. However, processing of the raw image data is laborious and time-consuming for most sophisticated vision algorithms. Feature extraction provides one way of dimensionality reduction, which plays an important role in most vision applications, such as image alignment, image classification, object detection and tracking, recognition, 3D scene reconstruction, robot localization, etc. The performance of such applications significantly depends on the selection of features.

Generally, image features can be categorized into two categories: global features and



(a) Global features encode the whole image into single descriptors.

(b) Local feature descriptors are computed multiple-times for small regions in an image.

Figure 2.2: Image features: Global vs. Local

local features. Global features (e.g., color and texture) aim to describe an image as a whole and can be interpreted as a particular property of the image involving all pixels [114]. Local features are representations of some salient regions in the image, such as corners, interestingly shaped patches, viewpoint invariant structures, etc. These local features are usually called keypoints or feature (interest) points. Figure 2.2 illustrates the basic concept of image representation with global and local features.

The global features represent the image by one multidimensional feature vector, whose values are computed by measuring the image attributes such as intensity, color, texture, or shape. Global features are compact and fast to compute, while the requirement of memory is relatively small. These advantages make global features good for applications on a large datasets, such as image retrieval and scene recognition [114]. Moreover, global features are also useful in classification tasks where a rough segmentation of the object of interest is available [179]. However, for applications like object detection and image registration (an image processing technique used to align multiple scenes into a single integrated image), global features meet their limitations. It is impossible for global features to distinguish foreground objects from the background or to find corresponding parts of two images, because they are invariant to significant transformations and sensitive to clutter and occlusion. These limitations can be overcome by introducing some extra steps, such as image segmentation algorithms and sliding window strategies.

Local features are distinctive and stable, while they are invariant to some transformations and local changes of illuminations. Properties of local features enable researchers to find image correspondences regardless of occlusion, changes in viewing conditions, or the presence of clutter [227]. Such advantages make local features perform well in various applications, such as object detection and recognition, image registration, stereo matching, video stabilization, etc. Moreover, local features have been used to generate descriptions of image contents for image retrieval and scene recognition tasks. They can even achieve a higher performance than global features in large-scale image search [114]. However, applications using local features usually need a significant extra amount of memory compared to global features.

In practice, there are no strict rules for utilizing global or local features. The global feature descriptor can be applied on parts of an image, while the local features are also able to represent the whole image. In some applications, the combination of global and local

features is more beneficial. "Good" features should meet the requirements and constraints of a certain application. The truth is that a "good" feature for one application can be useless in the context of a different problem. For applications such as image classification, camera calibration and object tracking/recognition, it is important to find features which are robust to changes in brightness or viewpoint and to image distortions (e.g. noise, blur, or illumination) [114]. For real time applications, features need to be fast to compute. For navigation applications, the feature descriptor must be robust to clutter and occlusion. In the literature, a rich variety of feature extraction methods have been proposed to meet specific requirements [106, 139, 167, 242]. In this thesis, we focus on feature extraction techniques, which have been successfully applied in object detection and recognition applications. A brief description of the state-of-art feature descriptors is given immediately below.

2.2.1. HISTOGRAM OF ORIENTED GRADIENTS

The Histogram of Oriented Gradients (HOG) proposed by Triggs and Dalal [70] measures the orientation and strength of image gradients within an image region. The basic idea of HOG is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. It uses a block-pattern for normalizing gradient histograms in cells of an image to compute the feature descriptor.

HOG is invariant to small shifts and rotations, and has been widely used in object detection and recognition applications, such as pedestrian detection and video surveillance, face detection and recognition, etc.[196, 227]. The limitation of HOG features is that they are not able to represent richer patterns.

2.2.2. SCALE INVARIANT FEATURE TRANSFORM

The Scale Invariant Feature Transform (SIFT) algorithm detects and describes local features in images, which was proposed in [151]. It has proved to be an efficient and robust way of detecting points of interests, which is useful in object detection and recognition [152]. The standard SIFT algorithm has four steps:

1) identification of potential keypoints from "scale-spaces",
2) keypoint localization,
3) orientation assignment,
4) keypoint description.

In the SIFT algorithm, a scale-space is first generated from the original to ensure scale invariance and potential keypoints are detected from the scale-space using the difference-of-Gaussian (DoG) [136, 148]. Then the bad key points are eliminated in the second step. In the third step, the orientation is calculated for each keypoint in order to make it rotation invariant. In the last step, the feature vector of a keypoint is constructed based on the orientation histograms computed in the third step.

SIFT features are invariant to image scaling and rotation, and robust to large amounts of pixel noise [152]. Because of the scale-invariant properties and the high distinctive feature expression, SIFT features are usable in object recognition [151, 227].

Compare to many other descriptors of that period, the standard SIFT descriptor has been shown better matching performance [167]. The limitation of standard SIFT is that the

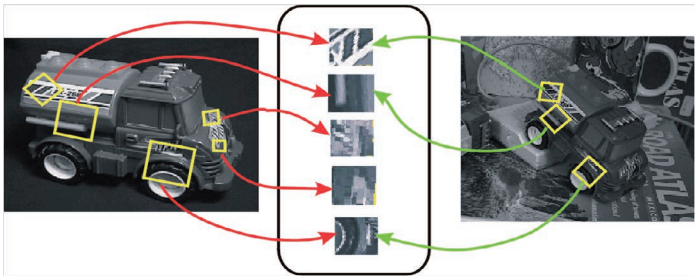


Figure 2.3: SIFT keypoints are invariant to affine transformations

computation of feature vectors is complicated and slow. Moreover, it is only invariant to minor affine changes (within 50 degrees). A number of variants and extensions of standard SIFT have been developed in recent decades, for more robustness and distinctiveness with scaled-down complexity [22, 132, 167, 171]. Nowadays, SIFT has become one of the most popular local feature extraction techniques used in computer vision tasks.

2.2.3. SPEEDED-UP ROBUST FEATURES

The Speeded-Up Robust Features (SURF) detector-descriptor algorithm is an efficient alternative to SIFT [21]. It is developed for the purpose of achieving a higher speed for local feature detection and description than SIFT methods. The SURF algorithm is based on the SIFT algorithm and follows a similar scheme of keypoint detection and description. The differences are:

1) The localization of potential keypoints is achieved using integral images, which is faster than DoG in SIFT.
2) The orientation is computed by the Haar wavelet in horizontal and vertical directions for a neighbourhood, instead of computing gradient histograms as in SIFT.
3) The feature vector is constructed based on the summation of Haar wavelet responses, which results in a descriptor with a total of 64 dimensions.

In short, the new characteristics of SURF improve the speed in every step compared to SIFT. Analysis shows it is 3 times faster than SIFT while performance is comparable to SIFT [21]. However SURF is not fully affine invariant, it is not good at handling viewpoint change. Moreover, it is not as stable as SIFT. A detailed comparison of the two methods can be found in [130].

2.2.4. LOCAL BINARY PATTERNS

Local Binary Patterns (LBP) is a texture operator which builds the spatial structure of a texture by creating the ordering relationship of each center pixel and its neighboring pixels [114, 178]. LBP features are invariant to monotonic transformations of the gray-levels, which makes it robust to illumination changes. Another advantage is its computational simplicity. Due to these advantages, LBP has become a popular approach for many applications, such as texture classification, real-time recognition, etc.

LBP also has some limitations. It is not invariant to rotation and also not robust on flat

image areas. Several variations of LBP have been proposed to increase the applicability of LBP. A survey of different versions of LBP can be found in [40].

2.2.5. FEATURES FROM ACCELERATED SEGMENT TEST

The Features from Accelerated Segment Test (FAST) algorithm was proposed by Rosten and Drummond [201] based on the Smallest Univalue Segment Assimilating Nucleus (SUSAN) corner detector [220]. Experiments have shown that corners are important for the human vision system, and that removing the corners from images impedes human recognition. Corners in feature extraction techniques correspond to areas in 2D images with high curvature, and can be found at various types of junctions, on highly textured surfaces, at occlusion boundaries, etc.

SUSAN provides a criterion to identify a corner, by comparing the similarity of a centre pixel and the pixels in its circular neighbourhood [220, 242]. FAST compares pixels on a circle of fixed radius around the point, and determines whether it is a corner based on the definition of the SUSAN corner. Instead of testing all pixels in the neighbourhood, FAST uses a strategy to test fewer pixels, which increases the processing speed.

The FAST corner detector has proved to be faster than many existing detectors, which makes it suitable for real-time video processing applications [201]. However, it also suffers from some disadvantages: it is not robust to noise and not invariant to scale changes, and it depends on a threshold.

2.3. SEGMENTATION

Segmentation is the process of dividing the input data into groups of interest, where each group contains the data points with similar attributes. This problem is known as cluster analysis in statistics, which has been studied extensively [227].

Research in image segmentation rose with the development of computer vision since the 1960s [199, 227]. Image segmentation is the essential step between low-level image processing and high-level image analysis (semantic representation of features, objects, and scenes). The objective of image segmentation is to partition an image into regions of interest, which are meaningful to the subsequent processes on higher levels. Early research on image segmentation addressed a single image, where visual attributes (intensity, colour, edge, texture shape, etc.) and their spatial relationships, were taken into consideration. Such image based segmentation can detect precise contours of feature regions. However, such approaches often lack the ability of locating a specific object without additional information because an object can be split into several regions corresponding to different appearance properties.

In recent decades, there is a rising need of segmenting out moving objects from backgrounds in videos due to the development of video processing applications. A video, which is composed of a sequence of consecutive images, provides additional cues of temporary variations such as the motion in the scene. Motion based segmentation aims at segmenting images into parts corresponding to their motions. Thus motion segmentation is able to segment out the objects of salient motions using proper motion models. However, motion based segmentation algorithms are plagued by fundamental limitations associated with motion estimation: occlusion and aperture problems [227].

Nowadays, image based and motion based segmentation techniques are both used for object detection and recognition systems. Image based segmentation algorithms are often used in detecting particular objects, where the descriptions of such objects are given in advance. For example, in a face recognition system, segmentation algorithms may generate separately regions as skin, eyes, nose and lip. These sub-regions can be merged together to get a face region if a model of a face (which describes the face features and their spatial relationship) is provided [269]. Motion segmentation is suited for detecting objects with salient motions where the category of present objects is unknown, for instance in robot navigation and video surveillance [270]. A brief review of the state-of-art techniques for image based segmentation and motion based segmentation is given in subsections 2.3.1 and 2.3.2.

2.3.1. IMAGE-BASED SEGMENTATION

Techniques and algorithms for image based segmentation vary considerably because research in this field has explored different directions. A common criterion divides segmentation algorithms into two categories according to the basic properties used for segmentation: discontinuity and similarity [101]. In the first category, algorithms aim to detect the discontinuous properties in an image, i.e. the edges, to extract contours explicitly. Segmentation algorithms in the second category aim to detect regions with similar properties. Figure 2.4 shows some examples of image segmentation based on different properties. In this section, we will introduce some principal approaches in both categories, as well as techniques combining the two kind of properties.

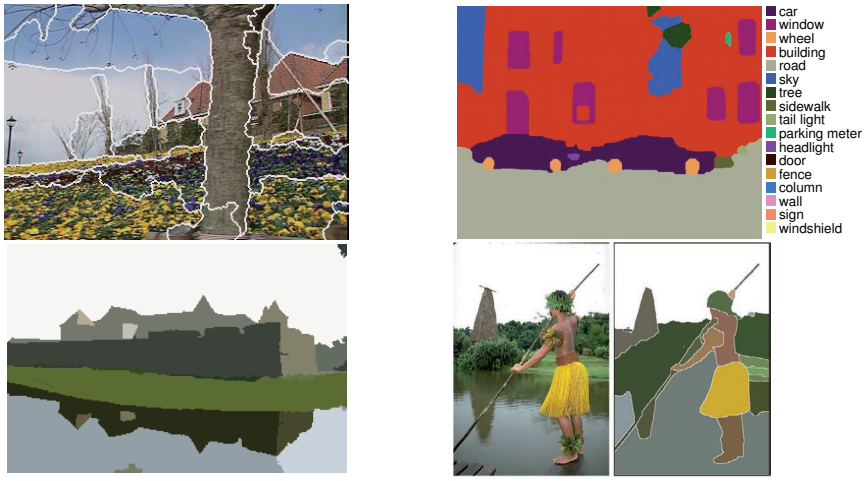


Figure 2.4: Some image segmentation examples: (a) segmentation of color-texture regions [72], (b) semantic segmentation using per-exemplar detectors [230], (c) mean-sift based segmentation of a landscape image [65], (d) segmentation based on contour detection [14]

EDGE-BASED SEGMENTATION

Edge-based segmentation methods find the edges between the regions and determine the segments as regions within these edges, based on edge detection techniques. Edges are

local changes in the image intensity, which are useful to distinguish boundaries of objects or homogeneous regions in a scene. Segmentation based on edges often relies on detecting high quality edges. Edge detection is an active research area in which many techniques and algorithms have been developed that are perceptive to certain types of edges.

A common type of edge detection techniques uses a gradient operator to identify locations of large intensity changes. The most important ones in this class are the Robert operator, the Sobel operator, and the Prewitt operator [215, 219]. These methods are efficient for detecting edges and their orientations. They are also sensitive to noise because of using first order derivative operators.

Some approaches use second order derivatives to detect edges based on a Laplacian filter. However they are extremely sensitive to noise and no directional information about the edge is given [101]. The Marr and Hildreth combine a Laplacian detector with Gaussian filtering for edge detection, which aims to reduce the noise from images before the edge detection process [56]. Marr-Hildreth method is known as the Laplacian of Gaussian (LoG) operator for edge detection, which is more likely to find the correct places of edges because it tests a wider area around the pixels [41, 159]. The disadvantage is its malfunctioning at corners, curves and areas where the grey-level intensity function varies [215, 219].

The edge detection technique proposed by Canny [52] uses a Gaussian filter for image smoothing, and adopts a double thresholding strategy for detecting and linking edges. The canny detector ensures good noise immunity and at the same time detects true edge points with minimum error. It outperforms most of the gradient and Laplacian methods, but it is limited in real-time applications by its higher computational complexity [159, 215, 219].

THRESHOLDING

Thresholding is the simplest segmentation technique, which separates an image into regions with thresholds based on the distribution of pixel values (intensity values or color). The key problem of thresholding is how to automatically determine an adequate threshold value to separate out desired objects. Thresholding can be viewed as a statistical decision problem which objective is to minimize the average error incurred by assigning pixels into two or more groups [101].

Segmentation can be done by thresholding using global information (e.g. the grey-level histogram of the whole image) or local information [110]. For global thresholding, threshold values are computed over the whole image. Another type of thresholding technique uses adaptive threshold values for different local areas, while different strategies are proposed to choose local areas and determine the proper threshold.

Based on the information the algorithm manipulates, thresholding segmentation techniques and algorithms can be categorized into six groups as follows [214]:

1) histogram based methods,
2) clustering based methods,
3) entropy based methods,
4) attribute based methods,
5) spatial methods using higher-order probability distributions, and
6) local methods of adaptive threshold values.

Descriptions and comparisons of these techniques can be found in the literature [110, 181, 214].

Thresholding is effective in segmenting images where the contrast between object and background is significant. However, image segmentation based on thresholding often fails to detect shape coherence because it is based on pixel-level attributes. Regions (or objects) of complex texture can be divided into multiple parts in thresholding based segmentation. Moreover, the performance of thresholding methods is sensitive to noise and non-uniform illumination. Image smoothing techniques such as Gaussian filtering, and edge detectors are often used to improve thresholding [101].

REGION-BASED SEGMENTATION

Region-based segmentation algorithms aim to find the regions directly according to the similarity properties of image pixels. Such algorithms aim to separate regions in which the pixels have similar values, where the selection of the similarity criterion plays an important role. Region-based segmentation techniques can be divided into two categories according to the strategy used for generating regions, i.e. region growing (merging) and region splitting and merging [101].

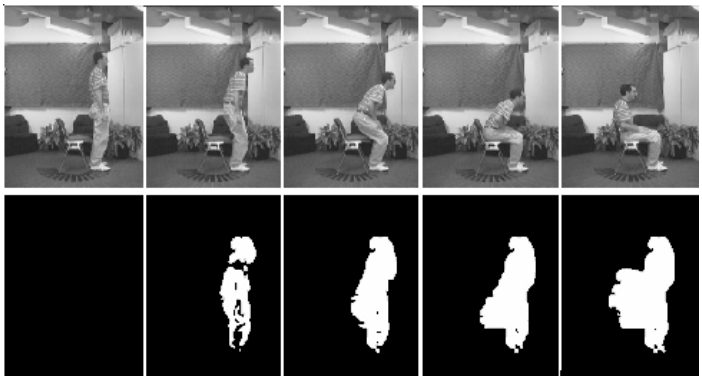
The region growing approaches follow a bottom up scheme, which groups pixels and small regions into larger regions based on pre-defined criteria. The basic region growing starts by choosing a set of "seeds" and the regions are grown by adding in neighbouring pixels that are similar to their own "seeds" [3]. The seed points can be selected either by a human or automatically, by avoiding areas of high contrast (large gradients). Lin et al. proposed an algorithm which can generate the seeds automatically [147]. Region growing techniques are conceptually simple and can achieve good segmentation results according to defined properties. They are, however, local methods and suffer from the initial seed-point problem, which means that different sets of initial seed points cause different segmentation results. The fast scanning algorithm [74] was developed to address the initial seed-point problem. It scans the whole image and determines if one can merge pixels into an existing clustering, thus seed points are not needed. This approach is good at matching shapes of real objects in an efficient way.

Another category of algorithms first divides an image into a set of disjoint regions and then merges and/or splits the regions, based on distinguishing the homogeneity of the image [119]. A combination of region merging and splitting was developed to avoid over-segmentation. The advantage of such approaches is that the image can then be segmented properly according to the demands of applications by giving splitting criteria. The limitation is that blocky segments might be produced [101].

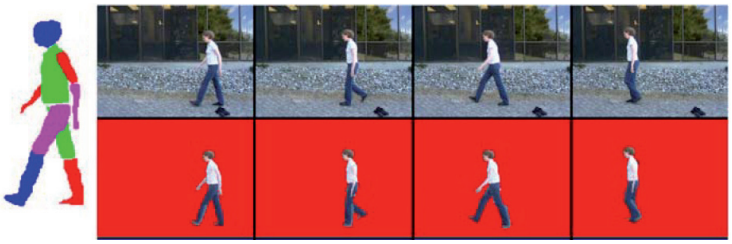
2.3.2. MOTION-BASED SEGMENTATION

A video sequence has much richer information than a single image. Motion is a primary feature in video, which carries a lot of information in spatiotemporal relationships between image objects [41]. In addition, image properties have a high correlation in the direction of motion, for example the color of a moving car keeps the same in images of the video sequence. Motion based segmentation techniques take advantage of these characteristics to locate regions or objects according to the motion information.

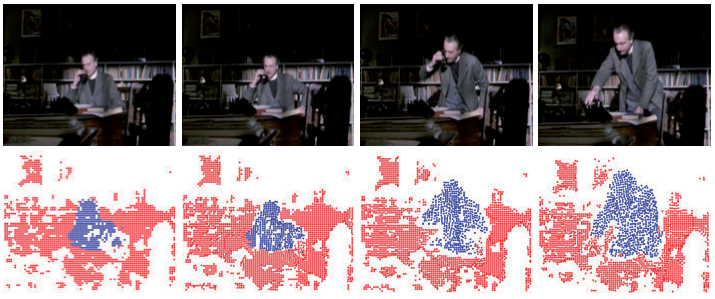
The procedure of a motion based segmentation application consists of two basic tasks: *motion estimation* and *motion segmentation* [41]. The *motion estimation* task aims to estimate the motion information by identifying interest points in images and measuring how



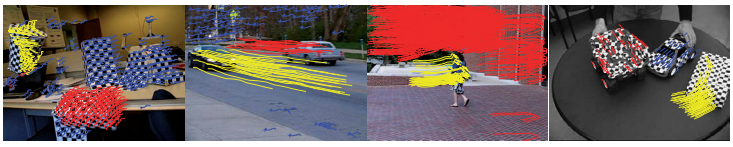
(a) Motion segmentation based on image difference [33]



(b) Motion segmentation based on layers [140]



(c) Motion segmentation based on particle filters [49]



(d) Motion segmentation based on manifold clustering [236]

Figure 2.5: Examples of motion segmentation

they move. Next a *motion segmentation* algorithm is used for partitioning the interest points into groups with similar motions.

MOTION ESTIMATION

Given a sequence of images, motion estimation aims to determine the motion of image elements, such as pixels, interest points, or regions [41, 101]. It is a typical technique in video processing, and has been widely applied in various applications, such as video compression and motion segmentation. Depending on the goals of motion estimation, algorithms of motion estimation show a rich diversity. In this thesis, we focus on motion estimation techniques for motion-based segmentation.

Since motion is described as changes between image frames, motion estimation techniques are based on comparing the similarity and dissimilarity of image features. The typical challenges of motion estimation are occlusions and illumination changes. A variety of schemes and solutions have been developed to detect motions in video sequences, which can be divided into 2 categories: *pixel-based methods* and *feature-based methods* [123, 233].

The pixel based methods are also called "direct" methods, because they aim at recovering the motion of each pixel directly from measurable image quantities [123]. Most pixel based methods are based on the "brightness constancy constraint", which is derived from the observation that the intensity values of a small region usually remains the same while its location may change [29]. The block matching algorithm, phase correlation, optical flow estimation and maximum a posteriori (MAP) estimation are typical approaches in this category [41]. Among them, optical flow estimation is one of the most popular techniques of motion estimation, for which a number of algorithms have been developed [118]. Horn and Schunck proposed a global constraint of smoothness and the computation of optical flow is formulated as a global energy minimization problem [118]. Lucas and Kanade proposed the local uniformity constraint and solved the optical flow using a coarse-to-fine approach [153]. The Horn-Schunck and Lucas-Kanade methods are at the core of two major directions of research for optical flow estimation, while both approaches have been extended by many researchers [8, 29, 30, 47, 108, 224]. There are several surveys of optical flow techniques in the literature [91, 165, 227], where more details can be found. Pixel based methods can achieve a high accuracy for recovering the pixel motions, and provide the advantage of accounting for subtle image details. However detection of long-range motion is difficult because the brightness constraint is only valid in limited situations (when the surfaces can be approximated as Lambertian) [46].

Feature based methods focus on establishing point correspondences between images by feature matching [233]. These methods first localize specific feature points, which should have good discriminative properties and there should be a high probability that the same point is selected in both images [242]. Therefore, the extraction of features plays an significant role in establishing accurate point motion [168, 233]. Popular feature detectors, such as HOG, SIFT and SURF, see Section 2.2, are widely used in capturing large motions [40, 46, 142, 216, 233]. Feature based methods can track salient feature points across multiple frames, thus they can handle arbitrary large displacements as long as the features can be tracked. However, the established motion vectors are often very sparse because the local feature descriptors are reliable only at salient locations and are locally rigid [257].

In recent decades, many researchers have extensively worked on extending the sparse matching of features to dense matching of features [15, 18, 46, 76, 142, 149, 207, 226, 250, 252, 257]. Some methods compute dense point motion by matching descriptors extracted in from local regions (generally square), which is computationally expensive [15, 18, 46, 149, 257]. Some other approaches combine feature matching with optical flow techniques to extract dense long-term trajectories from a video sequence [207, 226, 252]. Nowadays, feature matching techniques are also used to improve the optical flow estimation in some approaches [142, 250]. Some researchers propose to use a neural network to learn to estimate the optical flow between two images [76].

MOTION SEGMENTATION

Strategies and techniques for motion segmentation differ from that developed for image segmentation because of the distinct property of motion vectors compared to other image features (intensity, color, texture, etc.). Generally, the goal of motion segmentation is to partition the tracked points (or pixels) from multiple moving objects into different groups based on their motions [41], which can be considered as a clustering problem. Thus principles and achievements in other research areas, such statistics, machine learning, data analysis, etc., contribute to the development of motion segmentation approaches. A variety of schemes and strategies have been proposed in the literature for solving the motion segmentation problem [36, 121, 217, 247, 259, 260, 270].

According to the type of motion representation, motion segmentation algorithms can be divided into *pixel based* and *feature-point based* [271, 272]. Pixel-based motion estimation directly recovers image motion at each pixel, so they are also called direct methods [123, 228]. Thus segmentation algorithms based on pixel-wise motion vectors are able to segment out precise regions of objects. Pixel-based motion estimation is sensitive to environment variations, such as illumination variation, occlusion [123, 233], etc. Consequently, pixel based segmentation is also sensitive to environment variation. Feature based motion estimation tries to detect and track a set of salient feature points, which results in a sparse matrix describing the trajectories of these points [233]. An object is then represented by a group of feature points. Segmentation based on feature points is in general able to handle occlusion [270]. A variety of schemes and strategies has been developed for motion segmentation in both categories. In this section we will review some state-of-the-art techniques and algorithms.

Zappella et al. divided the existing approaches into the following groups: image difference, statistical, wavelets, optical flow based methods, layers based methods and manifold clustering [270]. Figure 2.5 shows some examples of motion segmentation techniques. This division is not meant to be tight, in fact some of the algorithms could be placed in more than one category [271]. Image difference is one of the simplest techniques which computes the segmentation by thresholding the pixel-wise intensity difference between two consecutive images [57, 64]. Statistical approaches regard the segmentation as a clustering (or classification) problem where each pixel (point) is assigned a class label. Techniques in statistics, such as Maximum A Posteriori (MAP), Particle Filter (PF) and Expectation Maximization (EM), are commonly used in pixel-wise motion segmentation [192, 193, 259]. These approaches can deal with multiple objects and occlusions. Some methods exploit the usage of wavelets for reducing noise and analyzing frequency compo-

nents of images. These wavelets-based methods are often used as a complementary part for improvement of other approaches, such as image difference and optical flow [61].

Optical flow estimation is a traditional method to compute pixel-wise motion. Approaches based on optical flow aim to segment moving objects by analysing discontinuities in the optical flow field [36, 36, 67, 182, 254]. Optical flow based motion segmentation can deal with 2D motions and segment the objects from the background. However, these methods are sensitive to noise, computationally expensive, and perform poorly in long-term analysis, which make them not able to deal with real time applications.

Layer based techniques aim to divide images into layers with uniform motion, while each layer is associated with a depth level and a transparency level that determines the behaviour of the layers in case of overlapping [48, 140, 225, 253, 270]. Approaches in this category are quite robust to occlusion.

Manifold clustering, or subspace clustering in some literature, is a new trend in motion segmentation, which is mainly used on feature based motion trajectories, i.e. feature points tracked over multiple frames. These algorithms assume that the rigid-body motion of each object can be projected to a linear subspace [236]. Agglomerative Lossy Compression (ALC) proposed by Ma et al. [156] segments the trajectories by minimizing a cost function. The Local Subspace Affinity (LSA) method proposed by Yan and Pollefeys [266] estimates the subspaces by building an affinity matrix, which is able to deal with articulated, non-grid and non-degenerate motions. Vidal and Hartley [247] use generalized principal component analysis (GPCA) to cluster projected trajectories based on the fact that trajectories of rigid and independent motions generate subspaces of at most dimension four. Elhamifar and Vidal developed a sparse representation to cluster trajectories from multiple linear or affine subspaces [79]. Manifold clustering techniques generally perform well in segmenting rigid motions. Some of them can be adapted for estimating 3D motion models. A common limitation of these methods is the need for a sufficiently large set of complete trajectories, which is a challenge. This drawback can be compensated, to some extent, by introducing matrix completion algorithms to recover missing data [218, 268]. However the quality of trajectories is still the major bottleneck of these algorithms. Moreover, most of the subspace clustering algorithms require knowledge about the number of subspaces.

2.3.3. SEMANTIC IMAGE SEGMENTATION

In recent decades, semantic image segmentation approaches have developed rapidly in computer vision research. Traditional image segmentation approaches, based on low-level cues, partition an image into “coherent” regions, which may not be semantically meaningful [227]. Semantic image segmentation, which attempts to segment an image into different objects and parts with semantic meaning, is a more challenging problem. The semantic segmentation problem is ill-posed because the definition of “objects” or “meaningful parts” is ambiguous. Moreover, how to effectively represent the “objects” is also a challenge [276]. Therefore, as of today semantic segmentation largely remains an open issue and solutions vary with respect to the problem definition. In early research, some approaches addressed segmentation and recognition simultaneously, by searching the image locations based on a learned model [24, 84, 143, 172, 183]. Over the past two decades, semantic segmentation based on a self-training framework has boosted [60, 98, 150, 184].

There are several surveys reviewing the semantic segmentation approaches in the literature from different points of view [103, 276].

2.4. CLASSIFICATION

Object recognition refers to determining the categories of detected object instances, which is a classification problem. Classification is a supervised task, which is achieved by comparing the candidates with models/representations of known objects. Classification methods are widely used in object recognition systems [227]. Classification techniques can also be used in some detection tasks. For example, to detect a specific object in an image, we can divide the image into small patches and apply classifiers on each of the patches to find the target object [98, 141]. Moreover, classifiers are also helpful for tasks such as feature extraction and segmentation [126, 138, 255].

Objects in computer vision are regarded as higher-level features compared to the visual features and keypoints discussed in Section 2.2. The object representation defines the composition and structure of an object based on low-level features. A good representation of an object is crucial for an efficient and robust object recognition system. A brief review of representation models is given in subsection 2.4.1.

A variety of classifiers has been developed in the literature with the help of theories and techniques from statistics, geometry and topology, pattern recognition, machine learning, etc. In subsection 2.4.2, we will introduce some classical classifiers. For a particular classifier, various versions have been proposed to meet different requirements in applications. Surveys and introductions of these approaches are available in a large volume of literature [11, 89, 227, 246].

2.4.1. OBJECT REPRESENTATION

An object present in an image, covers part of the image, thus we can represent the object by describing the features of this patch of the image. To represent the objects with features, we need to analyze the relationships of features based on their appearance, shapes, spatial information, etc. The basic components of object models can be either pixels or detected feature points. Researchers have proposed several important criteria for object representation: scope, uniqueness, stability, sensitivity, and accessibility [116].

Methods of object representation fall into two categories according to the information used for modeling: *geometry* or *appearance* [234]. Geometric representations describe the 3D geometric relationships between the parts of an object. There are point models, edge models, surface models and volumetric models [87]. Geometric methods used for object representation and recognition have been well developed since it has been an active topic in computer vision and graphics. Most geometric methods are object centered, i.e. the information about the position of shape elements is affixed to a single-object coordinate system [174, 234].

Appearance-based approaches only use the appearances of objects, which are captured by different two-dimensional views of the objects of interest [202]. Such approaches aim to describe an object with its appearance features and their relationships using feature descriptors introduced in Section 2.2. Based on the applied features, appearance-based representations can be divided into *local* and *global* approaches. Dimensionality

reduction techniques, such as principal component analysis (PCA), independent component analysis (ICA) and non-negative matrix factorization, etc., are often applied on global features for object representation [202]. A common local approach is to construct the features and their relationships in the form of graph [19]. A codebook based approach, which is based on the Bag of Words (BoW) technique [128], uses clustering techniques to obtain high-dimensional feature vectors for a classifier. It became popular due to its simplicity and efficiency, as well as robustness to clutter, occlusion, viewpoint change, and even non-rigid deformations [69]. Deformable Part Models (DPM) represent an object by a collection of parts arranged in a deformable configuration. Approaches combining a DPM and a sliding-window technique can detect complex objects, and research shows they achieve state-of-the-art results on difficult benchmarks such as the PASCAL datasets [83, 183]. In the last two decades, deep learning also has shown outstanding performance on image classification tasks. Deep Neural Networks (DNN) are able to learn more complex models efficiently without the need of hand-designed features [138].

2.4.2. CLASSIFICATION

Classification is the process of classifying observations into a given set of categories. It has been a key concept in computer vision because many vision problems can be viewed as a classification task. Over the years many classification methods have been developed. An overview can be found in the book [77]. In this section, we focus on the approaches and techniques which have been widely used in object recognition applications. Choosing an appropriate classifier is a complex problem, where the feature descriptor, the object representation and the size of the training set must all be taken into consideration.

Naive Bayes classifiers The probabilistic framework proved to be a convenient tool in classifier design. Naive Bayes (NB) classifiers are based on conditional probabilities and Bayes' Theorem [198]. NB is known as a fast and space efficient classifier, and not sensitive to irrelevant features [205]. However, NB requires the independence of features.

Support vector machines Support vector machines (SVMs) are the most popular method in traditional object detection systems [205]. SVMs are examples of so-called kernel-based classification methods [37]. Combined with sliding window and local feature techniques such as HOG, SIFT and SURF, it performs well in human face detection and pedestrian detection [227].

Decision trees Decision trees (DT) are non-parametric approaches based on hierarchical rules [77]. DTs determine the class of an object by creating a tree according to the attributes of object descriptors. DTs are fast to compute and can address non-linearity [205]. They are sensitive to noise.

Neural Networks Neural Networks is a state-of-the-art technique, and have been used in object recognition systems for decades [227]. Their attraction lies in their ability to partition the feature space using nonlinear boundaries for classes. They use a non-parametric approach. Neural networks are utilized not only for classification but also for automatic

feature learning (from the raw data of the image) [209]. One of the popular approaches is to use Convolutional Neural Networks (CNN) [93, 127, 194]. CNNs have achieved remarkably performance for hand-written digit recognition [27]. There is a variety of classifiers based on CNNs, such as Deep CNN and Recurrent Neural Networks [138, 261]. The performance of NNs depends upon the network structure and the amount of training examples. Generally, a large number of examples are needed to achieve good performance, which make the training stage a time consuming task [107]. One has to balance the trade-off between computation and accuracy in the applications of NNs.

2.5. CONCLUSION

Object detection and recognition bridges the low-level image processing and high-level vision tasks. It is applied in a wide variety of applications. Each of these applications has specific requirements, such as: the detection of particular object instances or general object detection from multiple categories, processing on-line or off-line, robustness to occlusions, invariance to rotations, detection under different view points, etc. For example, a face detection and recognition system needs to identify the objects from the class of "face". A navigation system needs to identify the present object instances, which are from a range of classes. Nevertheless, these applications have a common purpose, i.e., to extract object information from input images. Therefore, there are common components in current object detection and recognition systems.

In the previous sections we introduced the key components and related techniques of a common object detection and recognition system. These techniques were developed to address the basic problems existing in any object detection and recognition system. In this thesis, we develop a system for the purpose of moving object detection and recognition from a video sequence. It is designed for detecting moving objects from multiple categories. We focus on motion estimation and motion segmentation for moving object detection. We also investigate how the results of moving object detection can be used to identify objects in static images. In the following chapter, we will discuss the set up of our approach, solutions to the common key problems, and the applied techniques.

3

PRELIMINARIES

In this chapter, we focus on the fundamental concepts, algorithms and necessary equations needed for our research. Since our research focuses on moving object detection in a video sequence, a variety of techniques and approaches are of interest. Sections 3.1 and 3.2 present two approaches related to motion detection and estimation. Section 3.3 introduces the basic geometry of camera projection and the transformation from 3D rigid-body motion to 2D image motion. Section 3.4 provides a brief introduction to the singular value decomposition. In Section 3.5, two fundamental statistics techniques are presented: the expectation maximization (EM) algorithm and Bayes theorem. The EM algorithm is widely used for data clustering and Bayes theorem is a powerful tool for sequential updating. An efficient classifier called support vector machine is explained in Section 3.6.

3.1. OPTICAL FLOW

Optical flow is a successful approach to motion estimation, and has been one of the most active research domains in computer vision. Optical flow is defined as the distribution of apparent velocities of brightness patterns in an image sequence [118]. In this section, we will introduce the basic concepts and original formulation of optical flow estimation.

The estimation of optical flow is based on the assumption that the intensity of a pixel, corresponding with a point on an object surface, does not change significantly when the object or the camera is moving. This is called *brightness constancy* [96].

Let $I(x, y, t)$ be the intensity of a pixel at position (x, y) and at time t. Suppose this pixel is translated to position $(x + \Delta x, y + \Delta y)$ at time $t + \Delta t$. Based on the assumption of brightness constancy we then have that

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) \tag{3.1}$$

Expanding the left-hand side of Equation (3.1) with a first-order Taylor series, we find

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + I_x \Delta x + I_y \Delta y + I_t \Delta t + \mathcal{O}^2 \tag{3.2}$$

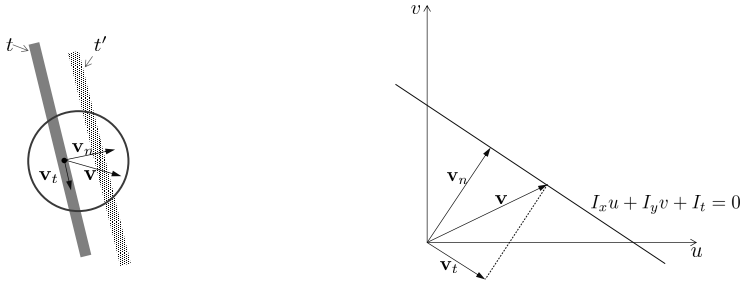


Figure 3.1: (a) The aperture problem. $\mathbf{v} = (u, v)$ is the true displacement (velocity) from t to t'. Only its normal component $\mathbf{v}_n$ can be estimated, but the tangential component $\mathbf{v}_t$ cannot. (b) Equation (3.3) yields a line in the (u, v) space of equal intensities; the true velocity vector $\mathbf{v}$ is on this line.

where I_x and I_y denote the spatial and I_t the temporal first-order partial derivatives of I. The notation $\mathcal{O}^2$ represents all the terms in Δx, Δy and Δt of order 2 and higher; they are assumed negligible for small displacements.

Combining Equation (3.1) and Equation (3.2), and defining the velocity vector by $(u, v) = (\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t})$, we obtain the equation of optical flow (u, v) per pixel:

$$I_x u + I_y v + I_t = 0 \tag{3.3}$$

This is an equation in two unknowns and can not be solved for u and v directly; this is known as the aperture problem, as visualized in Figure 3.1. As a result, we can only uniquely recover the component in the direction of the spatial image gradient, while the motion that is orthogonal to the spatial image gradient direction cannot be estimated accurately in a local region.

In practice, we will be dealing with discretized information: images that consist of pixels and which are recorded (sampled) at discrete moments in time. Without loss of generality, we can choose the time unit to equal the sampling time (the reciprocal of the frame rate), whence $\Delta t = 1$. We will use this convention in the rest of the thesis. The brightness constancy relation Equation (3.1), then takes the form

$$I(x + u, y + v, t + 1) = I(x, y, t) \tag{3.4}$$

The pixel coordinates x and y, but also the velocity components u and v, may or may not attain integer values, depending on the model that is used.

3.1.1. THE METHOD OF HORN AND SCHUNCK

To resolve the redundancy in Equation (3.3), additional constraints should be introduced. Horn and Schunck [118] proposed the *spatial smoothness* constraint, which is used as the fundamental basis of most optical flow estimation techniques. The spatial smoothness constraint comes from the observation that neighboring points on objects have similar velocities and the velocity field of the brightness patterns in the image varies smoothly almost everywhere [118]. In this approach, the estimated optical flow field is the solution which also shows the most smoothness.

Horn and Schunck use the square of the magnitude of the gradient of the velocity (denoted by the first-order partial derivatives u_x, u_y, v_x, v_y of u and v) to measure the spatial smoothness at each pixel (x, y). Then the optical flow estimation becomes the problem of minimizing the sum of the squared errors in the equation for the rate of change of image brightness. As squared velocity allows for an energy interpretation, an energy function with a penalty term, related to the brightness constancy assumption and the smoothness constraint respectively, is defined over the image domain as follows:

$$\begin{aligned} e &= \iint (e_b + \lambda e_c)\, dx\, dy \\ &= \iint ((I_x u + I_y v + I_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_y^2))\, dx\, dy \end{aligned} \tag{3.5}$$

Here $e_b = (I_x u + I_y v + I_t)^2$ denotes the energy mismatch at a location (x, y) regarding the brightness constancy assumption, $e_c = u_x^2 + u_y^2 + v_x^2 + v_y^2$ measures the change in the velocity (non-smoothness of the optical flow) at (x, y), and $\lambda > 0$ is a balancing parameter; integration takes place over the spatial image domain. This set-up uses a continuous model for the spatial dimension, which needs to be discretized depending on the pixel density (resolution) of actual images. The optical flow is then solved by minimizing the expression in Equation (3.5). Horn and Schunck's solution is global, because the computation of the flow vector at a specific location is based on the entire image. It results in dense flow fields, but is also known to be sensitive to noise.

3.1.2. THE METHOD OF LUCAS AND KANADE

Lucas and Kanade [153] proposed a local method based on the *spatial coherence* assumption, which assumes that neighboring pixels move coherently and (largely) share the same flow. Thus Equation (3.3) must hold in good approximation within the neighborhood of each pixel. For example, if we use the pixels within a 5×5 window around a given pixel (x, y) as its neighborhood $\Omega_{(x,y)}$, there are 25 equations to build for each pixel. We then can build an error function for the pixels within the neighborhood $\Omega_{(x,y)}$ of the chosen pixel:

$$E_{(x,y)}(u, v) = \sum_{(x',y') \in \Omega_{(x,y)}} (I_x(x', y')\, u + I_y(x', y')\, v + I_t(x', y'))^2 \tag{3.6}$$

Then the optical flow $(\hat{u}, \hat{v})$ of the chosen pixel at (x, y) can be solved by minimizing the error function:

$$(\hat{u}, \hat{v}) = \operatorname*{argmin}_{u,v \in \mathbb{R}} E_{(x,y)}(u, v) \tag{3.7}$$

The Lucas-Kanade (LK) method provides a simple but efficient solution to estimate the optical flow locally. However, it can only be used reliably when the pixel motion between two images is small enough for the spatial coherence assumption to hold. In practice, true flow vectors in a neighborhood could be inconsistent for some pixels, which then causes errors in estimation. Lucas and Kanade assign a weight to each neighbor to diminish the importance of distant neighbors, thus aiming to decrease the estimation error. Usually, such weights are set by a Gaussian function of the distance between the neighbor pixel and the center pixel. Then the weighted error function at (x, y) becomes:

$$E^{w}_{(x,y)}(u,v) = \sum_{(x',y')\in\Omega_{(x,y)}} w(x'-x,y'-y)(I_x(x',y')u + I_y(x',y')v + I_t(x',y'))^2 \tag{3.8}$$

where $w(x,y)$ is the Gaussian weight function.

There are many improved versions of LK optical flow estimation, to obtain more accurate flow estimates of large displacements. We will briefly discuss the affine LK model, the iterative optimization method, and the pyramidal computation of optical flow.

AFFINE LUCAS-KANADE OPTICAL FLOW ESTIMATION

The basic LK method of optical flow estimation assumes that a pixel and its neighbors in a local window all undergo the same translation. However, when for instance rotation occurs, a single translation vector (u,v) is inadequate to represent the motion patterns of all the pixels in the local window. Generally, an affine motion model with a rotation matrix and a translation vector is more accurate to represent the all the pixel motions [29, 88]. Equation (3.9) specifies such an affine motion model, which is used to jointly describe the movements of all pixels within a local window Ω, for one time step:

$$\begin{pmatrix} x+u \\ y+v \end{pmatrix} = \begin{pmatrix} 1+a_1 & a_2 \\ a_4 & 1+a_5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_3 \\ a_6 \end{pmatrix} \quad \text{for } (x,y) \in \Omega \tag{3.9}$$

Here (x,y) denotes a pixel position (within the window Ω) in one image, and $(x+u, y+v)$ denotes its new position in the next image. The two coordinates u and v constitute the flow vector of this pixel. As illustrated in Figure 3.2, u and v now vary over the neighborhood Ω. It directly follows from Equation (3.9) that u and v are affine functions of x and y given by

$$\begin{aligned} u(x,y) &= a_1 x + a_2 y + a_3 \\ v(x,y) &= a_4 x + a_5 y + a_6 \end{aligned} \tag{3.10}$$

By substituting Equation (3.10) into Equation (3.8), we obtain the modified error func-

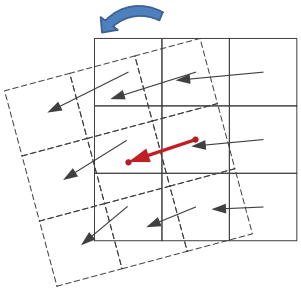


Figure 3.2: A window of 3 × 3 pixels undergoing a rotation. The optical flow vector clearly varies with the pixel locations.

tion below:

$$E_{(x,y)}(\mathbf{a}) = \sum_{(x',y')\in\Omega_{(x,y)}} (w(x'-x,y'-y)(I_x(x',y')(a_1x'+a_2y'+a_3) + I_y(x',y')(a_4x'+a_5y'+a_6)+I_t(x',y'))^2) \tag{3.11}$$

This function depends on 6 unknowns (the entries of $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5, a_6)$), which correspond to the 6 parameters of the affine motion model. The goal is to estimate the optical flow (u, v) at the center (x, y) of Ω according to Equation (3.10), by first finding the best estimate of $\mathbf{a}$ that minimizes this error. By minimizing this weighted least-squares criterion, the optimized estimates of the parameters are obtained; this can e.g. be achieved by setting the first-order partial derivatives of Equation (3.11) with respect to each parameter equal to zero and solving the corresponding system of equations. Then, a minimum of six pixels is at least required for obtaining a unique solution. Once the affine model parameters are determined for a pixel, the corresponding flow vector at (x, y) is computed from Equation (3.10).

