MANIFOLD CLUSTERING FOR MOTION SEGMENTATION

Luca ZAPPELLA

Dipòsit legal: GI-1083-2011
http://hdl.handle.net/10803/34765

Universitat de Girona

Ph.D. Thesis

Manifold Clustering for Motion Segmentation

Luca Zappella

2011

Thesis submitted for the degree of

Ph.D. in Technology

Thesis Supervisors: Dr. Xavier Lladó and Prof. Joaquim Salvi

Authorship Declaration

I hereby declare that his thesis contains no material which has been accepted for the award of any other degree or diploma in any university. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference has been made.

Luca Zappella

Abstract

In this study the problem of *motion segmentation* is discussed. Motion segmentation aims to decompose a video into the different objects that move throughout the sequence. In many computer vision algorithms this decomposition is the first fundamental step. It is an essential building block for robotics, inspection, video surveillance, video indexing, traffic monitoring and many other applications. The vast amount of literature on motion segmentation testifies to the relevance of the topic. However, the performance of most of the algorithms still falls far behind human perception.

In this thesis a review of the main motion segmentation approaches is presented. The main features of motion segmentation algorithms are analysed and a classification of the recent and most important techniques is proposed.

Specific attention is given to the motion segmentation algorithms that use feature trajectories. These algorithms assume the position of tracked points (features) in each frame of the video sequence as an input. The aim is to group together features that belong to the same motion. The main principle at the base of the proposed algorithms is that trajectories that belong to different motions span different subspaces. Therefore, if the subspace generated by each trajectory could be estimated, and if the different subspaces could be efficiently compared to measure their similarity, the segmentation problem could be cast into a manifold clustering problem.

In this study some of the most challenging issues related to trajectory motion segmentation via manifold clustering are tackled. Specifically, new algorithms for the estimation of the rank of the trajectory matrix are proposed. A new measure of similarity between subspaces is presented. Some problems related to the behaviour of principal angles are discussed. Furthermore, a generic tool for the estimation of the number of motions is also developed. The final algorithms that combine the previous proposals are among the very few in motion segmentation literature that do not require any prior knowledge nor manual tuning of their parameters.

Finally, the last part of the study is dedicated to the development of an algorithm for the correction of an initial motion segmentation solution. Such a correction is achieved by bringing together the motion segmentation and the structure from motion problems. The proposed solution not only assigns the trajectories to the correct motion, but it also estimates the 3D structure of the objects and fills the missing entries in the trajectory matrix.

All of the proposed algorithms are tested and compared with state of the art techniques on synthetic and real sequences. Tests show robust behaviour of the proposed algorithms and constant improvements over the state of the art.

Resum

En aquesta tesi s'estudia el problema de la segmentació del moviment (motion segmentation). L'objectiu de la segmentació del moviment és descompondre un vídeo en els diferents objectes que es mouen al llarg de la seqüència. Aquesta descomposició és un primer pas fonamental per a molts algoritmes de visió per computador, convertint-se en una part essencial de la robtica, la inspecció, la vdeo vigiláncia, la vídeo indexació, el seguiment de tráfic i moltes altres aplicacions. La gran quantitat de literatura en segmentació del moviment en testifica la seva rellevància, sense oblidar peró, que la percepció humana encara ofereix més garanties que els algoritmes existents.

La tesi presenta una revisió dels principals algoritmes de segmentació del moviment, s'analitzen les característiques principals i també es proposa una classificació de les tècniques més recents i importants.

S'ha donat una atenció especial als algoritmes de segmentació del moviment que utilitzen trajectòrias. Les trajectòrias estan formadas per punts de seguiments a cada imatge de la seqüència de vídeo. L'objectiu és agrupar trajectòrias que pertanyen al mateix moviment. Per tal d'aconseguir-ho, els algoritmes proposat utilitzen el fet que trajectòries que pertanyen a moviments diferents generen subespais diferents. Llavors, si s'estima el subespai generat per cada trajectòria i es mesura la similitud, el problema de la segmentació es pot entendre com un problema d'agrupament d'espais (manifold clustering).