ITERATIVE OPTIMIZATION

Recall that the basic LK approach is based on a first-order Taylor approximation, which neglects the higher-order terms denoted by $\mathcal{O}^2$ in Equation (3.2); which is only valid when movements are small enough. To obtain a more accurate estimate of the optical flow in case of larger displacements, we can use a coarse-to-fine strategy based on an iterative procedure similar to Newton's method [39, 43, 153].

Suppose we already have an estimate $(\hat{u}, \hat{v})$ of the true optical flow (u, v) for a pixel at location (x, y), with residual (δ_u, δ_v):

$$\begin{aligned} u &= \hat{u} + \delta_u \\ v &= \hat{v} + \delta_v \end{aligned} \tag{3.12}$$

Then the left-hand side of Equation (3.4) can be written as:

$$I(x+u, y+v, t+1) = I(x+\hat{u}+\delta_u, y+\hat{v}+\delta_v, t+1) \tag{3.13}$$

This suggests that we can generate a warped image J, by applying the estimated motion $(\hat{u}, \hat{v})$ to each of the pixels of $I(x, y, t)$, which gives

$$J(x+\hat{u}, y+\hat{v}, t) = I(x, y, t) \tag{3.14}$$

The brightness constancy equation between the warped image $J(x+\hat{u}, y+\hat{v}, t)$ and $I(x+u, y+v, t+1)$ then is as follows:

$$I(x+\hat{u}+\delta_u, y+\hat{v}+\delta_v, t+1) = J(x+\hat{u}, y+\hat{v}, t) \tag{3.15}$$

which can be viewed a function of the residual vector (δ_u, δ_v). Then this residual vector can again be solved by the Lucas-Kanade method. The estimates of residual flows $(\hat{\delta}_u, \hat{\delta}_v)$ are subsequently used to further refine the optical flow estimates:

$$\begin{aligned} \hat{u}_{new} &= \hat{u} + \hat{\delta}_u \\ \hat{v}_{new} &= \hat{v} + \hat{\delta}_v \end{aligned} \tag{3.16}$$

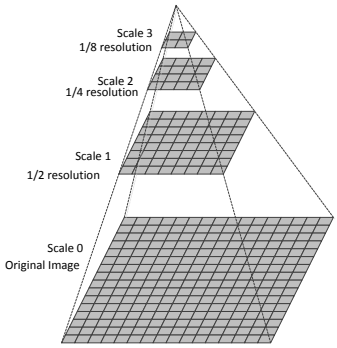


Figure 3.3: The pyramidal representation of images. Images at a larger scale are more smoothed and have lower resolution.

This new optical flow estimate can be used to re-warp the original image $I(x, y, t)$ again – and then to estimate a new residual flow. This procedure can be applied in an iterative manner until $\sqrt{\hat{\delta}_u^2 + \hat{\delta}_v^2}$ is smaller than a threshold.

PYRAMIDAL COMPUTATION OF THE OPTICAL FLOW

The Lucas-Kanade optical flow computation is based on local uniformity, thus the estimation accuracy is related to the size of the local window. Normally, a smaller window will lead to higher accuracy, because the chances then are smaller that it contains pixels that move in different patterns. On the other hand, for a larger window size the estimated optical flow is expected to be less sensitive to variations, such as light changes and large displacements that exceed the window size. In this sense, a larger local window is preferable to obtain a more robust estimate of optical flow. Therefore, there is a natural trade-off between accuracy and robustness when choosing the window size [39]. To address this problem, one can construct an image pyramid, which consists of multi-scale representations of a given image. The images in the pyramid have increased smoothing but decreased resolution when going from a smaller to a larger scale. When we go up in the pyramid, small motions are removed and large motions take the role of small motions. We first estimate the LK optical flow on the largest scale (at the top of the pyramid), then propagate and refine the resulting motion flows by applying the LK algorithm again on the next scale down. This procedure continues by going down the pyramid until we arrive at the smallest scale (the original image scale) [42]. In this way, the pyramidal LK method is able to properly track larger motions than the basic LK algorithm.

Given a pair of images, we first establish a set of pyramidal representations for each image, in the same way as illustrated in Figure 3.3. An image pyramid consists of multiple images of different resolution—all arising from the same single original image. The layers (scales) in a pyramid of $L+1$ scales are numbered from 0 to L, from bottom to top as indicated, where the image at scale 0 is the raw image. The image in each layer is obtained by smoothing and sub-sampling the image at the previous (lower) scale, using a scaling factor; thus the resolution of the images decreases as long as the scale increases. A variety

of different smoothing kernels can be used for generating pyramids [4, 50].

The flow vectors are first computed by the iterative affine Lucas-Kanade algorithm, starting at the top scale L. The results are then propagated to the lower scale $L-1$ as the initial guess of the flow vectors, which are then further refined with the iterative affine Lucas-Kanade algorithm at scale $L-1$. This procedure runs iteratively down to scale 0.

The pyramidal scheme of optical flow estimation can achieve high accuracy and robustness for relatively large pixel motions, i.e. when the displacement of a pixel is slightly larger than the local window size. However, it still suffers from the common limitation of optical flow methods — they can be erroneous because the brightness constancy or velocity smoothness assumptions can be violated. Thus the ability of dealing with large displacements is limited for the pyramid algorithm. Moreover, the depth of the image pyramid, the choice of L, affects the quality of estimated optical flow.

3.2. SCALE-INVARIANT FEATURE TRANSFORM

The scale-Invariant Feature Transform (SIFT) is an algorithm to detect and describe local features in images, which was proposed by David Lowe [151]. Lowe's method transforms the image into a large collection of local features, each of which is invariant to image scale (feature size) and rotation, and with partial invariance to affine distortion, noise and illumination changes. It is proved to be an efficient and robust way of detecting points of interest, which is useful in object detection and recognition [152].

The original SIFT algorithm involves two stages: feature detection and description. The feature detection refers to the process of locating salient features in image, which is achieved with the help of a scale space. The SIFT description represents the features in a manner invariant to scaling and rotating. This property makes it possible to match corresponding interest points between different images. The SIFT algorithm works well for tasks such as object categorization, texture classification and image alignment.

There are four main steps which constitute the SIFT algorithm according to Lowe [151]:

1) scale-space extrema detection,
2) keypoint localization,
3) orientation assignment, and
4) keypoint description.

Scale-space Extrema Detection

This first step attempts to identify the potential feature points over images of different scales. To obtain scale invariance, a scale space is constructed by applying Gaussian blur filters to the original image at different octaves and scales. An image pyramid is built and each level in the pyramid is called an octave. The image resolution decreases when the octave number increases in the pyramid. At each octave, a group of images is composed by blurring the raw image on different levels, where each such level is called a scale. Octaves and scales are indexed starting from 0, as shown in Figure 3.4. The number of octaves and scales are defined by the user in the implementation. Lowe [152] suggests that 4 octaves and 5 blur levels are ideal for the algorithm.

Suppose original image is $I(x,y)$ for a scale space of N_o octaves and N_s scales, then we define the image at scale s ($s=0,1,\ldots,N_s$) and octave o ($o=0,1,\ldots,N_o$) as $L(x,y,\sigma_s^o)$,

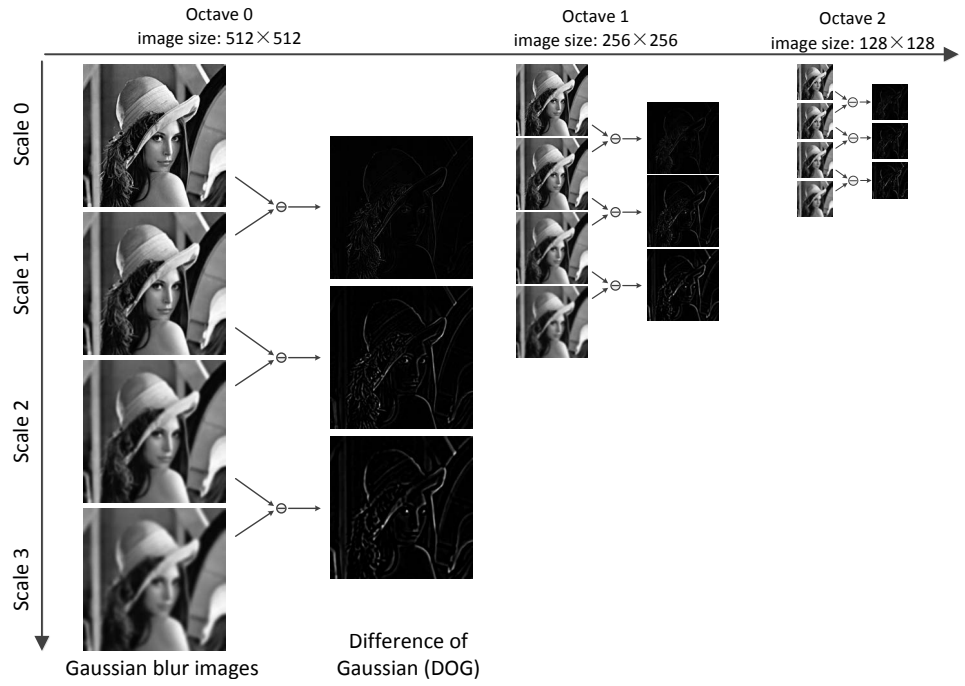


Figure 3.4: An example of scale space which consists of 3 octaves and 4 scales per octave. The image resolution halves from an octave to the next. Each octave has 4 images blurred by a Gaussian function with increasing scales.

where σ_s^o is the blur level. $L(x, y, \sigma_s^o)$ is generated by computing the convolution of the original image $I(x, y)$ with a Gaussian kernel $G(x, y, \sigma_s^o)$ as Equation (3.17),

$$L(x, y, \sigma_s^o) = G(x, y, \sigma_s^o) * I(x, y) \tag{3.17}$$

The Difference-of-Gaussian (DoG) images are generated by computing the difference of adjacent Gaussian-blurred images per octave, as Equation (3.18). Figure 3.4 illustrates the DoG images in the example scale space.

$$D(x, y, \sigma_s^o) = L(x, y, \sigma_{s+1}^o) - L(x, y, \sigma_s^o) \tag{3.18}$$

The key locations are obtained by detecting the local maxima and minima of the DoG images over scales and octaves. One pixel in a DoG image is compared with its 8 neighbors as well as 9 pixels in next scale and 9 pixels in previous scale, as shown in Figure 3.5. If this value is the minimum or maximum of all these points then this point is an extrema.

Keypoint Localization

The keypoint candidates detected in previous step contain points of low contrast and points on the edge. Both kinds of points are sensitive to noises thus they are regarded as unstable. Such points are eliminated in this step. The low contrast points can be found

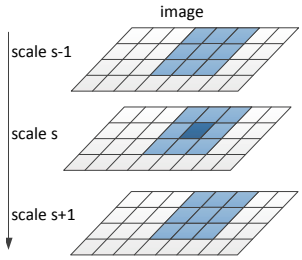


Figure 3.5: The search area of one pixel (the dark blue lattice) contains 26 adjoining pixels (the light blue lattices) in DoG space.

by simply checking their intensities. If the absolute value of the DoG image at a detected point is less than a threshold value, it is rejected as a low contrast point. Since the image intensity gradient will be big across the edge and small along the edge, a 2×2 Hessian matrix is used to reject the edge points based on the Harris corner detector [111]. This Hessian matrix is computed at the location and scale of a point. If the ratio between the largest and the smallest eigenvalues of the Hessian matrix is larger than a threshold, the corresponding point is rejected as an edge point.

Orientation Assignment

The keypoints are located at different octaves and scales. Next, the keypoints are assigned a consistent orientation to achieve invariance to image rotation. Given a keypoint located at (x, y) in the image $L(x, y, \sigma_s^o)$ at scale s and octave o, the calculation of orientation is based on the image intensity gradient magnitude in a neighborhood around its location. Without loss of generality, we simplify the notation of $L(x, y, \sigma_s^o)$ to $L(x, y)$ because σ_s^o is fixed once the keypoint is chosen. Then the gradient vector at (x, y) is approximately proportional to $(L(x+1, y) - L(x-1, y), L(x, y+1) - L(x, y-1))$. Its magnitude $m(x, y)$ and its orientation (angle) $\theta(x, y)$ are then computed as follows:

$$\begin{aligned} m(x, y) &= \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \\ \theta(x, y) &= \text{atan2}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))) \end{aligned} \tag{3.19}$$

An orientation histogram with 36 bins covering 360 degrees is created and the highest peak of the histogram is selected as the orientation of the keypoint. Note that if there is another peak that ranks above 80% of the highest peak in the orientation histogram, a new keypoint at the same location but of different orientation is created as well. This contributes to stability of matching.

Keypoint Description

The last step is to build a descriptor for each keypoint. The SIFT descriptor is generated from a statistical analysis of the gradient magnitude in the neighborhood of keypoints. By dividing the neighborhood into sub-blocks, a set of histograms is computed to form the SIFT descriptor. Typically the SIFT method takes a 16×16 neighborhood around each keypoint and divides it into 16 sub-blocks of size 4×4. For each sub-block a histogram

of 8 bins is generated. This histograms of all sub-blocks are used to form a vector of 128 elements, which is the keypoint descriptor.

3.3. CAMERA GEOMETRY

Videos are obtained by cameras, which map the 3D world to 2D images. In this thesis, we are interested in the geometry of camera projection and the relationship between 3D rigid body motion in the real world and 2D motion in images and videos.

3.3.1. 3D RIGID BODY MOTION

In kinematics, a rigid body is an idealization of a body that does not deform under the action of applied forces. Formally it is defined as a collection of particles with the property that the distances between the particles remain unchanged during the motions of the body [162]. In the real world a perfectly rigid body does not exist, but for many objects, the deformation is negligible compared to the overall motion of the body. Based on this rigid body approximation, all particles that define the object undergo the same motion model, which is called a rigid transformation [38]. Such a rigid transformation is an affine transformation in 3D space, which can be decomposed into a 3D rotation followed by a translation. Suppose a particle moves from (X, Y, Z) to (X', Y', Z'), then its motion can be modelled as

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} \tag{3.20}$$

where R is a 3D rotation matrix and $\mathbf{t}$ is a translation vector. A rotation matrix R in 3D space can be parameterized as in Equation (3.21). The translation vector $\mathbf{t} = (t_1, t_2, t_3)^\top$ indicates displacements in the directions of the three standard coordinate axes.

$$\begin{aligned} R &= R_z(\varphi_z)R_y(\varphi_y)R_x(\varphi_x) \\ &= \begin{bmatrix} \cos\varphi_z & -\sin\varphi_z & 0 \\ \sin\varphi_z & \cos\varphi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi_y & 0 & \sin\varphi_y \\ 0 & 1 & 0 \\ -\sin\varphi_y & 0 & \cos\varphi_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_x & -\sin\varphi_x \\ 0 & \sin\varphi_x & \cos\varphi_x \end{bmatrix} \end{aligned} \tag{3.21}$$

where $\varphi_z \in (-\pi, \pi]$, $\varphi_y \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, and $\varphi_x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ are yaw, pitch, and roll angles respectively. By matrix multiplication, we have

$$R = \begin{bmatrix} \cos\varphi_x\cos\varphi_y & \cos\varphi_x\sin\varphi_y\sin\varphi_z - \sin\varphi_x\cos\varphi_z & \cos\varphi_x\sin\varphi_y\cos\varphi_z + \sin\varphi_x\sin\varphi_z \\ \sin\varphi_x\cos\varphi_y & \sin\varphi_x\sin\varphi_y\sin\varphi_z + \cos\varphi_x\cos\varphi_z & \sin\varphi_x\sin\varphi_y\cos\varphi_z - \cos\varphi_x\sin\varphi_z \\ -\sin\varphi_y & -\cos\varphi_y\sin\varphi_z & -\cos\varphi_y\cos\varphi_z \end{bmatrix} \tag{3.22}$$

For simplicity, we use the notation r_{ij} to indicate the element of R in the i^{th} row and j^{th} column in Equation (3.22), thus R is represented in the rest of the chapter by:

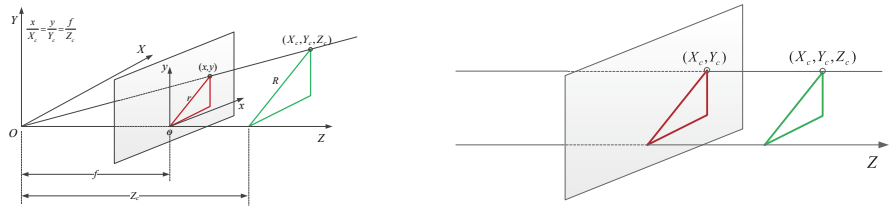
$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.23}$$

3.3.2. PROJECTIONS

A physical camera projects the 3D world onto a 2D image according to some projection model. In general, the lens system in a real camera is too complex to be modelled in detail in practice. Instead, simplified camera models have been developed for different applications depending on given assumptions.

GENERAL PERSPECTIVE PROJECTION

Perspective projection provides an idealized mathematical model of a real camera, which is widely used in computer vision applications. It is based on the assumption that the camera is small enough compared to the viewed scenes in most situations.



(a) The general perspective projection of a pinhole camera model

(b) The parallel projection of an orthographic camera

Figure 3.6: Camera models: (a) perspective projection; (b) orthographic projection.

A general perspective projection maps a point in 3D space to a point on the 2D image plane. Figure 3.6a shows the simplest central-projection camera model: the pinhole camera model. The XYZ coordinate frame is centered at the camera, with the Z-axis being the principal axis of the camera. The projected image plane locates at the focus plane, and is characterized by the xy coordinate frame. The origin of image frame o is the projection of the camera center O on the image plane. A point (X, Y, Z) in the camera frame is mapped to the point (x, y) in the image frame by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.24}$$

ORTHOGRAPHIC CAMERA MODEL

Assuming the camera has an infinite focal length f, the points in the camera frame are mapped to the image by parallel lines, which is illustrated in Figure 3.6b.

In the orthographic camera model, a point (X, Y, Z) is mapped to an image point (x, y), eliminating the Z-coordinate. A general orthographic projection from the camera frame to the image frame has the form:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.25}$$

3.3.3. 2D AFFINE TRANSFORMATION

The natural scenes of 3D objects are projected to a sequence of 2D images by the camera. A point (particle) on the surface of an object is mapped to a pixel in the image through the camera model. Assume a point moves from (X, Y, Z) to (X', Y', Z') in camera space, which are mapped to the 2D points (x, y) and (x', y') in two different images, using a perspective camera as in Equation (3.24), then we have the following relationships:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{3.26}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{f}{Z'} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \frac{f}{Z'} \begin{bmatrix} X' \\ Y' \end{bmatrix} \tag{3.27}$$

Substituting these two equations and Equation (3.23) into Equation (3.20), we have

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{Z}{Z'} \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \frac{fZ}{Z'} \begin{bmatrix} r_{13} \\ r_{23} \end{bmatrix} + \frac{f}{Z'} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{3.28}$$

In case of an orthographic camera, the factors $\frac{f}{Z}$ and $\frac{f}{Z'}$ equal 1, and Equation (3.28) becomes

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + Z \begin{bmatrix} r_{13} \\ r_{23} \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{3.29}$$

Equation (3.29) reveals that all points belong to an object have the same transformation matrix regarding their 2D images based on an orthographic camera. The translation vector varies for different points because it is affected by Z. Suppose the center point of an object is (X_0, Y_0, Z_0), the component Z can be written as $Z_0 + \delta_Z$. Thus for all points belonging to the object, the differences of the translation part in Equation (3.29) is determined by δ_Z, as Equation (3.30) shows.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} + Z_0 \begin{bmatrix} r_{13} \\ r_{23} \end{bmatrix} + \delta_Z \begin{bmatrix} r_{13} \\ r_{23} \end{bmatrix} \tag{3.30}$$

Normally, the fast frame rate of a video makes that the changes between two consecutive frames are relatively small, which enables the human eyes to perceive the continuity of its content. In this case, the changes caused by the component δ_Z are small, and Equation (3.29) approximates a 2D affine transformation, which holds for all points on an object. Therefore, we can assume that when a moving object in 3D space is projected to a video sequence through an orthographic camera, the movements of the image points belonging to the object preserve their continuity in the 2D video frames.

Based on the affine motion assumption, a 6-parameter affine transformation model is able to describe the motion of an object between two consecutive frames, which can capture the motions of translation, scaling, and rotation in the 2D plane. If a point is detected at position $\mathbf{x} = (x, y)^\top$ in one frame and at position $\mathbf{x}' = (x', y')^\top$ in the next frame, then Equation (3.31) is assumed to hold in a good approximation for all points belonging to the same object.

$$\mathbf{x}' = A\mathbf{x} + \mathbf{b}; \tag{3.31}$$

where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, and $\mathbf{b} = [\, b_1 \; b_2 \,]^\top$.

Given a group of points belonging to the same object together with their motion vectors, we can recover the affine model of the object by solving Equation (3.31) for all points jointly. At least 3 points with independent motion vectors are needed to compute the affine matrix because there are 2 equations and the same 6 unknowns for each point. Given an affine model of a moving object, we can determine whether a point belongs to the object by checking its motion with the affine model.

3.4. SINGULAR VALUE DECOMPOSITION

Singular value decomposition is a technique to factor a matrix [166]. It provides a useful way to analyze the rectangular matrices.

For any matrix $M \in R^{m \times n}$ of rank r, there exist orthogonal matrices $U_{m \times m}$ and $V_{n \times n}$ and a non-negative diagonal matrix $\Sigma_{m \times n}$ such that

$$M = U \Sigma V^\top \tag{3.32}$$

Such a factorization is called a singular value decomposition of the matrix M. If the nonzero singular values $(\sigma_1, \sigma_2, \ldots, \sigma_r)$ are arranged in non-increasing order along the diagonal of Σ. The columns of U and V are called the left and right singular vectors of M respectively. The SVD has some basic properties:

- The sequence of singular values is unique for any M.
- The columns of V are the eigenvectors of $M^\top M$.
- The columns of U are the eigenvectors of $MM^\top$.
- The rank of matrix M is equal to the number of nonzero singular values.

The SVD provides a useful way of analyzing matrix and is applicable in many fields. It is used for computing the pseudoinverse of a matrix, solving homogeneous linear equations, solving least squares problems, principle component analysis and dimensionality reduction, etc. [99, 133, 134, 251]. Applications of the SVD in computer vision exists in every field, such as face recognition, 3D reconstruction, data compression, and so on [44, 137, 173, 227, 244]. In the field of motion segmentation, SVD is useful for the factorization of motion matrix.

3.5. STATISTICAL METHODS

A common problem in computer vision is that one is given a collection of measurements (image, feature positions, motion fields, etc.), and one has to estimate the values of some unknown structure or parameters (camera motion, object shape, etc.). These problems are called *regression problems* because we are trying to estimate a continuous quantity from noisy inputs, as opposed to a discrete classification task [27]. In computer vision, they are called *inverse* problems, which is the opposite of the *forward* problem as in computer graphics, because they involve estimating unknown model parameters instead of simulating the forward formation equations [227].

To address the *inverse* problem, the statistical models and methods provide powerful tools. Approaches based on statistical inference have been extensively developed with

varying degrees of success over the last decades. In this section, two kinds of statistical methods are discussed.

3.5.1. EXPECTATION-MAXIMIZATION ALGORITHM

The expectation-maximization (EM) algorithm [71] is an effective and popular technique of solving maximum likelihood estimation (MLE) problems when the given data is incomplete. It has been widely used for statistical estimation problems with missing data, as well as for mixture estimation problem which can be posed in a similar form.

Assume a model with parameters $\boldsymbol{\theta}$ is associated with a given set of observed data $\mathbf{X} = x_1, x_2, \ldots$ and some unobserved latent variables (or missing data) $\mathbf{Z} = z_1, z_2, \ldots$. We may want to estimate two things: (1) the parameters and (2) the latent variables. The EM algorithm is intuited by the fact that each of these two steps is easy assuming that the other one is solved.

The basic idea of the EM algorithm is to find the parameter $\boldsymbol{\theta}$ that maximizes the likelihood $L(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Z}) = P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$ in an iterative procedure. By starting with some initial guesses, the EM algorithm alternatively executes two steps to update the estimation of parameters by increasing the expected value of $\log L(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Z})$ until it converges to a local maximum, as shown in Equation (3.33),

$$\begin{aligned} \boldsymbol{\theta}^{i+1} &= \operatorname*{argmax}_{\boldsymbol{\theta}} E_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^i}[\log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z})] \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{z} P(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^i) \log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z}) \end{aligned} \tag{3.33}$$

The component $E_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^i}[\log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z})]$ is defined as a function $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^i)$, which is constructed conditioned on the distribution of the hidden variables with respect to the observed data and the current estimation of the parameters. The construction of $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^i)$ is called the *expectation step* (E-step) and the computation of $\boldsymbol{\theta}^{i+1}$ is called the *maximization step* (M-step).

A variant of the EM algorithm, which simply estimates the latent variable $\mathbf{Z}$ in the E-step instead of computing the conditional distribution $P(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^i)$ over all possible values of $\mathbf{Z}$, is called classification EM [58, 104]. The two basic steps of classification EM in the i^{th} iteration are:

Classification step: find $\mathbf{Z}^i = \operatorname{argmax}_{\mathbf{Z}} P(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^i)$;
Maximization step: compute $\boldsymbol{\theta}^{i+1} = \operatorname{argmax}_{\boldsymbol{\theta}} P(\mathbf{Z}^i, \mathbf{X} \mid \boldsymbol{\theta})$;

The classification EM is usually faster to converge and easier to implement than the original EM, thus it is widely used in practice. One of the famous applications of classification EM is k-means clustering [211].

3.5.2. BAYESIAN UPDATING

Bayes' theorem is part of a fundamental probabilistic theory proposed by Bayes and Laplace in the 18th century [23]. It has been developed and applied by statisticians and philosophers in a variety of research areas, such as human cognition, neural computation,

machine learning, etc.[115]. Bayes' theorem provides a rule to update the belief of a hypothesis $\mathcal{H}$ given some observed data $\mathcal{D}$, which can be described by

$$P(\mathcal{H}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{H})P(\mathcal{H})}{P(\mathcal{D})} \tag{3.34}$$

where $P(\mathcal{D})$ is the marginal likelihood of $\mathcal{D}$, which acts as a normalizing constant. So we can understand Bayes' theorem as a form of proportionality between the posterior and the prior times the likelihood, as Equation (3.35).

$$P(\mathcal{H} \mid \mathcal{D}) \propto P(\mathcal{D} \mid \mathcal{H})P(\mathcal{H}) \tag{3.35}$$

The conditional probability $P(\mathcal{D} \mid \mathcal{H})$ is called the likelihood. $P(\mathcal{H})$ is the prior probability and describes the information we have about the hypothesis $\mathcal{H}$ before observing the data $\mathcal{D}$. After the data is observed, the prior distribution is updated to the posterior probability $P(\mathcal{H} \mid \mathcal{D})$ according to Equation (3.35).

Suppose that the observed data arrives sequentially, e.g. $(D_1, D_2, \ldots, D_N)$. Assume also that the observed data is conditionally independent of the hypothesis $\mathcal{H}$. Then the computation of the posterior given all observed data can be written as

$$\begin{aligned} P(\mathcal{H} \mid D_1, D_2, \ldots, D_N) &\propto P(D_1, D_2, \ldots, D_N \mid \mathcal{H})P(\mathcal{H}) \\ &= P(D_1, \ldots, D_{N-1} \mid \mathcal{H})P(D_N \mid \mathcal{H})P(\mathcal{H}) \\ &\propto P(D_1, \ldots, D_{N-1} \mid \mathcal{H})P(\mathcal{H} \mid D_N) \end{aligned} \tag{3.36}$$

In Equation (3.36), $P(\mathcal{H} \mid D_N)$ is the posterior having observed D_N, and it can be used as the new "prior" for the remaining data $(D_1, \ldots, D_{N-1})$. Thus we can update the prior and compute the new posterior iteratively when new data is observed. The process of improving the prior probability to the posterior probability is called Bayesian updating.

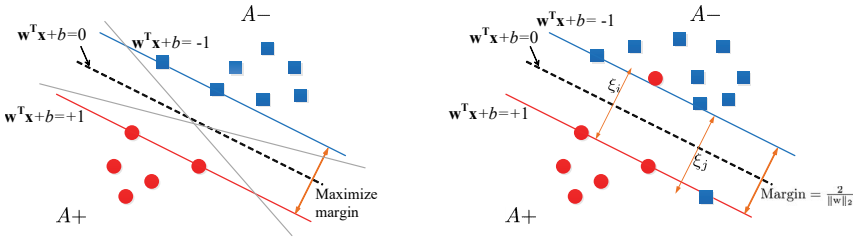
The essential characteristic of Bayesian methods is their explicit use of conditional probabilities for quantifying uncertainty in inferences based on statistical data analysis [53]. Bayes' theorem provides a powerful tool for solving problems in model estimation, decision making, missing data inference, prediction, learning of parameters, etc.[17, 53, 205].

3.6. SUPPORT VECTOR MACHINES

Support vector machines (SVMs) introduced by Boser, Guyon and Vapnik [37], are supervised learning models for classification and regression analysis. The main idea behind the SVM is the construction of an optimal hyperplane that separates the given patterns. SVM can be used both for linear and for non-linear classification, which will be briefly discussed in this section.

Linear SVMs

Given a set of N example points $\mathbf{x}_i$ from two classes $A+$ and $A-$, each associated with a class label y_i ($y_i \in \{1, -1\}$ where 1 and -1 indicate the class $A+$ and $A-$ respectively), the dataset can be written as $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$. Assume the data is linearly separable;



(a) Linearly separable data (b) Non-linearly separable data

Figure 3.7: Linearly separable and non-linearly separable data.

i.e. there exists a hyper plane which separates the positive from the negative examples. Figure 3.7 illustrates linearly separable and non-separable situations.

As shown in Figure 3.7, a linear separating hyperplane is a set of points x satisfying $\mathbf{w}^\top \mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal vector to the hyperplane and b is the bias. The data from each class is bounded by a parallel bounding plane $\mathbf{w}^\top \mathbf{x} + b = +1$ or $\mathbf{w}^\top \mathbf{x} + b = -1$. Thus it gives the following constraints for all data point,

$$\begin{aligned} \mathbf{w}^\top \mathbf{x} + b \geq +1, \quad \text{for} \quad \mathbf{x} \in A+ \\ \mathbf{w}^\top \mathbf{x} + b \leq -1, \quad \text{for} \quad \mathbf{x} \in A- \end{aligned} \tag{3.37}$$

Combined with the label description, we have

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for} \quad i = 1, \ldots, N \tag{3.38}$$

When two classes are linearly separable, there normally a family of separating hyperplanes, as well as bounding planes, between the two classes. According to the statistical learning theory [243], the optimized bounding planes are those with a maximal margin between them. Thus SVMs search for the separating hyperplane by maximizing the margin between two bounding planes, which can be formulated as a convex optimization problem as shown in Equation (3.39).

$$\begin{aligned} &\min \frac{1}{2} \|\mathbf{w}\|_2^2 \\ &\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \text{for} \quad i = 1, \ldots, N \end{aligned} \tag{3.39}$$

For non-separable data, we can define a "soft margin" between the bounding planes by introducing a nonnegative *slack variable* for each data point,

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \text{for} \quad i = 1, \ldots, N \tag{3.40}$$

The 1-norm of the slack variables $\sum_{i=1}^{N} \xi_i$ is called the penalty term. Then the optimiza-

tion problem of SVMs for non-separable data becomes:

$$\begin{aligned} &\min(\frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i) \\ &\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \\ &\qquad\qquad \xi_i \geq 0 \quad \text{for} \quad i = 1,\ldots,N \end{aligned} \tag{3.41}$$

there C is a positive parameter to balance the weight of the penalty term, which can be determined with a tuning procedure [68].

Equation (3.41) is a standard convex quadratic program, which is called the primal problem. It turns out that in most cases the primal problem can be solved more easily in its dual formulation, as shown in Equation (3.42) [68].

$$\begin{aligned} &\max(\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i y_i \mathbf{x}_i^\top \cdot \mathbf{x}_j \alpha_j y_j) \\ &\text{subject to} \sum_{i=1}^{N} y_i \alpha_i = 0, \quad \text{and} \quad 0 \leq \alpha_i \leq C \quad for \quad i = 1,\ldots,N \end{aligned} \tag{3.42}$$

Nonlinear SVMs

The nonlinear situation can be handled by mapping the data into a higher dimensional space as shown in Figure 3.8. Suppose the input data in space $\mathbb{R}^n$ is mapped to a higher dimensional feature space $\mathcal{F}$ by some mapping Φ. Thus the data point $\mathbf{x}_i$ is mapped to $\Phi(\mathbf{x}_i)$ for $i = 1,\ldots,N$. Then SVMs can be applied to the mapped data for classification. This approach can easily increase the additional computational complexity due to the computation of a mapping and the generated higher dimensional data.

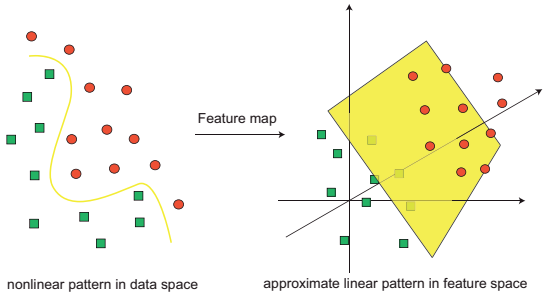


Figure 3.8: Transform linearly non-separable data to linearly separable data.

Boser, Guyon and Vapnik suggested to apply the kernel trick, which avoids the explicit nonlinear map Φ [37]. Note that only the dot product between the input data vectors $\mathbf{x}_i^\top \mathbf{x}_j$ is needed in the dual SVM formulation as shown in Equation (3.42). If we know the dot product of the mapped data $\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$ for all $i, j \in (1,\ldots,N)$, it is possible to solve the dual problem in the feature space $\mathcal{F}$ without computing the mapped data explicitly. Suppose we have a kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$, then the dual nonlinear SVM is given by

$$\begin{aligned} &\max(\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}_j)\alpha_j y_j) \\ &\text{subject to} \quad \sum_{i=1}^{N} y_i\alpha_i = 0, \quad \text{and} \quad 0 \le \alpha_i \le C \quad \text{for} \quad i = 1,\ldots,N \end{aligned} \tag{3.43}$$

Given the training data (and kernel functions in the nonlinear cases), we can obtain the predicted separating hyperplane ($\mathbf{w}, b$) by solving the SVM formulations (linearly separable data for Equation (3.42) and nonlinear for Equation (3.43)). Then we can formulate the classifier as

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top\mathbf{x} + b) \tag{3.44}$$

SVMs became popular because of their good performance in many computer vision tasks, such as face and character recognition, image segmentation, image classification, etc. Experimental results have shown that the SVM can achieve higher performance than K-Nearest-Neighbor(KNN), Hidden Markov models (HMM), particular Artificial Neural Networks (ANN), Naive Bayes (NB) and Decision Trees (DT) in image classification and other classification tasks. Although the speed of the learning period of a SVM is typically slower than for DT, NB and KNN, it can compute as fast as the other algorithms in the classification stage.

4

SYSTEM FRAMEWORK

In this chapter, we investigate the general pipeline and propose a framework for segmenting out moving objects from video sequences. Our system processes videos, which consist of sequences of consecutive images. Specifically, the proposed system focuses on processing and analysing the "motion" information in the video sequence in an unsupervised way.

Given a sequence of successive images, which captures a dynamic scene containing multiple moving objects, the system should be able to determine the number of moving objects in the sequence, and extract the locations and representations of these objects. In this chapter, we introduce the design of the proposed system and briefly describe the main modules.

4.1. OVERVIEW

Object detection approaches can broadly be classified into 3 main streams: top-down, bottom-up and a combination of both [35]. The top-down approach processes images starting from the highest level, by analyzing semantic information from a global pipeline. A top-down scheme for object detection typically includes two stages: a learning stage for obtaining global descriptors of objects, and a verifying stage to localize (segment out) the hypothesized objects from images based on the knowledge obtained in the training stage [24, 51, 183, 193]. Bottom-up approaches focus on local features, which analyze the videos from low-level information (such as contrast, color, intensity, orientation, texture, and motion). Approaches based on a bottom-up scheme first segment the images into salient regions that are relatively homogeneous. A recognition process is then used to classify the regions to corresponding object classes [7, 28, 36, 55, 217]. The top-down approach can often detect the precise object regions, but for this it requires a large amount of training data. The bottom-up approach can be applied to any image sequence without prior knowledge, but it can not solve the ambiguities of objects. For example, an object can be segmented into parts due to the discontinuities of local features in these parts, and two objects may be merged if they have similar appearance. Some approaches combine the top-down and bottom-up processing within a single scheme, which is inspired by the achievements of

the human recognition system [78]. These methods utilize both global and local features to improve the segmentation accuracy and recognition performance. A variety of approaches are proposed based on particular strategies of combining the two schemes, involving various techniques, such as feature extraction, motion analysis, data clustering and machine learning, etc. [35, 144, 186, 239, 274]. There is no general answer to the question: "What is the best method for moving object detection?", nor a standard methodology for evaluating the methods. The underlying reason for this is that the detection of objects is complicated and varies subtly in different situations.

In this thesis, we address the problem of moving object detection in a video sequence, without giving prior knowledge of the objects. The detection of moving objects mostly relies on the motion information which is implied in the changes between images. A bottom-up strategy is applicable for detecting regions with "coherent" motion. To clarify this issue, we employ the following definition of a moving object:

- A moving object moves independently of other objects in the scene.
- A moving object can move in some frames of the sequence and can be present in the following frames without movement. If an object has moved in some frames, it should always be detected in the rest of the video sequence
- A moving object can be a combination of several parts that undergo different relative motions but move together. For example, a walking person is regarded as an moving object, even though his legs and arms move differently.
- The background region including all static objects is also regarded as an object, of which the motion reflects the camera movement.

Objects satisfying these conditions should be segmented out. Due to a lack of prior knowledge of the scene objects, we can not immediately distinguish between the background and foreground objects. The fourth definition above emphasizes the fact that the static background can only move as a result of the camera motion. If the regions of moving objects and background are separated correctly, it is usually fairly easy to tell their classes through a classification step. Therefore, our goal is to segment the video frames into meaningful regions, each corresponding to a moving foreground object or the background. To avoid any ambiguity in the rest of the thesis, the term "foreground moving object" is used to denote the object moving independently from background (such as a walking person, a moving car, etc.), and the term "moving object" can refer to either a foreground moving object or the background.

We propose a moving object detection system, that can automatically segment out the moving objects from a video which contains multiple rigid objects moving independently, either with a static or moving camera. The proposed system consists of a bottom-up scheme of segmenting out moving objects based on their motions and a top-down process of tracking the detected objects in case they stop moving. It does not require prior knowledge of the objects present in the video (i.e., the number of objects and descriptors of the objects), nor is it dependent on any predefined object models at the image level. As described in Chapter 2, there are three fundamental issues in the design of a motion based detection system: motion estimation, motion segmentation, and segmentation without motion. These three issues correspond to the three main modules of our system, as shown in Figure 4.1.

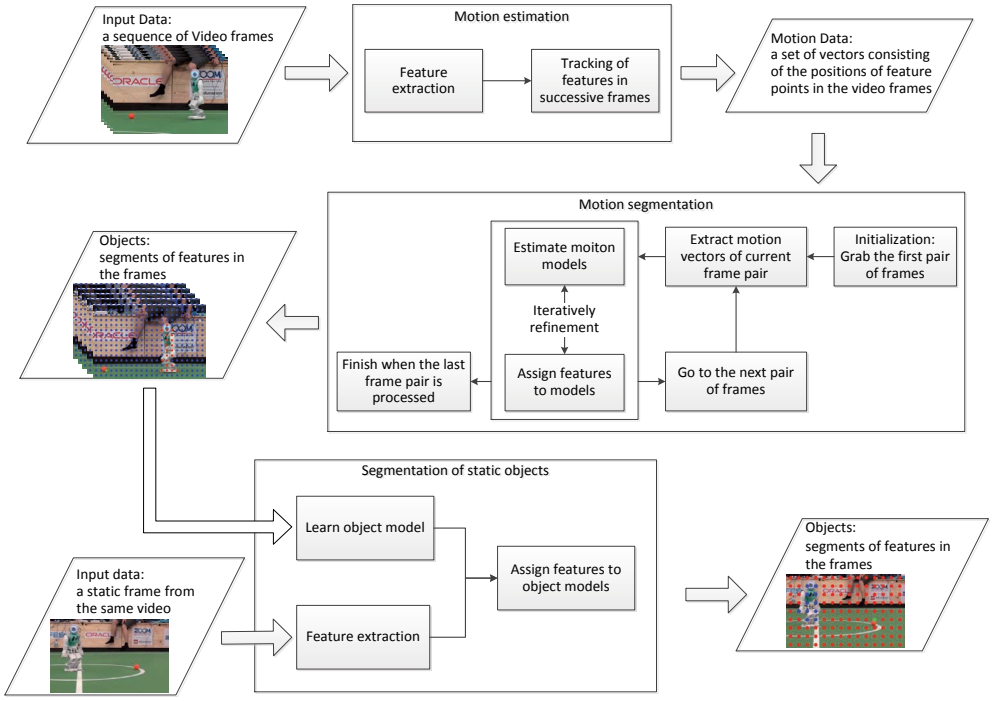


Figure 4.1: The framework of our system. The picture shows at the top, from left to right, the extraction of motion information. The picture shows in the middle, from right to left, the motion-based segmentation process. The picture shows at the bottom, from left to right, the segmentation of objects that are no longer moving using information learned when they were moving.

Motion estimation is considered as a low level vision problem because it processes pixels or local features. In this thesis, we extract 2D motion information from a video, which is more challenging than motion estimation between two images. The choice of an adequate motion estimation approach is important because the acquired data is used as the input of the next module: motion segmentation. 2D motion estimation determines the movements of image elements in the two dimensional image plane. However, the scenes and objects in the real world are three dimensional.

Motion segmentation based on 2D motion data is challenging because the consistency of 3D motions could be broken in 2D images due to the camera projection. Motion segmentation solves the problem of detecting objects from a video sequence when they are moving. These objects should also be detected when they are no longer moving. Based on the obtained information from motion segmentation, we investigate static object detection in the last module.

Because the proposed method is a combination of solutions to different sub-tasks, there is neither a benchmark data set, nor a universal evaluation methodology, for it. To address this issue, we choose specific videos from several benchmark datasets based on the description in Section 1.2. The performances of individual modules in the system, i.e., the motion segmentation and object detection, are measured by different metrics based

on the data acquired from the given videos.

In Section 4.2, a brief description of the data sets used in the experiments is given. Section 4.3 presents the problem and the solution of motion estimation in our system. The motion segmentation based on the results of motion estimation, is introduced in Section 4.4. Section 4.5 gives a brief discussion of static object segmentation based on motion segmentation results.

4.2. DATA SETS

In line with the problem definition in Section 1.2 and the above mentioned moving objects definition for segmentation in this thesis, the proposed system aims to process videos that satisfy the following conditions:

1. There are multiple objects moving independently in the video;
2. The objects can only be rigid bodies or articulated rigid-bodies.
3. The camera can move or be stationary;
4. Any motion can take place on a subsequence of a video;

Based on these principles, we chose specific videos from several benchmark datasets. To evaluate the performance of the various stages of the proposed system, the ground truth segmentation is provided for the given data. Descriptions of the used datasets are provided in the remainder of this section.

ROBOCUP 2014

We chose 7 videos from one of the competition videos of Robocup 2014 [1]. Videos in this dataset record scenes of robots playing football. Foreground moving objects in these videos are robots and balls. The camera can be stationary or moving. Each video in this dataset has a length of 31 frames. These videos are standard HD (1280 × 720 pixels). We generated a pixel accurate segmentation mask for every video frame. Figure 4.2 illustrates some frames and segmentation masks of a video in the dataset. The frame rate is 25 fps.

A SUBSET OF CDNET 2014 DATASET

The CDnet 2014 data set is a benchmark dataset for change detection [256]. It provides videos with different kinds of objects captured in a variety of situations, including some extreme conditions, such as videos captured by infrared cameras, videos in the night with low-visibility of objects, videos with a very low frame rate, etc. We chose 16 videos that satisfy the above mentioned conditions, from the category Baseline. These videos include scenes of moving pedestrians and moving cars with unchanged global illumination. The camera can move slowly or be stationary. Each video has a length of 31 frames, with a frame rate from 24 to 30 fps. Each video in this dataset has a resolution of 320 × 240 pixels.

This dataset also provides pixel accurate masks of moving and static regions for each frame, including the curves of motion boundaries. However, individual moving objects are not distinguished, as illustrated in the second row of Figure 4.3. Based on the provided ground truth data, we generated the required segmentation masks, by assigning different values to regions of different objects and eliminating the motion boundaries, as illustrated in the third row of Figure 4.3.

[1] `https://www.youtube.com/watch?v=dhooVgC_0eY`

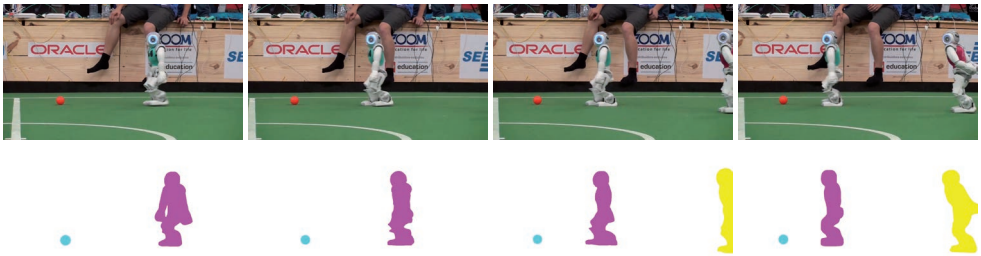


Figure 4.2: Some frames from a video in the Robot soccer videos, along with the segmentation masks. Objects with different motions are annotated in different colors.

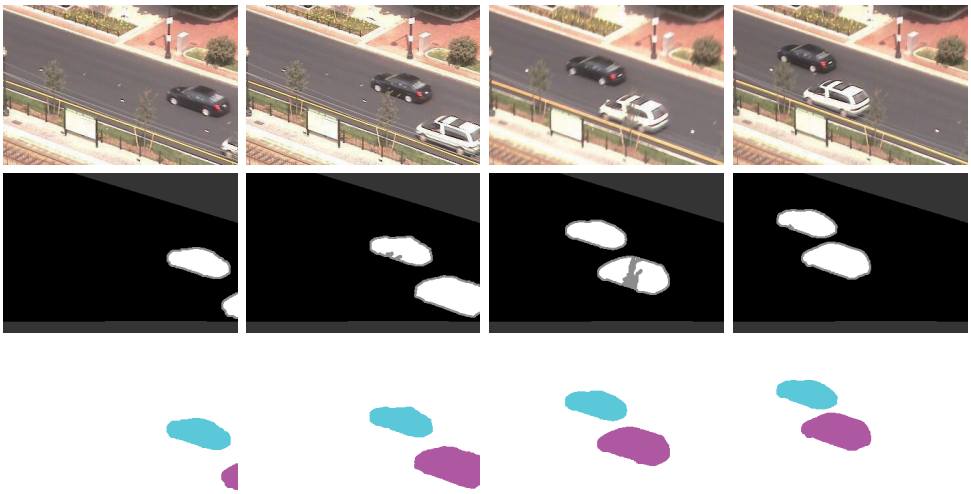


Figure 4.3: A video of a traffic scene refers to CDNet 2014: the first row shows some frames in the video; the second row displays the annotations of corresponding frames provided by the dataset; the third row gives the generated segmentation masks .

HOPKINS155 DATASET

The Hopkins155 dataset is a benchmark dataset built for evaluating feature based motion segmentation [236]. It contains 50 videos, which are divided into three categories. The category named "checkerboard" contains several objects covered with a uniform checker board surface, which make 3D rotations and translations. Videos in this category are captured by a handhold camera. The camera can be in one of four conditions: stationary, translating, rotating, or a combination of translating and rotating. The "traffic" sequences contain moving vehicles in outdoor traffic scenes. The remaining sequences which contain motions constrained by joints, head and face motions, people walking, etc., are put in the category named "others". Videos in the last two categories are taken using either a stationary or a moving camera. The resolution varies between 320×240 pixels and 720×480 pixels.

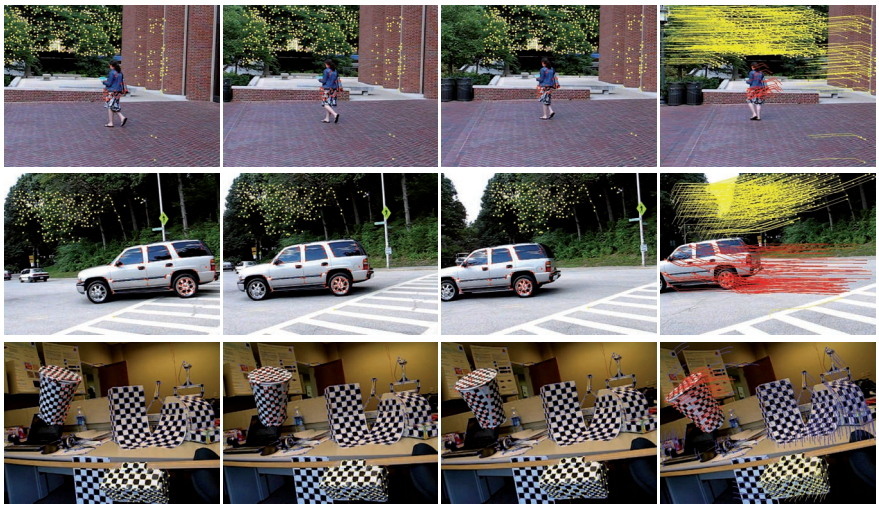


Figure 4.4: Examples of three videos from Hopkins155 dataset: each row illustrates four frames of a video, as well as the provided feature points. The last frame of a row shows the trajectories of the points in the provided data. Points and trajectories are marked by different colors according to the provided segmentations.

Table 4.1: The information of 155 sequences in Hopkins155 dataset [236]. The table shows for each of the aforementioned category, the number of sequences, the average number of tracked features and the average number of frames, for videos with two and three moving objects respectively.

	2 Motions			3 Motions		
	# Seq.	Points	Frames	# Seq.	Points	Frames
Checkerboard	78	291	28	26	437	28
Traffic	31	241	30	7	332	31
Others	11	155	40	2	122	31
All	120	266	30	35	398	29

The dataset provides video sequences along with the feature points extracted and tracked in all frames for each video, as illustrated in Figure 4.4. The provided trajectories are manually corrected, thus they are regarded as the true motion of these feature points. The Hopkins 155 dataset also provides the ground-truth segmentation of the trajectories for each sequence. For each video sequence, several subsequences of point trajectories are extracted based on the number of moving objects, see [249]. There are in total 155 sequences of point trajectories for the Hopkins155 videos. Table 4.1 gives some information of the provided motion sequences.

FREIBURG-BERKELEY MOTION SEGMENTATION DATASET (FBMS-59)

The Freiburg-Berkeley Motion Segmentation Dataset (FBMS-59) is a benchmark dataset for motion segmentation [177]. It consists of 59 videos, including videos of traffic scenes, animals moving and people walking. The length of the video sequences varies from 18 to 719 frames. The resolution of the videos varies between 350×288 pixels and 960×540

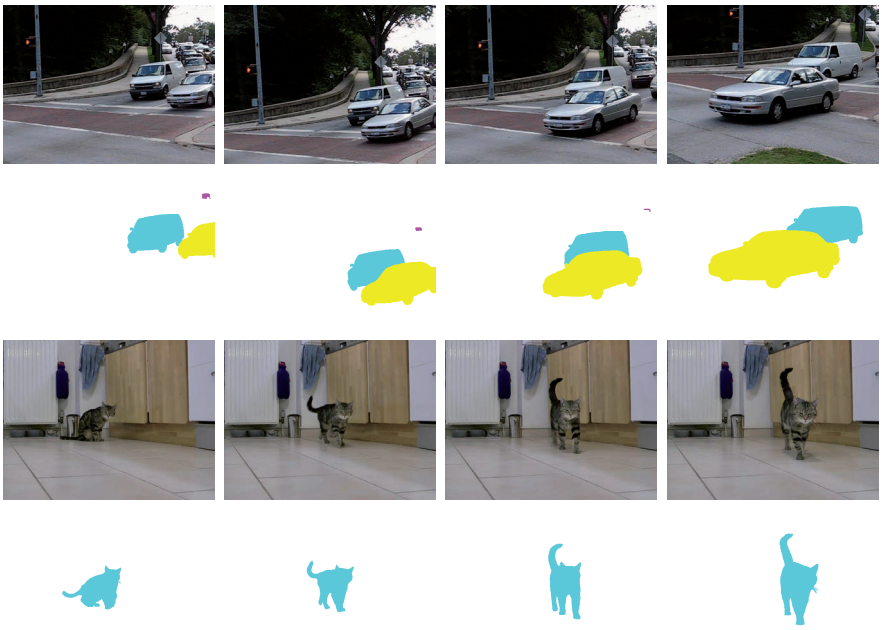


Figure 4.5: Some frames along with their segmentation annotations of 2 videos from the FBMS-59 dataset.

pixels. This dataset provides pixel-accurate segmentation annotation of moving objects for some key frames in a sequence, as illustrated in Figure 4.5. A total of 720 frames is annotated in the dataset.

4.3. FEATURE EXTRACTION AND MOTION ESTIMATION

Given a video sequence, the first step is to extract motion information from the input video frames for our system. As mentioned in Subsection 2.3.2, motion estimation approaches generally fall into two categories: pixel-based methods and feature-based methods [41, 123, 228, 233]. There are two corresponding representations of motions. The pixel-based methods compute the per-pixel correspondences between two images and represent the motions as a collection of displacement vectors, which can be called the motion (or displacement) field [228]. The feature-based methods determine corresponding feature points over multiple frames. The trajectory of a feature point is represented as a vector consisting of its locations in the frames in which it has been tracked.

A variety of motion estimation approaches have been proposed in the literature and applied in different areas. The choice of motion estimation approach in our system is related to the motion segmentation algorithm used in the next step. The design of motion segmentation algorithm must take the motion representation into consideration because it uses the motion data as inputs. Segmentation based on feature trajectories can be more reliable than the segmentation based on pixel motion fields, since feature trajectories can

provide more information on the motions in a video than two-frame motions [49]. However, segmentation based on pixel motion fields provides per-pixel precise regions, which carry more information of the object than a few of feature points obtained by feature-based motion segmentation approaches.

In this thesis, we first address the problem of obtaining motion information in a video sequence, and the obtained motion data is used as input to the motion segmentation module. As we proposed a motion segmentation algorithm that can be generally applied to both two-frame motions and feature trajectories, which will be described in Chapter 5, the motion estimation approach in our system is a user choice. We will investigate the differences of segmentation results based on different types of motion data, that are obtained from different motion estimation approaches. In Chapter 5, we will introduce three representative motion estimation approaches from the two categories, i.e., the pixel-based methods and the feature-based methods, and apply the chosen approaches to obtain different types of motion data.

4.4. MOTION-BASED SEGMENTATION

Suppose a scene containing multiple rigid moving objects is captured by a moving camera. We investigate the issue of detecting moving regions in the video sequence. This research is based on the assumption that the foreground objects are rigid bodies, and each is accompanied by a unique motion pattern [121]. The object detection is therefore the problem of segmenting out the moving regions with respect to their motion coherence.

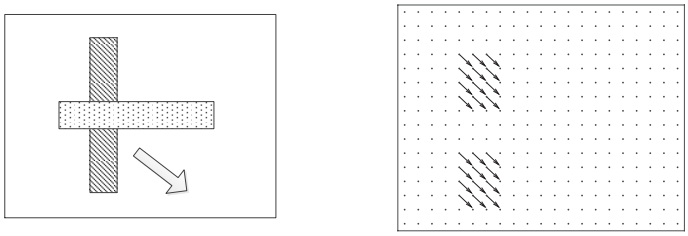
The motion estimation module produces a collection of points (or pixels) and their motion information in a sequence. Generally, if we know the number of objects and how they move, the segmentation can be achieved by simply assigning each point to the motion model it fits. However, it is unrealistic to acquire such information for a dynamic scene under unpredictable variations. Our goal is to find a solution for more general situations, i.e., the number of moving objects and their motion patterns are all unknown. Thus the motion segmentation module in our system is required to segment the video frames into a number of motion regions using the detected motion data and their implied geometric properties, while no additional information of the scene motions is available. Thus motion segmentation needs to address three issues given the feature points (pixels) and their motions:

1. How to determine the number of moving objects in the data?
2. How to determine the motion pattern of each object in the data?
3. How to assign each point to a related object?

Motion segmentation is difficult: not only because of the above issues, but also due to the complexity of camera projection and three-dimensional motions. An image is a 2D mapping of the 3D scene obtained by camera projection, in which we lose one dimension [113]. The 3D motion of a scene object gives rise to the 2D motion of corresponding regions in image plane. Motion segmentation based on the 2D motion in images is called 2D motion segmentation [155]. Research has shown 2D motion segmentation is effective for the static scenes with simple motions, i.e., the camera or a scene object undergoes a single 3D motion [155]. However, when the scene is dynamic, i.e., there are multiple motions, the 2D motion segmentation approaches often fail to give correct segmentations [155]. This

happens because the 3D geometric continuity may be lost in the 2D motion fields after camera projection [157, 236]. For example, a moving object could be broken into different 2D motion fields, because of depth discontinuities, occlusions, perspective effects, etc. (as shown in Figure 4.6).

Segmentation based on 3D motion consistency is meaningful since most of the applications need to address dynamic scenes. 3D motion segmentation mainly focuses on the issue of recovering 3D motion consistency from the 2D motion data in an image sequence [155].



(a) The moving vertical bar is partly occluded by the horizontal bar.

(b) The detected motion of the vertical bar is inconsistent.

Figure 4.6: An example of a moving object that is broken into two parts because of occlusion.

Another problem is that motion data often contains noise due to the errors produced in feature extraction and motion estimation [41, 106]. For example, there can be inaccuracies in the observation of feature points. Points can be mismatched between two frames due to the similarity of repetitive patterns. The environment changes, such as illumination changes, occlusion, etc., can cause errors in pixel motion estimation.

Over the past 30 years, considerable progress has been made in the field of motion segmentation and a variety of techniques and algorithms have been proposed to meet different requirements [7, 155, 232, 248, 262]. In this thesis, we focus on the motion segmentation of dynamic video sequences and we investigate the segmentation using different types of motion data. The two motion estimation methods discussed in Section 4.5 provide two types of motion data, each having its own merits and limitations. Our purpose is to develop a motion segmentation algorithm which can use the motion information between successive frames and in a long sequence of frames. We investigate motion segmentation with respect to both 2D motion consistency in the image plane and 3D motion consistency in world space. The algorithm should be applicable for both pixel-based and feature-based motion data and be able to deal with missing data.

In Chapter 6, we will propose a motion segmentation algorithm based on 2D motion consistency. We will evaluate the performance of our algorithm using different types of motion data. We will also compare our approach to some state-of-the-art methods using data from benchmark datasets. In Chapter 7, we will discuss 3D based motion segmentation, and evaluate its performance in experiments.

4.5. SEGMENTATION WITHOUT MOTION

Motion segmentation groups together pixels or feature points with respect to their 2D (or 3D) motion consistency. The segmentation of images intends to produce meaningful information of the scene, since each segment can correspond to individual surfaces, objects, or natural parts of objects in the scene. However, motion segmentation can only deal with motion data. We can not segment out a stationary object from the background based on its motion because the points on the object share the same motion with the points in the background.

If an object is moving in some frames and is present without movement in the next frames of a video sequence, we can apply motion segmentation on the frames where it is moving and obtain the segmentation annotations of these frames. Can we use this information to segment the other frames in which the object is present? This can be regarded as a supervised segmentation task, which learns a segmentation model from the training images, to segment the other images under similar conditions.

In this thesis, the training images are the frames annotated by the motion segmentation algorithm, and the objective images to be segmented are those in which the objects stop moving. The training data is limited in amount and biased, because the images to be segmented and images with annotations are from the same video sequence. But it also implies that the appearance of an object will not change largely in the images of the video. Thus learning a segmentation model from these training images is likely to be possible. Nevertheless, this problem is difficult because the knowledge is "imperfect", since segmentations obtained by motion segmentation are not as accurate as the ground-truth data. Moreover, the evaluation and comparison of our method with other methods is complicated because of the variations in definitions of objects and ground truths. In general, for segmentation without motions, the system has to address the following problems:

1. How to represent an object?
2. How to learn a segmentation model from the obtained segmented images?

In Chapter 8, non-moving object segmentation is investigated by learning the motion segmentation results.

4.6. CONCLUSION

In this chapter, we concretized some of the research goals of the thesis and proposed a system design for detecting moving objects from videos, which tackles research question 1. The proposed system processes video sequences, to turn low-level image features (pixels) into semantic level information (objects). Without any prior knowledge of the objects present in the videos, the key idea is to make use of the motion information implied in the video sequences. The proposed detection system extracts the motion information from a video sequence, and then uses it for segmenting images into regions of objects. Due to the complexity of motions in the real world, defining the moving objects in a video sequence is a difficult and ambiguous problem [26, 94]. We define the moving objects to be detected based on the assumption that the objects can be segmented out based on their motions, see Section 4.1. We chose videos that contain defined objects from several benchmark datasets, to evaluate the proposed system. The chosen datasets are described in Section 4.2.

The proposed system contains three main modules: motion estimation, motion segmentation, and segmentation without motion. The motion estimation module processes the image pixels to extract motion data, which captures the spatial and temporal aspects of image points. Motion segmentation is then applied to separate the image points into subsets based on their motion information, and each subset represents an object moving in the scene. Segmentation without motions deals with the images when no motion information is obtained, by learning a segmentation model from the images segmented by motion. We discussed the problems to be solved in each module and provided a brief introduction of the solutions, in sections 4.3 to 4.5.

In the following chapters, each module will be introduced in more detail, including their solutions to the corresponding problems and evaluations of their performance.

5

FEATURE EXTRACTION AND MOTION ESTIMATION

The first step of our system is to extract the motion data from a video sequence. A digital video can be represented as a time-varying sequence of images, called frames. Consecutive frames in a video are visually similar because of temporal coherence. The changes between frames are observed as "apparent motion" in the 2D image plane, and caused by the relative 3D motions between the camera and the objects in the scene. Motion estimation refers to determining the underlying motions, either 2D planar motions or 3D motions, from a video.

This chapter addresses the problem of estimating the 2D image-plane motion that is observed in a video sequence. As mentioned in Section 4.3, 2D motion estimation can be approached as the estimation of pixel-wise correspondence, or the estimation of image feature displacement. There are two corresponding categories of motion estimation methods, i.e. "pixel-based" and "feature-based". In Section 5.1, a brief introduction to the motion estimation problem is given. Then we present three methods, one is "pixel-based" and two are "feature-based", in Sections 5.2 and 5.3. In Section 5.4, we use these methods for obtaining motion data from a video sequence. We will show the differences between the data obtained by the chosen methods, by visualizing the results. The obtained data can be analyzed and processed further by the motion segmentation algorithms in the next chapter.

5.1. INTRODUCTION

"Motion estimation" in literature may refer to the 2D planar motion in the projected image plane or to the 3D motion in the world space [228]. In this thesis, we use this term to denote 2D motion estimation in the image plane. Thus our purpose is to determine the projected motion of 3D points on the 2D image plane, by locating the projected positions in different video frames. Motion estimation viewed in this way is therefore a problem of estimating frame-to-frame correspondences in a video. Efficient and accurate motion es-

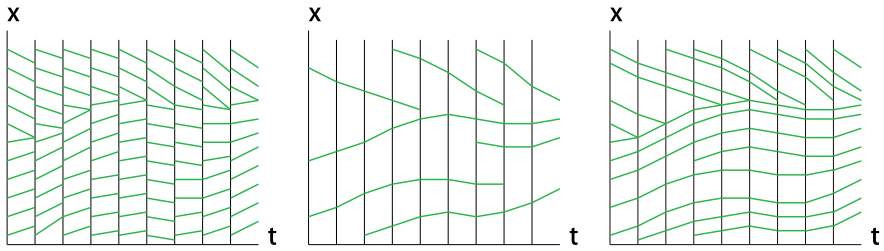
timation is challenging because the video sequences contain noise caused by the camera and the dynamics of the real world. For instance, illumination changes, background movements, shadows, camouflage effects (photometric similarity of objects and background) and ghosting artifacts (delayed detection of a moving object after it has moved away), etc. [41, 129].

Motion estimation approaches can generally be categorized into pixel-based and feature-based, as introduced in Subsection 2.3.2. Pixel-based methods produce dense motion fields at the pixel level, and therefore keep track of subtle image details. However, these methods are sensitive to noise and lighting variations, and can not deal with occlusion. Feature-based methods allow for detecting large displacements and long-term trajectories over multiple video frames. The long-term point trajectories contain more motion information than the motion field of one video frame. Because feature points are tracked over multiple frames in a video, such long term analysis decreases the motion's intra-object variance relative to the inter-object variance [177]. Estimation of point trajectories relies mostly on feature selection and tracking techniques, which can be divided into sparse tracking and dense tracking methods based on the sparsity of tracked features The sparse feature tracking only focus on salient features in the video, which improves the computational efficiency of motion segmentation. However, a sparse set of feature points reduces the precision of representations of the segmented objects. A dense set of feature points carries more information of the processed images than a sparse set of points. However, accurate estimation of dense point trajectories needs more computational resources. The density of trajectories can be controlled by sub-sampling the detected feature points.

The result of motion estimation is called "motion data" in this thesis. The motion data forms the input of the motion segmentation algorithms, which will be addressed in the next chapter. The type of motion data; i.e. dense motion fields, sparse trajectories and dense trajectories, as well as the quality of the data, influences both the type of motion segmentation algorithms that can be used, and the quality of the segmentation result. For each type of motion data, several algorithms have been proposed in the literature. To compare the motion segmentation algorithms in the following chapters, one algorithm is chosen for each type of motion data. Algorithms that are often used in the literature were selected. The following sections describe the algorithms that were chosen, and their characteristics.

5.2. PIXEL-BASED MOTION ESTIMATION

Pixel-based methods compute the pixel-wise correspondences between two images, and result in a 2D motion vector field for all pixels. Most contemporary pixel-based motion estimation methods are based on optical flow techniques. As introduced in Section 3.1, optical flow is a well-known technique to compute the motion of pixels between two images. Over decades of development, significant progress has been made and a variety of approaches have been proposed. For nice reviews of this field, see the papers [16, 91, 165, 223]. In this thesis, we are only interested in the type of motion data that can be obtained from a video. Therefore, we choose a widely used approach, the pyramidal LK method based on affine motion approximation that has been described in Subsection 3.1.2, to compute the optical flows between frames. This method is a successful improvement of the classic LK method, which can handle larger displacements.



(a) Pixel-wise optical flows (b) Sparse point trajectories (c) Dense point trajectories

Figure 5.1: The point correspondences between frames of a hypothetical w.r.t. three motion estimation methods [207].

An optical flow algorithm computes the motion field between two images. Given a video sequence, this method can be applied sequentially to pairs of frames in the sequence. In this way, the motion data extracted from a sequence is formed as a series of two-frame motion vectors, which describes the instantaneous velocity of each pixel from each frame [97, 207, 229]. These motion vectors are temporally discontinuous over frame pairs in a sequence, as illustrated in Figure 5.1a [97, 207], because a pixel of a frame need not be map exactly to a pixel position in the next frame.

Optical flow-based approaches require that the illumination remains similar between two consecutive frames and the frame-to-frame displacements are not too large. These constraints are usually met for videos captured in daily scenes by a digital camera, such as an outdoor scene in daylight, or an indoor scene in unchanged illumination. In such situations, the observed optical flows can be reliable estimations of the actual 2D motions in the image plane [229].

5.3. FEATURE-BASED MOTION ESTIMATION

Feature-based methods extract and track local features from multiple images [123, 233]. As defined by Tuytelaars and Mikolajczyk, a local feature is "an image pattern which differs from its immediate neighborhood" [242]. An extracted feature is associated with an unique descriptor and its location in the image, which we call the feature point. Feature-based motion estimation establishes the correspondences of feature points by matching their feature descriptors, which results in a set of point trajectories [97]. Due to occlusion, a point might be unobserved in some frames. Moreover, errors in feature extraction and matching can also lead to missing data in the process of tracking feature points. Thus the trajectories might have different lengths, see Figures 5.1b and 5.1c. The missing data increases the difficulty of estimating a precise motion model.

The quality of obtained trajectories is affected by feature extraction and tracking techniques. "Good" feature points should have discriminative properties and there should be a high probability that the same point is selected in multiple images [242]. Only a few feature points can be extracted from an image based on this criterion. There is a range of whole feature extraction approaches in the literature, a brief overview is given in Section 2.2.

Feature-based motion estimation generally falls into two categories: the sparse tracking approaches, and the dense tracking approaches. Sparse tracking is the traditional method, which builds the point correspondences based on descriptor matching. Only a sparse set of salient features that have unique descriptors can be tracked. These methods are more likely to generate wrong motions than optical flow based methods, because the geometric constraints between matched points are missing [46]. More recently, some methods apply the geometric constraints on local features, to solve the dense matching problem. In the dense tracking methods, point correspondence is built based on both descriptors and geometric constraints. Thus much denser and more accurate point trajectories can be established, than for sparse tracking approaches.

SPARSE POINT TRACKING

Motion estimation based on sparse tracking normally has two main stages: feature detection and tracking. Similar to Section 5.2, we chose a popular method for obtaining sparse trajectories. We use the scale-invariant feature transform (SIFT) to extract feature points, and a matching strategy based on the Euclidean distance measure is used for tracking the detected feature points [152].

As introduced in Section 3.2, SIFT is a well-developed algorithm for object detection and recognition. SIFT features are invariant to image scaling and rotation, and partially invariant to affine distortion, noise and illumination changes. Moreover, they are tested to be robust to large amounts of pixel noise. The SIFT descriptor is highly distinctive, and is therefore suited for feature matching. These properties make them easy to be tracked in images and usable for object recognition. The SIFT detection and description is described in Section 3.2 in more detail.

The movements of SIFT feature points can be identified by matching the corresponding feature points of two frames. This can be done by computing the similarity of two feature points. The matching algorithm in [152] is used. There, the similarity of two points is measured by the Euclidean distance of their feature descriptors. A point in an image is matched to its nearest neighbor, which is defined as the feature point with minimum Euclidean distance for the SIFT descriptor, in another image. Nevertheless, matches could be incorrect for some points when they have similar descriptors. This often happens for points located in a cluttered background. To eliminate the "bad" matches, a reliability measure is evaluated for every match, as in [152]. Based on the observation that the closest neighbor is usually significantly closer than others for a correct match, the reliability of a match is defined by the ratio of the distance between a point and its nearest neighbor and the distance between the point and its second nearest neighbor. Matches with reliabilities that are greater than a chosen threshold are considered to be incorrect. The threshold value provided by [152] is used in the experiments described in this thesis. Furthermore, the random sample consensus (RANSAC) is used on top of it to discard "bad" matches [85].

Starting from the first frame, all detected points are tracked by matching them with points in the next frame. If a point is tracked in multiples frames, a trajectory is established for it. Tracking of a point is stopped if no reliable match can be found in a frame.

DENSE POINT TRACKING

Dense tracking approaches often combine the basic ideas of optical flow and feature tracking, by applying geometric constraints (as in optical flow estimation) on local features, to

solve the dense matching problem [149, 207, 226, 257].

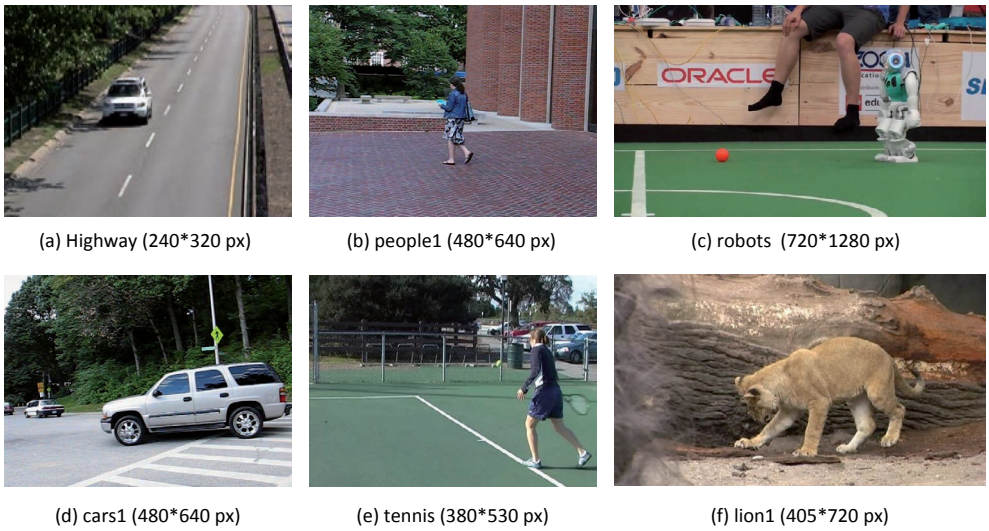
We use the approach proposed in [226], to obtain dense trajectories from video sequences. This method is based on one of today's high quality variational optical flow methods [46]. It selects pixels located at the areas of significant structures in the first video frame as the initial points, because areas without any structure are problematic for tracking. In addition, one can reduce the density of initial points by spatially sub-sampling the pixels. Each of the selected points is tracked to the next frame by using the large displacement optical flow algorithm (LDOF) in [46]. LDOF is a technique that integrates the optical flow constraints and feature matching to estimate the point correspondences. Each point in the initial set is assigned a descriptor based on the local features within a neighborhood centered at this point. The feature descriptor is computed by the histograms of oriented gradients (HOG) [70]. HOG features have proved to be faster than SIFT features in computation in [46].

The computation of optical flow at the selected points is based on brightness constraints, smoothness assumption and descriptor matching [46, 47, 70]. Starting with the first frame, the selected points are tracked to the next frame by using the estimated optical flow field. The tracking should be stopped when a point gets occluded. As in [226], a forward-backward consistency checking of optical flow is used to detect occlusions. For those points located at the motion boundary, the estimated optical flow is less accurate [258]. Therefore, points on motion boundaries are no longer tracked [226]. Besides the tracked points propagated from the previous frame, new points are also initialized in each frame in the tracking process to fill the empty areas, using the same strategy as for the first frame [226].

5.4. VISUALIZATION OF THE MOTION DATA

As discussed above, there are two representations of motion data that can be obtained, either a series of two-frame motion vectors or a set of point trajectories. The point trajectories can be sparse or dense, using different kinds of tracking methods. Therefore three types of motion data, i.e. two-frame motion vectors, sparse trajectories and dense trajectories, can be extracted from a giving video. In this chapter, we chose three motion estimation methods for obtaining these three types of motion data from the video sequences. As introduced in Section 4.2, four datasets were chosen to evaluate the proposed system. The chosen motion estimation methods are applied to the videos to obtain motion data that can be processed by the motion segmentation algorithms in the next chapter.

As we only aim to demonstrate how the methods work and which kind of data they produce here, we visualize the results to show the differences of the three data types. For demonstration, we chose 6 videos that were captured in different scenes with different motions. In Figure 5.2, we display an example frame for each of the 6 videos. The results of optical flow, sparse trajectories and dense trajectories are shown and discussed in Subsections 5.4.1 to 5.4.3 respectively.



(a) Highway (240*320 px) (b) people1 (480*640 px) (c) robots (720*1280 px)

(d) cars1 (480*640 px) (e) tennis (380*530 px) (f) lion1 (405*720 px)

Figure 5.2: Frames from 6 videos in the dataset: (a) "highway" is from CDnet 2014; (b)"robots" is from Robocup 2014; (c) "people1" and (d) "car1" are from "Hopkins 155"; (e) "tennis" and (f) "lion1" are from FBMS-59.

5.4.1. TWO-FRAME OPTICAL FLOW

As mentioned in Section 5.2, we used a pyramidal optical flow method to extract optical flow fields from a video sequence [39] (one can find a quantitative evaluation of this method in [16]). The parameters are set to the values suggested by [39], with a window size of 7, a pyramidal level of 4, and 100 as the maximum number of iterations.

To visualize the obtained motion vectors, a color coding scheme proposed in [16] is used. Each flow vector is coded to a HSB color value based on its direction and magnitude, as shown in Figure 5.3. Based on the color coding scheme, a two-frame optical flow field can be visualized as a colored image. Figure 5.4 illustrates the optical flows extracted from 6 successive frames of each of the 6 videos.

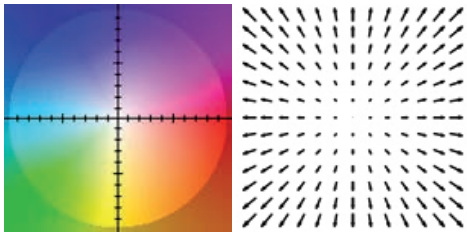
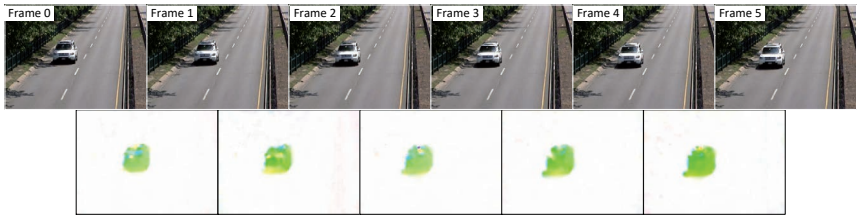
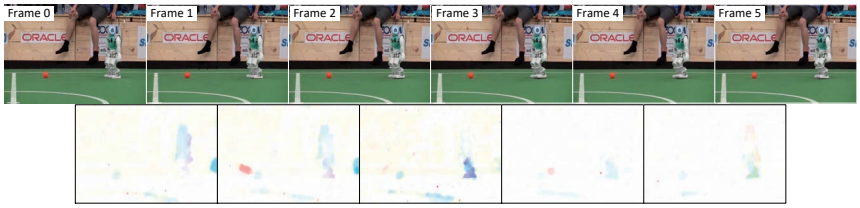


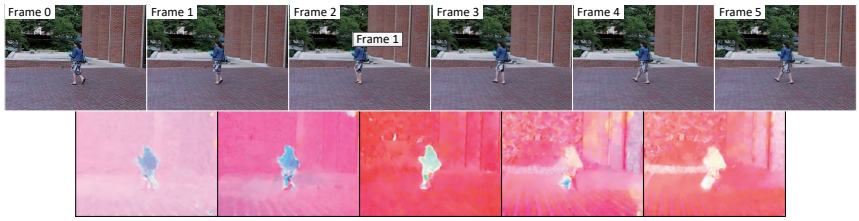
Figure 5.3: Color coding of the flow vectors: each pixel denotes a flow vector where the orientation and magnitude are represented by the hue and saturation of the pixel, respectively [16].



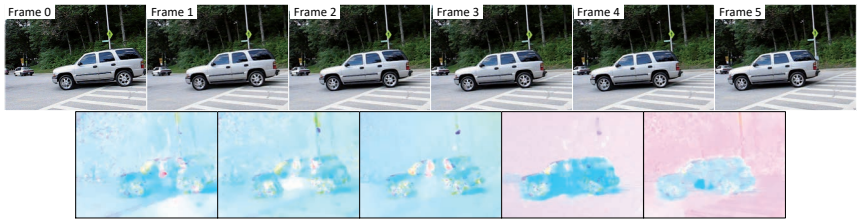
(a) "highway" video



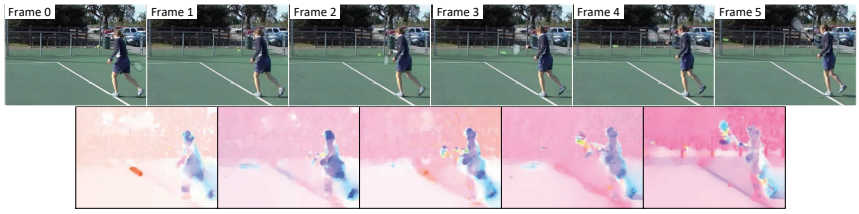
(b) "robot" video



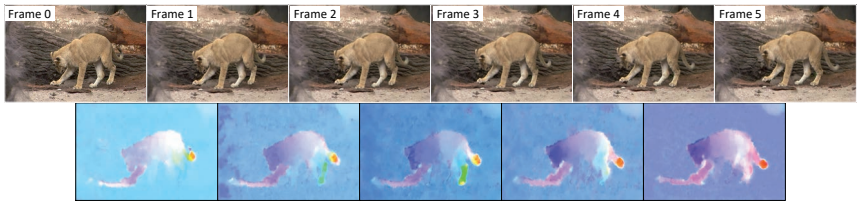
(c) "people1" video



(d) "car1" video



(e) "tennis" video



(f) "lion1" video

Figure 5.4: For each video, 6 successive frames are shown in the upper row; the obtained optical flow fields between two successive frames are shown in the bottom row, using the color coding method in [16].

5.4.2. SPARSE POINT TRACKING

SIFT detection and tracking algorithms, introduced in Section 5.3, are used to detect a sparse set of point trajectories from a video sequence. We used the VLFeat library for SIFT detection [245]. For SIFT feature detection in each image, the edge threshold is set to 10 as in [152], and the peak threshold is set to 2. Figure 5.5 illustrates the detected feature points in two successive images of a video. Points between two images are linked by descriptor matching. The points that are successfully matched are drawn in cyan, and the points that are not matched are drawn in yellow. Table 5.1 gives the statistics for the detected points and matches in Figure 5.5.

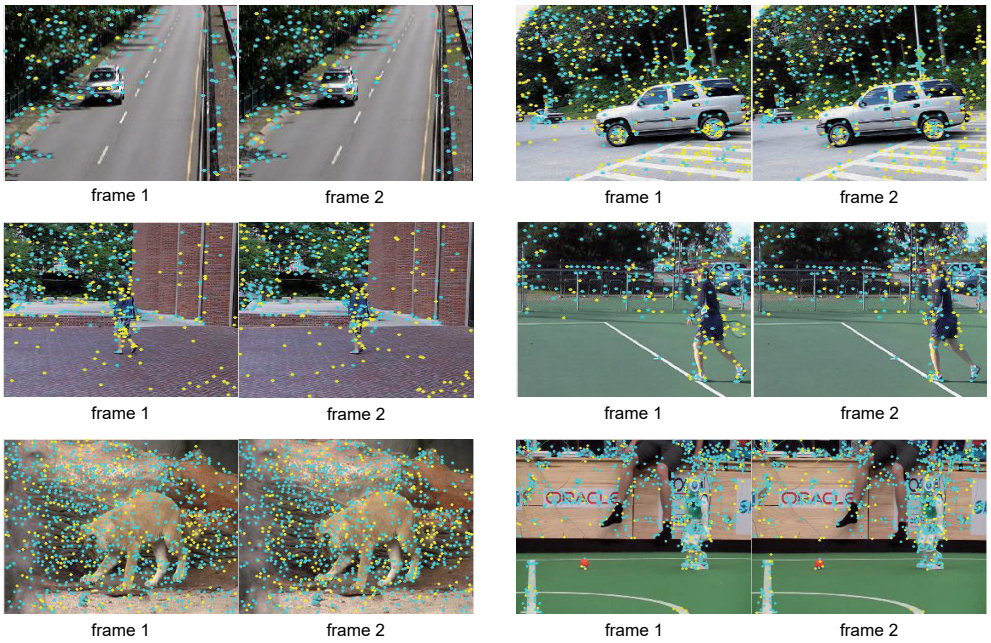


Figure 5.5: For each of the 6 videos, a pair of successive frames with the detected SIFT points are illustrated. Points in cyan are success matches between the two frames, and the yellow ones are not matched.

# of points	frame 1	frame 2	matches
highway	190	200	159
cars1	689	670	337
people1	419	427	211
tennis	362	341	234
lion1	1159	1176	806
robot	1001	988	761

Table 5.1: The number of points and matches in two successive frames (Figure 5.5), based on SIFT tracking.

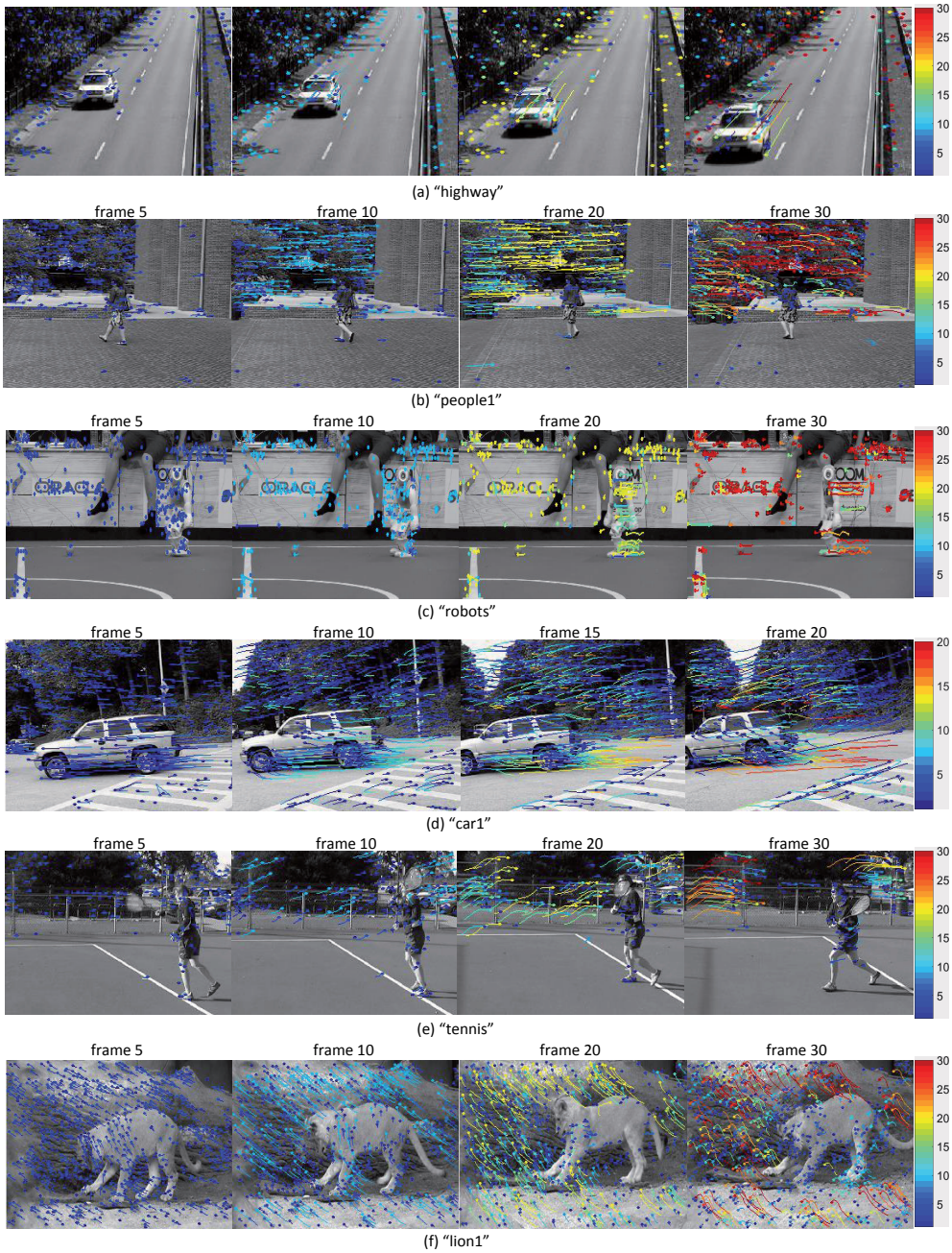


Figure 5.6: The sparse trajectories extracted from 6 example videos. For each video, the tracked points together with their trajectories in four frames at different time steps, are illustrated. Trajectories are drawn in different colors based on their lengths, using a color map shown on the right side for each video.

If a point is successively matched in multiple frames, its trajectory is constructed by its positions in these frames. Tracking of a point is stopped if no match can be found. The length of a point trajectory is calculated as the number of frames in which the point is continuously tracked. Figure 5.6 illustrates the tracked points with their trajectories in frames at particular time steps (frame indices). These trajectories can have different lengths, because a point tracked in a frame at time step t (i.e. frame index t) can be first observed in any frame before t. To show the differences in length, the trajectories are drawn in different colors corresponding to these lengths. Figure 5.7 shows the number of all trajectories and the number of complete trajectories detected at frame t when processing a video sequence of 31 frames.

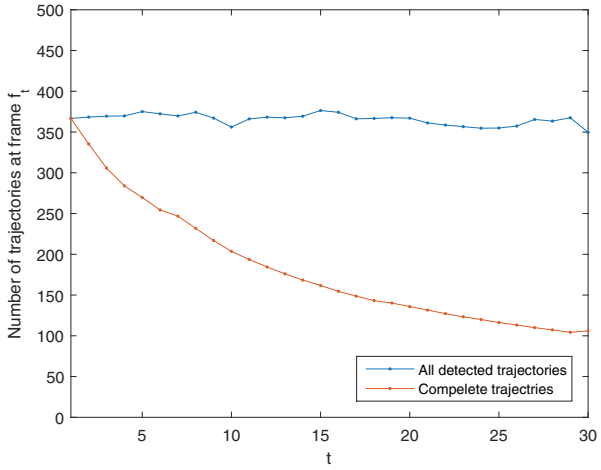


Figure 5.7: Using the SIFT tracking method, the numbers of all trajectories and of complete trajectories detected at frame f_t when processing a video of 31 frames are given, for $t \in [1,30]$. A complete trajectory at frame t means this feature point is tracked over all frames from f_0 to f_t.