Aquest estudi aborda alguns dels reptes més difícils pel que fa a la segmentació de trajectòries a través de l'agrupament d'espais. Concretament, s'han proposat nous algoritmes per a l'estimació del rang de la matriu de trajectòries, s'ha presenta una nova mesura de similitud entre subespais, s'han abordat problemes relacionats amb el comportament dels angles canònics i, també s'ha desenvolupat una eina genèrica per estimar quants moviments apareixen en una seqüència. Els algoritmes finals combinen les propostes prèvies, por lo tanto son uns dels pocs algoritmes, dins el camp de segmentació del moviment, que no requereixen cap coneixement a priori ni cap ajust manual.

Lúltima part de l'estudi es dedica a la correcció de l'estimació inicial d'una segmentació del moviment. Aquesta correcció es du a terme ajuntant els problemes de la segmentació del moviment i de l'estructura a partir del moviment. La solució proposada no només assigna las trajectòrias als moviments correctes, sinó que també estima l'estructura 3D dels objectes i ompla les entrades buides dins la matriu de trajectòries.

Els algoritmes proposats han estat analitzats i comparats amb tècniques de l'estat de l'art utilitzant seqüències sintètiques i reals. Els resultats demostrem un comportament robust dels algoritmes proposats i una constant millora respecte l'estat de l'art.

Resumen

Esta tesis estudia el problema de la segmentación del movimiento (motion segmentation). La segmentación del movimiento tiene como objetivo principal la descomposición de un vídeo en los diferentes objetos que se mueven a lo largo de la secuencia. Esta descomposición es un primer paso fundamental para muchos algoritmos de visión por computador, convirtiéndose en una parte esencial de la robótica, la inspección, la video vigilancia, la video indexación, el seguimiento del tráfico y muchas otras aplicaciones. La gran cantidad de literatura en segmentación del movimiento demuestra la relevancia del tema, aunque la percepción humana sigue ofreciendo más garantías que los algoritmos existentes.

En la presente tesis se realiza una revisión de los principales algoritmos de segmentación del movimiento, se analizan las caractersticas principales y se propone una clasificación de las técnicas más recientes e importantes.

Se ha prestado especial atención a los algoritmos de segmentación del movimiento que utilizan trayectorias. Dichas trayectorias están formadas por puntos de seguimiento en cada imagen de la secuencia de vídeo. El objetivo es agrupar trayectorias que pertenecen al mismo movimiento. Los algoritmos propuestos se basan en el hecho de que las trayectorias que pertenecen a diferentes movimientos generan subespacios diferentes. Estimando el subespacio generado por cada trayectoria y midiendo su similitud el problema de la segmentación del movimiento se puede entender como un problema de agrupamiento de espacios (manifold clustering).

En este estudio se analizan algunos de los retos más difíciles relacionados con la segmentación de trayectorias a través de la agrupación de espacios. En concreto, se proponen nuevos algoritmos para la estimación del rango de la matriz de trayectorias, se presenta una nueva medida de similitud entre subespacios, se abordan problemas relacionados con el comportamiento de los ángulos cannicos y, además, se desarrolla una herramienta genérica para estimar cuantos movimientos aparecen en una secuencia. Los algoritmos finales combinan las propuestas previas, por lo cual son unos de los pocos, dentro del campo de la segmentación del movimiento, que no requieren ningún conocimiento a priori ni ningún ajustamiento manual.

La última parte del estudio se centra en la corrección de la estimación inicial de una segmentación del movimiento. Esta corrección se lleva a cabo juntando los problemas de segmentación y de estructura a partir del movimiento. La solución final no solamente clasifica las trayectorias correctamente, sino que también estima la estructura 3D de objetos y llena las entradas vacías en la matriz de trayectorias.

Los algoritmos propuestos han sido probados y comparados con técnicas del estado del arte, utilizando tanto secuencias sintéticas como reales. Los resultados demuestran un comportamiento robusto de los algoritmos y una constante mejora sobre el estado del arte.

Acknowledgements

This thesis was made possible thanks to the contribution of many persons.