5.4.3. DENSE POINT TRACKING

Dense point trajectories are obtained by the dense tracking method in [226], which has been introduced in Section 5.3. This method initializes a dense field of points in each frame by choosing and subsampling pixels in the areas of rich structures, and then tracks these points by computing the large displacement optical flow using the algorithm in [46]. All parameters are set to the values given in [226].

Figure 5.8 illustrates the points in two successive images of a video. Points that are successfully matched are drawn in cyan, and the points that are not matched are drawn in yellow. Table 5.2 gives the statistics of the detected points and matches in Figure 5.8.

Figure 5.9 illustrates the tracked points with their trajectories in different frames of a sequence based on the dense tracking method. These trajectories are also shown in colors corresponding to these lengths, using the same color mapping strategy as for sparse point trajectories in Subsection 5.4.2. Figure 5.10 shows the number of all trajectories and the number of complete trajectories detected at frame t when processing a video sequence of 31 frames.

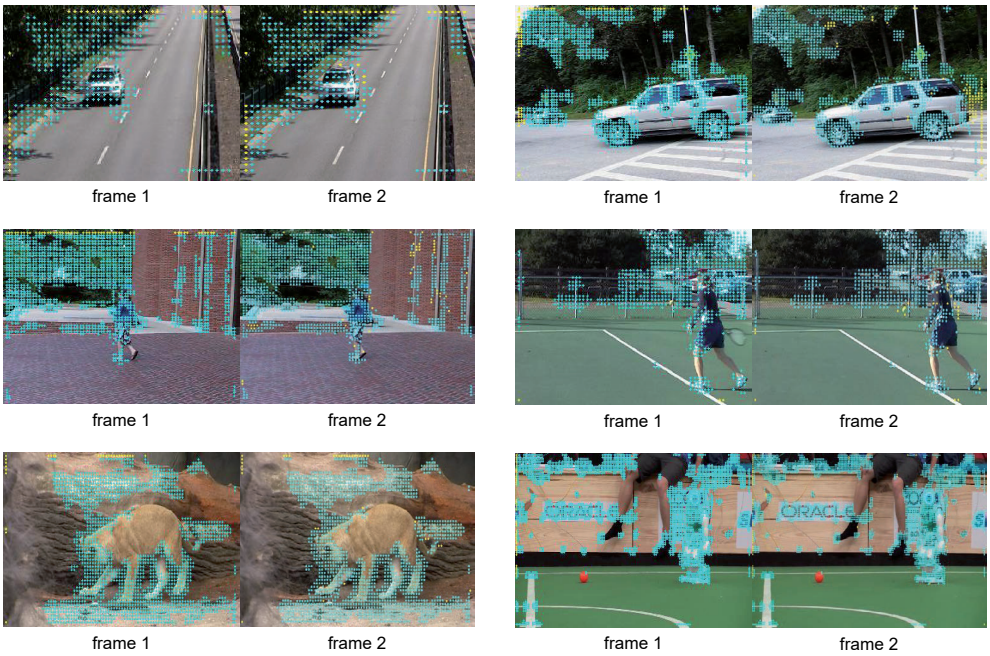


Figure 5.8: For each of the 6 videos, a pair of successive frames with the initial dense points are illustrated. Points in cyan are success matches between the two frames, and the yellow ones are not matched.

# of points	frame 1	frame 2	matches
highway	372	378	337
cars1	1305	1326	1235
people1	1443	1438	1390
tennis	782	803	779
lion1	1609	1617	1581
robot	3240	3260	3197

Table 5.2: The number of points and matches in two successive frames, based on dense points tracking.

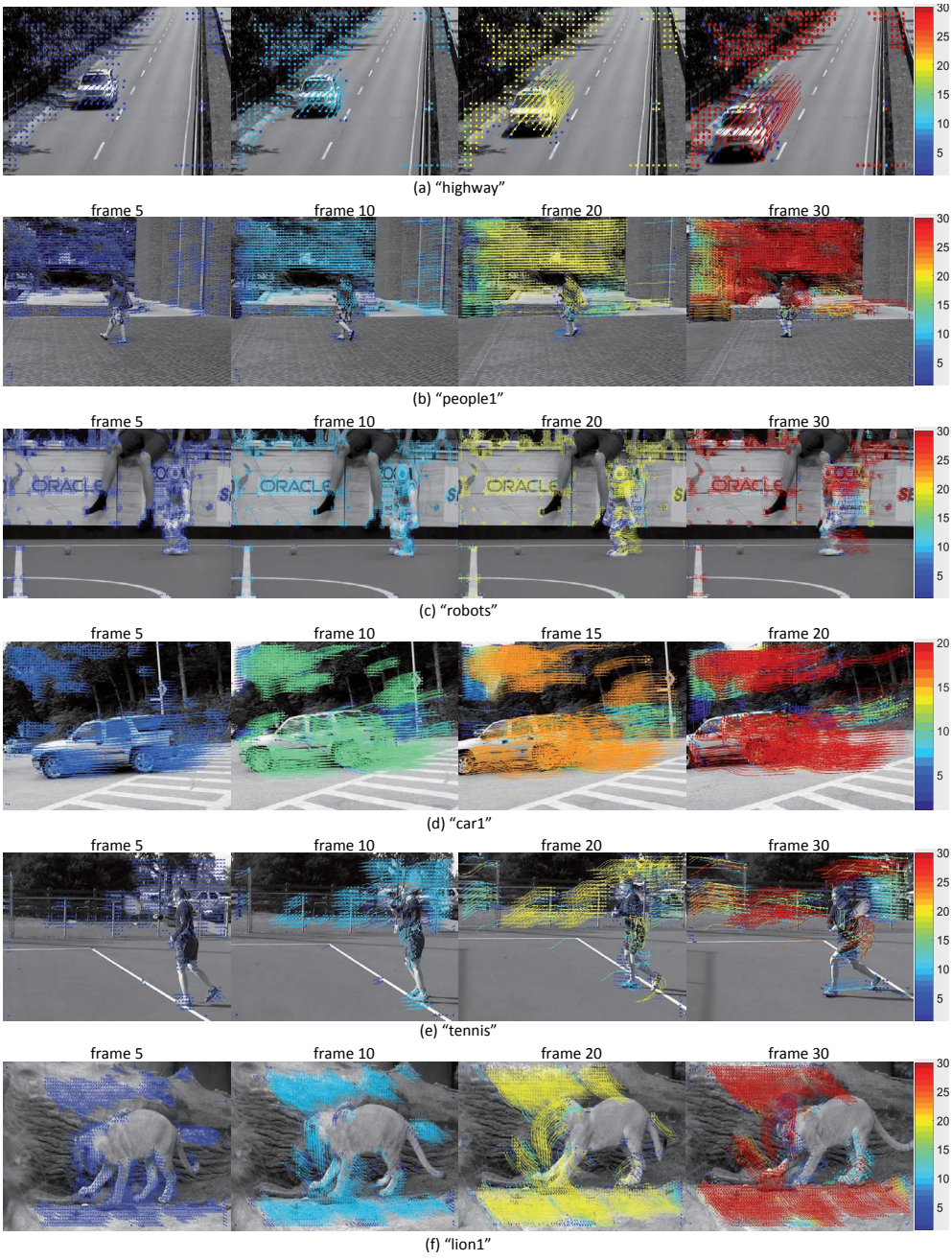


Figure 5.9: The dense trajectories extracted from 6 example videos. For each video, the tracked points together with their trajectories in four frames at different time steps, are illustrated. Trajectories are drawn in different colors based on their lengths, using a color map shown on the right side for each video.

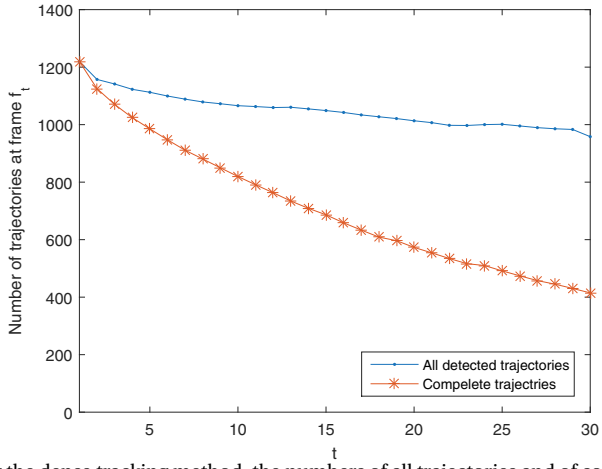


Figure 5.10: Using the dense tracking method, the numbers of all trajectories and of complete trajectories detected at frame f_t when processing a video of 31 frames are given for $t \in [1,30]$. A complete trajectory at frame t means this feature point is tracked over all frames from f_0 to f_t.

5.5. CONCLUSION

This chapter discussed the problem of extracting motion data from a video sequence. Because it is the first step of our video analysis system, the obtained motion data substantially affects the performance of following tasks, since it will be used as the input data of the next module. We discussed three types of motion data that can be extracted from a video, i.e. the optical flow fields, sparse point trajectories and dense point trajectories. Three methods are selected for obtaining the three types of data respectively. We chose 6 example videos to visualize the results.

Figure 5.4 shows that the two-frame optical flow can reflect the actual 2D motion over a short period (between two successive frames in a regular video whose frame rate is about 24 fps) with a high accuracy. The motion vectors within an object show high similarity if the object undergoes a major translation, as in video Figure 5.4.(a) and in some frames of video Figure 5.4.(c) and Figure 5.4.(d). However, the motion coherence of pixels within an object can be inconspicuous if the object moves in an articulated way, as in the video of Figure 5.4.(e) where the robot only moves one leg in some frames, and in the video of Figure 5.4.(f) where the lion plays with branches by moving its legs, while the tail and head move in different ways. Moreover, the motion vectors can be inaccurate for pixels on a texture-less surface, as the car body in video Figure 5.4.(b).

Figure 5.6 shows that the long-term trajectories of points on the same object show higher similarity than two-frame motion vectors. When the number of tracked frames increases, these trajectories of points on the same object become more similar, as shown in the results from frame 5 to frame 30 in Figure 5.6. However, the extracted points are mainly located on the areas that have obvious local structures. Many points are missing in the regions with insignificant structure, such as the legs of people in Figure 5.4.(c) and Figure 5.4.(d), the floor in Figure 5.4.(c) and Figure 5.4.(d), and the lion's body in Figure 5.4.(f). Moreover, the number of "complete" trajectories decreases when the number

of tracked frames increases, as shown in Figure 5.7, because many points are missing during the tracking process (a complete trajectory means the point is tracked in all frames that have been processed).

Figure 5.9 shows the trajectories for the dense tracking method are much denser and more accurate than for sparse tracking. Moreover, the number of "complete" trajectories of dense tracking is larger than that of sparse tracking, see Figure 5.10.

The observations above, suggest that the moving objects can be segmented out by analyzing the similarity of motion vectors (or trajectories). In the next chapter, we will discuss a newly proposed motion segmentation algorithm and compare the performance of motion segmentation on the three types of motion data.

6

2D MOTION SEGMENTATION

In the previous chapter, we discussed how to determine the movements of pixels (or feature points) in the images of a video subsequence. This chapter addresses the issue of partitioning the extracted pixels (or feature points) into groups that undergo independent motions. This is called the 2D motion segmentation problem. We investigate the 2D motion segmentation when the number of moving objects is unknown. Given the positions of 2D points in a pair of successive images, the underlying independent motions are modeled by 2D affine transformations. The segmentation of a video sequence is obtained by iteratively processing successive pairs of frames in this sequence.

In Section 6.1, a brief introduction of the 2D motion segmentation problem is given. A motion segmentation algorithm based on 2D motion coherence is proposed in Section 6.2, with a discussion of possible improvements for the algorithm. Section 6.3 introduces the evaluation methodology, and the experimental results are provided in Section 6.4. Section 6.5 concludes this chapter.

6.1. INTRODUCTION

As introduced in Chapter 5, the motion information obtained from an image sequence is represented as the movements of points (pixels or feature points) in the 2D image plane. The 2D points in the image plane are projections of the 3D points in the real world, and the obtained 2D motions in the images are caused by the 3D motions of the camera and the moving objects. A rigid motion in the 3D world consists of a rotation and a translation of a rigid 3D object. Such a motion can be described by a 3D transformation. In other words, points on the same object undergo the same motion in the 3D world. In this thesis, we only consider the rigid motions.

Motion segmentation refers to the task of labeling image points that undergo the same motion in a video sequence containing multiple moving objects, for the purpose of extracting object-level descriptions from a video sequence [228]. The key issue of motion segmentation is the definition of the "same motion", which describes how an object moves in the images of a sequence. The "same motion" in the 3D real world is different from the

"same motion" in the 2D image plane. A rigid motion of a rigid 3D object in the real world can result in multiple 2D motions in the image plane. Each 2D motion in the image plane can be modeled as an affine transformation [155]. Therefore the 2D motion segmentation can divide the projection of a 3D object into multiple regions, which is then called over-segmentation. 3D motion segmentation is more likely to segment out the integral objects than 2D motion segmentation. But the estimation of 3D motion models is more complex than 2D motion models, because the depth value is missing due to the camera projection.

In this chapter, we investigate 2D motion segmentation, given the motion data obtained in Chapter 5. As introduced in Section 3.3, we use a parametric motion model to describe an independent 2D motion between two images. Due to the over-segmentation problem, segmentation based on two-frame 2D motion coherence may not reflect the actual segmentation into objects in the 3D world. The trajectories over multiple frames of points on the same object show a higher similarity than the motion vectors between just two frames, and the differences between independent motion patterns become more distinguishable, as observed in Section 5.4. Based on this observation, we analyze the 2D motions over a video subsequence consisting of sufficiently many images, by integrating the results obtained from the two-frame motions in the sequence.

Any motion segmentation approach relies on the representation of extracted motion data. As introduced in Chapter 5, the motion information obtained from an image sequence can be represented as either the two-frame motion fields at the pixel level, or as point trajectories over the sequence. Therefore there are feature-based and pixel-based motion segmentation approaches [271]. We investigate a general approach that can be used for both types of motion data. Since both types of motion data represent the movement of either a pixel or a feature point by its location in the images, we use "points" to indicate both pixels and feature points in this chapter when we do not need to distinguish between them.

In general, we investigate the 2D motion segmentation by addressing the following questions:

- How to model the 2D motion of a single object in a sequence of images?
- How to determine the parameters of motion models?
- How to determine the assignment of points to different segments?
- How to determine the number of objects?

6.2. 2D MOTION SEGMENTATION

Regardless of the specific representation used, the obtained motion data is computed frame by frame from a video sequence, as discussed in Chapter 5. We perform segmentation on the motion vectors between each pair of successive frames and accumulate the information in the sequence gradually.

As introduced in Subsection 3.3.3, an affine model of 6 parameters can be used for representing the motion of a rigid object between two images when the displacements in the z direction are small enough. Given the motions extracted from a video sequence, the point correspondences are built between every pair of successive frames. The motion of a single object in a video sequence can be modeled as a series of affine motion models, each corresponding to the movement between two successive frames.

We perform segmentation on the motions between two successive frames based on the affine motion assumption, as discussed in Section 3.3. A classification EM algorithm introduced in Subsection 3.5.1 can be used to estimate the parameters of a motion model and to determine the assignments of the points through an iterative process. However, the number of moving objects is usually unknown for the given video. To estimate the number of moving objects, a divisive scheme is used for hierarchically segmenting the motion data until eventually the number of moving objects is found. The segmentation information based on a pair of successive frames, is propagated to the next pair of frames as prior knowledge using a Bayesian updating process. This makes it possible to further improve the segmentation results in each frame of a sequence.

In the following subsections, we describe the details of the proposed algorithm, by addressing the problems introduced in Section 6.1.

6.2.1. PARAMETRIC MOTION MODEL

As discussed in Section 3.3, the projected motion of an object between two images can be modeled as a 2D affine transformation under an orthographic projection, when the changes between two frames are small (see Equation (3.29)). For the videos with standard frame rate (24fps), the affine motion assumption usually holds, especially for rigid objects.

An affine motion model is parameterized by 6 parameters, 4 for the matrix A and 2 for the translation vector $\mathbf{b}$. Given the movement of any 3 points of the object, $(A,\mathbf{b})$ can generically be computed. In practice, it is difficult to determine the correct solution of the affine parameters because the motion data is noisy. When we have $n \geq 3$ points and their movements, we can compute an approximation of the parameters by solving the overdetermined system, given by Equation (3.31) over all points. More specifically, given the motion vectors of $n \geq 3$ points belonging to an object, the affine motion model $(A,\mathbf{b})$ of the object can be estimated by solving the optimization problem:

$$\begin{gathered}(A,\mathbf{b}) = \operatorname{argmin}_{(A,\mathbf{b})} \sum_{i=1}^{n} r_i ||\boldsymbol{\varepsilon}_i||^2 \\ \text{where } \boldsymbol{\varepsilon}_i = \mathbf{x}'_i - A\mathbf{x}_i - \mathbf{b}\end{gathered} \tag{6.1}$$

where $\mathbf{x}_i$ and $\mathbf{x}'_i$ are locations of point i in two frames, $\boldsymbol{\varepsilon}_i$ measures the error of point i's movement w.r.t. the motion model $(A,\mathbf{b})$, and r_i is a weighting factor which defines the reliability that point i belongs to the object. The value of r_i varies between 0 and 1 for all $i \in \{1,2,\ldots,n\}$, with 0 indicating that the point does not belong to the object and 1 indicating that it does, with certainty. With an appropriate reliability weighting, the points that are more likely to be in the group will contribute more to the estimation of motion parameters. The computation of such reliabilities will be discussed in Subsection 6.2.4.

In practical applications, we can not always get 3 or more points from an object as are needed to estimate its affine motion model. For example, for a small monochrome ball, SIFT can only detect 1 or 2 feature points on the ball. To address this situation, we assume that the affine transformation degenerates to a translation in case just one point is available, and to a combination of a translation and a scaling in case of 2 available points, respectively. The matrix A is then reformulated as shown in Equation (6.2).

$$A = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \text{for} \quad n = 1 \\ \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix}, & \text{for} \quad n = 2 \\ \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, & \text{for} \quad n \geq 3 \end{cases} \tag{6.2}$$

6.2.2. SEGMENTATION BASED ON TWO-FRAME MOTION

The 2D motion of image points between two images can be modeled as a set of 2D motion models described in Subsection 6.2.1. Each model corresponds to an independent (object) motion in the images. Ideally, an object is described by one independent motion. However, because of the projection on the 2D image plane, multiple independent motions are possible.

We do not know the number of independent motions, nor which points undergo the same motion. To determine the number of independent motions, we propose a divisive segmentation algorithm to address these issues jointly.

When the number of motion models is known, a classification EM algorithm can estimate the latent motion models and the assignments of the points to the various models simultaneously [58, 104]. Given an initial segmentation of the points, the EM algorithm executes two steps repetitively to refine the segmentation:

1. Estimating the 2D motion models
   Given a partition of points, the affine motion model of each group can be estimated by minimizing a weighted least squares criterion, see Equation (6.1).
2. Assignment of points
   Given the current estimation of motion models, each point is reassigned to the group that it fits best. A point is reassigned based on the probability that the point belongs to a group, which is described in Subsection 6.2.3.

These two steps are repeated until convergence happens to a locally optimal partition of points. Suppose there are K independent motions, this optimal partition minimizes the sum of group errors:

$$E = \sum_{k=1}^{K} E_k \tag{6.3}$$

Here the group error E_k of group G_k is defined by:

$$E_k = \sum_{i \in G_k} ||\boldsymbol{\varepsilon}_{i,k}||^2 \tag{6.4}$$

where G_k is a group of points assigned to the same motion model, and $\boldsymbol{\varepsilon}_{i,k}$ is the error of point $(\mathbf{x_i}, \mathbf{x_i}')$ given the motion model $(A_k, \mathbf{b}_k)$:

$$\boldsymbol{\varepsilon}_{i,k} = \mathbf{x}_i' - A_k \mathbf{x}_i - \mathbf{b}_k \tag{6.5}$$

Algorithm 1 formalizes the process of the EM algorithm used for 2D motion segmentation. This EM algorithm requires an initial estimation of the segmentation. However,

Algorithm 1 The EM motion segmentation algorithm

Input: A set of N pairs of points. $M := \{(\mathbf{x}_i, \mathbf{x'}_i)\}_{i=1}^{N}$; An initial segmentation of the points in M: $\mathcal{G}' =< G'_1, G'_2, \ldots, G'_K >$.

Output: The estimated segmentation of points in M: $\mathcal{G}$.

1: **repeat**
2: $\mathcal{G} := \mathcal{G}'$;
3: Compute $(A_k, \mathbf{b}_k)$ of $G_k \in \mathcal{G}$ for $k = 1, 2, \ldots, K$;
4: Reset $\mathcal{G}' =< \emptyset, \ldots, \emptyset >$, $|\mathcal{G}'| = K$
5: **for all** $(\mathbf{x}_i, \mathbf{x'}_i) \in M$ **do**
6: Compute the probability of $(\mathbf{x}_i, \mathbf{x'}_i)$ fits the model $(A_k, \mathbf{b_k})$, for $k = 1, \ldots, K$ (see Subsection 6.2.3);
7: Assign point i to group G'_j if the probability of $(\mathbf{x}_i, \mathbf{x'}_i)$ to group model $(A_j, \mathbf{b}_j)$ is maximal (see Subsection 6.2.3);
8: **end for**
9: **until** $\mathcal{G}' = \mathcal{G}$ {convergence}
10: **End**

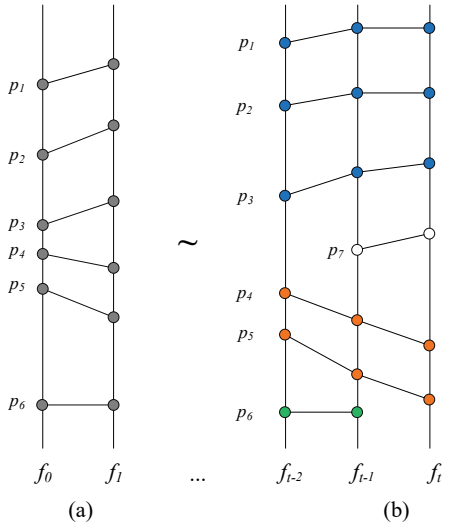


Figure 6.1: Two cases for initialization: (a) Points exacted and tracked from first two frames, are initialized as one big group, as p_1 to p_6 are marked with the same color. (b) Points tracked in subsequent frames are initialized by their prior estimates. For example, when processing frame pair (f_{t-1}, f_t) with $t > 1$, p_1 to p_5 are initialized by their assignments in the previous step i.e. processing frame pair (f_{t-2}, f_{t-1}). Points in the same color indicate that they are assigned to the same group. p_6 is no longer detected in f_t, thus the information of the group in green is missing in this step. p_7 is firstly detected in f_{t-1}, so it is initialized as an unassigned point, which is drawn as a white dot in the figure.

the number of motions is unknown, thus a proper initialization for the EM algorithm is difficult.

To determine the number of objects, we use a divisive segmentation scheme. As we

sequentially process two-frame motion vectors extracted from a video sequence, two situations are considered, as shown in Figure 6.1. In the very first step of the process, i.e. when the motions are extracted from the first two frames of a sequence, no prior knowledge is available. In this case, we initialize all points in one group. In all further steps of the process, some points might have been tracked and segmented in the previous pair of frames. In this case, the prior segmentation of these points can be used to initialize the current step. Given the initialization, we recursively choose a group and split it, until a given stopping criterion is met. In this process, we have to address the following three issues:

Which group to split?

We have to determine which group to split when there are multiple groups. Suppose that the points are partitioned into K groups. We choose to split the group with the largest outliers. More precisely, for each group G_k, we determine the sorted list of errors $\mathcal{E}_k$:

$$\mathcal{E}_k = \text{sort decending}\langle ||\boldsymbol{\varepsilon}_{i,k}||^2 \mid i \in G_k \rangle = \langle e_{1,k}, e_{2,k}, \ldots, \rangle$$

where $\boldsymbol{\varepsilon}_{i,k}$ is the error of point $(\mathbf{x_i}, \mathbf{x_i}')$ with respect to the model $(A_k, \mathbf{b}_k)$. The sorting achieves that $e_{1,k} \geq e_{2,k} \geq e_{3,k}, \ldots$. Next for a user-selected value of q, we choose the first q points in the sorted list $\mathcal{E}_k$ to determine the sum O_k^q of the q largest errors in group G_k:

$$O_k^q = \sum_{i=1}^{q} e_{i,k} \tag{6.6}$$

Finally, we select the group G_k for which O_k^q is maximal as the one that will be split.

How to split the chosen group?

We split the selected group G_k by splitting off the q points with the largest errors. Once the group is split, the number of groups K increases by 1 and the EM algorithm is then used to estimate the segmentation for the new number of groups. Note that in some cases the q points that are split off could belong to different objects. Since the EM algorithm can handle this indirectly, there's no need for additional steps for refining the splitting.

When to terminate the division procedure?

The divisive procedure increases the number of groups by 1 in each iteration, and then evaluates the segmentation based on current estimate. The optimal segmentation is considered to be achieved when it is better than the results in the previous and in the next iteration. The problem then is how to determine the quality of a segmentation result. A quantitative metric is needed to evaluate this.

We measure the intra-group variations and inter-group distances for a segmentation result. The intra-group variation of G_k is measured by the average error $\mu_k = E_k/|G_k|$. The inter-group variation of group G_k w.r.t. G_l is denoted as $\eta_{k|l}$, and is defined by the average error of group G_k w.r.t. the affine transformation of group G_l:

$$\eta_{k|l} = \frac{1}{|G_k|} \sum_{i \in G_k} ||\boldsymbol{\varepsilon}_{i,l}||^2 \tag{6.7}$$

where $\boldsymbol{\varepsilon}_{i,l}$ is the error of point $(\mathbf{x_i}, \mathbf{x_i}')$ to the model $(A_l, \mathbf{b}_l)$. We accept a split of a group if for every pair of groups G_k and G_l, μ_k is sufficiently smaller than $\eta_{k|l}$. Formally:

$$\frac{\mu_k}{\eta_{k|l}} \leq w, \quad \text{for every } 1 \leq k \leq K \text{ and } 1 \leq l \leq K \tag{6.8}$$

where $w \in (0,1)$ is a user defined threshold.

At each stage of the division procedure, the algorithm chooses a group and splits off some points to form a new group, and the estimated number of groups is increased by 1. Next the EM algorithm is then applied to determine a segmentation based on the groups after splitting. It may happen that the number of groups goes down after applying the EM-based segmentation because there is a chance that a group $(A, \mathbf{b})$ will not be assigned any points. Because some initial groups fed into the EM algorithm could consist of points that represent the same motion. Note that the splitting criterion can not guarantee that the split-off points belong to the same object. The new group split from the selected group can contain points from different objects.

The segmentation result produced by EM algorithm is then compared with the segmentation result before splitting. If the current segmentation is better than the previous one, the splitting stage continues. If the current segmentation is worse than the previous one, the previous segmentation is therefore regarded as the optimal estimation and the splitting process is stopped. In this way, the number of objects and the segmentation of points are estimated simultaneously. The main steps of the EM-based divisive segmentation algorithm are shown in Algorithm 2. Note that some objects can have the same motion in some frame pairs in the video, and the divisive algorithm is not able to distinguish them from each other. Nevertheless, if these objects show different motions in other frame pairs, they can be segmented out through a updating scheme, which is going to be discussed in Subsection 6.2.3.

6.2.3. SEGMENTATION OF A SEQUENCE OF FRAMES

Given a sequence of frames, points may be tracked over multiple frames, thereby providing us with long term motion data. We have discussed how to segment the points based on the two-frame motion vectors. In this section, we will investigate how to segment points if these points are present in a sequence of frames.

For a pair of successive frames, the observed motion between two frames is termed as the evidence e. Let $z_i = k$ denote the assignment of a point i to a group k. We wish to determine the likelihood $L(z_i = k)$ that point i belongs to group k given the evidence e. This likelihood is proportional to the probability density $p(e|z_i = k)$:

$$L(z_i = k) \propto p(e|z_i = k) \tag{6.9}$$

So, we need to determine the probability density function $p(\cdot\,|z_i = k)$ of the possible evidence values given the assignment $z_i = k$. Since the evidence is given and the assignment of a point to a group is variable, we may estimate a relative value for the probability density $p(e|z_i = k)$. The idea is that the smaller the error of a point i w.r.t. the affine model of a group k compared to the errors of point i w.r.t. the affine model of all the groups, the more likely it is that point i belongs to the group k. Let $\epsilon_{i,j}$ denote the error of point i w.r.t. the

Algorithm 2 The EM-based divisive motion segmentation algorithm

Input: A video sequence; A set of feature points described by their trajectories in this video sequence.

Output: Segmentation $\mathcal{G}$ of points in every successive frame pair (I, I') of the video.

fetch the first frame I and the second frame I';
$P := \{p_1, p_2, \ldots, p_i, \ldots | p_i \in I, p_i \in I'\}$; {extract points present in both I and I'.}
$\mathcal{G} := \{G_1 | G_1 = P\}$; {segmentation $\mathcal{G}$ is initialized by assigning the points to a single group.}
repeat
  $M := \{(\mathbf{x}_i, \mathbf{x'}_i) | \mathbf{x}_i := \text{Position}(p_i, I), \mathbf{x'}_i := \text{Position}(p_i, I')\}_{i=1}^{|P|}$; {fetch two-frame motion vectors of points in P, function Position(p_i, I) gives the coordinates of point p_i in the image frame I, $|P|$ is the number of points in P.}
  $\mathcal{G}' :=$ EM motion segmentation $(M, \mathcal{G})$; {applying Algorithm 1}
  repeat
    $\mathcal{G} := \mathcal{G}'$;
    select a group G_k in $\mathcal{G}$ ($k = \{1, 2, \ldots, |\mathcal{G}|\}$), if O_k^q (see Equation (6.6)) is maximal; {$|\mathcal{G}|$ is the number of groups in $\mathcal{G}$.}
    split G_k into $G'_{k,1}$ and $G'_{k,2}$;
    $\mathcal{G}' := (\mathcal{G} \setminus \{G_k\}) \cup \{G'_{k,1}, G'_{k,2}\}$;
    $\mathcal{G}' :=$ EM motion segmentation $(M, \mathcal{G}')$;
  until the stopping criterion of divisive procedure holds (see Equation (6.8))
  $I := I'$;
  $I' :=$ fetch the next frame;
  $P := \{p_1, p_2, \ldots, p_i, \ldots | p_i \in I, p_i \in I'\}$; {extract points present in both I and I'.}
  $\mathcal{G} := \{G \cap P | G \in \mathcal{G}$; {initial segmentation of P.}
until the end of the sequence
End

affine model of group j. Then, the likelihood $L(z_i = k)$ is *inversely* proportional to $\frac{\epsilon_{i,k}}{\sum_{j=1}^{K} \epsilon_{i,j}}$. It may happen that the error $\epsilon_{i,j}$ is 0 w.r.t. all affine models $j \in \{1, \ldots, K\}$. To avoid dividing by 0, we add a constant δ to the denominator and a constant δ/K to the numerator. The exact value of δ is not very important, as long as it is small enough. Since the error w.r.t. an affine model is measured in pixels, the value 0.1 was chosen for δ. The resulting fraction, $\frac{\epsilon_{i,k} + \frac{\delta}{K}}{\sum_{j=1}^{K} \epsilon_{i,j} + \delta}$ is inversely proportional to the likelihood $L(z_i = k)$ and has a value from the interval $[0, 1)$. Therefore, we define the likelihood that point i belong to group k as:

$$L(z_i = k) = 1 - \frac{\epsilon_{i,k} + \frac{\delta}{K}}{\sum_{j=1}^{K} \epsilon_{i,j} + \delta} \tag{6.10}$$

Given an image sequence of $T + 1$ frames $f_0, f_1, \ldots f_T$, a segmentation is determined for each pair of successive frames (f_{t-1}, f_t). Assuming that the evidence $E_\tau = (e_1, \ldots, e_\tau)$ over τ $(0 < \tau < T)$ pairs of frames in the sequence is *independent*, we may use Bayesian update to determine the probability that point i belongs to group k given the evidence E_τ of the past

$\tau+1$ frames:

$$P(z_i = k|E_\tau) \propto P(z_i = k) \prod_{t=1}^{\tau} L_t(z_i = k) \tag{6.11}$$

where $L_t(z_i = k)$ is the likelihood of assigning point i to group k using the evidence e_t of the frame pair (f_{t-1}, f_t).

An important issue is the choice of the a priori probability $P(z_i = k)$. Here, different choices are possible. If the length T of the sequence is long enough, all choices will converge to the same a posteriori probability. For short sequences, the choice of the a priori probability can have a significant impact. Here we assume that, a priori, we have no preference for assigning i to any particular group. Since we have a finite number of groups, we use $P(z_i = k) = \frac{1}{K}$. Therefore,

$$P(z_i = k|E_\tau) \propto \prod_{t=1}^{\tau} L_t(z_i = k) \tag{6.12}$$

This implies that the a posteriori probability is completely determined by the likelihood values.

When processing the video sequence, the segmentation algorithm is sequentially applied to the two-frame motions over the sequence. If a point i is continuously detected in the video frames, the probability $P(z_i = k|E_\tau)$ is updated frame by frame.

6.2.4. RELIABILITY MEASUREMENT

When we determine the assignments of points, it is based on the errors to the estimated group motion model. Meanwhile, the motion models are estimated based on an estimates of point assignments. Misclassified points during the process can reduce the accuracy of the motion models, thus reduce the quality of final segmentations. To reduce the impact of misclassified points, we define a reliability value that evaluates the assignment of each point. As argued in Subsection 6.2.1, it is reasonable to use reliabilities as weights, which determine how much each point in a group influences the estimates of the affine motion model, when solving Equation (6.1) [175]. When a point is assigned to a group with a higher reliability, it will contribute more to the computation of the motion model of the group. If the reliabilities are chosen appropriately, we can find more accurate estimates of the motion models, as well as improve the segmentations.

Points assigned to the same group may have different likelihood. If $L_t(z_i = k)$ is higher than $L_t(z_j = k)$, the assignment of point i to G_k has a higher chance to be correct than the assignment of point j to G_k. Thus we can measure the reliability of an assignment based on its likelihood. We propose three ways to compute the reliabilities. Suppose a point i is assigned to G_k based on the motion information of the current frame pair (f_{t-1}, f_t)

1. Suppose a point i is assigned to G_k, we measure the distance of $L_t(z_i = k)$ to the nearest $L_t(z_i = l)$ for any $l \neq k$, as:

$$r_i^{(1)} = \min_{l \neq k} (L_t(z_i = k) - L_t(z_i = l)) \tag{6.13}$$

   A larger distance means the assignment is more reliable.

2. Similar to $r^{(1)}$, we can measure the distance of $L_t(z_i = k)$ to the average of $L_t(z_i = l)$ for all other groups:
$$r_i^{(2)} = L_t(z_i = k) - \frac{1}{K-1} \sum_{l \neq k} L_t(z_i = l) \tag{6.14}$$
where K is the total number of groups.

3. $L_t(z_i = k)$ gives the probability of a point i belonging to a group G_k, which can be directly used as a reliability measure:
$$r_i^{(3)} = L_t(z_i = k) \tag{6.15}$$

6.2.5. CAMERA MOVEMENT

Experiments in Subsections 6.4.1 and 6.4.2 show that the camera movement significantly influences the performance, especially when the camera is rotating. When Algorithm 2 is applied to the Hopkins 155 dataset with all reliabilities set to 1, a drop of 5% in accuracy was observed when the camera is rotating (see Table 6.1). In these videos, the camera rotation causes large point movements compared to the moving objects. This makes it difficult to distinguish differences between the movements caused by different moving objects. As a result, the chance of incorrect assignments in Algorithm 2 increases.

To eliminate the effect of camera movement, we introduce an extra step to compensate for the movement of the camera. The key problem is to find the motion model of camera. Since we can assume that the background does not move, we can use the background movement to determine the camera movement. So we need to identify the background points. Usually the background covers the largest range in the image, so it is easy to identify the "background" by segmenting the feature points and comparing the geometric size of the obtained groups using the non-stabilized motion data. However, the segmentation contains misclassified points for the non-stabilized motion data. Therefore, only the points with a high reliability to belong to the background are used for computing the camera motion. We use a threshold strategy to eliminate the points that are less likely to belong to the background.

The following steps are introduced to compensate for the camera movement.

1. Run the segmentation algorithm to obtain an initial segmentation of the feature points.
2. Select the "background", based on the assumption that the "background" group has the largest inner distance among all groups.
3. Refine the "background" by focusing on reliable points with a chosen reliability measure. Points that have high reliability values for the "background" are used to compute the camera motion. We can select the points which reliability values are larger than a threshold α (0.5 for example).
4. Compensate all feature points for the movement of the camera using the estimated camera motion. Given the motion vector of a point, which is described by its positions in two successive frames $(\mathbf{x}, \mathbf{x}')$, we replace $\mathbf{x}'$ by $\hat{\mathbf{x}}'$, which is computed by:
$$\hat{\mathbf{x}}' = \mathrm{B}_b^{-1}\mathbf{x}'$$
where $\mathbf{x}'$ and $\hat{\mathbf{x}}'$ are homogeneous coordinates, $\mathrm{B}_b = \begin{bmatrix} A_b & \mathbf{b}_b \\ 0^T & 1 \end{bmatrix}$ is the homogeneous affine matrix of the background group.

Since the divisive segmentation algorithm computes the segmentations in an iterative manner, the motion models and partitioning of points are updated and refined iteratively. We need to determine when to apply the background compensation during this procedure. There are several possibilities for adding the background compensation to Algorithm 1:

Option 1 In the inner loop of EM: in Algorithm 1 between lines 2 and 3. In each iteration of the EM loop, the motion vectors are first calibrated by compensating the camera motion. Then the motion models of objects and the assignments of points are estimated based on the calibrated motion vectors

Option 2 In Algorithm 2, right after the loop of EM: between lines 7 and 8. The motion vectors of points are calibrated after the EM algorithm generated a converged result, but before the splitting step.

Option 3 In Algorithm 2, after the splitting step: between lines 13 and 14. The calibration takes place after one step of segmentation and splitting in the algorithm.

The accuracy of estimated camera motion relies on the quality of the identified "background" group, which is from an estimated segmentation. If there are too many incorrect assignments in the "background" group, the camera compensation can have a negative impact of the segmentation. Option 1 tries to dynamically change the estimated camera motion with the segmentations in the EM loop, it will also affect the convergence procedure of EM segmentation. Option 2 utilizes the segmentation result after the EM loop converged, which is regarded as an optimal estimation in this iteration. Option 3 uses the segmentation after splitting, which goes one step further than Option 2. If the "background" is split, it means that the outliers are split off. Otherwise, the "background" group is assumed to be sufficiently accurate. The camera motion estimation is also affected by the method of computing reliabilities. In Subsection 6.4.1, we compared the results of using different options combined with different reliabilities measures, to find a good setting of these parameters based on experiments.

6.3. EVALUATION OF THE METHODOLOGY

As introduced in Chapter 5, the motion implied in an image sequence can be represented in different ways. Correspondingly, there are different types of motion data that can be extracted. Motion segmentation approaches rely on the type of motion data they analyze. Therefore, there is no general dataset that can be used for evaluation and comparison of all existing motion segmentation methods, neither a generally applicable measurement metric [26]. The existing datasets are proposed for different goals, where the definition of motion segmentation and the type of utilized motion data varies.

In this thesis, we provided a detailed definition of moving objects to be segmented out in videos in Chapter 4. Based on the definitions, we chose four benchmark datasets (see Section 4.2) to test the proposed method. The obtained segmentation results are evaluated against the provided ground truth. In Subsection 6.3.1, we summarize the provided data types in the chosen dataset, which is the basis for the experimental set-up.

To evaluate the quality of the segmentation results, we utilize several widely used metrics in the field of motion segmentation These metrics demonstrate the performance from different perspectives, which is further described in Subsection 6.3.2.

6.3.1. DATASETS

As in Chapter 4, we presented four datasets that will be used to test the proposed method in the experiment. One of them (the Hopkins155 dataset) provides the accurate point trajectories extracted from the videos in this dataset, which can be regarded as the ground-truth motion in the video frames. Such data can be directly used as the input of motion segmentation algorithm, thus is regarded as "flawless" motion data. The other three datasets only provide videos, which requires a pre-processing step of extracting motion data from these videos that can be used as the input of motion segmentation algorithm from these videos. The extracted data normally contain errors, which is called "realistic" motion data in this thesis.

Based on the provided data, the experiment of evaluating the motion segmentation algorithm is naturally divided into two parts: the first part uses the "flawless" motion data as input, and the second uses the "realistic" motion data as input. The "flawless" data gives a standard baseline for evaluating motion segmentation approaches. However, in practice, "flawless" motion data is often unobtainable. The evaluation on realistic motion data better reflects the robustness of a motion segmentation algorithm in practice.

THE "FLAWLESS" MOTION DATA

For each video in the Hopkins 155 dataset, a set of sparse trajectories, of the same length as the video sequence, is given. This dataset also gives the groundtruth segmentation of the provided trajectories.

Hopkins 155 dataset is specific for the trajectory based motion segmentation, and provides only sparse trajectories. The provided data in the Hopkins 155 dataset is biased. Firstly, videos in the dataset are short sequences (of 31 frames). Secondly, only full trajectories are considered. Thirdly, half of the videos are from unnatural scenes (the "checkerboard" videos). Fourthly, the number of independent motions is fixed in a video sequence, and equals two or three. Experiments on the "flawless" data are given in Subsection 6.4.2. Since our method dose not focus on point trajectories of the same (full) length, and addresses more general situations (see Chapter 4), results on the data in the Hopkins 155 dataset do not give a fair evaluation that can be extrapolated to a more general practical setting.

THE "REALISTIC" DATA

The remaining three datasets, i.e. Robocup 2014, CDNet 2014 and FBMS-59, provide videos with more variations than Hopkins 155. They contain different scenes recorded in the real world, and the provided videos are of different lengths (with a minimum of 19 frames and a maximum of 800 frames). Since they do not provide the groundtruth motion data, a pre-processing step is needed to extract motion data from the videos.

The proposed motion segmentation algorithm has a flexible scheme, which can deal with not only continuous trajectories in multiple frames, but also the two-frame motion point correspondences. Therefore, in the experiments, we evaluate the proposed algorithm given different types of motion data. In Chapter 5, we discussed three methods to extract motion data from videos: one based on optical flow estimation, and two based on trajectory extraction. These methods result in three types of data:

(1) a series of two-frame motion vectors at the pixel level,

(2) a set of sparse trajectories of detected feature points in the video sequence,
(3) a set of dense trajectories of detected feature points in the video sequence.

In Subsection 6.4.3, we evaluate the performance of the segmentation algorithms on the three types of motion data. Regardless of how the motion is obtained, the extracted motion data is partially erroneous. The number of moving objects in each sequence is between 2 and 7. In some sequences, the number of moving objects varies over the frames. The extracted trajectories can have different lengths (see Section 5.4). More details of the video properties are given in Section 4.2.

The three data sets, i.e. Robocup 2014, CDNet 2014 and FBMS-59, provide the pixel-accurate ground truth segmentation of moving objects for each video sequence, see Section 4.2. The provided segmentations can directly be used to evaluate the pixel-based segmentation results. For the segmentation based on point trajectories, the groundtruth can be obtained by reading the pixel annotations of corresponding locations of the feature points. It is worth to mention that the provided segmentation concerns whole objects even if their parts have different motions. Also note that groundtruth segmentation of points on the edge is debatable because points may only partially belong to an object, which means the "groundtruth" segmentation can in fact contain errors on the edge.

6.3.2. EVALUATION METRICS

The segmentation results are evaluated by the amount of overlap with the provided ground truth. To quantitatively demonstrate how well a segmentation result matches the ground truth, we use four metrics that have been applied widely in motion segmentation [26, 92, 177, 236, 247]. These evaluation metrics provide different interpretations of the quality of segmentation results.

Generally, the segmentation results are evaluated on a per object basis. Given a video sequence containing multiple moving objects, we evaluate the quality of each object in the segmentation results. Suppose there are N points in a set to be segmented, the groundtruth segmentation assigns the points to K^* clusters $\mathcal{C} = \{C_1, \ldots, C_{K^*}\}$, where each cluster C_k represents an object. The segmentation algorithm results in a partition $\mathcal{G} = \{G_1, \ldots, G_K\}$ of the N points, in which each group $G_{k'}$ represents a set of points that is estimated to be an object. We define a mapping g from $\mathcal{C}$ to $\mathcal{G} \cup \{\emptyset\}$, thus for every $C_k \in \mathcal{C}$, there is a $g(C_k) \in \mathcal{G} \cup \{\emptyset\}$ which represents the same object as C_k. Based on the value of K^* and K, there are three cases:

1. If $K^* = K$, g is bijective from $\mathcal{C}$ to $\mathcal{G}$;
2. If $K^* < K$, g is injective from $\mathcal{C}$ to $\mathcal{G}$;
3. If $K^* > K$, g is non-injective. We divide the $\mathcal{C}$ into two subsets: $\mathcal{C}_K$ containing K clusters and $\mathcal{C}_{K^*-K}$ containing the remaining $K^* - K$ clusters. The mapping g is bijective from $\mathcal{C}_K$ to $\mathcal{G}$. And for every cluster C_k in $\mathcal{C}_{K^*-K}$, $g(C_k) = \emptyset$.