The help and guidance of my Ph.D. supervisors Dr. Xavier Lladó and Prof. Joaquim Salvi was essential for the completion of this work. I would like to thank also the reviewers who helped me to improve this thesis with precise and meaningful suggestions. An important role was played by Dr. Alessio Del Bue who supervised my work during the wonderful six months I spent in Lisbon, and then kept advising me after I moved back to Girona. A special thanks goes to Dr. Edoardo Provenzi whose friendship, mathematical knowledge, scientific advice, ping-pong thinking, awesome cooking, and brilliant film suggestions made this thesis more sound and fun.

I would like to thank also all the VICOROB group, especially the people of the P-IV lab. You have always been very kind and made me feel at home since the beginning. During these three years I have always felt "in my place" also thanks to the very good friends I made in Girona, inside and outside the lab. I really hope I will be able to keep in touch with you all.

I am grateful to all of the VIBOT family: secretaries, professors and students. Everything started back then during the masters, and this thesis is also the result of that experience. Among all of the students, I would like to thank especially Josep for his kindness (please you should swear sometimes, and without saying that's a joke!) and for sharing with me this masters plus Ph.D. experience, Mario, Stephen and Chloe for having made the masters so much fun and the Kiwis also for showing me the wonderful land of New Zealand.

Thanks to my flatmates Fanny and Jan for keeping up with my crazy rules in the house, and a special thanks to Jan for teaching me how to look at life with a positive light and do not start every sentence with "The problem is...".

I am so glad to thank Rebecca for bringing peace, warmness, love and a smile into my life. I have also to thank you for your infinite patience with my constant requests for English advice. After all your English is not that bad.

Without being very original I conclude my acknowledgements thanking my parents and my little brother for all their help, support and love. To you I dedicate this thesis.

Contents

List of Figures

List of Tables

Abbreviations and Notation

In alphabetical order.

- $\mathtt{A}$: affinity matrix, usually of size $P \times P$
- ALC: Agglomerative Lossy Compression
- ASA: Adaptive Subspace Affinity
- A-ASA: Automatic-ASA
- ELSA: Enhaned Local Subspace Affinity
- EMS: Enhanced Model Selection
- F: total number of frames
- $\mathtt{G}_n$: binary mask matrix; entries equal to 0 correspond to missing data, entries equal to 1 corresponds to known data
- GMC: Grassmannian Maximum Consensus
- GPCA: Generalized Principal Component Analysis
- k: parameter of the model selection formula
- k_g: parameter of the model selection formula for global space size estimation
- k_s: parameter of the model selection formula for local space size estimation
- LDA: Linear Discriminative Analysis
- LSA: Local Subspace Affinity
- M: minimum dimension between two subspaces
- $\mathtt{M}$: aggregate motion matrix
- $\tilde{\mathtt{M}}$: aggregate affine motion matrix
- $\mathtt{M}_n$: motion matrix of object n
- N: total number of motions
- NNs: Near Neighbours
- NSI: Normalized Subspace Inclusion
- P: total number of tracked points
- P_n: number of tracked points for object n, note that $P = \sum_{n=1}^{N} P_n$
- PA: Principal Angle

- PAC: Principal Angles Clusterization
- r: rank of $\mathtt{W}$
- r_h: highest local subspace size
- $\mathtt{R}_{fn}$: rotation matrix of object n in frame f
- $\mathtt{S}$: aggregate structure matrix
- $\tilde{\mathtt{S}}$: aggregate affine structure matrix
- $\mathtt{S}_m$: structure matrix of object n
- $S(j)$: subspace generated by a generic trajectory j
- SCbA: Sum of the Clusterization-based Affinity
- SVD: Singular Value Decomposition
- SSC: Sparce Subspace Clustering
- $\mathtt{W}$: trajectory matrix, usually of size $2F \times P$
- $\bar{\mathtt{W}}$: registered trajectory matrix $\bar{\mathtt{W}} = [\bar{\mathtt{W}}_1 | \dots | \bar{\mathtt{W}}_N]$
- $\mathtt{W}_n$: trajectory matrix that contains only points of motion n, note that $\mathtt{W} = [\mathtt{W}_1 | \dots | \mathtt{W}_N]$
- $\bar{\mathtt{W}}_n$: registered trajetory matrix of object n, i.e. $\mathtt{W}_n$ minus the centroid of object n
- $\mathcal{A}$: number of attempts given to the JESS algorithm
- $\mathcal{M}$: number of misclassified points
- $\mathcal{P}$: parameter of the ASA algorithm that indicates the percentage of PAs used to compute the μ_{PAC} for the PAC function
- $\mathcal{F}$: parameter of the ASA algorithm used as boosting factor of the auto tuned β parameter in the SCbA measure
- $\theta_i(r)$ same as θ_i^r: i^{th} generic PA given rank of $\mathtt{W} = r$
- $\theta_i^r(S_j, S_i)$: i^{th} PA between the subspaces S_j and S_l given rank of $\mathtt{W} = r$
- Θ_i^r: is the collection of PAs such that $\Theta_i^r = \{\theta_i^r(S_j, S_l),\ j, l = 1, \dots, P\}$
- Θ_i: is the collection of PAs such that $\Theta_i = \bigcup_{r=1}^{r_{\max}} \Theta_i^r$