However, the mapping g is as yet unknown given only $\mathcal{C}$ and $\mathcal{G}$. In case 1 and 2 where $K^* \leq K$, there are $K \cdot (K-1) \cdot (k-2) \cdots (K-K^*+1)$ ways of mapping g, and in case 3 there are $K^* \cdot (K^*-1) \cdot (K^*-2) \cdots (K^*-K+1)$ ways. We investigate all possible mappings, and choose the mapping that maximizes the segmentation accuracy. We define the **segmentation accuracy** as the ratio of correctly assigned points to all points in the set [236]. Given

a matching $\{(C_1, g(C_1)), \ldots, (C_{K^*}, g(C_{K^*}))\}$ of the groundtruth segmentation and the estimated segmentation

$$\text{accuracy} = \frac{1}{N}\sum_{i=1}^{K*}|C_i \cap g(C_i)| \tag{6.16}$$

where $|\cdot|$ denotes the size of a set.

The segmentation accuracy represents the fraction of points which are segmented correctly compared to all points in the segmentation. If the groundtruth segments all contain an almost equal number of points, the segmentation accuracy is often a reasonable measure of the segmentation quality. However, when the sizes of clusters are unbalanced, the segmentation accuracy can not reflect the quality of segmentation results well. For example, if an object contains 90% of the points, the segmentation accuracy will be 90% even if we assign all points to a single group. If there are two objects with an equal number of points, the segmentation accuracy is only 50% if we put all points in a single group, which we find reasonable since object is not segmented out.

For better understanding of the segmentation results, we also determine the **precision** and **recall** of each object; i.e., every matched pair $(C_k, g(C_k))$ [177]. Given a matched pair $(C_k, g(C_k))$, The precision P_k and recall R_k are defined as:

$$P_k = \frac{|C_k \cap g(C_k)|}{|g(C_k)|} \tag{6.17}$$

$$R_k = \frac{|C_k \cap g(C_k)|}{|C_k|} \tag{6.18}$$

If $g(C_k) = \emptyset$, we define $P_k = 1$ following the paper [177], and R_k is 0 in this case because $|g(C_k)| = 0$ and $|C_k| \neq 0$. The precision represents the fraction of points in a segmented group that are correctly assigned. A higher precision indicates that the obtained group contains less outliers. The recall measures the fraction of a groundtruth cluster covered by the estimated group. A higher recall indicates that more points of a cluster are correctly identified. For each video sequence, the average precision $\bar{P}$ and recall $\bar{R}$ are computed on a per object basis. The average **F-measure** is computed based on $\bar{P}$ and $\bar{R}$, as in [177]. The F-measure is a standard way of combining precision and recall. A higher F-measure indicates a better performance for the combination of precision and recall. It is given by:

$$F = \frac{2\bar{P}\bar{R}}{\bar{P} + \bar{R}} \tag{6.19}$$

Since our method is designed to also estimate the number of moving objects in the provided sequence, we additionally evaluate the results by a fifth evaluation method, which considers the number of estimated objects. Suppose the groundtruth number of objects is K^*, and the estimated number of objects is K in the segmentation result. Then we define the deviation of the estimated number ΔK as,

$$\Delta K = K - K^*; \tag{6.20}$$

Note that $\Delta K = 0$ needs not correspond with a correct segmentation. One object can be split into two, while two other objects are combined into one. This measure is very coarse.

$\Delta K < 0$ indicates that at there is at least some under segmentation and $\Delta K > 0$ indicates there is at least some over-segmentation. Only of the F-measure is high and clusters have similar sizes, ΔK indicates under and over segmentation.

To evaluate the computational efficiency, we also measured the **average computation time** of processing sequences with length of 31 frames. If a video contains more than 31 frames, we evaluated the computation time for processing the first 31 frames.

6.4. EXPERIMENTAL RESULTS

In Section 6.2, we proposed a 2D motion segmentation algorithm based on the EM algorithm and Bayesian updating, this algorithm is called AEM-b in this thesis. We also proposed an improved version that is called AEM-b$^+$, which measures the reliability of point assignments and compensates for the camera motion. Several parameters in AEM-b and AEM-b$^+$have to be set by the users, which is discussed in Subsection 6.4.1.

We have applied AEM-b and AEM-b$^+$to the provided datasets and evaluated the segmentation results. As mentioned in Chapter 4, the motion segmentation algorithm uses the motion data extracted from a video sequence as the input data. The quality and type of the motion data varies with the method used for motion estimation, as discussed in Chapter 5. Based on the quality of the input data, these datasets were divided into two categories, "flawless" data and "realistic" data. Using the "flawless" data, we evaluate the ability of dealing with videos containing various selected motions (i.e. rotation, translation, and a combination of them). Using "realistic" data, we investigate the performance of using different types of motion data that are obtained from a video, i.e. the optical flow, sparse point trajectories and dense point trajectories, see Chapter 5. The performance on "realistic" data reveals the ability of dealing with noisy data in practice.

For comparison, we also evaluated the proposed algorithms with 6 motion segmentation approaches from the literature. Four of the comparison methods are well-known trajectory clustering methods, i.e. the Local Subspace Affinity (LSA) method [266], random sampling and consensus (RANSAC) method [85], generalized principal component analysis (GPCA) [247], and the sparse subspace clustering (SSC) [79] method. These methods are specifically designed for segmenting complete trajectories, which requires that the points are detected in all frames of a sequence. The number of objects is required as an input for these methods. Implementation codes of these methods were downloaded from the site of Hopkins 155 [1]. . We also chose two approaches for motion-based video segmentation, i.e. the method proposed by Fragkiadaki, et al. [92] and the method proposed by Ochs, et al. [177]. These methods are proposed for "realistic" trajectories obtained by the feature tracking approaches as [46, 226]. These methods are more flexible in dealing with incomplete trajectories, and can determine the number of objects automatically. The code of Fragkiadaki's method is available on the author's web page [1]. The results of Ochs' method are reported in [177].

In the following subsections, Subsection 6.4.1 discusses the parameter configuring of AEM-b and AEM-b$^+$. The evaluation of segmentation results on "flawless" data and "realistic" data are given in Subsections 6.4.2 and 6.4.3 respectively.

[1] http://www.vision.jhu.edu/data/hopkins155/

[1] https://www.cs.cmu.edu/~katef/videoseg.html

All experiments were done on a standard PC: CPU: Intel Core i5-2400 3.10GHz, OS: Window 7 Enterprise, and using Matlab 2015a 64bit.

6.4.1. PARAMETER CONFIGURATION

In this experiment, we investigated the proper parameter settings of the improved algorithm AEM-b$^+$. we used the "flawless" data provided by the Hopkins 155 dataset. The Hopkins 155 data set provides videos that contain different conditions of motions, together with "flawless" motion data. With the flawless data, we can investigate the influences of parameters on segmentation results, to avoid the effects of noise in the motion data. As discussed in Subsection 6.3.2, segmentation accuracy is sufficient to evaluate the segmentation results because all objects contain similar numbers of points in a sequence. Thus we compared the average segmentation accuracy in this experiment.

As discussed in Section 6.2, parameters in the algorithm are categorized into three classes: 2 for the splitting of groups, 1 for the reliability measure, and 2 for the camera compensation. The following subsections discuss the settings of these parameters.

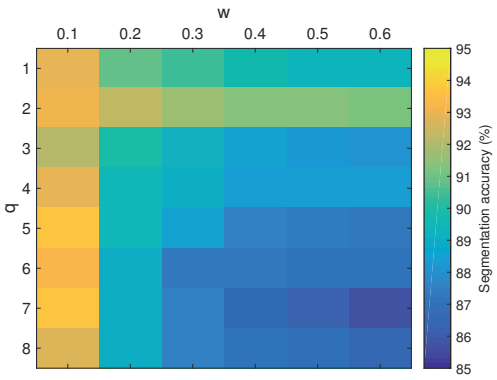


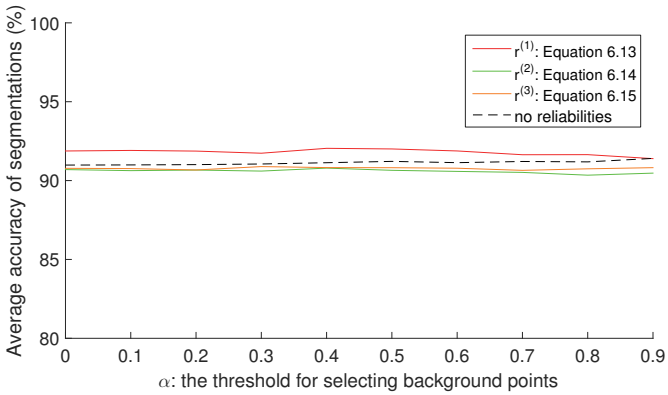
Figure 6.2: Segmentation accuracy of AEM-b on the Hopkins 155 dataset using different settings of q and w for splitting.

PARAMETER SETTING OF AEM-B

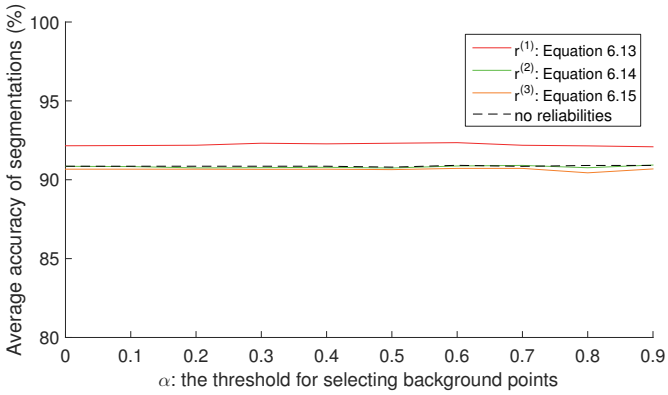
As discussed in Subsection 6.2.2, we have to determine which group to split when applying the divisive segmentation algorithm (AEM-b). A segmentation is evaluated by computing the sum of the value of q largest errors of each group, and the q needs to be set. Moreover, a threshold w is used to determine when to terminate the division procedure. We varied q from 1 to 8, and w was chosen in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. The results are shown in Figure 6.2. From this figure, the best performance is obtained when $w = 0.1$, and no big differences are observed for different values of q. Note that an object point on the edge may have big error, a small q can increase the possibility of wrong splitting. So we chose $q = 5$ and $w = 0.1$ as the default setting for AEM-b .

PARAMETER SETTING OF AEM-B$^+$

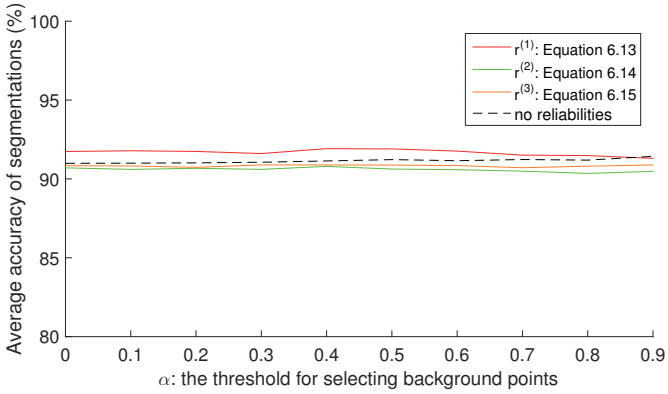
We investigate the parameter setting of AEM-b$^+$, including the splitting parameters q and w and the parameters for computing the reliabilities and compensating the camera mo-



(a) *Option* 1: compensate the camera motion in the inner loop of EM segmentation algorithm (between line 2 and 3 in Algorithm 1)



(b) *Option* 2: compensate the camera motion right after EM loop in the divisive motion segmentation algorithm (between line 7 and 8 in Algorithm 2).



(c) *Option* 3: compensate the camera motion after the splitting step in the divisive motion segmentation algorithm (between line 11 and 12 in Algorithm 2).

Figure 6.3: (a), (b) and (c) respectively show the performance of compensating camera motion in three different ways. The curves reflect the variation of segmentation accuracy when the threshold value changes from 0 to 0.9. The colors of curves represent different reliability functions used. The dashed line represents the case of no reliability computation.

tion. As in Subsections 6.2.4 and 6.2.5, we present three ways for reliability computation and three options for camera motion compensation. In the process of compensating the camera motion, a threshold α is used to eliminate the background points with low reliabilities when computing the camera motion.

Based on the default setting of splitting parameters, i.e. $w = 0.1$ and $q = 5$, we investigate the influences of applying different settings for reliability computing and camera compensation. Using one of the three options for camera motion compensation given in Subsection 6.2.5, we vary the reliability computation by using Equations (6.13) to (6.15), We also evaluate the performance when no reliability is computing. To select a proper value of α for camera motion estimation, we investigated the performances when α changes between 0 and 0.9. The results are illustrated in Figure 6.3. Of all results in this figure, the best performance is obtained when using Equation (6.13) for reliability computation and compensating camera motion after the EM loop (*Options* 2 in Subsection 6.2.5), which are chosen as the default setting of AEM-b$^+$ in this thesis. The influence of α value is insignificant, but generally a value between 0.3 and 0.6 is better than others. We choose the value of 0.5 as the default setting of α for AEM-b$^+$.

AEM-b$^+$ addresses to improve the performance of segmenting moving objects from videos captured by moving cameras. We additionally investigate its performance on videos under different camera motions. The Hopkins 155 dataset provides some videos captured by a handheld camera under controlled conditions, in the category called "checkerboard". We divided the videos in the "checkerboard" category into 4 sub-categories based on the type of camera motion, as shown in Table 6.1. We compared the performance of AEM-b and AEM-b$^+$ on videos from the four sub-categories, and the results are shown in Table 6.1. From this table, AEM-b$^+$ and AEM-b perform almost equally on videos with static camera, while AEM-b$^+$ performs better than AEM-b in case that the camera is moving.

checkerboard	AEM-b	AEM-b$^+$
Static camera: 20 sequences	96.15	96.05
Translating camera: 20 sequences	92.67	94.91
Rotating camera: 24 sequences	90.78	92.56
Both rotating and translating camera: 40 sequences	91.15	92.17

Table 6.1: Average segmentation accuracy (%) of on the "checkerboard" category in Hopkins 155 dataset.

We also investigate the influences of changing w and q for AEM-b$^+$, when the parameters for reliability measure and camera compensation were fixed as the default value. The results are shown in Figure 6.4. By comparing Figure 6.4 with Figure 6.2, the AEM-b$^+$ generally improves the performance of AEM-b for different values of w and q, and the best performance of AEM-b$^+$ and AEM-b are both achieved when $w = 0.1$. We still choose $w = 0.1$ and $q = 5$ for AEM-b$^+$.

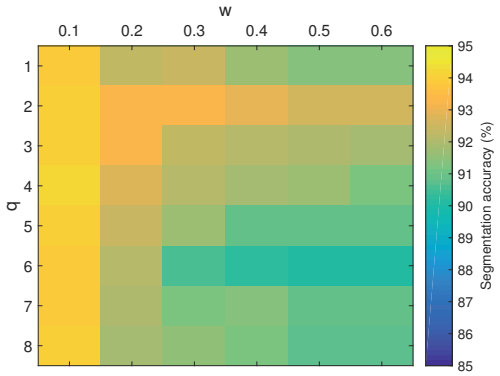


Figure 6.4: Segmentation accuracy of AEM-b$^+$ using different settings of q and w for splitting.

6.4.2. SEGMENTATION ON FLAWLESS DATA

In a next experiment, we compared AEM-b and AEM-b$^+$ using the "flawless" data provided by the Hopkins 155 dataset. The provided motion data is a set of complete trajectories for each video sequence, which means that each feature point is tracked in every frame of the sequence. A trajectory of a feature point over F frames is represented as a sequence of F 2D-coordinates. Trajectories of N detected feature points are represented by a matrix $\boldsymbol{X} \in \mathbb{R}^{2\times F\times N}$. In each sequences, there are 2 or 3 moving objects. Each motion corresponds to a moving object in the video. The number of frames F is 31 and the number of feature points N is 296 on average for the provided sequences. More details of these videos are given in Section 4.2.

Following [236], we evaluate the accuracy of the segmentation results per sequence. Since all objects in a video are represented by more or less the same number of feature points in the given data, the segmentation accuracy is a reasonable measure of the quality of segmentation results, as mentioned in Subsection 6.3.2. We compare the algorithms AEM-b and AEM-b$^+$ with the four trajectory clustering algorithms, LSA [266], RANSAC [85], GPCA [247] and SSC [79], and with Fragkiadaki's method (FM) [92]. The algorithms LSA, RANSAC, GPCA and SSC are specific for dealing with complete trajectories, and require the number of objects as an input. Fragkiadaki's method (FM) estimates the number of individually moving objects. The results are given in Table 6.2.

We also evaluated the average computation time of a sequence, and the results are shown in Table 6.3. Figure 6.5 illustrates some of the segmentation results, by visualizing the point trajectories in different groups by different colors. Since our algorithms and FM can estimate the number of objects, for these we evaluated the error ΔK of the estimated number of objects, shown in Figure 6.6.

We can draw the following conclusions from the results:

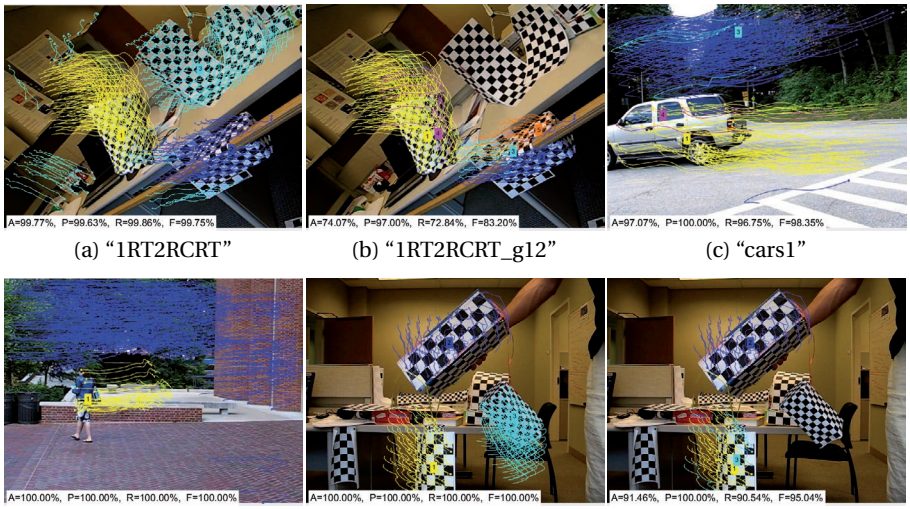
- SSC outperforms all other methods on each category of the videos from Hopkins 155 dataset, as the segmentation accuracy in each category is above 97% for all categories. However, SSC requires the number of objects to be specified. AEM-b and AEM-b$^+$ do not, and these are better than LSA, RANSAC and GPCA in general, since the average segmentation accuracies of them are lower than for AEM-b or AEM-b$^+$.

		LSA	RANSAC	GPCA	SSC	FM	AEM-b	AEM-b$^+$
2 objects	Checkerboard(78)	93.91	92.01	79.11	**98.37**	73.13	91.89	93.52
	Traffic(31)	**98.62**	92.14	73.2	**99.42**	75.09	**97.71**	**98.18**
	Others(11)	**96.93**	90.45	72.52	**98.19**	93.33	92.44	**95.37**
	All	**95.37**	91.9	76.98	**98.73**	75.51	93.44	94.44
3 objects	Checkerboard(26)	68.06	72.23	80.4	**97.4**	72.57	93.76	93.50
	Traffic(7)	80.2	88.28	53.1	**99.2**	74.72	**97.78**	**98.01**
	Others(2)	83.19	76.98	78.9	**98.9**	**100**	90.86	94.62
	All	71.35	75.71	74.85	**97.85**	74.57	94.40	94.47
Average of all videos		89.98	88.24	76.50	**98.45**	75.30	93.66	94.80

Table 6.2: Segmentation accuracy (%) on Hopkins 155 motion data. The number in brackets means the number of video sequences in this category. The values of segmentation accuracies ⩾95 are printed in bold.

LSA	RANSAC	GPCA	SSC	FM	AEM-b	AEM-b$^+$
4.32s	0.09s	0.14s	3.8s	1.20s	0.31s	2.6s

Table 6.3: Average computation time (seconds per sequence) on the Hopkins 155 dataset. The number of processed points is 296 on average.



(a) "1RT2RCRT" (b) "1RT2RCRT_g12" (c) "cars1"

(d) "people1" (e) "articulated" (f) "articulated_g12"

Figure 6.5: Visualized segmentation results: 6 examples are shown by their names in the dataset. (a) and (b) are from the same video in "checkerboard" category: (a) is composed by all 3 objects (a cuboid, a frustum cone and the background) from the original video and (b) is composed by choosing the cuboid and the frustum cone in (a); (c) is from the category of "traffic"; (d) is from the category of "others"; (e) and (f) are originated from the same video in the category of "others".

- Both AEM-b and AEM-b$^+$ perform well when the objects are undergoing translations, as either of them achieves a segmentation accuracy around 98% on the "traffic" category. Such degree of segmentation accuracy implies that the segmented objects can be well recognized (see Figure 6.5c). Note that some traffic videos are taken by a shaking camera, which means AEM-b and AEM-b$^+$ can handle small camera motions. Moreover, as observed from the result of the walking people videos (see Figure 6.5d), AEM-b and AEM-b$^+$ can handle an articulated object when the articulated parts move as an integral unit.

  When the object motion is more complicated, as in videos from the categories "checkerboard" and "others", the segmentation accuracy of AEM-b and AEM-b$^+$ drops, which is not unexpected. AEM-b has a performance between 90% and 95%, and AEM-b$^+$ has a performance between 93% and 96%.

- In general, AEM-b$^+$ has better performance than AEM-b . The improvement of AEM-b$^+$ in the category of "others" and in the category of "checkerboard" with 2 objects, is more significant than that in the "traffic" videos. It reveals that AEM-b$^+$ is better at handling complex motions than AEM-b .

  For the videos containing 3 objects in the "checkerboard" category, the segmentation accuracy of AEM-b$^+$ is 0.26% lower than AEM-b . This is likely due to compensation of the camera motion which can deteriorate the performance when the camera is actually static, as discussed in Subsection 6.4.1. However, note that a decrease of 0.26% means that AEM-b$^+$ has 1.14 more points that are misclassified than AEM-b for a sequence contains 437 points (the average value over these videos). Such a small decrease does not affect the quality of object representations much.

- Our method outperforms FM using the Hopkins 155 dataset. FM is proposed specially for segmenting the dense trajectories obtained by [92]. This might explain the low average accuracy in this experiment. The experiment with dense trajectories, reported in Subsection 6.4.3, shows a better performance for FM.

- AEM-b is almost 10 times faster than SSC, while AEM-b$^+$ is 1.5 times faster than SSC. Compared to the fastest method, i.e. RANSAC, both AEM-b and AEM-b$^+$ can achieve more than 5 percentage points higher segmentation accuracy on average.

- AEM-b and AEM-b$^+$ both have a chance above 50% to correctly estimate the number of objects, i.e. $\Delta K = 0$, while chance of correctly estimation for FM is only 10%. The over-segmentation cases of AEM-b and AEM-b$^+$ mostly happen for $\Delta K = 1$, while FM often produce over-segemntations with $\Delta K \geqslant 2$. We can conclude that AEM-b and AEM-b$^+$ are less likely to produce over-segmentations than FM, using the "flawless" motion data.

6.4.3. SEGMENTATION ON REALISTIC DATA

In the following set of experiments, we evaluated the results when using motion data extracted from video sequences in Chapter 5, which contains the videos in Robocup 2014, CDNet 2014 and FBMS-59 datasets. Such data is corrupted by noise, such as incorrect

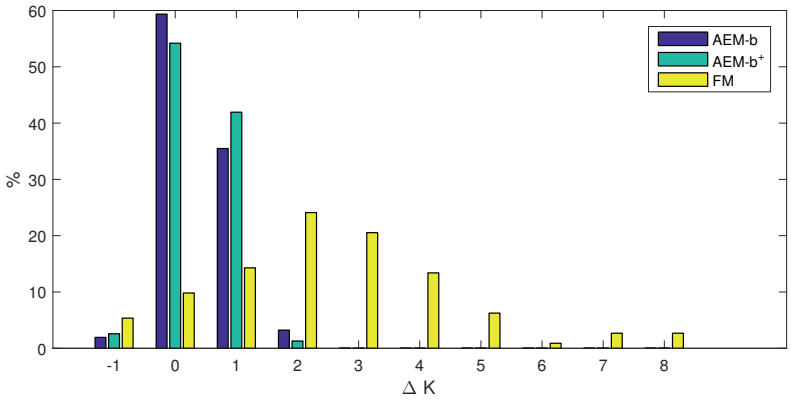


Figure 6.6: The occurrences (%) of the mismatches in the estimated number of objects ΔK, for the methods AEM-b , AEM-b$^+$and FM, when using the "flawless" motion data provided by the Hopkins155 dataset.

trajectories and erroneous point motions, and also suffers from missing points. Moreover, the point trajectories in a video sequence might have unequal lengths. We have considered three types of motion data in Chapter 5, consisting of optical flows, sparse point trajectories and dense point trajectories. The segmentation algorithms are now applied to these three types of motion data. We evaluate the performance with respect to the following four aspects.

Firstly, while most of the existing approaches in the literature are proposed for dealing with either two-frame motion fields [57, 64, 140, 192, 193, 259], or with point trajectories over multiple frames [79, 92, 156, 177, 247, 266], the methods AEM-b and AEM-b$^+$combine the analysis of two-frame motion vectors with the analysis of long-term trajectories in a video sequence. The segmentation accuracy is expected to increase when processing more frame pairs while the scene objects do not change. Thus we investigated how the segmentation accuracy varies with the number of frame indices in processing a video sequence.

Secondly, since the quality of motion data is affected by the frame rate (see Chapter 5), we evaluated the effect of frame rates on the segmentation results. For comparison, we extracted sub-sequences with different frame rates from a given video, by sub-sampling the frames. Let $\{f_0, f_1, \ldots, f_T\}$ denote a video of $T+1$ frames, we define a sample factor t to denote the time step between two sampled frames. For the original video, t obviously equals 1. For any integer $t > 1$, we can formulate a new sequence $S_k = \{f_0, f_t, \ldots, f_{t \cdot k}\}$ by repeatably jumping over $t-1$ frames in the original sequence. The displacements of points between two successive frames become larger and the length of the subsequence becomes shorter when t increases. In our experiments, the extracted sub-sequences have the same first and last frame as the original sequence. This allows us to compare the segmentation accuracies for the last frame. In this experiment, we chose frame f_0 and f_{30} to be the first and last frame for every sequence in test. The value of t is chosen from $[1, 2, 3, 5, 10]$, and k has the corresponding values $[30, 15, 10, 6, 3]$. We compared the segmentation accuracies to illustrate the influences of frame rates on the segmentation results.

Since these two evaluations above require the groundtruth segmentation for every frame in a video sequence, we only used motion data from Robocup 2014 and CDNet 2014

dataset.

Thirdly, we compared the proposed algorithms AEM-b and AEM-b$^+$ to SSC, FM and Ochs' method (OM), by measuring the average segmentation accuracy, precision, recall and F-measure. We only choose SSC as a representative of trajectory clustering algorithms in comparison, because the experiments in Subsection 6.4.2 revealed that our methods achieve better performance than LSA, RANSAC and GPCA on flawless motion data. We evaluated the performance on all videos from the Robocup 2014, CDNet 2014, and FBMS-59 datasets.

Fourthly, we compared the computation time of all applied methods. We evaluated the computation time of processing a sequence of 31 frames, together with the amount of processed data, to give an intuitive measure of the computational efficiency.

The segmentation results of the three types of data, i.e. optical flows, sparse point trajectories and dense point trajectories, are evaluated in the following subsubsections. Trajectory clustering methods (SSC, RANSAC, GPCA, LSA), FM and OM can not process optical flows in the video sequence, as they require that the motion data to be consistent trajectories over the video frames. We do not compare with them when using optical flows as the input.

SEGMENTATION ON OPTICAL FLOWS

In these experiment, the motion data is represented as a sequence of two-frame motion vectors, as obtained by the optical flow algorithm introduced in Section 5.2. We evaluate the performance of our methods AEM-b and AEM-b$^+$ on this type of data.

# of objects	Sequence $\{f_0, f_t, f_{2t}, \ldots, f_{30}\}$, for $t =$				
	1	2	3	5	10
fixed	81.6	82.0	82.2	78.9	72.0
increasing	77.1	78.6	72.8	70.3	58.8
decreasing	78.3	76.9	73.0	62.1	55.6
all	80.4	80.5	79.4	75.3	67.0

Table 6.4: The segmentation accuracy (%) of sequences with different frame rates (indicated by sample factor t) and the trends for the number of objects present in the sequence, using per-pixel motion vectors obtained from Robocup+CDNet.

Firstly, we illustrate the variations of segmentation accuracy with respect to the number of frames that has been processed, for segmenting a sequence of 31 frames. As shown in Figure 6.7, the curves show how the segmentation accuracy varies over time, i.e. with respect to the index of frames. The results show that the segmentation accuracy is fluctuating but generally slightly increasing over time. For the sequences with varying number of objects, the accuracy fluctuates near the frames where the number of objects changes.

Secondly, we investigated the effects of frame rates on the segmentation results. The segmentation accuracy of sub-sequences $\{f_0, f_t, \ldots, f_{30}\}$ with different sub-sampling factors t is shown Table 6.4. A larger t represents a lower frame rate, which means larger displacements between two frames. From this table, the segmentation accuracy generally decreases when t increases. It is no surprise that the optical flows are more accurate when

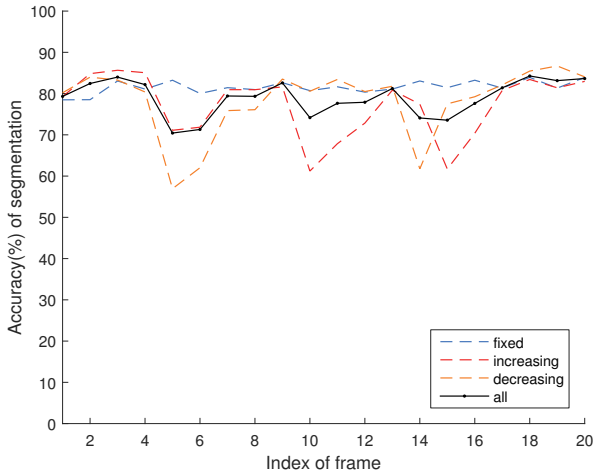


Figure 6.7: The segmentation accuracy of segmenting optical flows varies w.r.t. the index of the frames, using the Robocup 2014 and CDNet 2014 datasets. The curves in different colors show results on videos containing fixed, increased and decreased numbers of objects respectively.

the displacements between two frames are small, as shown in Subsection 5.4.1. The influence of t is more significant when dealing with dynamic scenes containing a changing number of objects, than dealing with videos with fixed number of objects. For small values of t (1 or 2), the differences in performance is not significant – as in some cases, the segmentation accuracy even increases by 1 percentage point when t increases from 1 to 2. Since the frame rate of the original video is 30 fps on average, we can conclude that a frame rate between 15 and 30 is preferred for the proposed segmentation algorithm using optical flows.

SEGMENTATION ON SPARSE POINT TRAJECTORIES

In this experiment, the motion data is represented as a set of sparse point trajectories, which is obtained by the SIFT tracking approach described in Section 5.3. These trajectories can have different lengths, i.e. points can be present only in a subset of the frames. The average number of tracked points in each frame is about 366 for a video with resolution of 640×320. For a video sequence of 31 frames, the number of extracted trajectories is 1429 on average, and 145 of them are full trajectories. More details of the sparse point trajectories can be found in Chapter 5.

Firstly, we evaluated the per-frame segmentation accuracy over a sequence of 31 frames, as illustrated in Figure 6.8. The segmentation accuracy generally increases over time, except at the frames where the number objects changes.

Secondly, we compared the segmentation accuracies of sequences with different frame rates and containing different numbers of objects, as shown in Table 6.5. As the segmentation accuracy stays around 85% when t is less than 5, sparse trajectories are more robust than optical flows with respect to a small t (≤ 5). However, the performance drops by 10% when t increases to 10. Since the frame rate of the original video is 30 fps on average, the proposed segmentation algorithm can achieve stable performance when the frame rate is

# of objects	Extracted sequence $\{f_0, f_t, \ldots, f_{30}\}$, for $t =$				
	1	2	3	5	10
fixed	86.2	86.7	85.6	85.5	78.4
increasing	81.2	83.2	83.5	80.1	66.9
decreasing	81.2	80.9	82.1	77.9	63.9
all	84.3	85.0	84.6	83.1	73.6

Table 6.5: The segmentation accuracy (%) w.r.t. sequences with different frame rates (indicated by sample factor t) and the trends for the number of objects present in the sequence, using sparse trajectories extracted from Robocup+CDNet.

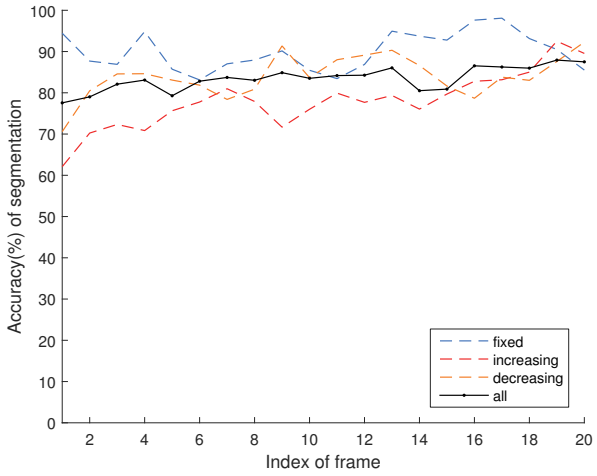


Figure 6.8: The segmentation accuracy of segmenting sparse trajectories varies w.r.t. the index of the frames, using the Robocup 2014 and CDNet 2014 datasets. The curves in different colors show results on videos containing fixed, increased and decreased numbers of objects respectively.

higher than 10 fps, using sparse point trajectories.

Thirdly, we compared our methods with SSC and FM, by measuring the segmentation accuracy, precision, recall and F-measure. The results are shown in Table 6.6 and Table 6.7, for the Robocup 2014 and CDNet 2014 dataset and the FBMS-59 dataset, respectively. Generally, AEM-b and AEM-b$^+$have higher accuracy, precision, recall and F-measure than SSC and FM. Only for specific data (the FBMS-59 dataset), FM has about 0.8% higher accuracy than AEM-b and AEM-b$^+$.

We also computed the occurrences (%) of different deviations ΔK in estimating the number of objects. From the results shown in Figure 6.9, AEM-b and AEM-b$^+$both are comparable to correctly estimate the number of objects, but clearly outperform FM. FM is more likely to generate over-segmentation than AEM-b and AEM-b$^+$.

	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
SSC	74.38	60.05	50.00	54.56
FM	77.20	62.52	51.04	56.19
AEM-b	84.27	**73,75**	**57.36**	**64.53**
AEM-b$^+$	**85.67**	71.05	56.07	62.11

Table 6.6: Segmentation accuracy, precision, recall and F-measure on sparse trajectories obtained from Robocup+CDNet dataset.

	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
SSC	64.25	55.15	60.44	57.68
FM	**85.63**	68.89	53.96	60.51
AEM-b	84.88	**71.48**	60.49	65.53
AEM-b$^+$	84.84	70.10	**63.06**	**66.39**

Table 6.7: Segmentation accuracy, precision, recall and F-measure on sparse trajectories obtained from FBMS-59 dataset.

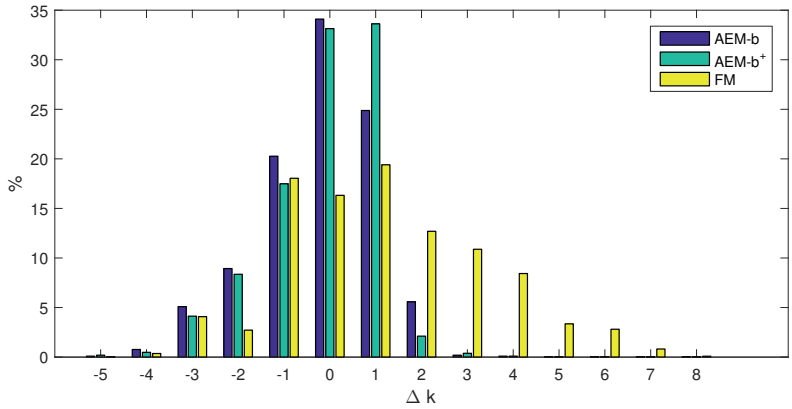


Figure 6.9: The occurrences (%) of mismatches in the estimated number of objects ΔK, in the segmentation results for the methods AEM-b , AEM-b$^+$ and FM, when using the sparse trajectories.

The computation time is shown in Table 6.8. We evaluated the average computation time for processing a sequence with 31 frames. FM can segment 1305 trajectories for each sequence, while SSC can only deal with the 145 full trajectories. As AEM-b and AEM-b$^+$ process part of the trajectories present in every frame pair, the amount of data is counted by the number of processed points in one sequence. AEM-b is 6.5 times faster than AEM-b$^+$, and 1.5 times faster than FM. Considering that SSC processed the least number of trajectories, it is also slower than AEM-b and AEM-b$^+$.

Method	SSC	FM	AEM-b	AEM-b$^+$
computation time (s)	0.89	1.83	1.21	7.84
processed data	145 trajectories	1305 trajectories	366*30 points	366*30 points

Table 6.8: Average computation time (seconds) per sequence of 31 frames, for segmenting sparse trajectories.

SEGMENTATION ON DENSE POINT TRAJECTORIES

In this experiment, the motion data is represented as a set of dense point trajectories extracted by the dense tracking method in Subsection 5.4.3. Compared to the sparse trajectories, more feature points are tracked in each frame. Take a video of a resolution of 640×320 for example, then the number extracted trajectories is around 1300 for a sequence of 31 frames. More information of this type of motion data is given in Chapter 5.

Firstly, the per-frame segmentation accuracies are illustrated in Figure 6.10. From this figure, using dense trajectories has better performance for the first frame pair than using sparse trajectories, with an increase of more than 10 percentage points. In general, the increasing trend of the average segmentation accuracy is not as significant as in the results of sparse trajectories (Figure 6.8), but this is also due to the accuracy being higher. For videos containing a varying number of objects, the segmentation accuracy drops when the number of object changes, and then increases after several frames.

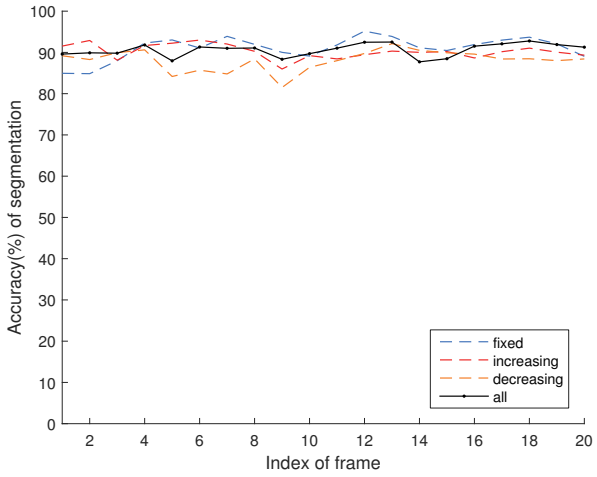


Figure 6.10: The segmentation accuracy of segmenting dense trajectories varies w.r.t. the index of the frames, using the Robocup 2014 and CDNet 2014 datasets. The curves in different colors show results on videos containing fixed, increased and decreased numbers of objects respectively.

Secondly, Table 6.9 compares the performance on sequences with different frame rates, also considering the number of moving objects present in a video. Although the segmentation accuracy decreases when t increases, this is less than 4%, which is smaller than when using sparse trajectories and optical flows. In general, using dense trajectories achieves

# of objects	Sequence $\{f_0, f_t, \ldots, f_{30}\}$, for $t =$				
	1	2	3	5	10
fixed	92.0	91.4	91.5	91.8	91.2
increasing	88.1	87.8	88.6	86.9	83.4
decreasing	85.7	85.4	85.3	84.0	81.5
all	90.5	90.1	90.2	89.9	88.0

Table 6.9: The segmentation accuracy (%) of sequences w.r.t. different frame rates (sample factor t) and number of objects present in the sequence, using dense trajectories extracted from Robocup+CDNet.

higher segmentation accuracy than using sparse trajectories and optical flows. But the preferable frame rate is at least 6 fps for dense trajectories segmentation, for which the segmentation accuracy remains above 90%.

Thirdly, Tables 6.10 and 6.11 show the segmentation accuracy, precision, recall and F-measure, using videos in the Robocup 2014 + CDNet 2014 datasets and the FBMS-59 dataset respectively. Here, AEM-b and AEM-b$^+$ outperform SSC and FM in all datasets.

Table 6.11 also shows the OM results for the FBMS-59 dataset, reported in the [177] OM has a similar precision, recall and F-measure as AEM-b and AEM-b$^+$. AEM-b$^+$ has better performance than AEM-b in the Robocup 2014 + CDNet 2014 datasets. In contrast, it has a slightly lower recall and F-measure than AEM-b on the FBMS-59 dataset.

The evaluation of identifying the number of objects is shown in Figure 6.11. We measured the differences ΔK between the estimated number and the groundtruth number of objects in a video. The performance of AEM-b and AEM-b$^+$ are comparable, while both outperform FM in estimating the correct number of objects. FM is more likely to produce over-segmentations than AEM-b and AEM-b$^+$.

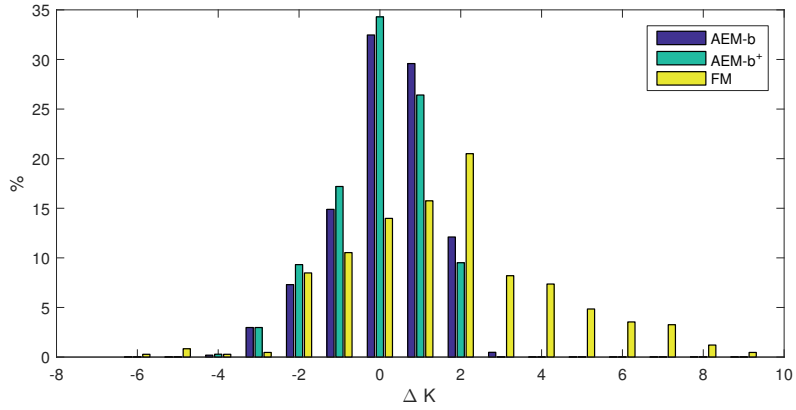


Figure 6.11: The occurrences (%) of different mismatches in the estimated number of objects ΔK, in the segmentation results for the methods AEM-b , AEM-b$^+$ and FM, when using the dense trajectories.

Finally, the computation time per sequence is shown in Table 6.12. FM processes about

	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
SSC	78.91	74.80	39.24	51.47
FM	85.63	68.89	53.96	60.51
AEM-b	90.51	72.73	**64.53**	68.39
AEM-b$^+$	**92.53**	**77.68**	62.42	**69.22**

Table 6.10: Segmentation accuracy, precision, recall and F-measure on dense trajectories obtained from Robocup+CDNet dataset.

	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
SSC	87.92	81.35	40.74	54.29
FM	**86.90**	79.37	56.65	66.11
OM	-	**82.45**	61.70	**70.56**
AEM-b	85.91	79.86	**62.40**	70.06
AEM-b$^+$	86.88	81.75	61.57	70.24

Table 6.11: Segmentation accuracy, precision, recall and F-measure on dense trajectories obtained from FBMS-59 dataset, the accuracy of OM is not valid.

Method	SSC	FM	OM(*)	AEM-b	AEM-b$^+$
Computation time (s)	20.7	5.61	0.8	2.25	12.70
processed data	945 trajectories	13744 trajectories	2127 trajectories	1050*30 points	1050*30 points

Table 6.12: Average computation time (seconds) per sequence of 31 frames, for segmenting dense trajectories.(* The results of OM is reported in [177], which is produced in a different environment.)

13744 trajectories for a sequence of 31 frames. SSC can only deal with the 945 complete trajectories. AEM-b and AEM-b$^+$processes about 1050 points per frame pair, and in total 1050*30 points per sequence. AEM-b is 5.6 times faster than AEM-b$^+$and 2.5 times faster than the FM. SSC is 1.2 times slower than AEM-b$^+$, even though it processes only 7% of the extracted trajectories. Based on the computation times reported in [177], OM is around 3 times faster than AEM-b . However, OM is executed in a different environment that uses GPUs.

CONCLUSION

From these experimental results over three types of motion data, we can draw the following conclusions

- Segmentation based on point trajectories outperforms the optical flow based method in two ways. Firstly the point trajectories contain less points than the optical flows. Therefore, the segmentation of trajectories requires less computational resources than optical flow. Secondly, the segmentation of point trajectories always achieves higher segmentation accuracy than optical flow.

- Segmentation of dense point trajectories is more stable than segmentation of sparse point trajectories, i.e. the variations of segmentation accuracy with respect to the frame index are smaller, comparing Figure 6.10 to Figure 6.8. Moreover, the segmentation accuracy of dense trajectories is generally higher than that of sparse trajectories. This might because the dense tracking method tracks more points and the obtained points trajectories are more accurate than those obtained by sparse tracking methods. As a result, the estimated motion models based on dense trajectories are more accurate.

- AEM-b$^+$ is a little more powerful than AEM-b in dealing with videos that are captured by a moving camera (see Table 6.2). In general, the improvement of AEM-b$^+$ is statistically insignificant, see Tables 6.10 and 6.11. In some cases, compensating the camera motion even worsens the performance, see Table 6.6. Moreover, the computation time of AEM-b$^+$ is around 6 times higher than that of AEM-b , see Tables 6.8 and 6.12.

- While SSC is a powerful method in dealing with "flawless" motion data, it is much poorer than our methods when processing the more "realistic" data that contain errors. Moreover, our method is more flexible in dealing with a whole set of motion data, while SSC can only deal with a subset of full trajectories.

- Comparing to two state-of-the-art video segmentation methods, i.e. FM and OM, our methods show competitive performance over all datasets. As the F-measure reflects the balance between precision and recall, it provides a qualitative justification of the extracted objects. AEM-b and AEM-b$^+$ always achieve a higher F-measure than FM, which means that the objects obtained by AEM-b and AEM-b$^+$ cover more points of the groundtruth objects. Moreover, AEM-b and AEM-b$^+$ have a lower chance of suffering from the over-segmentation problem than FM. In general, OM just outperforms our methods from the results. But here we must note that the F-measures of AEM-b and AEM-b$^+$ are quite close to that of OM, with a difference within 0.5%. And the result here of OM is not valid for all datasets.

6.5. CONCLUSION

In this chapter, we addressed the second research question: how to segment out the moving objects from a video sequence based on the extracted motion data? Assuming that the point motions in successive images can be modeled by a 2D affine transformation, a motion segmentation algorithm based on classification EM algorithm and Bayesian updating is proposed, which is named by AEM-b . AEM-b is able to segment a video sequence, by considering the motion information since the beginning of the sequence. Moreover, an improved version of AEM-b is proposed with the name of AEM-b$^+$, by compensating the camera movement and measuring the reliabilities of segmentation results during processing the sequence. Since the performance of proposed algorithm depends on several parameters, we discussed the best parameter settings of the proposed algorithm using a benchmark dataset. Experiments also show that, with the chosen parameters, this algorithm performs well on other datasets compared to the reference methods as worked out in Section 6.4.

Most of the existing approaches prefer to deal with a specific type of motion data, either the sparse trajectories or the dense trajectories, and they are sensitive to noise and errors in the motion data. Our approach shows high flexibility in dealing with optical flows, sparse and dense trajectories. And the performance of our method stays in the upper level, no matter the motion data is flawless or corrupted by noise and errors.

Moreover, our method is able to deal with arbitrary numbers of frames regardless of the computational limit. The other methods usually require that the sequences are shorter than a certain length. Our method also has the ability to handle videos in which the number of objects varies, and on this aspect they are superior to the compared methods that have the same functionality in the experiments.

The two versions of the proposed algorithm, i.e. AEM-b and AEM-b$^+$, both have advantages and disadvantages. The original algorithm (AEM-b) is faster than AEM-b$^+$, which computes the segmentation reliabilities and compensates for the camera motion. AEM-b$^+$is better in dealing with videos with significant camera motion than AEM-b. Moreover, AEM-b$^+$requires the motion data to contain background points for obtaining a better performance than AEM-b. Both algorithms are good in dealing with translations, like cars moving and people walking.

The proposed segmentation algorithm is based on 2D motion coherences in frames, which neglects the 3D motion and structural coherences of the objects in real world. As mentioned in Section 6.1, 2D based motion segmentation often suffers from the over-segmentation problem, especially when segmenting two-frame motion vectors. We used a Bayesian updating procedure to obtain long-term motion coherences of a series of 2D motions, which reduces the over-segmentation problem in some situations. However, the results reveal that the proposed segmentation algorithms tend to produce over-segmented objects in some cases.

One possible way to address the over-segmentation problem is to analyze the 3D motion coherences of the given 2D points motions. In the next chapter, we will discuss motion segmentation based on 3D motion models.

7

3D MOTION SEGMENTATION

In Chapter 6, a motion segmentation algorithm is present based on the motion of 2D points in a video sequence. Given a set of 2D points that belong to the same object, their movements between two successive frames in a video is modeled by a 2D affine transformation. This algorithm performs well as long as the rotation around the x (or y) axis is limited. However, this method often fails to segment out an object if the rotation about the x (or y) axis is significant.

In this chapter, we investigate how to segment the projected 2D image points based on the motion consistency of the corresponding points in the 3D space, since the 2D image motion is the projection of 3D motion in the real world. Section 7.1 briefly introduces the issue of this chapter. In Section 7.2, we analyze the motion of a set of 2D points between two image frames based on matrix factorization technique and derive two theorems for retrieving the 3D consistency and the 3D rigid body motion from 2D motions. In Section 7.3, we applied the proposed theorems for 3D consistency measure, and to recover the 3D motion of a rigid 3D body. Section 7.4 concludes this chapter.

7.1. INTRODUCTION

The motion data implied in two successive images is acquired by tracking some 2D points in these images, see Chapter 5. These 2D image points are projections of corresponding points in the 3D world. As points from the same object move consistently in the 3D world, the corresponding 2D image points preserve the consistency in the 2D video frames if their movements are small enough (see Subsection 3.3.3). In Chapter 6, we investigated motion segmentation based on modeling the object motions in the 2D image plane as 2D affine transformations. However, the 2D motion of an object in the video frames can not be simply represented by one 2D affine model in many cases, due to the perspective effects, depth discontinuities, occlusions, transparent motions, etc.[155] As a result, a 3D motion in the real world might be broken into different 2D motions in the 2D images, and segmentation based on 2D models tends to segment one object into multiple segments, which is called over-segmentation. Experiments reveal that the 2D motion segmentation algorithm

proposed in Chapter 6 is not good at dealing with videos in which the objects rotate significantly about the x (or y) axis.

To address the over-segmentation, we investigate 3D motion segmentation in this chapter. That is, given the trajectories of 2D points in a video sequence, we intend to segment out the points that move consistent with a rigid 3D motion in the world space. In literature, the 3D motion segmentation approaches often capture the 3D motion consistency with the help of geometric constraints derived from physical models, such as rigidity of an object, 2D homography, an epipolar constraint, or a trilinear constraint [66, 102, 112, 272]. Some of these approaches factor the point trajectories into different motion subspaces, assuming an affine projection model [79, 249, 266]. Other methods assume a perspective projection, and therefore segmentation is a problem of clustering the point trajectories into different multi-linear varieties [112, 208, 231].

7.2. 3D MOTION CONSISTENCY

To analyze 3D motion consistency, we address the situation in which we have a given set of matched pairs of feature points from two image frames. We aim to find a 3D rigid body motion consistent with all those matched pairs. Combining such a 3D motion with the camera projection, we can set up an equation relating the coordinates of each matched pair. With sufficiently many points from the same object, an overdetermined system of equations will be obtained. Due to the rigid body motion assumption, this system will have certain structural properties. By using matrix factorization techniques we then can analyze how to recover a 3D rigid body motion in the best possible way.

7.2.1. ANALYSIS

Consider a point on an object undergoing a rigid body motion. Suppose it moves, in the camera coordinate system, from some position $\mathbf{x} = (x, y, z)^\top$ at time τ to another position $\mathbf{x}' = (x', y', z')^\top$ at time τ'. Then according to Equation (3.20) we have that $\mathbf{x}' = R\mathbf{x} + \mathbf{t}$, where R is a rotation matrix and $\mathbf{t} = (t_1, t_2, t_3)^\top$ is a translation vector. Since all points of a rigid body have the same movement, the translation vector can be eliminated by working relative to a selected point at $\mathbf{x_0} = (x_0, y_0, z_0)^\top$ (e.g., a center of mass or any other point on the object):

$$\mathbf{x}' - \mathbf{x}'_0 = R(\mathbf{x} - \mathbf{x}_0). \tag{7.1}$$

If the scene is far away from the camera, and focal length is small compared to the distance of the object to the camera, then for every two points $\mathbf{x}$ and $\mathbf{x}'$ at distances Z_c and Z'_c, we can assume $\frac{f}{Z_c} \approx \frac{f}{Z'_c}$ (see Subsection 3.3.2). Hence, we can ignore the effect of the scaling factor $\frac{f}{Z_c}$ and the camera projection can be approximated by an orthographic projection. Therefore the point at position $(x, y, z)^\top$ in the camera frame is mapped (up to a fixed factor) to position $(x, y)^\top$ in the image frame. The following theorems apply, subject to this orthographic projection assumption. The general situation is discussed in Subsection 7.2.3.

Theorem 7.2.1. *A set of $m+1$ matched pairs of 2D points $(x_i, y_i)^\top$ and $(x'_i, y'_i)^\top$ (with $i = 0, \ldots, m$) can consistently be interpreted as the 2D coordinates of orthographic projections onto the image plane of $m+1$ pairs of 3D points in camera space which are related by a*

single 3D rigid body motion, if and only if the $m \times 4$ data matrix

$$M = \begin{pmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{x}'_1 & \tilde{y}'_1 \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{x}_m & \tilde{y}_m & \tilde{x}'_m & \tilde{y}'_m \end{pmatrix} \tag{7.2}$$

where $\tilde{x}_i = x_i - x_0$, $\tilde{y}_i = y_i - y_0$, $\tilde{x}'_i = x'_i - x'_0$ and $\tilde{y}'_i = y_i - y_0$, has a nontrivial null space containing a vector $\mathbf{v} = (v_1, v_2, v_3, v_4)^\top$ of which the four entries satisfy

$$v_1^2 + v_2^2 = v_3^2 + v_4^2. \tag{7.3}$$

Proof. ($\Rightarrow$) From Equation (3.21) and Equation (7.1), we have:

$$\begin{pmatrix} \tilde{x}'_1 \dots \tilde{x}'_m \\ \tilde{y}'_1 \dots \tilde{y}'_m \\ \tilde{z}'_1 \dots \tilde{z}'_m \end{pmatrix} = \begin{pmatrix} c_z & -s_z & 0 \\ s_z & c_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_x & -s_x \\ 0 & s_x & c_x \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \dots \tilde{x}_m \\ \tilde{y}_1 \dots \tilde{y}_m \\ \tilde{z}_1 \dots \tilde{z}_m \end{pmatrix} \tag{7.4}$$

Where s_x, c_x, s_y, c_y, s_z and c_z are shortcuts for $\sin\varphi_x$, $\cos\varphi_x$, $\sin\varphi_y$, $\cos\varphi_y$, $\sin\varphi_z$ and $\cos\varphi_z$ respectively.
After applying the orthographic projection to the result of the rotation $\tilde{\mathbf{x}}' = \mathbf{x}' - \mathbf{x}_0$, we get:

$$\begin{pmatrix} \tilde{x}'_1 \dots \tilde{x}'_m \\ \tilde{y}'_1 \dots \tilde{y}'_m \end{pmatrix} = \begin{pmatrix} c_z & -s_z \\ s_z & c_z \end{pmatrix} \begin{pmatrix} c_y & s_y s_x & s_y c_x \\ 0 & c_x & -s_x \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \dots \tilde{x}_m \\ \tilde{y}_1 \dots \tilde{y}_m \\ \tilde{z}_1 \dots \tilde{z}_m \end{pmatrix} \tag{7.5}$$

After transposition of this equation, we have:

$$\begin{pmatrix} \tilde{x}'_1 & \tilde{y}'_1 \\ \vdots & \vdots \\ \tilde{x}'_m & \tilde{y}'_m \end{pmatrix} = \begin{pmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\ \vdots & \vdots & \vdots \\ \tilde{x}_m & \tilde{y}_m & \tilde{z}_m \end{pmatrix} \begin{pmatrix} c_y & 0 \\ s_y s_x & c_x \\ s_y c_x & -s_x \end{pmatrix} \begin{pmatrix} c_z & s_z \\ -s_z & c_z \end{pmatrix} \tag{7.6}$$

Note that every row $(\tilde{x}'_i, \tilde{y}'_i)$ is a linear combination of $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)$, implying that M has a rank of at most 3.

Equation (7.6) can be rewritten as:

$$\begin{pmatrix} \tilde{x}'_1 & \tilde{y}'_1 \\ \vdots & \vdots \\ \tilde{x}'_m & \tilde{y}'_m \end{pmatrix} \begin{pmatrix} c_z & -s_z \\ s_z & c_z \end{pmatrix} = \begin{pmatrix} \tilde{x}_1 & \tilde{y}_1 \\ \vdots & \vdots \\ \tilde{x}_m & \tilde{y}_m \end{pmatrix} \begin{pmatrix} c_y & 0 \\ s_y s_x & c_x \end{pmatrix} + \begin{pmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{pmatrix} \begin{pmatrix} s_y c_x & -s_x \end{pmatrix} \tag{7.7}$$

or equivalently:

$$\begin{pmatrix} \tilde{x}_1 & \tilde{y}_1 \\ \vdots & \vdots \\ \tilde{x}_m & \tilde{y}_m \end{pmatrix} \begin{pmatrix} c_y & 0 \\ s_y s_x & c_x \end{pmatrix} - \begin{pmatrix} \tilde{x}'_1 & \tilde{y}'_1 \\ \vdots & \vdots \\ \tilde{x}'_m & \tilde{y}'_m \end{pmatrix} \begin{pmatrix} c_z & -s_z \\ s_z & c_z \end{pmatrix} = \begin{pmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{pmatrix} \begin{pmatrix} -s_y c_x & s_x \end{pmatrix} \tag{7.8}$$

We can combine the matrices $\begin{pmatrix} \tilde{x}_1 & \tilde{y}_1 \\ \vdots & \vdots \\ \tilde{x}_m & \tilde{y}_m \end{pmatrix}$ and $\begin{pmatrix} \tilde{x}'_1 & \tilde{y}'_1 \\ \vdots & \vdots \\ \tilde{x}'_m & \tilde{y}'_m \end{pmatrix}$ to form the matrix M:

$$M\begin{pmatrix} c_y & 0 \\ s_y s_x & c_x \\ -c_z & s_z \\ -s_z & -c_z \end{pmatrix} = \begin{pmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{pmatrix}\begin{pmatrix} -s_y c_x & s_x \end{pmatrix} \tag{7.9}$$

After multiplying the result by $\begin{pmatrix} c_x & s_x \\ -s_y s_x & s_y c_x \end{pmatrix}$, we get:

$$M\begin{pmatrix} c_y c_x & c_y s_x \\ 0 & s_y \\ -c_z c_x - s_z s_y s_x & -c_z s_x + s_z s_y c_x \\ -s_z c_x + c_z s_y s_x & -s_z s_x - c_z s_y c_x \end{pmatrix} = \begin{pmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{pmatrix}\begin{pmatrix} -s_y & 0 \end{pmatrix} \tag{7.10}$$

Because the second column of $(-s_y, 0)$ in this equation is 0, we have a nontrivial vector $\mathbf{v} = (c_y s_x, s_y, -c_z s_x + s_z s_y c_x, -s_z s_x - c_z s_y c_x)^\top$ such that $M\mathbf{v} = 0$. The elements of the vector $\mathbf{v}$ satisfy: $v_1^2 + v_2^2 = v_3^2 + v_4^2$, which proves one implication of Theorem 7.2.1.

If $s_y = 0$ and $c_x \neq 0$, the first column of $(-s_y, 0)$ in Equation (7.10) is also 0, and we have another nontrivial vector $\mathbf{v} = (1, 0, -c_z, -s_z)^\top$ such that $M\mathbf{v} = 0$. The elements of this vector $\mathbf{v}$ also satisfy: $v_1^2 + v_2^2 = v_3^2 + v_4^2$.

($\Leftarrow$) Conversely, if a non-zero vector $\mathbf{v} = (v_1, v_2, v_3, v_4)^\top$ is given in the kernel of M which happens to satisfy $v_1^2 + v_2^2 = v_3^2 + v_4^2$, we can proceed by the following two cases with respect to the value of v_2:

Case 1. *Assume that $v_2 \neq 0$. Then let φ_y have an arbitrary nonzero value in the interval $(-\arctan\left|\frac{v_2}{v_1}\right|, \arctan\left|\frac{v_2}{v_1}\right|)$, where the range is set to $(-\frac{\pi}{2}, \frac{\pi}{2})$ if $v_2 = 0$. Next, compute $\varphi_x = \arcsin\left(\frac{v_1}{v_2}\tan(\varphi_y)\right)$. Let $\lambda = \frac{v_2}{\sin\varphi_y}$ be a scaling factor, which is nonzero. Then $v_2 = \lambda s_y$ and $v_1 = \lambda c_y s_x$. Consequently λ can be computed from $v_1^2 + v_2^2 = \lambda^2(1 - c_y^2 c_x^2)$.*

Note that $\begin{pmatrix} v_3 \\ v_4 \end{pmatrix} = \lambda\begin{pmatrix} -c_z & s_z \\ -s_z & -c_z \end{pmatrix}\begin{pmatrix} s_x \\ s_y c_x \end{pmatrix}$ should hold, which can be rewritten in terms of c_z and s_z: $\begin{pmatrix} -v_3 & -v_4 \\ -v_4 & v_3 \end{pmatrix}\begin{pmatrix} c_z \\ s_z \end{pmatrix} = \lambda\begin{pmatrix} s_x \\ s_y c_x \end{pmatrix}$. The values of s_z and c_z are obtained, which uniquely specify $\varphi_z \in (-\pi, \pi]$.

Case 2. *Assumes that $v_2 = 0$. Then let $\varphi_y = 0$ and note that $v_1 \neq 0$. Now choose φ_x to have an arbitrary nonzero value in the interval $(-\frac{\pi}{2}, \frac{\pi}{2})$. Set $\lambda = v_1$. Then φ_z is determined through $\begin{pmatrix} v_3 \\ v_4 \end{pmatrix} = \lambda\begin{pmatrix} -c_z \\ -s_z \end{pmatrix}$. It follows that $v_1^2 + v_2^2 = v_3^2 + v_4^2 = \lambda^2$.*

In either of the two cases Case 1 and Case 2, a nonzero scaling factor λ and suitable values for φ_z, φ_y and φ_x are obtained which make that the vector $\mathbf{v}$ is of the form $\mathbf{v} = \lambda\begin{pmatrix} c_y s_x \\ s_y \\ -c_z s_x + s_z s_y s_x \\ -s_z s_x - c_z s_y c_x \end{pmatrix}$, or simplified of the form $\mathbf{v} = \lambda\begin{pmatrix} 1 \\ 0 \\ -c_z \\ -s_z \end{pmatrix}$ when $s_y = 0$.

In *Case 1* (where $s_y \neq 0$), this allows one to construct a corresponding vector $(\tilde{z}_1, \ldots, \tilde{z}_m)^\top$ to satisfy the required identity. Because $\begin{pmatrix} c_x & s_x \\ -s_y s_x & s_y c_x \end{pmatrix}$ is invertible, $(\tilde{z}'_1, \ldots, \tilde{z}'_m)^\top$ can be obtained by reconsidering the omitted third row of R. Clearly, translations in the z-direction cannot be observed at all, while coordinate values in all directions can only be obtained relative to an arbitrarily chosen origin. For z_0 and z'_0 one can introduce arbitrary values, which shows that the entry t_3 of translation vector $\mathbf{t}$ is completely free.

In *Case 2* (where $\varphi_y = 0$), the matrix:

$$\begin{pmatrix} c_y c_x & c_y s_x \\ 0 & s_y \\ -c_z c_x - s_z s_y s_x & -c_z s_x + s_z s_y c_x \\ -s_z c_x + c_z s_y s_x & -s_z s_x - c_z s_y c_x \end{pmatrix}$$

takes the form:

$$\begin{pmatrix} c_x & s_x \\ 0 & 0 \\ -c_z c_x & -c_z s_x \\ -s_z c_x & -s_z s_x \end{pmatrix}$$

Both columns are of the form $k(1,0,-c_z,-s_z)^\top$ because k can be either c_x or s_x. With φ_x from the indicated range (which ensures that c_x and s_x are both nonzero), we have that both columns are collinear, and that the relationship in Equation (7.10) is properly satisfied. However the matrix $\left(\begin{smallmatrix} c_x & s_x \\ -s_y s_x & s_y c_x \end{smallmatrix}\right)$ is no longer invertible, so to rewind our steps, we should reconsider Equation (7.9), which takes now the form:

$$M \begin{pmatrix} 1 & 0 \\ 0 & c_x \\ -c_z & s_z \\ -s_z & -c_z \end{pmatrix} = \begin{pmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{pmatrix} \begin{pmatrix} -0 & s_x \end{pmatrix} \tag{7.11}$$

because $\varphi_y = 0$. The first column represents a vector in the kernel we just dealt with. With $s_x \neq 0$ it follows that:

$$\begin{pmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{pmatrix} = \frac{1}{s_x} M \begin{pmatrix} 0 \\ c_x \\ s_z \\ -c_z \end{pmatrix} \tag{7.12}$$

Then we can proceed as in Case 1 to construct a rotation and translation which is consistent with the given observed data. This proves the converse implication of Theorem 7.2.1. □

Theorem 7.2.2. *If the condition under Theorem 7.2.1 is satisfied, then there exists a family of rigid body motions, consistent with the data, having at least one real degree of freedom for the translation (corresponding to an arbitrary translation in the z-direction) and at least one real degree of freedom for the 3D rotation.*

Theorem 7.2.2 is also proved according the proof of Theorem 7.2.1, noting that in both *Cases 1* and *2* a real degree of freedom for R (for the angles φ_y and φ_x, respectively) and for the coordinate t_3 was encountered. In special cases, i.e., when the rank of M is less than 3, more degrees of freedom may occur.

7.2.2. APPLICATION

Theorems 7.2.1 and 7.2.2 show the properties of point pairs between two images that following the same 3D motion. These theorems can be applied to analysis the consistency of a given set of point pairs, or estimate some of the 3D motion parameters, as shown in the following subsections.

CONSISTENCY OF 3D RIGID BODY MOTION

In computer vision applications, motion based image segmentation is an important and fundamental topic. The aim is to partition visual elements (pixels or feature points) into groups, based on their motion features. Segmentation algorithms are used in tasks like object detection and tracking, where objects are represented by groups of points (or pixels). For videos from a monocular camera, the key challenge of motion segmentation is to segment the points w.r.t. their 3D motions, while only 2D projection-coordinates of points are available.

Theorem 7.2.1 can be used to determine whether the movements of a group of 2D points (matched point from consecutive images) are consistent with a 3D rigid body motion. Giving $m+1$ pairs of points, we can decompose the $m \times 4$ data matrix M using the SVD:

$$M = UDV^\top \tag{7.13}$$

in which U is an $m \times m$ orthogonal matrix, V is a 4×4 orthogonal matrix, and $D = \text{diag}\{d_1, d_2, d_3, d_4\}$ is an $m \times 4$ diagonal matrix with entries $d_1 \geq d_2 \geq d_3 \geq d_4 \geq 0$ on its main diagonal. Theorem 7.2.1 establishes that at least $d_4 = 0$ should hold if the movement of 2D points is consistent with a 3D rigid body motion. However, when working with real data, deviations may occur for various reasons, such as inaccuracies in feature extraction and motion detection. Moreover, the orthographic projection hypothesis - which disregards the perspective - is an approximation.

The value of d_4 can be taken as a measure for the (lack of) quality of 3D rigid body motion consistency, for the group of points being analyzed. According to Theorem 7.2.1, in case of a rigid 3D body motion, every vector $\mathbf{v}$ in the kernel satisfies $v_1^2 + v_2^2 = v_3^2 + v_4^2$ if $d_3 > 0$. This property can be also used as a quality measure for the rigid body motion consistency. Note that a vector $\mathbf{v}$ is obtained as the last column of matrix V if $d_4 = 0$ and $d_3 > 0$.

RECONSTRUCTION OF 3D RIGID BODY MOTION

Theorems 7.2.1 and 7.2.2 also enable us to estimate the parameters of a 3D rigid body motion for a given set of matched pairs. Starting from data matrix M (Equation (7.2)) with a 1-dimensional null space, using Equation (7.3), there will be one real degree of freedom when computing the 3D rotations $\varphi_z, \varphi_y, \varphi_x$. There is also one degree of freedom (the translation in the z direction) in determining $\mathbf{t}$. However, the values $\tilde{z}_1, \ldots, \tilde{z}_m$ completely depend on the degree of freedom for the 3D rotation.

We may determine the value of φ_y or φ_x by minimizing a criterion function, such as the sum of squares of values $\tilde{z}_1, \ldots, \tilde{z}_m$. The idea is that the norm of the vector of changes in the (unobserved) z-direction, consistent with the computed rotation and translation, is minimized. So, no unnecessarily movement in the unobserved z direction is included in the rigid body motion.

7.2.3. ERROR ANALYSIS

The proposed theorems are based on the orthographic projection, which is an approximation of the perspective projection. In this subsection, we analyze the errors of orthographic projection w.r.t. to the perspective projection.

Suppose a 3D point is moving from $(X_c, Y_c, Z_c)^\top$ at time t to $(X'_c, Y'_c, Z'_c)^\top$ at time t', and two images are captured at the two time points. The coordinates of a projected point at time t and t' under the perspective projections are $(x_p, y_p)^\top$ and $(x'_p, y'_p)^\top$ respectively. The coordinates of the same projected point under orthographic projection are $(x, y)^\top$ and $(x', y')^\top$. According to the Equations (3.24) and (3.25): $(x, y)^\top = (X_c, Y_c)^\top$, $(x_p, y_p)^\top = \frac{f}{Z_c}(X_c, Y_c)^\top$, $(x', y')^\top = (X'_c, Y'_c)^\top$ and $(x'_p, y'_p)^\top = \frac{f}{Z'_c}(X'_c, Y'_c)^\top$. The perspective projection scales the $(X_c, Y_c)^\top$ with a factor $\frac{f}{Z_c}$. We can compensate for the scaling of $(x_p, y_p)^\top$ by multiplying $(x_p, y_p)^\top$ with $\mu = \frac{Z_c}{f}$. So, $(x, y)^\top = \mu(x_p, y_p)^\top$. By applying the same scaling to $(x'_p, y'_p)^\top$, we can compute the error caused by orthographic projection:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} - \mu \begin{pmatrix} x'_p \\ y'_p \end{pmatrix} = (1 - \frac{Z_c}{Z'_c}) \begin{pmatrix} X'_c \\ Y'_c \end{pmatrix} \tag{7.14}$$

If the changes in z direction caused by translation and rotation are small, then $\frac{Z_c}{Z'_c} \approx 1$, and the error is approximately 0.

7.3. EXPERIMENTS

In this section, we evaluate the applicability of the proposed theorems on both synthetic data in subsection and real video data in Subsection 7.3.1. These theorems can also be used to estimate the 3D rigid motion of an object with one degree of freedom. We evaluate this aspect in Subsection 7.3.2.

7.3.1. IMPROVING THE SEGMENTATION QUALITY

As the theorems proposed in Section 7.2 can be used to evaluate the 3D consistency of a given set of 2D projected points pairs, we use them for improving a generated segmentation result. There are two possible ways to apply our results in motion segmentation,

1. Giving the result of a segmentation, an object is represented as a group of points. Usually there are miss-classified points in each group, which reduces the precision of the segmentation. We can use the results of the previous section to find the miss-classified point in the group of points.
2. Given a group of points that are belonging to an object, and a set of new points without assignments, we can use the results of the previous section to identify whether the new points belong to the object. Failing to identify these points reduces the recall of the segmentation.

We measure the 3D consistency error by $error = \left\| d_4 + \sqrt{v_1^2 + v_2^2 - v_3^2 - v_4^2} \right\|$, which is named by "3DM" in the experiments. A point is assigned to a group if its error is lower than a threshold, which is 0.005 in this experiment. This threshold is determined by investigating the average 3D consistency error of the objects from the Hopkins155 dataset. In the experiments, we evaluate the performance of 3DM in two ways as above. Firstly, given some points from one object and some noise does not belong to this object, we aim at identifying the labels, i.e. "object" and "noise", of these points using 3DM. By varying the ratio of noise in the given set, we evaluate the classification accuracy, which is defined as the percentage of points that are successfully classified. Secondly, given a set of points that

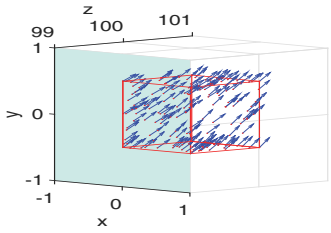


Figure 7.1: 3D motion flows

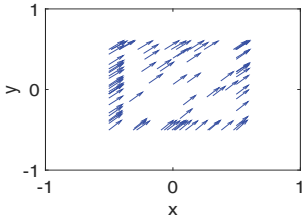


Figure 7.2: 2D motion fields

are known to be belong to an object, we aim at determining the class of some unlabeled points. We investigate the classification accuracy of the unlabeled points, by varying the size of the known set. We use synthetic data and real data to do the experiments in the following subsections.

IMPROVING MOTION SEGMENTATION USING SYNTHETIC DATA

We generated a 3D synthetic scene containing a cube, which follows a combination of rotation and translation. Randomly chosen points on the surface of the cube are tracked. We also randomly generate some noise points that have arbitrary 3D motions. The motion of each point is represented by its initial position and the new position after transformation. Figure 7.1 illustrates the 3D motion flows of the points on the cube surface in camera space, while Figure 7.2 shows the orthographic projection of these motion vectors on the image plane that is parallel to the xy plane.

For the first experiment, we choose 100 points from the cube object and n noise points. The objective is to divide these points into two subgroups: the "objects" and "noises" using the results of the previous section. We did the experiment multiple rounds by varying the number of noise points n from 0 to 50. In each round, we redid the experiment 100 times by changing the chosen points, and plotted the average result.

We compared the 3DM with the sparse subspace clustering method (SSC) [79]. Figure 7.3 illustrates the accuracy with respect to different n, i.e. the number of noise points. The classification accuracy of 3DM is higher than SSC. We also investigated the false negatives (FNs) and false positives (FPs) in the results, as shown in Table 7.1. The 3DM can correctly identify the object points since the false negatives remains zero, and the errors are caused by the misclassified noise, i.e. the false positives. Note that the noise points are randomly generated, and it might have similar movement as the object points. Therefore, it is no surprise that those noise points are misclassified. SSC generates false negatives while the 3DM does not, and generates more false positives than 3DM. As a trajectory clustering algorithm, SSC requires that the input points fit with a given number of motions and it is sensitive to noise [79]. It can be observed from Table 7.1 that both FP and FN increase with the number of noise in the set. We can conclude that our 3DM outperforms SSC when dealing with one single cluster with noise from the experiment results.

For the second experiment, m ($m \in [4, 100]$) points on the cube are chosen to represent the object, which are used to determine the classification of other 200 unlabeled points (half from the object and while the other half are randomly generated noise points) using the results of the previous section. We compared the results of the proposed approach

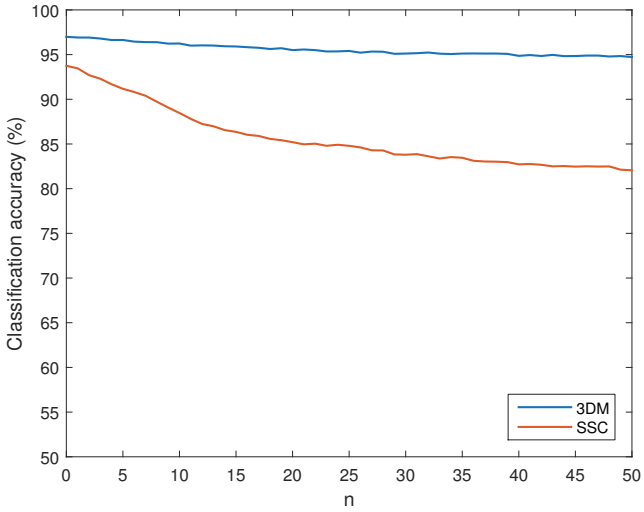


Figure 7.3: The average classification accuracy of classifying the object points and the noise from a set of points representing a moving object.

n		0	1	5	10	20	30	40	50
FN	3DM	0	0	0	0	0	0	0	0
	SSC	16.2	15.4	18.5	23.4	24.6	23.9	25.5	25.0
FP	3DM	0	0.15	0.72	1.45	2.36	3.23	3.75	4.69
	SSC	0	0.1	0.6	1.6	3.1	4.0	5.4	6.5

Table 7.1: The average false negatives (FNs) and false positives (FPs) in the results of classifying the object points and the noise from a set of points representing a moving object.

3DM, with that of an affine model based method, which computes the error w.r.t. the affine motion model of the m points using the approach described in Chapter 6. For each m, we redid the experiment 100 times by changing the unlabeled points each time, and plotted the average result.

The classification accuracy of unlabeled points is shown in Figure 7.4. The classification accuracy of 3DM is higher than that of the 2D affine model. Table 7.2 shows the false negatives and false positives of the result. Using the 3DM, the number of false negatives is 0 and the number of false positives decreases when m increases. It shows that the 3DM is better able to identify the object points than the noise points. The method based on 2D affine model results in lower classification accuracy than the 3DM. When the number of given labeled points increases, the 2D method generate more false positives and less false negatives. It means that the 2D method is sensitive to the size of the given set, as a larger set does not guarantee a better performance. The results show that the 3D consistency measure described in this chapter outperforms classification of points based on the affine motion model.

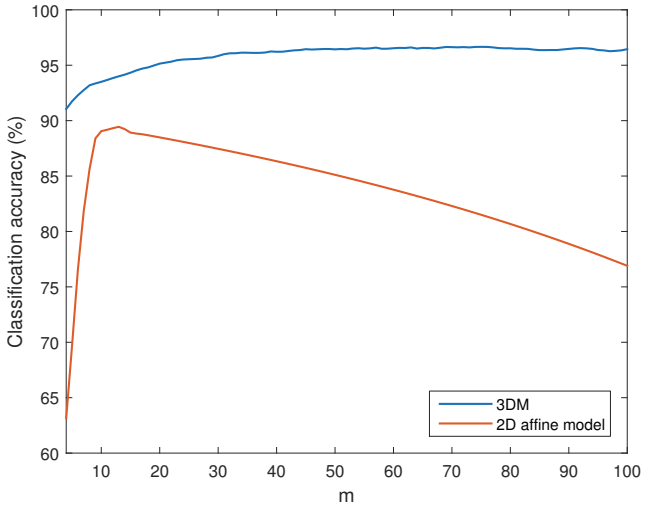


Figure 7.4: The average classification accuracy of assigning 200 unlabeled points to a given set of m points. The unlabeled set contains 100 object points and 100 noise.

m		4	10	20	30	40	50	60	70	80	90	100
FN	3DM	0	0	0	0	0	0	0	0	0	0	0
	2D model	33.9	15.5	12.0	11.4	10.8	10.4	10.4	10.4	10.3	10.0	9.7
FP	3DM	20.2	14.2	8.5	7.3	6.1	5.6	5.1	4.7	4.6	4.6	4.4
	2D model	3.0	3.6	5.3	8.0	11.7	14.8	17.6	20.0	21.6	24.5	25.5

Table 7.2: The average false negatives (FNs) and false positives (FPs) in the results of assigning unlabeled points to a given set of m points based on 3D consistency measure.

IMPROVING MOTION SEGMENTATION USING VIDEO DATA

In this experiment, we used the real video sequences from the Hopkins155 benchmark data set [236]. The Hopkins155 benchmark data set provides for each video in the data set, a set of feature points and their motions on the image plane. We chose 25 video sequences from the category named "checkerboard". Each video contains 29 frames, which records a scene with 3 objects following distinct 3D motions (rotation and translation). There are 75 objects in total. In each experiment we chose one object and used the motion vectors between the frame pair $\{f_1, f_i\}$ $(i \in (2, 29))$.

In the first experiment, for each video in the dataset, we chose all points of one object to be the "object", and points from other objects to be the "noise". We changed the number of noise points n_{noise}, thus the ratio $n_{\text{noise}} : n_{\text{object}}$ varied between 0% and 50%, where n_{object} is the number of object points. We took the average result of all objects in the experiment, with respect to different frame pairs and noise ratios. Figure 7.5 shows the average accuracy with respect to different frame pairs and noise ratios. The results show that the accuracy decreases when the noise rate increases. The distance between frame pairs has no significant influence for low noise ratios, and has a small influence, around 7% for

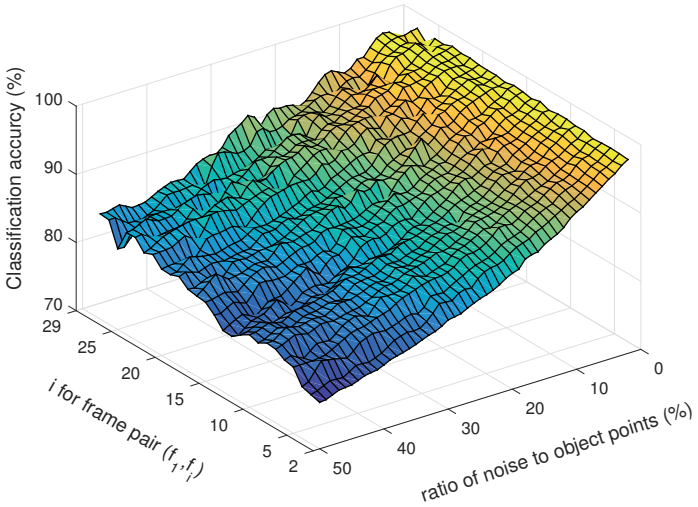


Figure 7.5: The accuracy of classifying the noise and object points in a given set by 3DM, using the data provided by the Hopkins 155 dataset.

high noise ratios. When there are more noises, the accuracy increases with the distance of frame pairs. As was to be expected, a larger frame distance results in a higher accuracy. Since displacement between two frames is smaller when the frames are closer, 3DM is better able to deal with frame pairs of relatively large distance. Note that the video sequences used in the experiment are of at most 31 frames, the time slot of a frame pair is smaller than 1.25 seconds for a standard 24 fps video. To investigate the affects of frame distance, more experiments are needed using sequences longer than 1.25 seconds. Table 7.3 shows the average false negatives (FNs) and false positives (FPs) in the result. When the noise increases, both false positives and false negatives increase. When the distance between the frame pair increases, the false negatives increase and the false positives decreases. Generally, the performance of 3DM on real video data is affected by the frame distance and noise rate. The 3DM performs well when the noise is less than 10% of the objects points.

In the second experiment, we computed the average accuracy of allocating an unlabeled point to a given group representing an object, with respect to the group size m, i.e. the number of points in the group, and the distance between frames. Figure 7.6 and Table 7.4 shows the average results. The results show that a larger group size is beneficial in identifying whether a point belongs to a group. However, the accuracy levels off at a maximum of 85%. The distance between frames does not have a significant influence on the results, although a smaller frame distance seems to be beneficial for smaller group sizes. The false negatives increases when the distance between frames becomes larger, although the false positives decreases. If more points are given in the group, the false negatives decrease but false positives increases. The results also suggest that 3DM on its own will not be sufficient to create a segmentation algorithm that out performs the segmentation algorithm described in Chapter 6.

Average number of points in the set								
n_{object}	146	146	146	146	146	146	146	146
n_{noise}	0	1.5	7.3	14.6	29.2	43.8	58.4	73.0
i for (f_1, f_i)	FN							
2	3.2	5.5	5.1	8.4	13.1	14.9	13.0	14.5
5	5.0	8.7	7.2	12.7	21.3	23.1	24.0	22.5
10	5.8	9.7	8.3	14.2	23.9	25.8	25.5	21.9
15	6.0	10.1	8.5	14.4	24.3	26.0	25.1	24.7
20	6.8	10.2	9.6	15.9	25.8	25.7	28.8	30.0
29	7.8	11.9	10.7	17.7	27.2	27.8	28.4	27.7
i for (f_1, f_i)	FP							
2	0	0.1	3.1	4.3	5.2	8.0	10.1	20.0
5	0.0	0.2	4.7	6.5	7.8	12.0	15.2	30.0
10	0.0	0.0	3.2	6.0	9.0	14.4	16.1	18.0
15	0.0	0.0	3.6	6.5	6.9	9.2	15.2	18.8
20	0.0	0.0	2.9	4.4	7.2	9.9	14.9	15.6
29	0.0	0.2	0.9	3.3	3.0	4.8	13.1	13.4

Table 7.3: The average false negatives (FNs) and false positives (FPs) of classifying the object points and the noise from a set.

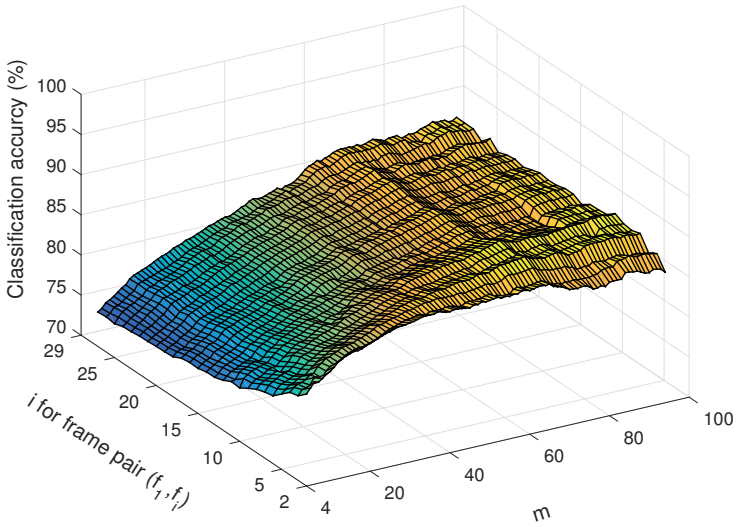


Figure 7.6: The accuracy of classifying unlabeled points to a given group by 3DM, using the data provided by the Hopkins 155 dataset.

Average number of points in the unlabeled set											
n_{object}	142	136	126	116	106	96	86	76	66	56	46
n_{noise}	291	291	291	291	291	291	291	291	291	291	291
m	4	10	20	30	40	50	60	70	80	90	100
i for (f_1, f_i)	FNs										
2	11.0	12.4	10.7	7.5	6.5	1.2	1.3	0.2	0.4	0.1	0.2
5	48.0	33.7	32.9	28.4	24.2	14.3	12.7	7.2	4.9	3.3	6.4
10	55.8	52.0	43.4	36.8	32.7	24.4	19.1	13.9	9.6	6.3	7.2
15	82.7	59.5	51.9	43.3	38.6	28.9	24.1	25.4	14.6	9.6	10.1
20	85.3	69.7	58.3	49.4	43.7	33.8	28.4	28.1	18.1	13.4	13.6
29	87.9	65.5	47.9	47.6	43.3	34.0	35.6	35.3	23.5	10.9	11.4
i for (f_1, f_i)	FPs										
2	127.4	106.9	99.4	88.2	79.7	76.1	72.9	70.8	70.3	71.1	69.3
5	96.4	89.2	61.0	51.1	61.6	58.8	65.2	68.2	69.0	68.3	68.9
10	79.0	47.2	40.0	24.4	25.9	31.3	37.6	38.7	40.4	39.6	41.7
15	55.5	48.0	36.5	26.6	29.2	33.8	40.0	41.3	36.9	50.1	43.1
20	53.4	33.1	18.5	21.9	33.4	37.1	43.0	43.1	47.3	44.0	43.5
29	53.3	66.0	67.9	66.6	66.6	71.8	59.5	31.0	39.4	48.5	41.7

Table 7.4: The average false negatives (FNs) and false positives (FPs) of assigning unlabeled points to a given group.

7.3.2. RECOVERING THE 3D RIGID BODY MOTION

In this experiment, we investigated whether it is possible to handle the one degree of freedom for the 3D rotation by minimizing the sum of squares of $\tilde{z}_1, \ldots, \tilde{z}_m$. We generated an synthetic 3D object as in Subsection 7.3.1 and moved it by a 3D rotation with a 3D translation. Based on the assumption of the presented theorems, we did the experiment only for small motions. The rotation angles are chosen randomly in the interval $(-\frac{\pi}{4}, \frac{\pi}{4})$. The translation distance in a direction is also randomly selected, of which the absolute value is no more than the size of object in this direction.

We estimate the rotation angles based on the method in Subsection 7.2.2. The idea is that the norm of the vector of changes in the (unobserved) z-direction, consistent with the computed rotation and translation, is minimized so that no unnecessarily large deviations are included in the rigid body motion.

Our initial experiments with points on the surfaces of a cube in the synthetic scene showed that for randomly chosen rotations smaller than $\pi/4$ rad, we can recover the rotation angles φ_x, φ_y and φ_z with average accuracies of 74.3%, 74.3%, 94.6% respectively. The high accuracy for the rotation around the z axis is not surprising because this rotation is does not depend on the (unobserved) z-direction. The rotations around the x and y axis do depend on the (unobserved) z-direction. Minimizing the sum of squares of $\tilde{z}_1, \ldots, \tilde{z}_m$ results in a reasonable estimate of these rotation angles.

7.4. CONCLUSION

This chapter presented two theorems specifying properties of a 2D projection of a 3D rigid body movement. The theorems state that the data matrix of 2D projection of points on a 3D rigid body making a 3D movement, has a non-trivial kernel with a specific structure. The theorems also show that we can reconstruct the original 3D body movement with one degree of freedom for the translation in z-direction and one degree of freedom for the 3D rotation.

We used the theorems to measure the 3D rigid motion consistency of a group of 2D projection points. We applied the 3D motion consistency measure to refine a given segmentation, by removing the outliers in a given group of points that are supposed to move consistent in 3D world. The experimental results show that the classification accuracy increases if the noise rate decreases. We also used the theorems to determine whether an unlabeled point belong to a given object. When more points are given for the know object, the unlabeled points are more likely to be identified correctly. These results suggest that the theorems can be used to improve the segmentation accuracy of existing motion segmentation algorithms.

Recovering the 3D rotation angle of a moving object has also been evaluated. Due to the one degree of freedom in computing the 3D rotations, the estimated rotation angles can be inaccurate. The experimental results are promising. The estimation of rotation angle about z direction an accuracy of 94.6%, which is higher than the estimation of rotation angles about x and y direction.

In general, the proposed 3D motion consistency measure shows promising potential. In the future research, we can investigate to vary the threshold flexibly based on the data. The applicability of the presented theorems can be widened by developing a 3D based motion segmentation algorithm. One can investigate the 3D consistency constraint for

mixture motions, as the presented 3DM is only applied for a single motion. For recovering the 3D motion models, further research is required to address the one degree of freedom when computing the 3D rotations. Moreover, further studies can be aimed at investigating the 3D consistency of a long-term trajectory.

8

LEARNING STATIC SEGMENTATION FROM THE MOTION SEGMENTATION RESULTS

The previous chapters of this thesis addressed the segmentation of moving objects in each frame of a video sequence. Two approaches were investigated, one that describes the movements of objects using affine models, and one that tries to exploit properties of rigid moving 3D objects. An advantage of movement-based object segmentation is that no a priori knowledge about the moving object is needed. This makes it possible to learn objects from videos in an unsupervised way. However, as the objects in a video may not always move, motion segmentation does not work when an object is static. In the context of the thesis, it is interesting to investigate whether we can unsupervisedly learn to segment an static image as well. Moreover, it is interesting whether we can learn to identify objects.

Learning to segment objects that do not move is a very useful ability. In robot soccer for instance the ball or some robots may stop moving for some period. Having learned to identify an object and keeping track of it when it does not move, will be important for playing the soccer game. Being able to identify different appearances of one and the same object in different situations is also a useful ability in robotics. Moreover, it is necessary to consistently identify the same object in different situations. It allows for communication about the object on a higher level of abstraction.

In the following sections, we will first address the problem of segmenting non-moving objects using information that is learned while the object was moving. Next we will address the task of recognizing an object using the same learned information.

8.1. INTRODUCTION

Motion segmentation algorithms can segment out moving objects from the static background in an unsupervised way, but will fail to segment out objects that are not or no longer moving; i.e., no motion information is available. Segmentation of static images

relies on the visual representations of objects, which are classically obtained by a supervised learning approach. A variety of studies have addressed this problem using various machine learning techniques and successes have been achieved [60, 98, 103, 150, 183, 184, 276]. These methods typically require a large dataset of annotated images to train a learning model. However, annotation of these images is quite time-consuming even, frustrating and commercially expensive, while most of the methods need a pixel-wise accurate annotation [103]. In recent decades, weakly supervised methods have been proposed for semantic segmentation by less accurate annotations, such as bounding boxes and image-level labels [183, 184]. These methods have shown good performance on many state-of-the-art datasets, such as ImageNet [204] and PASCAL VOC [82].

As motion segmentation can segmented out the objects from video sequences, this suggests an unsupervised learning approach for static segmentations could be developed, which utilizes the results of motion segmentation as the training data. Human vision studies have shown that infants and newly sighted congenitally blind people tend to learn objects based on their motions [180, 221]. Motivated by such studies and the development of computer vision and machine learning techniques, research on the subject of learning from motion has boosted in recent years. A variety of methods have been proposed for unsupervised object detection or semantic image segmentation by learning from motions. Since learning from motion segmentation is a multi-task problem, its solution varies with the problem set-up. Techniques used for motion estimation, motion segmentation and feature representation, as well as learning methods, affect the design and performance of a solution. Some early methods in this class focused on learning a statistical model from the motion segmentation results [200, 222]. These methods often use pixel-wise segments based on two frame optical flows to construct appearance models for objects. Recently, due to the development of deep learning, more and more methods utilize the techniques, such as Convolutional Neural Networks (CNN), to learn high-level feature representations from the motion segmentation results [75, 109, 185]. As CNN has shown excellent ability of representing high-level features, these methods use CNN to simultaneously learn both the object representations and segmentation (detection) models, from motion segmentation results of videos [51, 60, 103, 204].

An object presents in different frames may differ in the appearances, which lead to the variations of object representations in the segmentation results. The video object segmentation in this thesis aims at segmenting out the objects from every frame of a video sequence, which requires that the same object to be identified in different frames. The object identification can be interpreted as an object recognition task. However, the classic learning based object recognition methods require a big training set and a supervised learning process. In this thesis, we investigate unsupervised object segmentation, and expect to perform object identification with less human intervention.

In this chapter, we utilize the segmentation results obtained in Chapter 6 as the learning data. Experiments in Chapter 6 showed that the segmentation algorithm performs best on dense trajectories. Furthermore, the dense features contain more visual information of the segmented objects than sparse features. Thus in this chapter, we utilize the dense feature extraction method of Subsection 5.4.3 for image representation and use the segmentation results on dense trajectories (see Subsection 6.4.3) as the learning data.

8.2. THE LEARNING DATA

To formally specify the learning method, we first must specify the learning data. We will use the following notation. An image is represented by a set of feature points, while each feature point consists of a position vector **x** and a description vector **d**. A segmentation partitions an image into multiple feature sets, each corresponding to a (moving) object (or the background) in the image. If an image contains K objects, the segmentation induces a partition $\{S_1,\ldots,S_K\}$ which consists of non-overlapping sets containing feature points belonging to the objects. Given a sequence of T segmented images, we collect $C_i = \{S_{i,0},\ldots,S_{i,T-1}\}$, a set of feature point sets per frame for each object i. Note that an object may be not present in every frame: $S_{i,j} \in C_i$ is empty when object i is not present in the j^{th} frame. For K objects, our learning data consists of $C_1,\ldots,C_K$.

8.3. STATIC OBJECT SEGMENTATION

Our aim is to segment objects from a video frame when they are static, by using information that was learned when the objects were moving. We will investigate this by learning from the first T frames from a sequence of L frames, i.e., the images $I_0,\ldots,I_{T-1}$; are subsequently used to segment the images $I_T,\ldots,I_{L-1}$. By varying the length T of the learning sequence from 2 to $L-2$, we can see the effect on the segmentation quality. By segmenting images $I_T,\ldots,I_{L-1}$, we can see to which extent the quality decreases when the images start to differ more from the images $I_0,\ldots,I_{T-1}$ used for learning.

Each feature point extracted from an image is represented by a description vector representing the feature and a position vector containing its location in this image. A feature may be present in multiple image frames of the video, because they are consecutively observed from the same scene. The same feature point extracted from consecutive images in a video, is identified by matching their description vectors, of which the locations compose the trajectory of this point in the images $I_0,\ldots,I_{T-1}$.

For each image J in the remaining images $\{I_T,\ldots,I_{L-1}\}$, we also extract feature points. We compare the extracted feature points of each image J with the feature points extracted form the images $I_0,\ldots,I_{T-1}$. Using the segmentation results of the images $I_0,\ldots,I_{T-1}$, we can associate an object label to the feature points of the each image J in the remaining images $\{I_T,\ldots,I_{L-1}\}$.

As our feature points are described by the scale-invariant feature transform algorithm (SIFT), two feature descriptors are matched by the Euclidean-distance based nearest neighbor approach in [152]. For each feature point in J, a best match is found by identifying the nearest neighbor from the features in learning images $I_0,\ldots,I_{T-1}$, and the feature point in J is correspondingly associated with an object class to which the nearest neighbor belongs. However, some of the matches found by the nearest neighbor approach can be incorrect, because ambiguous features can arise due to background clutter, illumination changes and motions, etc. Therefore, we evaluate the quality of matches and reject the less reliable matches using Lowe's method [152]. Lowe proposed an effective measure of SIFT matching, which is obtained by comparing the distance of the closest neighbor to that of the second-closest neighbor. To match the features in two images, matches in which the distance ratio is greater than 0.8 are rejected because they have a 90% probability [152]. Since the training data contains features from multiple image frames, a feature in image J

may have multiple correct matches that are from the same object but in different frames. Therefore, we compare the distance ratios of matches from different objects. In general, given a feature point, which is denoted by p in image J, we carry out the following steps to find the match from the learned data:

1. For each $C_i = \{S_{i,0}, \ldots, S_{i,T-1}\}$ in the learning data, we find a candidate matching $\hat{q}_i$ for p by
$$\hat{q}_i = \underset{q_i \in \cup_{j=0}^{T-1} S_{i,j}}{\operatorname{argmin}} \|p - q_i\| \tag{8.1}$$
where $\|\cdot\|$ is the l_2 norm.
2. Suppose there are K objects in the learning data, we have K candidates $\hat{q}_1, \ldots, \hat{q}_K$, each represent the possible matching feature of q to an object in the learning data. We take the nearest and the second-nearest neighbor of p in the K candidates. Let $\hat{q}_{j_1}$ denote the nearest neighbor, and $\hat{q}_{j_2}$ denote the second-nearest neighbor, we compute the distance ration
$$DR = \frac{\|p - \hat{q}_{j_1}\|}{\|p - \hat{q}_{j_2}\|} \tag{8.2}$$
If DR is smaller than a threshold ρ, here we take it to be 0.8 as in [152], the best match of p is taken to be the nearest neighbor $\hat{q}_{j1}$. Otherwise, the given feature does not have a match in the learning data.

8.4. OBJECT IDENTIFICATION

Object identification aims at identifying the same object in different images. Of course, if we track an object through a sequence of frames, we are fairly certain that it is the same object in each of these frames. In fact we exploit this in the learning process when we track features. However, tracking objects by their motions only works for a continuous video sequence. Suppose that there are two video sequences that reflect the same scene at different time, or from different views. It is possible that some objects can present in both sequences. But the approaches proposed in this thesis, for motion segmentation and non-moving objects segmentation, can not identify the same object in different sequences.

What we will now investigate is whether we can identify an object without using the motion information but only using the learned segmentations. Thus the robot (computer) can continually learn from the obtained information. We will investigate this by learning from a video sequence, which is called the training sequence, to identify the objects present in another video sequence of the same scene, which is called the test sequence. As a beginning, we start with the simple case. We divide a video sequence of length L into two parts: the first T frames form the training sequence, and the rest $L-T$ frames are used as the test sequence. The training sequence is segmented by the presented segmentation approach, so do the test sequence. Different objects in the training sequence are annotated by their class labels. Then we learn a SVM classifier from the training sequence, to identify the class labels of objects in the test sequence. To see what the effect is of the training sequence length on the identification task, we vary the value of T from 2 to $L-1$.

As an object instance is represented as a set of SIFT features in the segmentation results, the feature sets of different objects can contain different number of feature points

because the object appearance varies in different images. However, they can not be directly used as the input of SVM to object recognition, since the SVM requires that the input data to be identical dimension. We will generate uniform data representations for the input objects, and then perform object recognition based on the new representations.

DATA REPRESENTATION

We encode these feature sets into uniform object representations based on the Bag of Words (BoW) model [105, 227]. Specifically, we learn a set of basis vectors $\mathrm{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_M]$ which is called codebook, so that each SIFT feature vector $\mathbf{d}$ can be represented as a linear combination of these basis vectors

$$\mathbf{d} = \sum_{m=1}^{M} a_m \mathbf{b_m} \tag{8.3}$$

The coefficients a_m form a vector $\mathbf{a} = [a_1, \ldots, a_M]^\top$, which is called the sparse codes of d based on B. Sparse coding (SC), which is able to capture high-level features in the inputs, is used to learn the codebook and compute the sparse codes of the SIFT features in the input data. Let $D = [\mathbf{d}_1, \ldots, \mathbf{d}_N] \in \mathcal{R}^{N \times d}$ be the input set of SIFT features, where N is the number of feature in this set and d is the dimension of feature vectors, sparse coding solves the following problem

$$\begin{aligned} &\min_{\mathrm{A,B}} \textstyle\sum_{n=1}^{N} \|\mathbf{d}_n - \mathbf{a}_n \mathrm{B}\|^2 + \lambda\, |\mathbf{a}_n| \\ &\text{subject to} \quad \|\mathbf{b}_m\| \le 1, \quad \forall \mathbf{b}_m \in \mathrm{B} \end{aligned} \tag{8.4}$$

where $\mathrm{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_N]$, $\mathrm{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_M]$, $\|\cdot\|$ is the l^2 norm and $|\cdot|$ is the l^1 norm. Here M is chosen to be larger than d to make the codebook B an over-complete basis set.

The encoding of object representations consists of a training phase and a coding phase, as follows:

- In the training phase, we learn the codebook for sparse coding. We randomly chose a collection of SIFT features extracted from different images to be the training set. Then Equation (8.4) is solved with respect to both B and A given D as the training set. The training set of sparse coding requires more SIFT features that are extracted from various of images for generating a more representative codebook.
- In the coding phase, we repeat the following steps to encode every segmented object in the training and test sequences, based on the spatial pyramid representation in [267].

  Firstly we compute the sparse codes by solving Equation (8.4) given the features of an object and the learned codebook. Secondly, we construct spatial pyramids by partitioning an object into increasing finer sub-segments. For each sub-segment of the spatial pyramids, we determine a collection of sparse codes $\left[\mathbf{a}_1^*, \mathbf{a}_2^*, \ldots, \mathbf{a}_q^*\right]$. Next we do max pooling on each collection of sparse codes to compute an M-dimensional vector $\mathbf{z}^*$, of which the m^{th} element z_m^* is computed by:

  $$z_m^* = \max\{|a_{1,m}^*|, |a_{2,m}^*|, \ldots, |a_{q,m}^*|\} \tag{8.5}$$

  where $a_{i,m}^*$ is the m^{th} element of vector $\mathbf{a}_i^*$. The pooled representations $\mathbf{z}^*$ of all sub-segments are then concatenated to form a single vector $\mathbf{z}$, which is therefore the spatial pyramid representation of the object.

After the coding phase, all segmented objects are represented by vectors in a M-dimensional feature space.

CLASSIFICATION

As in Section 8.2, as the training sequence is specified as the learning data, the objects are naturally annotated by a serial of numbers, which can be used as their class labels. Once the segmented objects are encoded, we can train a SVM classifier based on the obtained objects in the training sequence. We use a linear SVM with kernel:

$$\kappa(z_i, z_j) = z_i^T z_j \tag{8.6}$$

where z_i and z_j are the spatial pyramid representations of object segments S_i and S_j respectively. [267] shows that spatial pyramids together with sparse coding makes it possible to classify images using a linear SVM. Then we use the learned SVM classifier to identify the class labels of objects in the test sequence.

It is worth to note that, SVM is only used to identify objects from different classes, and it is not able to distinguish objects which are belong to the same class. When we use the object number in the segmentation results as the class label, we assume that these objects are from different classes in the real world. In practice, however, there are usually multiple objects in a video that are belong to the same class. For example, there are multiple individuals of the object class "robot" in a robot soccer video, but the segmentation result differs them in number. To avoid ambiguity, we manually give the objects the same class label if they belong to the same class in the real world in the experiment.

8.5. EXPERIMENT

In this section, we evaluate the proposed static image segmentation and object identification on the Robocup 2014 and CDNet 2014 dataset. Because the two datasets provide groundtruth segmentation for every frame in a given video, which allow us to evaluate the segmentation quality for T varying from 1 to 30.

8.5.1. STATIC OBJECT SEGMENTATION

In this experiment, we evaluated the non moving object segmentation, as proposed in Section 8.3. Given the motion segmentation results of $I_0, \dots I_{T-1}$, we obtained the learning data $C_1, \dots, C_K$ for K "objects". And then we segmented the following images $I_{T+1}, \dots I_L$, based on matching feature descriptors with the leaning data. By varying T, we investigate how the quality of segmentation results changes by measuring the segmentation accuracy, precision, recall and F-measure.

Figures 8.1 to 8.4 illustrate the non-moving object segmentation quality for $T = 2, 10, 15, 20$ respectively. The motion segmentation quality of image frame I_{T-1} is also plotted in figures with round circles for comparison. The performance of static segmentation drops compared to the motion segmentation on I_{T-1}, of which the accuracy, precision, recall and F-measure drops about 10% to 20%. Generally, the segmentation quality of $I_T, \dots I_{L-1}$ decreases when the frame index increases. It is because of a moving object might appear with more differences in two distant frames than in two nearby frames. When the amount of training data increases, the decreasing trend becomes more smoothly. For example, for $T = 2$ shown in Figure 8.1, the segmentation accuracy from I_T to I_{T+5} drops

from 72% to 68%. But for $T = 25$ shown in Figure 8.4, the segmentation accuracy from I_T to I_{T+5} remains around 72%. The other measures, i.e. precision, recall and F-measure show similar trend as the accuracy.

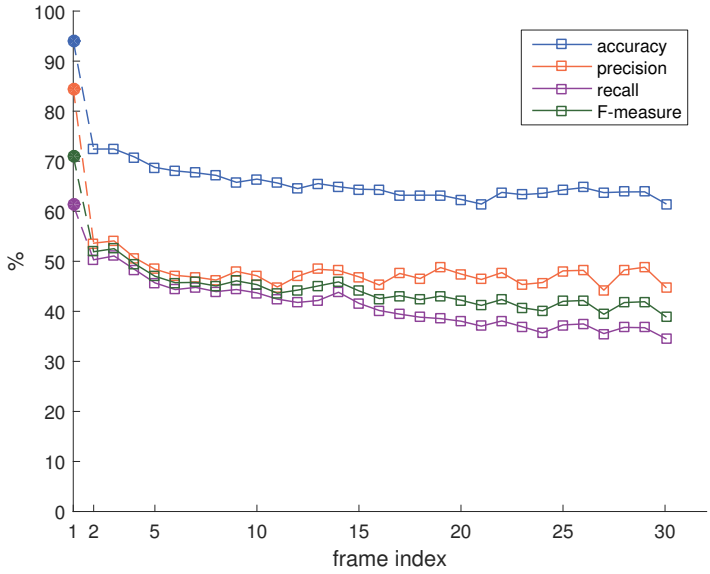


Figure 8.1: The accuracy, precision, recall, and F-measure of static segmentation of $I_T, \ldots, I_{L-1}$ learned from the motion segmentation results of $I_0, \ldots, I_{T-1}$ for $T = 2$ are plotted in squares. For comparison, the measures of motion segmentation results of I_{T-1} is plotted in circles.

8.5.2. OBJECT CLASSIFICATION

In this experiment, we evaluated the classification accuracy of the segmented objects.

To train the codebook, we use 10,000 SIFT descriptors randomly sampled from the feature points extracted from images in the datasets. It is sufficient to extract the main features from given videos, considering that there are 23 videos and each contains about 30 frames of images. The size of code book is chosen as 1024, which is verified for a good performance on sparse coded features [267].

Given a video sequence of length L in the dataset, we take the first T images $I_0, \ldots I_{T-1}$ as the training sequence, and the rest images $I_T, \ldots I_{L-1}$ as the test sequence. We perform motion segmentation to segment both training and test sequence, and then encode the segmented objects by sparse coding, as introduced in Section 8.4. Given the SC encoded object representations, an SVM classifier is learned based on the training sequence. There are 5 categories of moving objects that present in the dataset: car, truck, people, robot, ball. In addition, the static background is also regarded as an object class because all feature points from the non-moving objects are segmented out as an integral part. Therefore there are in total 6 classes of objects. The number of objects that are extracted from all videos in the dataset for each class is shown in Table 8.1.

The obtained SVM classifier is used to identify the class labels of the encoded object

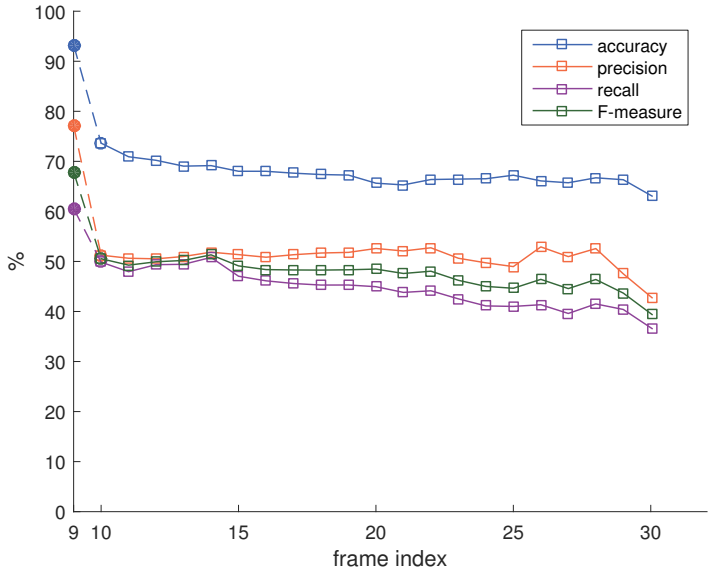


Figure 8.2: The accuracy, precision, recall, and F-measure of static segmentation of $I_T, \ldots, I_{L-1}$ learned from the motion segmentation results of $I_0, \ldots, I_{T-1}$ for $T = 10$ are plotted in squares. For comparison, the measures of motion segmentation results of I_{T-1} is plotted in circles.

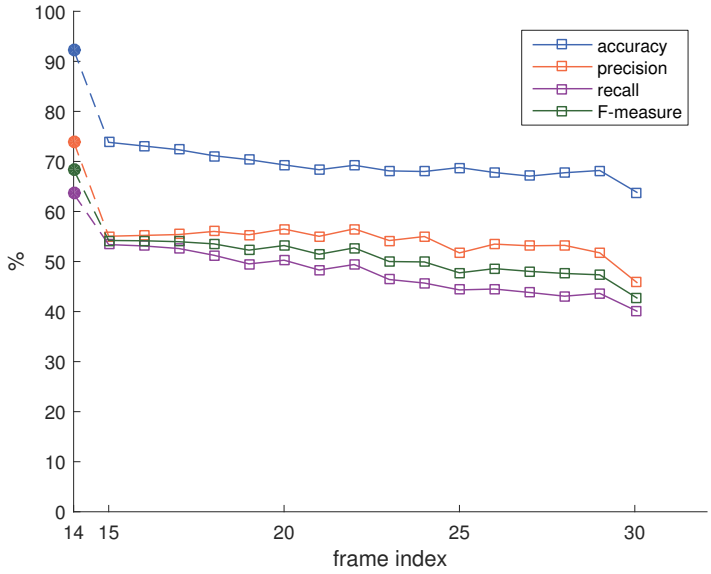


Figure 8.3: The accuracy, precision, recall, and F-measure of static segmentation of $I_T, \ldots, I_{L-1}$ learned from the motion segmentation results of $I_0, \ldots, I_{T-1}$ for $T = 15$ are plotted in squares. For comparison, the measures of motion segmentation results of I_{T-1} is plotted in circles.

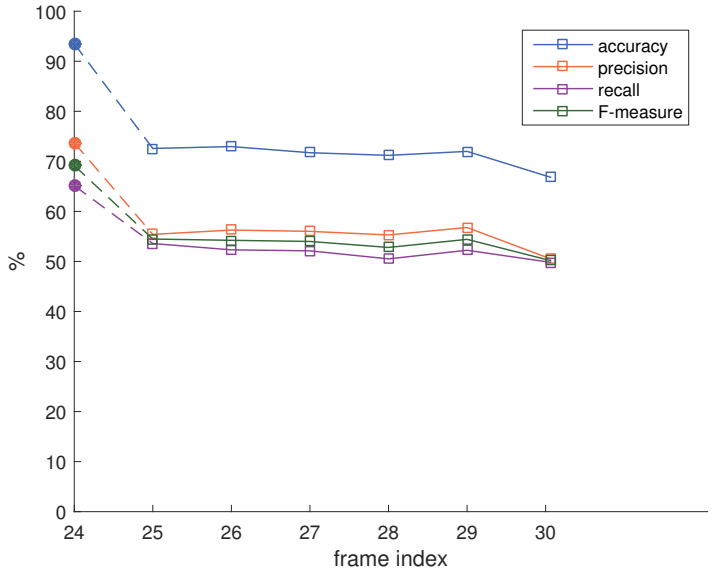


Figure 8.4: The accuracy, precision, recall, and F-measure of static segmentation of $I_T,\ldots,I_{L-1}$ learned from the motion segmentation results of $I_0,\ldots,I_{T-1}$ for $T=25$ are plotted in squares. For comparison, the measures of motion segmentation results of I_{T-1} is plotted in circles.

background	car	truck	people	robot	ball
713	859	103	124	542	142

Table 8.1: The number of object instances of each class can be extracted from all videos in the dataset.

representations in the test sequence. The ground-truth class label of a segment in the test sequence is manually assigned. Since some obtained segments may cover regions of multiple ground-truth objects, it is labeled by the object class of the largest part.

The classification accuracy is evaluated by the average percentage of corrected identified segments for all testing data. We varied the number of images in the training data, i.e. T, from 2 to 30. We visualized the results for $T=2,10,15,20$ in Figure 8.5.

When T is fixed, the classification accuracy is not affected by time. It means that although the motion segmentation accuracy usually increases over time, it does not have obviously influences on the identification of the segmented objects. Figure 8.5 shows that the classification accuracy depends on T, the number of images in the training data. For a particular frame, the classification accuracy is always higher when T is larger. It reveals that the object identification relies on the amount of training data. Note that it corresponds to only 1 second in a video if the training set is formed by 25 images, for a typical video with a frame rate of 24 fps. The results show that the segmented objects obtained by the presented motion segmentation algorithm can effectively represent the scene objects, since they can be identified by learning from a limited training set.

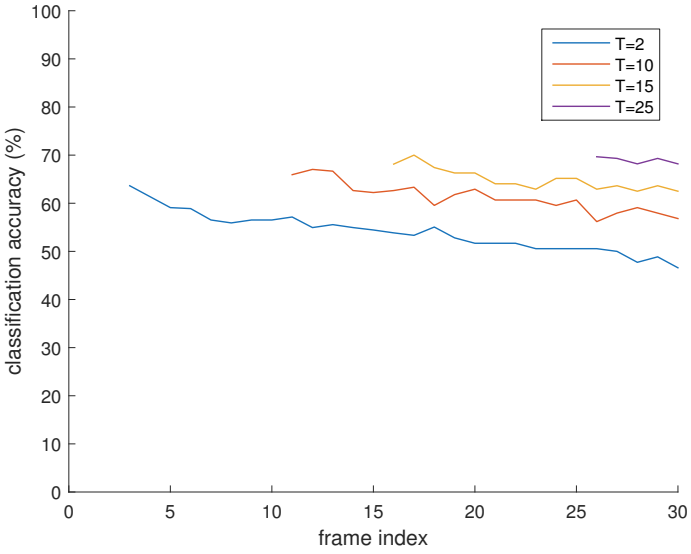


Figure 8.5: The classification accuracy of obtained segments from $I_T, \ldots, I_{L-1}$ by SVM learned from the objects extracted from $I_0, \ldots, I_{T-1}$, for $T = 2, 10, 15, 20$ respectively.

8.6. CONCLUSION

This chapter addressed the segmentation of non-moving objects in a video sequence by learning from the motion segmentation results. It answered the fourth research question: "How can the objects be segmented out without motion information?".

Motion segmentation integrate the features of various visual properties in an image, to obtain semantic meaningful segments in an unsupervised way. When no motion occurred, the knowledge acquired by motion segmentation can still be utilized for static object detection/segmentation in a similar scene. Therefore, we suggested an unsupervised structure for object segmentation from a given video, which focuses on segmenting the moving objects that present in the scene. As SIFT algorithm provides robust feature descriptions, the non-moving object segmentation is determined by matching the SIFT points from the static image with the learning data.

With more images, we have richer learning data. By varying the number of images in the learning data, we investigated its effect on the segmentation quality. Results show that more learning data results in a better performance in non-moving object segmentation. Note that the experiment uses sequences of only 31 frames, which correspond to only 1.2 seconds of a 24 fps video. It shows that learning from motion information within a short period (1.2 seconds) can provide effective knowledge for segmenting static images.

Compared with the manually managed datasets for supervised segmentation approaches, the learning data obtained from motion segmentation is less accurate, in which the wrong segmentation, over-segmentation and under-segmentation may occur. We have evaluated the segmentation results by comparing with the ground-truth segmentations. In this chapter, we additionally evaluated the quality of the segmentation results by investi-

gating whether we can correctly identify the object class of the obtained segments. Results show that more than 82% of the obtained segments can be identified correctly, if we use the learning data obtained from at least 25 frames of a video.

As the segmentation of static images can benefit from the motion segmentation results, it is worth to investigate if the resulted static segmentation can be used to refine the motion segmentation results in the future. It suggests an unsupervised system with the ability of improving the segmentation over time, by learning from both motion cues and visual features. Furthermore, it is worth to explore the performance on longer sequences.

9

CONCLUSIONS AND FUTURE WORK

This thesis investigated unsupervised object segmentation in a video sequence. When a video is given without any additional information, motion provides concrete cues for grouping visually distinct features into regions that then represent different objects in a video sequence. We focused on segmenting out "salient" objects from backgrounds by learning from the motion information in a video sequence. The salient object refers to an object which moves independently with respect to the background in a video sequence. Note that the movements are unpredictable, a salient object might move during only part of a given video. As videos are composed of successive images, the motion information is inherent in the changes of the images. To tackle the research problem, we needed to extract the motion data from a video sequence, and utilize the acquired motion data for object segmentation. In Chapter 1, we have posed five research questions that needed to be answered.

In Section 9.1, we present our results and conclusions to answer these research questions. In Section 9.2, we discuss promising directions of future research based on the achievements in this thesis.

9.1. CONCLUSIONS ON THE RESEARCH QUESTIONS

The research questions stated in Chapter 1 concern the data processing pipeline from low level to high level, (i.e. from pixels to semantic objects). In the following subsections, these research questions are addressed one by one.

9.1.1. SYSTEM DESIGN

As a multi-task problem, video object segmentation can not be solved within a single step. Starting from pixel-level image processing, multiple steps are involved to obtain the semantic level object representations and intermediate data, such as motion data, are generated during the middle steps. The acquirement and representation of motion data plays an important role in this process. When to extract the intermediate data, and what kind of data to generate, as well as how to process it, all affects the final results. This has led to the

first research question:

Research Question 1 How to design a framework for motion based video object segmentation?

In Chapter 4, we designed a framework, combining a bottom-up procedure, which extracts high-level object representations by analyzing the lower-level features in the video frames, with a top-down procedure, which segments the lower-level features by learning from the obtained high-level representation. Without knowledge, a video only provides a sequence of successive digital images. The direct information consists of the low-level visual features represented in the images, such as pixel intensity and color. Based on this direct information, we can extract local features, such as structures, orientations, and textures. By identifying the same local features in multiple images of the video, we can obtain the movements of these local features in the 2D images. As the 2D images are projections of the 3D world, the 2D motions of local features in the images reflect the projected motion of the corresponding 3D elements. As points on the same object should move consistently, those local features that move accordingly can be integrated together as a segment. Such a segment is assumed to be a potential object in the video. The above unsupervised process segments out objects from different images in the video sequence. It naturally follows a bottom-up scheme. However, this only works well if motions occur in the images. If an object stops moving, it can be no longer detected. To segment out static objects, prior knowledge of the object representations is required. As motion based segmentation can segment out objects while they are moving, it inspires us to learn object representations from the obtained segmentations. The learned knowledge can be used to segment out these objects also from static images, which is obviously a top-down procedure.

The bottom-up approach learns object segmentation from low-level features, while the top-down approach learns object segmentation from high-level object representations. With a combination of both bottom-up and top-down approaches, it is possible to tackle the research problem in an unsupervised way. In practice, our system is composed of three main steps based on the nature of the processed data. The first step processes the input video frames (pixels) to extract the motion data. The second step analyzes the motion data for segmenting out moving objects. The first two steps form the bottom-up procedure. The third step fulfills the top-down learning, which extracts high-level object representations from the motion segmentation results, and utilizes the learned information for segmenting out objects from static images. For each step, there are corresponding research questions to be answered, which will be discussed in the following subsections. The results show that the moving objects can be unsupervised segmented out from a video with this presented framework. It also has high flexibility in dealing with variations, such as the data types, video lengths, and the number of objects.

9.1.2. FEATURE EXTRACTION AND MOTION ESTIMATION

The first step of the system is to extract motion data from video sequences, which is also called motion estimation. As a video projects a 3D scene into dynamic 2D images, the motion information of the 3D scene is implied in the changes of successive frames. Therefore, motion estimation can refer to either analysis of the 2D image motions, or recovering the 3D motions in the real scene. In this thesis, we are interested in segmenting out the objects,

for which the recovering of 3D motions is not strictly necessary. In addition, the representation of extracted motion data is an issue as there are several methods to do this, which can have impact on the further processing. It has led to the following research question:

Research Question 2 How to extract and represent motion data from a video sequence?

We tackled the second question by tracking the movements of specific features in the frames of a video sequence. The motion data is represented as the movements of invariant features in these frames. Good features for motion estimation are supposed to be visually invariant in different images, thus we can track the same feature by matching their descriptions in different images; see Subsection 2.3.2 and Sections 3.1 and 3.2. However, noise and variations are caused by technical reasons, such as the camera, lighting conditions, etc., and they affect the quality and accuracy of this information. Therefore the answer to this research question also depends on the type of features to be tracked, which was discussed in Chapter 5.

We investigated pixel-based and feature-based methods, which are two basic categories of motion estimation methods. The pixel-based method can generate dense and accurate motions between two images, but can not track pixels in multiple frames. The feature based method tracks a set of local features, and generates longer trajectories of feature points over multiple frames. In the experiments, we applied one pixel-based method and two feature-based methods, one for sparse feature tracking and another for dense feature tracking. Three types of motion data are extracted from the used data sets, i.e. the two-frame pixel motions, sparse point trajectories and dense point trajectories. The three types of motion data all have their own advantages and limitations. The two-frame pixel motions is accurate at a pixel level, which is convenient in segmenting image regions. However, the similarity of two-frame pixel motions on the same object is less significant than for longer point trajectories. Compared to pixel-wise motions, point trajectories lose some visual information. The sparse tracking method is the fastest, but it tracks less points than the other two methods. The dense tracking method is also more accurate than sparse tracking method. The obtained motion data is subsequently used as the input for the motion segmentation method. In the following chapters, we further investigated the influences of different motion data types on the segmentation results in more details.

9.1.3. MOTION SEGMENTATION

The obtained motion data describes how 2D image points move in a video sequence. These 2D points can be pixels, or some local features that are invariant in a video sequence. The 2D image points are projections of corresponding 3D points on the objects in the real world. Assuming that the object is rigid, points from the same object should move consistently in the 3D world accordingly to a rigid body motion (such as translation and rotation). Thus the motion information provides clues for segmenting out objects that undergo different motions from the video sequences. However, retrieving the 3D object motions from the obtained 2D image motions is difficult, as the depth value is missing due to the camera projection. This has led to the following research question:

Research Question 3 How can the objects be segmented out based on the motion information restricting to 2D consistency in the images?

We developed a motion segmentation algorithm, called AEM-b in Chapter 6. The core intuition behind this algorithm is that long-term trajectories of points from the same object show higher similarity than two-frame motion vectors, as observed from the obtained motion data in Chapter 5. The proposed segmentation algorithm performs and updates the segmentation for every pair of successive frames in a video sequence. For each frame pair that is processed, this algorithm models the two frame object motion by a 2D affine transformation, and computes the segmentation by a classification EM algorithm. The segmentation result of each frame pair is propagated to the next frame pair by Bayesian updating. With this procedure, the segmentation is improved gradually by considering the long-term motion coherences. We also proposed an improved version of the algorithm AEM-b$^+$ by including reliability estimates of segmentation results and compensating for the camera movement.

To evaluate the algorithms AEM-b and AEM-b$^+$, we used the three types of motion data obtained in Chapter 5, i.e. the two-frame pixel motions, sparse trajectories and dense trajectories. We compared the AEM-b and AEM-b$^+$ with six motion segmentation approaches: four are well-known trajectory clustering algorithms and two are the state-of-the-art video segmentation methods. The compared methods have more requirements for the input motion data than our algorithm. Firstly, the compared methods can only deal with trajectories, while our algorithm is able to deal with all three types of motion data. Secondly, the trajectory clustering algorithms normally require that the trajectories are of the same length and the video segmentation methods need that the sequences are shorter than a certain length, while the our algorithm does not need that. Moreover, our algorithm can deal with a varying number of objects. The segmentation results were evaluated with four metrics: accuracy, precision, recall and the F-measure. Results showed that AEM-b and AEM-b$^+$ perform among the best for dealing with all three types of motion data. The other methods perform well for one type of the motion data but poorly for other types of motion data. AEM-b$^+$ outperforms AEM-b when dealing with videos with significant camera motion. We also evaluated the computation time of all methods. AEM-b is reasonably fast, and it usually outperforms other fast methods on segmentation quality. AEM-b$^+$ is slower than AEM-b . The results also showed that the segmentation quality for dense trajectories is better than for the other two data types (i.e. the two-frame motions and sparse trajectories).

The results of AEM-b and AEM-b$^+$ also showed that over-segmentation of objects remains a problem in the results. As the camera projection can break a 3D motion into multiple 2D motions, the over-segmentation is inevitable for 2D model based motion segmentation. Although the Bayesian updating reduces the number of over-segmentation areas, there are usually one or two over-segmented objects in the results for the experiments that were performed. To address the over-segmentation problem, the inherent 3D motion consistency of the motion data needs to be considered. This raised the fourth research question:

Research Question 4 How to retrieve 3D motion consistency based on the 2D motion data?

To address this question, we derived two theorems by investigating the 3D motion consistency of the 2D projected motion data, in Chapter 7. These theorems are based on an

orthographic projection assumption. We presented a metric for measuring the 3D motion consistency of a given set of two-frame 2D point motions based on the theorems. We also investigated to retrieve the 3D motion models using the theorems. The experimental results showed that the 3D motion consistency measure (3DM) can identify and reduce the noise from object points given a noisy segment. The 3DM is also useful for allocating unlabeled points to a known object. The results suggested that the theorems are helpful to improve the segmentation accuracy of an existing motion segmentation algorithm. However, the proposed theorems are insufficient for developing an accurate motion segmentation algorithm based on the 3D motion models alone. Results have shown that the retrieval of 3D motion models is not accurate as there is one degree of freedom for solving the problem. Moreover, the orthographic projection assumption ignores the perspectivity effects. And this chapter did not yet consider the 3D motion consistency of long-term trajectories which cover multiple frames. But this can be done and further research is needed to investigate a better motion segmentation algorithm based on 3D motion consistency.

9.1.4. LEARNING FROM THE MOTION SEGMENTATION

Motion segmentation addresses the detection of objects that move in a video sequence. However, the motions that occur in a video are changeable and unpredictable. One object can move in some frames and stop in others. Motion segmentation can not segment out an object once it stops. This has led to the fifth question:

Research Question 5 How to segment out objects from static frames using moving information from videos?

To address the answer of this question, we investigated in Chapter 8 how we can learn from the knowledge that is obtained from the motion segmentations. The motion segmentation provides the representations of objects in multiple frames. The segmented frames can be used as knowledge input for a learning model for segmenting static images in similar situations. We proposed an unsupervised object segmentation approach based on learning models from the motion segmentation results of a subsequence of a given video. This approach segments out static objects by matching the SIFT descriptors of the feature points in the learning data and the static images. We evaluated the quality of segmentation results by measuring the classification accuracy of the class labels of the obtained objects. As the segmentation of trajectories abstracts the images by a set of feature points, the obtained objects can miss some pixels. The object identification process measures the robustness of the obtained object representations in the segmentation results.

In the experiments, we evaluated this static object segmentation approach and the object identification using the segmentation results of dense trajectories, which has proved to be the best data type for motion segmentation in Chapter 6. We evaluated the segmentation quality by varying the number of images in the learning data. Results showed that static object segmentation improves if the learning data uses more images. Unfortunately, the errors in the learning data, such as wrong segmentation, over-segmentation and under segmentation, remain present in the static object segmentation results and decrease the segmentation quality. Although the quality of static object segmentation is less compared to the motion segmentation results, the object identification results have shown that the segmented object can still be identified with an accuracy higher than 80%. It means

that the decrease of segmentation quality does not significantly affect the representations of segmented objects. We conclude that the motion information is not only helpful for motion segmentation, but also beneficial to static object segmentation. Learning from motion data in this way achieved unsupervised object segmentation of video sequences, for both moving and static objects.

9.2. FUTURE RESEARCH

The research presented in this thesis indicates the following areas of interest for future research:

1. **Agglomerative motion segmentation**

   The motion segmentation algorithm presented in Chapter 6 follows a top-down scheme, which starts with putting all data points in one cluster and then iteratively splitting a chosen cluster until a stopping criterion is reached. The segmentation result is strongly affected by the splitting criterion, i.e. the choice of cluster to split, the splitting method and the stopping conditions. In the divisive procedure, the motion model of a cluster is computed by taking all points in this cluster into account, and the errors of points are computed according to the estimated model. This approximation can lead to wrong assignments in the splitting procedure, especially when the cluster is a mixture of multiple objects of almost equal size.

   An alternative method is to perform the segmentation in an agglomerative manner. An agglomerative approach is constructed bottom-up: it starts with building a singleton cluster for each data point and recursively merges clusters. The estimated models can be more accurate in the agglomerative procedure than in the divisive procedure, because the clusters are more likely to contain points from the same object.

2. **3D motion segmentation**

   To address the over-segmentation problem, segmentation based on 3D motion consistency is a potential option. Chapter 7 dealt with a cluster which undergoes one single motion and investigated 3D motion consistency of image motions between two frames. This research can be extended to deal with long-term trajectories of multiple motions over multiple frames. It has lead to two possible directions. On the one hand, one can investigate the 3D motion consistency of long-term trajectories, by introducing more strict geometrical constraints, such as the multiple view geometry under the perspective projection. On the other hand, retrieving 3D motion models from the obtained 2D projected motions can be investigated.

3. **Online learning**

   In this thesis, the segmentation is computed for each pair of frames by considering the motion information in a current frame pair and combining it with the knowledge learned from previous frames. The experimental results in Chapter 6 have shown that the segmentation can be improved if more frames are processed. Chapter 8 has revealed that the segmentation of a single frame can be helped by the motion segmentation results of previous frames in a video. This suggests the video object

segmentation can be achieved with an online learning procedure. We suggest that segmentation at each frame can be computed based on the obtained motion data and a learned model, and then the obtained segmentation is used to update the learned model, especially if a video is likely to generate many frames. Many state-of-the-art learning approaches, such as Deep convolutional neural networks (DCNN) and region-based convolutional neural networks (RCNN), can be investigated for training the motion segmentation model. The segmentation model can be continuously updated as long as new frames come in. The online learning segmentation can theoretically deal with videos of arbitrary length. And once an object has been learned, it can be recognized whenever it shows up again. Additionally, we suggest that a general model learned from multiple videos in a dataset can be investigated. This procedure mimics the memory mechanism of the human vision system, which helps for better and faster segmentation of an unknown video.

4. **Segmentation based on both motion and visual features**

   We investigated how to utilize the motion information for video object segmentation in this thesis. Over-segmentation remains a problem if we only use the motion information for object segmentation. To address this problem, we can consider to use both motion information and visual features for object segmentation. An object is often visually different from the background. However, the visual appearance of different regions on an object can be different too. We can partition the video frames into small regions by grouping the pixels of similar visual properties, and then perform motion segmentation on these small regions. Furthermore, the motion estimation and segmentation can be carried out simultaneously by considering both motion and visual features, for which high-level feature representations, such as super-pixel and super-voxels, can be investigated.

REFERENCES

[1] Andrea F Abate, Michele Nappi, Daniel Riccio, and Gabriele Sabatino. 2D and 3D face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885–1906, 2007. 5

[2] Malcolm Acock. Vision: A computational investigation into the human representation and processing of visual information. *The Modern Schoolman*, 62(2):141–142, 1985. 1

[3] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994. 23

[4] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984. 37

[5] Jake K Aggarwal and Quin Cai. Human motion analysis: A review. *Computer vision and image understanding*, 73(3):428–440, 1999. 5, 7

[6] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16:1–16:43, April 2011. 5, 6, 7

[7] Yucel Altunbasak, P Erhan Eren, and A Murat Tekalp. Region-based parametric motion segmentation using color information. *Graphical models and image processing*, 60(1):13–23, 1998. 49, 57

[8] Luis Alvarez, Rachid Deriche, Théo Papadopoulo, and Javier Sánchez. Symmetrical dense optical flow estimation with occlusions detection. *International Journal of Computer Vision*, 75(3):371–385, 2007. 25

[9] Yali Amit. *2D object detection and recognition: Models, algorithms, and networks*. MIT Press, 2002. 15

[10] Jens Christian Andersen, Morten Rufus Blas, Ole Ravn, Nils A Andersen, and Mogens Blanke. Traversable terrain classification for outdoor autonomous robots using single 2d laser scans. *Integrated Computer-aided engineering*, 13(3):223–232, 2006. 8

[11] Alexander Andreopoulos and John K Tsotsos. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8):827–891, 2013. 3, 28

[12] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016. 10

[13] Pablo Arbeláez, Bharath Hariharan, Chunhui Gu, Saurabh Gupta, Lubomir Bourdev, and Jitendra Malik. Semantic segmentation using regions and parts. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3378–3385. IEEE, 2012. 16

[14] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011. 21

[15] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4015–4023, 2015. 26

[16] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 62, 66, 67

[17] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012. 45

[18] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24:1–24:11, 2009. 26

[19] Harry G Barrow and RJ Popplestone. Relational descriptions in picture processing. *Machine intelligence*, 6:377–396, 1971. 29

[20] Harry G Barrow and J M Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer vision systems*, 2:3–26, 1978. 1

[21] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 19

[22] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. 19

[23] Mr. Bayes and Mr Price. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs. *Philosophical Transactions (1683-1775)*, pages 370–418, 1763. 44

[24] Elliot Joel Bernstein and Yali Amit. Part-based statistical models for object classification and detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '05*, pages 734–740. IEEE, 2005. 27, 49

[25] Massimo Bertozzi, Alberto Broggi, and Alessandra Fascioli. Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous systems*, 32(1):1–16, 2000. 171

[26] Pia Bideau and Erik Learned-Miller. It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In *European Conference on Computer Vision*, pages 433–449. Springer, 2016. 10, 58, 85, 87

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 30, 43

[28] Michael Black. Combining intensity and motion for incremental segmentation and tracking over long image sequences. In G. Sandini, editor, *Computer Vision—ECCV'92*, pages 485–493. Springer, 1992. 49

[29] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996. 25, 34

[30] Michael J Black, Yaser Yacoob, Allan D Jepson, and David J Fleet. Learning parameterized models of image motion. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 561–567. IEEE, 1997. 25

[31] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003. 4

[32] Aaron F Bobick. Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 352(1358):1257–1265, 1997. 7

[33] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001. 24

[34] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296, 2008. 7, 8, 9

[35] Eran Borenstein and Shimon Ullman. Combined top-down/bottom-up segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 30(12):2109–2125, 2008. 16, 49, 50

[36] Georgi D Borshukov, Gozde Bozdagi, Yucel Altunbasak, and A Murat Tekalp. Motion segmentation by multistage affine classification. *IEEE Transactions on Image Processing*, 6(11):1591–1594, 1997. 26, 27, 49

[37] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992. 29, 45, 47

[38] Oene Bottema and Bernard Roth. Theoretical kinematics. *North-Holland*, 1979. 40

[39] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001. 35, 36, 66

[40] Thierry Bouwmans, Caroline Silva, Cristina Marghes, Mohammed Sami Zitouni, Harish Bhaskar, and Carl Frelicot. On the role and the importance of features for background modeling and foreground detection. *Computer Science Review*, 28:26–91, 2018. 20, 25

[41] Alan C Bovik. *Handbook of image and video processing*. Academic press, 2010. 11, 12, 22, 23, 25, 26, 55, 57, 62

[42] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 4, 36

[43] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008. 35

[44] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. *Computer Vision—ECCV 2002*, pages 707–720, 2002. 43

[45] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 690–696. IEEE, June 2000. 5

[46] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2009. 25, 26, 64, 65, 70, 89

[47] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. *Computer Vision-ECCV 2004*, pages 25–36, 2004. 25, 65

[48] Thomas Brox, Andrés Bruhn, and Joachim Weickert. Variational motion segmentation with level sets. In *European Conference on Computer Vision*, pages 471–483. Springer, 2006. 27

[49] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pages 282–295. Springer, 2010. 24, 56

[50] Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983. 37

[51] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015. 15, 49, 124

[52] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. 22

[53] Bradley P Carlin and Thomas A Louis. *Bayesian methods for data analysis*. CRC Press, 2008. 45

[54] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012. 15

[55] Joao Carreira and Cristian Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3241–3248. IEEE, 2010. 49

[56] Vicent Caselles and J Morel. Introduction to the special issue on partial differential equations and geometry-driven diffusion in image processing and analysis. *IEEE transactions on image processing*, 7(3):269–273, 1998. 22

[57] Andrea Cavallaro, Olivier Steiger, and Touradj Ebrahimi. Tracking video objects in cluttered background. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(4):575–584, 2005. 26, 96

[58] Gilles Celeux and Gérard Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332, 1992. 44, 78

[59] H Chan and WW Bledsoe. A man-machine facial recognition system: some preliminary results. *Panoramic Research Inc*, 1965. 5

[60] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018. 27, 124

[61] Fang-Hsuan Cheng and Yu-Liang Chen. Real time multiple objects tracking and identification based on discrete wavelet transform. *Pattern recognition*, 39(6):1126–1139, 2006. 27

[62] Guangchun Cheng, Yiwen Wan, Abdullah N. Saudagar, Kamesh Namuduri, and Bill P. Buckles. Advances in human action recognition: A survey, 2015. 6

[63] Wongun Choi, Khuram Shahid, and Silvio Savarese. What are they doing? : Collective activity classification using spatio-temporal relationship among people. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1282–1289, 11 2009. 6

[64] Andrea Colombari, Andrea Fusiello, and Vittorio Murino. Segmentation and tracking of multiple video objects. *Pattern Recognition*, 40(4):1307–1317, 2007. 26, 96

[65] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002. 21

[66] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 1071–1076. IEEE, 1995. 108

[67] Daniel Cremers and Stefano Soatto. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, 2005. 27

[68] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. 47

[69] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, pages 1–22. Prague, 2004. 29

[70] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 18, 65

[71] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977. 44

[72] Yining Deng and BS Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE transactions on pattern analysis and machine intelligence*, 23(8):800–810, 2001. 21

[73] Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002. 8, 9

[74] Jian-Jiun Ding, CJ Kuo, and WC Hong. An efficient image segmentation technique by fast scanning and adaptive merging. *CVGIP, Aug*, 2009. 23

[75] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 124

[76] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 26

[77] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012. 29

[78] Krista A Ehinger, Barbara Hidalgo-Sotelo, Antonio Torralba, and Aude Oliva. Modelling search for people in 900 scenes: A combined source model of eye guidance. *Visual cognition*, 17(6-7):945–978, 2009. 50

[79] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797. IEEE, 2009. 27, 89, 93, 96, 108, 114

[80] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Camera-based navigation of a low-cost quadrocopter. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2815–2821. IEEE, 2012. 7

[81] James T Enns. *The thinking eye, the seeing brain: Explorations in visual cognition*. Recording for the Blind & Dyslexic, 2005. 2

[82] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. 124

[83] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010. 29

[84] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67(2):159–188, 2006. 15, 27

[85] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 64, 89, 93

[86] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973. 4

[87] Robert B Fisher. *From surfaces to objects: computer vision and three dimensional scene analysis*. Wiley New Jersey, 1989. 28

[88] David Fleet and Yair Weiss. Optical flow estimation. In *Handbook of mathematical models in computer vision*, pages 237–257. Springer, 2006. 34

[89] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003. 3, 28

[90] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach (2nd Edition)*. Pearson, 2012. 1

[91] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: a survey. *Computer Vision and Image Understanding*, 134:1–21, 2015. 25, 62

[92] Katerina Fragkiadaki, Geng Zhang, and Jianbo Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1846–1853. IEEE, 2012. 87, 89, 93, 95, 96

[93] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. 30

[94] Fabio Galasso, Naveen Shankar Nagaraja, Tatiana Jimenez Cardenas, Thomas Brox, and Bernt Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3527–3534, 2013. 58

[95] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001. 4

[96] James J Gibson. *The perception of the visual world*. Houghton Mifflin, 1950. 31

[97] Joel Gibson and Oge Marques. Optical flow and trajectory methods in context. In *Optical Flow and Trajectory Estimation Methods*, pages 9–23. Springer, 2016. 63

[98] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 15, 27, 28, 124

[99] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970. 43

[100] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001. 15

[101] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. 1, 5, 15, 21, 22, 23, 25

[102] Amit Gruber and Yair Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 707–714. IEEE, 2004. 108

[103] Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, pages 1–7, 2017. 28, 124

[104] Maya R Gupta, Yihua Chen, et al. Theory and use of the em algorithm. *Foundations and Trends® in Signal Processing*, 4(3):223–296, 2011. 44, 78

[105] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006. 127

[106] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 18, 57

[107] Martin T Hagan, Howard B Demuth, Mark H Beale, et al. *Neural network design*, volume 20. Pws Pub. Boston, 1996. 30

[108] Gregory D Hager and Peter N Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE transactions on pattern analysis and machine intelligence*, 20(10):1025–1039, 1998. 25

[109] Nazrul Haque, N. Dinesh Reddy, and K. Madhava Krishna. Joint semantic and motion segmentation for dynamic scenes using deep convolutional networks. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017) - Volume 5: VISAPP*, pages 75–85, 2017. 124

[110] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985. 22

[111] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*. Citeseer, 1988. 39

[112] Richard Hartley and René Vidal. The multibody trifocal tensor: Motion segmentation from 3 perspective views. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 769–775. IEEE, 2004. 108

[113] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2, 56

[114] M Hassaballah, Aly Amin Abdelmgeid, and Hammam A Alshazly. Image features detection, description and matching. In *Image Feature Detectors and Descriptors*, pages 11–45. Springer, 2016. 17, 18, 19

[115] Katherine A Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304. ACM, 2005. 45

[116] Donald D Hoffman and Whitman A Richards. Parts of recognition. *Cognition*, 18(1):65–96, 1984. 28

[117] Berthold Horn. *Robot vision*. MIT press, 1986. 1

[118] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981. 25, 31, 32

[119] Steven L Horowitz and Theodosios Pavlidis. Picture segmentation by a tree traversal algorithm. *Journal of the ACM (JACM)*, 23(2):368–388, 1976. 23

[120] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(3):334–352, 2004. 7

[121] Shih-Shinh Huang, Li-Chen Fu, and Pei-Yung Hsiao. Region-level motion-based background modeling and subtraction using mrfs. *Image Processing, IEEE Transactions on*, 16(5):1446–1456, 2007. 26, 56

[122] David H Hubel, Janice Wensveen, and Bruce Wick. *Eye, brain, and vision*. Scientific American Library New York, 1995. 1

[123] Michal Irani and P Anandan. About direct methods. In *International Workshop on Vision Algorithms*, pages 267–277. Springer, 1999. 25, 26, 55, 63

[124] Anil K Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20, 2004. 2

[125] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017. 170, 171

[126] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 28

[127] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 30

[128] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document, 1996. 29

[129] Pierre-Marc Jodoin, Sébastien Piérard, Yi Wang, and Marc Van Droogenbroeck. *Overview and benchmarking of motion detection methods*, chapter 24, pages 24–1–24–26. Chapman and Hall/CRC, 2014. 62

[130] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009. 19

[131] Takeo Kanade. Picture processing system by computer complex and recognition of human faces. *Doctoral dissertation, Kyoto University*, 3952:83–97, 1973. 4

[132] Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513. IEEE, 2004. 19

[133] Frank Kleibergen and Richard Paap. Generalized reduced rank tests using the singular value decomposition. *Journal of econometrics*, 133(1):97–126, 2006. 43

[134] Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE transactions on automatic control*, 25(2):164–176, 1980. 43

[135] Kristin Koch, Judith McLean, Ronen Segev, Michael A Freed, Michael J Berry, Vijay Balasubramanian, and Peter Sterling. How much the eye tells the brain. *Current biology : CB*, 16:1428–1434, 08 2006. 1

[136] Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984. 18

[137] Konstantinos Konstantinides, Balas Natarajan, and Gregory S Yovanof. Noise estimation and filtering using block-based singular value decomposition. *IEEE Transactions on Image Processing*, 6(3):479–483, 1997. 43

[138] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 10, 15, 28, 29, 30

[139] Gaurav Kumar and Pradeep Kumar Bhatia. A detailed review of feature extraction in image processing systems. In *Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on*, pages 5–12. IEEE, 2014. 16, 18

[140] M Pawan Kumar, Philip HS Torr, and Andrew Zisserman. Learning layered motion segmentations of video. *International Journal of Computer Vision*, 76(3):301–319, 2008. 24, 27, 96

[141] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 15, 28

[142] Manuel Lang, Oliver Wang, Tunc Aydin, Aljoscha Smolic, and Markus Gross. Practical temporal consistency for image-based graphics applications. *ACM Transactions on Graphics (ToG)*, 31(4):34:1–34:8, 2012. 25, 26

[143] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289, 2008. 27

[144] Anat Levin and Yair Weiss. Learning to combine bottom-up and top-down segmentation. *Computer vision–ECCV 2006*, pages 581–594, 2006. 50

[145] Stan Z Li, RuFeng Chu, ShengCai Liao, and Lun Zhang. Illumination invariant face recognition using near-infrared images. *IEEE Transactions on pattern analysis and machine intelligence*, 29(4):627–639, 2007. 10

[146] Zhe Lin and Larry S Davis. A pose-invariant descriptor for human detection and segmentation. In *European Conference on Computer Vision*, pages 423–436. Springer, 2008. 15

[147] Zheng Lin, Jesse Jin, and Hugues Talbot. Unseeded region growing for 3d image segmentation. In *Selected papers from the Pan-Sydney workshop on Visualisation-Volume 2*, pages 31–37. Australian Computer Society, Inc., 2000. 23

[148] Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994. 18

[149] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011. 26, 65

[150] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 27, 124

[151] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 18, 37

[152] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 18, 37, 64, 68, 125, 126

[153] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, 1981. 25, 33, 35

[154] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE transactions on medical imaging*, 31(2):474–486, 2012. 15

[155] Y. Ma, S. Sastry, and R. Vidal. *Generalized Principal Component Analysis*. Interdisciplinary Applied Mathematics. Springer New York, 2015. 11, 12, 56, 57, 76, 107

[156] Yi Ma, Harm Derksen, Wei Hong, and John Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE transactions on pattern analysis and machine intelligence*, 29(9), 2007. 27, 96

[157] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012. 57

[158] Muhammad Habib Mahmood, Yago Díez, Joaquim Salvi, and Xavier Lladó. A collection of challenging motion segmentation benchmark datasets. *Pattern Recognition*, 61:1–14, 2017. 10

[159] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11, 2009. 22

[160] Davide Maltoni, Dario Maio, Anil Jain, and Salil Prabhakar. *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009. 2

[161] Ana I Maqueda, Carlos R del Blanco, Fernando Jaureguizar, and Narciso García. Human–computer interaction based on visual hand-gesture recognition using volumetric spatiograms of local binary patterns. *Computer Vision and Image Understanding*, 141:126–137, 2015. 6

[162] Jerry B Marion. *Classical dynamics of particles and systems*. Academic Press, 2013. 40

[163] Richard H Masland and Paul R Martin. The unsolved mystery of vision. *Current Biology*, 17(15):R577–R582, 2007. 1

[164] Yoshio Matsumoto and Alexander Zelinsky. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 499–504. IEEE, 2000. 4

[165] Brendan McCane, Kevin Novins, D Crannitch, and Ben Galvin. On benchmarking optical flow. *Computer Vision and Image Understanding*, 84(1):126–143, 2001. 25, 62

[166] Carl D Meyer. *Matrix analysis and applied linear algebra*, volume 71. Siam, 2000. 43

[167] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630, 2005. 18, 19

[168] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72, 2005. 25

[169] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):696–710, 1997. 4

[170] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE transactions on pattern analysis and machine intelligence*, 23(4):349–361, 2001. 4

[171] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009. 19

[172] Greg Mori, Xiaofeng Ren, Alexei A Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR'04*, pages 326–333. IEEE, 2004. 27

[173] Neil Muller, Lourenço Magaia, and Ben M Herbst. Singular value decomposition, eigenfaces, and 3d reconstructions. *SIAM review*, 46(3):518–545, 2004. 43

[174] Joseph L Mundy. Object recognition in the geometric era: A retrospective. In *Toward category-level object recognition*, pages 3–28. Springer, 2006. 2, 28

[175] Mary Natrella. *NIST/SEMATECH e-handbook of statistical methods*. NIST/SEMATECH, 2010. 83

[176] Mark S Nixon and Alberto S Aguado. *Feature extraction & image processing for computer vision*. Academic Press, 2012. 1

[177] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1187–1200, 2014. 54, 62, 87, 88, 89, 96, 102, 103

[178] Timo Ojala, Matti Pietikainen, and David Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 582–585. IEEE, Oct 1994. 19

[179] Aude Oliva and Antonio Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006. 17

[180] Yuri Ostrovsky, Ethan Meyers, Suma Ganesh, Umang Mathur, and Pawan Sinha. Visual parsing after recovery from blindness. *Psychological Science*, 20(12):1484–1491, 2009. 124

[181] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993. 22

[182] Zailiang Pan and Chong-Wah Ngo. Selective object stabilization for home video consumers. *IEEE Trans. Consumer Electronics*, 51(4):1074–1084, 2005. 27

[183] Megha Pandey and Svetlana Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *2011 International Conference on Computer Vision*, pages 1307–1314. IEEE, 2011. 15, 27, 29, 49, 124

[184] George Papandreou, Liang-Chieh Chen, Kevin P. Murphy, and Alan L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1742–1750, Washington, DC, USA, 2015. IEEE Computer Society. 27, 124

[185] Deepak Pathak, Ross B Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6024–6033, 2017. 124

[186] Alexander Patterson, Philippos Mordohai, and Kostas Daniilidis. Object detection from large-scale 3d datasets using bottom-up and top-down descriptors. In *European Conference on Computer Vision*, pages 553–566. Springer, 2008. 50

[187] Alex Pentland, Baback Moghaddam, Thad Starner, et al. View-based and modular eigenspaces for face recognition. In *CVPR*, volume 94, pages 84–91, 1994. 4

[188] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016. 10

[189] Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS Comput Biol*, 4(1):e27, 2008. 3

[190] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. 6

[191] Marcus E Raichle. A brief history of human brain mapping. *Trends in neurosciences*, 32(2):118–126, 2009. 1

[192] Christopher Rasmussen and Gregory D. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):560–576, 2001. 26, 96

[193] Yogesh Rathi, Namrata Vaswani, Allen Tannenbaum, and Anthony Yezzi. Tracking deforming objects using particle filtering for geometric active contours. *IEEE transactions on pattern analysis and machine intelligence*, 29(8), 2007. 26, 49, 96

[194] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 512–519. IEEE, 2014. 30

[195] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, volume 1, pages 10–17, 2003. 15

[196] Xiaofeng Ren and Deva Ramanan. Histograms of sparse codes for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3246–3253, 2013. 18

[197] Robot Vacuum Cleaner Reviews. Robot vacuum cleaner navigation. `http://www.robotvacuumcleaner.org`. 8

[198] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, pages 41–46. IBM, 2001. 29

[199] Lawrence Gilman Roberts. *Machine perception of three-dimensional soups*. PhD thesis, Massachusetts Institute of Technology, 1963. 20

[200] Michael G Ross and Leslie Pack Kaelbling. Learning static object segmentation from motion segmentation. In *American Association for Artificial Intelligence*, 2005. 124

[201] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–119, 2010. 20

[202] Peter M Roth and Martin Winter. Survey of appearance-based methods for object recognition. *Inst. for Computer Graphics and Vision, Graz University of Technology, Austria, Technical Report ICGTR0108 (ICG-TR-01/08)*, 2008. 28, 29

[203] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998. 4

[204] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 124

[205] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009. 29, 45

[206] M. S. Ryoo and J. K. Aggarwal. UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA). `http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html`, 2010. 6

[207] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision*, 80(1):72–91, 2008. 26, 63, 65

[208] Konrad Schindler, U James, and Hanzi Wang. Perspective n-view multibody structure-and-motion through model selection. In *European Conference on Computer Vision*, pages 606–619. Springer, 2006. 108

[209] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 30

[210] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 3, pages 32–36. IEEE, 2004. 6

[211] Shokri Z Selim and Mohamed A Ismail. K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–87, 1984. 44

[212] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR2014)*, 2014. 15

[213] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 2809–2813. IEEE, 2011. 2, 15

[214] Mehmet Sezgin et al. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168, 2004. 22

[215] Mohsen Sharifi, Mahmood Fathy, and Maryam Tayefeh Mahmoudi. A classified and comparative study of edge detection algorithms. In *Proceedings. International Conference on Information Technology: Coding and Computing*, pages 117–120. IEEE, April 2002. 22

[216] Jianbo Shi et al. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. IEEE, 1994. 25

[217] Jianbo Shi and Jitendra Malik. Motion segmentation and tracking using normalized cuts. In *Sixth International Conference on Computer Vision*, pages 1154–1160. IEEE, 1998. 26, 49

[218] Yang Shi, Ji Huang, and Bo Yu. Robust tracking control of networked control systems: application to a networked dc motor. *IEEE Transactions on Industrial Electronics*, 60(12):5864–5874, 2013. 27

[219] GT Shrivakshan, C Chandrasekar, et al. A comparison of various edge detection techniques used in image processing. *IJCSI International Journal of Computer Science Issues*, 9(5):272–276, 2012. 22

[220] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997. 20

[221] Elizabeth S Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990. 124

[222] Andrew N Stein and Martial Hebert. Combining local appearance and motion cues for occlusion boundary detection. In *BMVC*, pages 1–10, 2007. 124

[223] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. 62

[224] Deqing Sun, Stefan Roth, J. P. Lewis, and Michael J. Black. Learning optical flow. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 83–97, Berlin, Heidelberg, 2008. Springer-Verlag. 25

[225] Deqing Sun, Erik B Sudderth, and Michael J Black. Layered segmentation and optical flow estimation over time. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1768–1775. IEEE, 2012. 27

[226] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision*, pages 438–451. Springer, 2010. 26, 65, 70, 89

[227] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 2, 3, 4, 5, 15, 17, 18, 20, 25, 27, 28, 29, 43, 127

[228] A Murat Tekalp. *Digital video processing*. Prentice-Hall, Inc., 1995. 26, 55, 61, 75

[229] A Murat Tekalp. *Digital video processing*. Prentice Hall Press, 2015. 63

[230] Joseph Tighe and Svetlana Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3001–3008, 2013. 21

[231] Philip HS Torr. Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998. 108

[232] Philip HS Torr, Richard Szeliski, and P Anandan. An integrated bayesian approach to layer extraction from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):297–303, 2001. 57

[233] Philip HS Torr and Andrew Zisserman. Feature based methods for structure and motion estimation. In *International workshop on vision algorithms*, pages 278–294. Springer, 1999. 25, 26, 55, 63

[234] Marco Alexander Treiber. *An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010. 2, 28

[235] Øivind Due Trier, Anil K Jain, and Torfinn Taxt. Feature extraction methods for character recognition-a survey. *Pattern recognition*, 29(4):641–662, 1996. 2

[236] Roberto Tron and René Vidal. A benchmark for the comparison of 3-D motion segmentation algorithms. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 24, 27, 53, 54, 57, 87, 93, 116

[237] Emanuele Trucco and Konstantinos Plakas. Video tracking: a concise survey. *IEEE Journal of Oceanic Engineering*, 31(2):520–529, 2006. 9

[238] Eduard Trulls, Iasonas Kokkinos, Alberto Sanfeliu, and Francesc Moreno-Noguer. Dense segmentation-aware descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2890–2897, 2013. 15

[239] Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of computer vision*, 63(2):113–140, 2005. 50

[240] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008. 5, 6, 7

[241] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991. 4

[242] Tinne Tuytelaars, Krystian Mikolajczyk, et al. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008. 18, 20, 25, 63

[243] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. 46

[244] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear (tensor) image synthesis, analysis, and recognition [exploratory dsp]. *IEEE Signal Processing Magazine*, 24(6):118–123, 2007. 43

[245] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008. 68

[246] Rodrigo Verschae and Javier Ruiz-del Solar. Object detection: current and future directions. *Frontiers in Robotics and AI*, 2:29, 2015. 2, 3, 28

[247] René Vidal and Richard Hartley. Motion segmentation with missing data using PowerFactorization and GPCA. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR'04*, pages 310–316. IEEE, 2004. 26, 27, 87, 89, 93, 96

[248] René Vidal and Shankar Sastry. Segmentation of dynamic scenes from image intensities. In *Proceedings of the Workshop on Motion and Video Computing*, MOTION'02, pages 44–49. IEEE, 2002. 57

[249] René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *International Journal of Computer Vision*, 79(1):85–105, 2008. 54, 108

[250] Sebastian Volz, Andres Bruhn, Levi Valgaerts, and Henning Zimmer. Modeling temporal coherence for optical flow. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV'11, pages 1116–1123. IEEE, 2011. 26

[251] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer, 2003. 43

[252] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013. 26

[253] John YA Wang and Edward H Adelson. Layered representation for motion analysis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 361–366. IEEE, June 1993. 27

[254] John YA Wang and Edward H Adelson. Representing moving images with layers. *Image Processing, IEEE Transactions on*, 3(5):625–638, 1994. 27

[255] Xiang-Yang Wang, Ting Wang, and Juan Bu. Color image segmentation using pixel wise support vector machine classification. *Pattern Recognition*, 44(4):777–787, 2011. 28

[256] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 393–400. IEEE, 2014. 52

[257] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013. 25, 26, 65

[258] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Learning to detect motion boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2578–2586, 2015. 65

[259] Yair Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, CVPR '97, pages 520–526. IEEE, 1997. 26, 96

[260] Yair Weiss and Edward H Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 321–326. IEEE, 1996. 26

[261] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 30

[262] Josh Wills, Sameer Agarwal, and Serge Belongie. A feature-based approach for dense segmentation and estimation of large disparity motion. *International Journal of Computer Vision*, 68(2):125–143, 2006. 57

[263] Patrick Henry Winston. The psychology of computer vision. *Pattern Recognition*, 8:193, 1976. 2

[264] Laurenz Wiskott, Norbert Krüger, N Kuiger, and Christoph Von Der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):775–779, 1997. 4

[265] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer vision and pattern recognition workshops (CVPRW), 2012 IEEE computer society conference on*, pages 20–27. IEEE, 2012. 10

[266] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision–ECCV 2006*, pages 94–106. Springer, 2006. 27, 89, 93, 96, 108

[267] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1794–1801. IEEE, 2009. 127, 128, 129

[268] Zhuoran Yang, Zhaoran Wang, Han Liu, Yonina C. Eldar, and Tong Zhang. Sparse nonlinear regression: Parameter estimation under nonconvexity. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *ICML'16*, pages 2472–2481. JMLR.org, 2016. 27

[269] Kin Choong Yow and Roberto Cipolla. Feature-based human face detection. *Image and vision computing*, 15(9):713–735, 1997. 2, 21

[270] Luca Zappella, Xavier Lladó, and Joaquim Salvi. Motion segmentation: a review. In *Proceedings of the 2008 conference on Artificial Intelligence Research and Development*, pages 398–407. IOS Press, 2008. 21, 26, 27

[271] Luca Zappella, Xavier Llado, and Joaquim Salvi. New trends in motion segmentation. In *Pattern Recognition*, chapter 3. IntechOpen, Rijeka, 2009. 26, 76

[272] Lihi Zelnik-Manor, Moshe Machline, and Michal Irani. Multi-body factorization with uncertainty: Revisiting motion consistency. *International Journal of Computer Vision*, 68(1):27–41, 2006. 26, 108

[273] Bo Zhang. Computer vision vs. human vision. In *9th IEEE International Conference on Cognitive Informatics (ICCI'10)*, pages 3–3. IEEE, July 2010. 1

[274] Liang Zhao and Larry S Davis. Closely coupled object detection and segmentation. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, volume 1 of *ICCV '05*, pages 454–461. IEEE, 2005. 50

[275] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003. 3

[276] Hongyuan Zhu, Fanman Meng, Jianfei Cai, and Shijian Lu. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, 34:12–27, 2016. 10, 16, 27, 28, 124

SUMMARY

This thesis addresses unsupervised video object segmentation based on motion information. As the motion information is implicit in a video sequence, we focus on acquiring, analyzing and utilizing motion data from a given video sequence. This research focuses on processing video and image data at different levels, using techniques and approaches from different fields. Chapter 2 and Chapter 3 give the introduction of related research fields, as well as the involved methodologies and techniques. Chapter 4 presents a general framework for addressing the research problem. In Chapter 5, we address the low-level image processing and discuss three types of motion data that can be extracted from a video sequence. The obtained motion data represent the 2D motions in the video frames. A motion segmentation algorithm is presented in Chapter 6, to segment the video frames based on the 2D motion data. In Chapter 7, we analyze the 3D motion consistency implied in the obtained 2D motions. Chapter 8 investigates how to segment non-moving objects based on information that was learned when the objects were moving. Chapter 8 also discusses the object identification based on motion segmentation.

In Chapter 4, we analyze the general pipeline of processing video sequences, for the purpose of video object segmentation. We investigate how to extract the implicit motion information hinted in the changes of video frames without using prior knowledge. A framework is presented for the system design that combines a bottom-up and a top-down scheme. At the bottom-up step, we investigate the extraction of motion data from the lower-level image features and the utilization of the obtained motion data for grouping image features into higher-level object descriptions. The top-down step then learns to segment non-moving objects by using the obtained object descriptions as the prior knowledge. Based on the scheme, we present three fundamental modules for building up the video object segmentation system: one for motion data extraction, one for motion based video objects segmentation, and one for non-moving object segmentation. We describe the explicit definitions of related terminologies in this system, and discussed the keypoints to be noticed for each module. We also introduced the datasets used in our experiments.

Chapter 5 focuses on the extraction of motion data based on the low level image features. We extract and track some salient image features in a video sequence, and represent the motion data as their position changes in the video frames. We investigate two ways for motion data extraction: one focuses on estimating the motion of elementary image pixels, another takes some local image patterns as the feature points to be tracked. Three approaches are described in this chapter. One of them focuses on the pixel-based motion estimation by using the optical flow technique. The other two approaches both extracts SIFT features from the images and tracks their movements, but differ in the methods used for feature extraction. One of the feature-based methods only focus on a sparse set of features in the images, while the other one adopts a dense field of features for tracking. As a result, three types of motion data are obtained from our data sets. As these data are used as the input for next step, i.e the motion segmentation, we can not conclude that which is

a better choice for our system in this chapter.

In Chapter 6, we investigate the segmentation of moving objects given the obtained motion data. An unsupervised 2D based motion segmentation algorithm is proposed. This algorithm focuses on segmenting the pixels or feature points into groups that undergo the unique 2D motion, based on the given motion data. Based on the assumption that points from the same object undergo the same motion, the segmented groups can be regarded as objects in the given videos. We model the points' motion between two successive image frames by a 2D affine transformation, and use a classification EM algorithm to segment points into groups of the same motion. Bayesian update is used to propagate the segmentation from one frame pair to the next. Thus the segmentation of the entire video sequence is the combination of segmentation of successive frames. In this way, the segmentation algorithm takes the motion information from the beginning to the end of a video sequence into account. The algorithm also automatically estimates the number of objects. The method has high flexibility in dealing with different types of motion data. We measure the reliabilities of the segmentation results and propose an optional step for compensating the camera motion, which leads to a different version of the proposed algorithm. The experimental results on the three types of motion data, which are described in Chapter 5, show that the performance of the proposed method is comparable with the state-of-the-art methods. The results also suggest that the proposed method prefers the motion data obtained by the dense feature tracking approach, on which the performance is more outstanding.

Chapter 7 complements Chapter 6, by analyzing the inherent 3D consistency in the obtained motion data, which is essentially the 2D movements in the successive images of a video sequence. As the 2D motions in images are a projection of the 3D motions in the real world carried out by a camera system, the assumption that the motion of points of an object can be described by an affine transformation, can be violated and leads to oversegmentation. Two theorems are presented in this chapter, which describes the properties of a given set of points from an object and their projected movements between two images. These theorems can effectively measure the 3D rigid motion consistency of a group of 2D projection points, and also estimate the original 3D body movement with one degree of freedom. Experiments shows that the segmentation obtained by our algorithm presented in Chapter 6 can be improved by using the proposed theorems. Unfortunately the recovering of 3D motion models only works well for the rotation about the z direction in the camera system. Nevertheless, the proposed theorems show promising potential, and further research is worth to take in the future.

Chapter 8 addresses the non-moving object segmentation, when motion segmentation is no longer possible. Without prior knowledge, we use the motion segmentation results to learn object descriptions from moving objects. These object descriptions are used to segment objects when they are no longer moving. We present a static object segmentation method based on the SIFT matching strategy, which obtains the segmentation by comparing the image data with the learned object descriptions. Additionally, we investigate object identification given the segmentation results based on a sparse coding algorithm. It addresses the problem of identifying the same objects without using the motion clue, and evaluates the quality of the segmented objects. The experimental results show that the non-moving object segmentation relies on the quantity of learning data that is obtained

by motion segmentation. With more images in the motion segmentation results, the non-moving object segmentation performs better. The results also show that the segmented objects can effectively represent the ground-truth, as they can be identified even with errors in the segmentation results.

Finally in Chapter 9, we discuss the results with respect to the research questions. We also present several directions for further research.

ADDENDUM: VALORIZATION

Valorization is "the process of creating value from knowledge, by making knowledge suitable and/or available for social (and/or economic) use and by making knowledge suitable for translation into competitive products, services, processes and new commercial activities" (Maastricht promotie regelement, 2013). Since 2013, an addendum about valorization to the PhD dissertation is required by Maastricht University. This chapter will comply with this requirement by addressing the potential value and social relevance of the work described in this thesis. It does not form as a part of the dissertation, and should not be assessed as part of the dissertation.

Since the digital era is coming, extremely large and increasing amounts of data and informatics are produced in the economy and society. However, grasping of the useful information from a massive data is beyond the ability of humankind. In modern society, computers and machines take the work of transferring the massive, redundant digital data into high-level information that can be understand by human. To deal with visual data, such as digital images and videos, computer vision systems have been broadly and extensively investigated in both academic and industrial communities. Indeed, an ever-increasing variety of computer vision products and services in industry have been created, as the computer vision technology become more mature. From machine inspection to video surveillance, from medical image analysis to unmanned vehicles, from robots industrial to intelligent man-machine communication, etc., computer vision technology has greatly benefited the modern society. These technologies enable human to acquire useful information from videos without watching them thoroughly—by just clicking the mouse.

This thesis address an fundamental problem in computer vision and video processing: the unsupervised video object segmentation. Video object segmentation aims at extracting and analyzing object-level information from videos that can be interfaced with other thought processes and elicit appropriate action. Such techniques supplement the traditional object detection and/or recognition technology that is based on supervised machine learning. In Section 1.1, we introduced some applications of the traditional object detection and recognition technology in existing society. This thesis is innovative in respect to the existing products from the following aspects:

- This thesis solves the problem in an unsupervised way. It means that the program can automatically learn useful information from the videos. The traditional products require a large dataset of manually annotated images for training the learning model. The process of manually annotating images is fulfilled by human, which is a costly work.

- This thesis focuses on analyzing the motion information, which is a kind of feature that implied in the videos. The traditional products only analysis the visual features that are directly acquired from the videos. The motion information captures the

spatial coherence of image features in a video, and provides more clues for extracting more representative information.

In conclusion, the results of this thesis addresses some short-comes of the traditional products in real-life, with potential for further development.

In general, the research results of this thesis are valuable for both academic and non-academic audiences. It can be applied in computer vision applications that requires generating and analyzing object-level information from digital videos or a sequence of images. In the following sections, we will introduce two non-academic domains that can be benefit from the research results of this thesis, as well as the representative products and/or services of these applications.

Autonomous Vehicles

Computer vision technology has been widely applied in the navigation systems for autonomous vehicles. Autonomous navigation has been extensively investigated and applied in the non-academic domain, due to its economic and social potentials. Autonomous navigation requires the vehicles or robots to perceive and understand the environment. As video data is the primary resources for autonomous navigation systems, computer vision currently plays the key role in a perception systems for autonomous vehicles or robots [125].

Figure 9.1: Object detection and recognition helps for understanding the environment in autonomous navigation: a snapshot of the vision system of Tesla Autopilot in driving [1].

A variety of projects has been started worldwide to explore intelligent transportation systems (ITS), by many governmental institutions and industrial groups. In Europe, the PROMETHEUS project started in 1986 was the largest project in the autonomous driving field, and defined the state of the art of autonomous vehicles [125]. Numerous universities and car manufactures, as well as research units from governments of 19 European countries participated in this project. The U.S. government established the National Automated Highway System Consortium (NAHSC) in 1995. Meanwhile Japan established the

Advanced Cruise-Assist Highway System Research Association in 1996 [25]. In the 21 century, several prototype vehicles have been developed and tested in real world, including ARGO, BRAiVE. The onboard system allows to detect obstacles, lane marking, ditches, berms and identify the presence and position of a preceding vehicle [125]. Google also started their self-driving car project in 2009. Their self-driving system equipped with different sensors, such as cameras, radars, LiDAR, wheel encoder and GPS, and can detect pedestrians, cyclists, vehicles, road work and more in all directions. Besides, many car manufactures, such as Tesla, GE, Toyota, etc., have established projects and institutes for developing autonomous driving cars. Although great progress has been made, the fully autonomous navigation cars is still in laboratory, as most of the existing computer vision systems produce errors at a rate which is not acceptable for the safety consideration. Still the achievements for autonomous vision, which refers to the computer vision technology in a autonomous navigation system, have benefited in the advanced driver assistant systems. More and more modern cars are equipped with the advanced driver assistant system. These systems help the driver in the driving process, and increase the car safety and the road safety. Some successful implementations are the Tesla Autopilot, Nissan ProPilot Assist, Mobileye, etc.

Besides the ground vehicles, like cars, computer vision system is also applied for autonomous vehicles used in different environments, such as the unmanned aerial vehicle (UAV) and autonomous underwater vehicle or robot. Vision systems for these vehicles need to deal with specific environment, and focus on specific features that are different from the ground objects as cars meet. UAVs move in 3D space and at high attitude, which requires to process videos with high resolution. UAVs have been developed and deployed many countries around the world, for applications in civilian, commercial, military, and aerospace domains. AUVs are created for various of undersea applications, such as inspection of sunken ships, sea life monitoring, military missions, undersea infrastructures or installations inspection and maintenance, etc. Vision system for AUVs deals with underwater environments, where the video quality is easily affected by the muddy or turbid waters . Currently the AUVs navigation systems are often the combination of the sonar based system and the vision based system [25].

Video Surveillance

Video surveillance systems are used to monitor security sensitive areas, such as banks, department stores, highways, crowded public places and borders. The implementation of artificial intelligence and computer vision technologies makes the video surveillance being smart: computer vision system takes over the work of human operator for tracking and detecting suspicious objects.

Video surveillance system needs to be sensitive to moving objects and providing automatic alarming function. The ability of segmentation, detection and recognition of moving objects is therefore required for an intelligent video surveillance system. Numerous of governmental institutes, universities, companies from worldwide have involved in video surveillance projects, and a variety of products and services have been produced and used in our daily life. The technologies for specific objects, such as face detection and recognition, people detection and tracking, are widely used in commercial products and offer

[1] https://www.tesla.com/autopilot

Figure 9.2: iRadar's iSense^{TM} Smart Video Surveillance System

reliable security solutions for the society. Nowadays, numerous smart security cameras are available in markets, provide high quality intelligence security services for families with a low cost, such as the Amazon Cloud Cam, the Ring Spotlight Cam, iBaby Monitor, etc.

ACKNOWLEDGMENTS

There are many people and institutes that have earned my gratitude for helping me to achieve this thesis.

First and foremost, I would like to express my sincere appreciation to my supervisors. My PhD dissertation would not have been possible without the support and nurturing of my supervisors. Nico, I am extremely grateful to for your guidance in the past seven years. You are always patient and supportive, to guide me with my research. I appreciate that you were never stop reviewing my papers when you got injured. Thank you as well for mentoring me on the precise scientific writing. With your help, I am growing as a research scientist. I am deeply indebted to Ralf for your insightful comments and effective advice in mathematics and thesis writing, especially your contributions in the theoretical part of my thesis. And most importantly, I appreciate your encouragement when I almost lose my confident during the process of thesis writing.

I would also like to express my appreciation to all the members of my assessment committee, for your time and valuable comments. Thank you all for your participation and contribution, to the special day of my life.

As my thesis was carried out at the Department of Data Science and Knowledge Engineering (DKE) in Maastricht University, I would like to thank DKE for offering the opportunity and providing me with a wide range of academic resources. I must also thank the China Scholarship Council (CSC) for financial supporting of my PhD research.

I would express my gratitude to the excellent research fellows at DKE that have taken some time to discuss and enrich my work. Some special words of thanks goes to Gijs, Rachel and Katerina, for your kind help of composing better sentences in propositions. Particularly helpful to me during this time was Haitham, with whom I published the first conference paper, and thanks for your brilliant ideas.

I am very grateful to the support staffs at DKE, KCIS and FSE Local Support, who help me a lot in the non-academic area. I would like to extend my sincere thanks to Jet, for your solicitude and encouragement, which had helped me to get through the hard times.

Special thanks to Katharina and Yiyong, for helping me with preparing the defense and being my paranymphs during the ceremony.

I would like to thank all my PhD colleagues, with whom I have shared unforgettable moments. Zhenglong, Frederik, Wenzhao, Siqi, Yiyong, Arjun, Monica and Maria, many thanks to you for the good moments shared in the office, and for the inspiration discussions. I am also grateful to Seethu, Amir, Kirill and Jordy, for the joyful moments of playing games. Bijian, Nasser, Firat and Chiara, I would like to thank you for the lunch-times and interesting discussions we have had. Katharina and Lucas, thank you for cheering me up, and for the invitations to many great events. Some special words of gratitude go to Hua, Li and Shuang, who have always been supporting me. I appreciate for your generous friendship, and for the life times we have spent together.

I am also grateful to my friends in Maastricht, Ai, Wenqing, Xiahong, Haiyan, Rui, Tianxiang, Yuzhe, Jie, Pei, Yuan, Mengmeng, Mengxing, Ancui, Yu, Xi and Tian, who have brought me precious memories of life in this city. I would like to acknowledge the help of Weiwei, for your useful advice and cocktails. Thanks should also goes to Yin and Shuang, thank you for your warm hospitality and the true Sichuan dishes.

Finally, my dearest Mom and Dad, thank you for showing faith in me and supporting me.

Wei Zhao
February 2019
Maastricht

ABOUT THE AUTHOR

Wei Zhao was born in April 10, 1985 in Sichuan, China. She received her B.Sc. degree in Computer Science and Technology at 2007, from the University of Electronic Science and Technology of China (UESTC). From 2008, she started her Master study in the same university and became a member of the Laboratory of Digital Media Technology in School of Computer Science and Engineering, UESTC. During her master studies, she joined the project of developing developing a real-time 3D computer graphics engine, which is supported by the National High-tech Research and Development Program of China (grant NO.2006AA01Z335), and led a team of 5 master students. She obtained her B.Sc. in Computer Application Technology in July 2007.

In January 2012, she started a PhD project at the Department of Data Science and Knowledge Engineering (DKE), Maastricht University. Since then she has been a member of Robotics, Agents and Interactions group (RAI). Her research was founded by the China Scholarship Council. During her PhD studies, she focused on object detection for a computer vision system, which is supported by the Robotlab project. Her scientific research contributes have led publications in scientific conferences and finally this dissertation.

LIST OF PUBLICATIONS

Wei Zhao, H Bou Ammar, and Nico Roos. Dynamic object recognition using sparse coded three-way conditional restricted boltzmann machines. In *25th Benelux Conference on Artificial Intelligence (BNAIC 2013)*, 2013.

Wei Zhao and Nico Roos. An EM based approach for motion segmentation of video sequence. In *24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 61–69, 2016.

Wei Zhao and Nico Roos. Motion based segmentation for robot vision using adapted em algorithm. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: VISAPP, (VISIGRAPP 2016)*, pages 649–656. INSTICC, SciTePress, 2016.

Wei Zhao, Nico Roos, and Ralf Peeters. 3D motion consistency analysis for segmentation in 2d video projection. In *Computer Analysis of Images and Patterns: 17th International Conference, CAIP 2017, Ystad, Sweden, August 22-24, 2017, Proceedings, Part II*, pages 440–452. Springer International Publishing, Cham, 2017.

Wei Zhao and Nico Roos. Moving object detection in video sequence. submitted, 2018.

Wei Zhao and Nico Roos. Detecting objects by learning from motion segmentation results. Unpublished Manuscript, 2019.