1

Introduction

This chapter provides an introduction to the motion segmentation problem. The difference between image segmentation and motion segmentation is explained in Section 1.1, the importance of motion segmentation and research motivations are highlighted in Section 1.2. In Section 1.3 a brief presentation of the laboratory where this thesis was developed is provided. The problem of motion segmentation is defined in Section 1.4, while the objectives of this thesis are presented in Section 1.5. Finally the structure of the thesis is described in Section 1.6.

1.1 From image segmentation to video segmentation

This thesis is focused on the *motion segmentation* problem. Motion segmentation belongs to a wider, and more generic, category of algorithms known as *segmentation*. The ideal goal of segmentation is to divide the image into semantically meaningful components. Typically, such a task is performed on still images, taking into account clues like: colours, edges, shapes and textures (or, in general, statistical descriptors). The list of segmentation approaches on still images is fairly long: region growing [1], split and merge [1], watershed [2], histogram-based algorithms [1], neural networks [3], active contours [4], graph partitioning [5], and level sets [6, 7], are only some of the most famous.

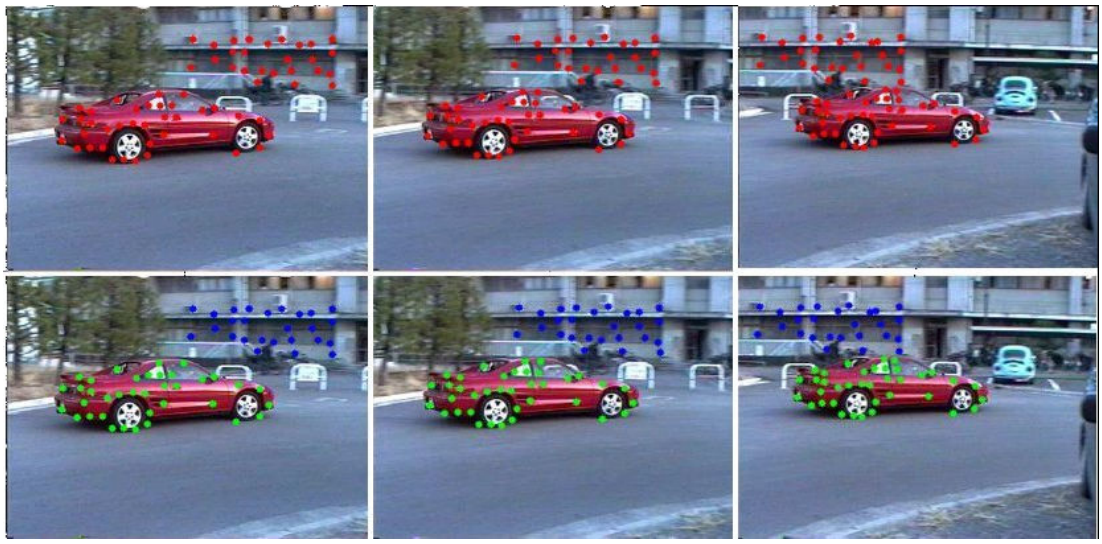


Figure 1.1: An example of motion segmentation. On the first row different frames of a video sequence where 2D image points in the background (which is not still due to the camera motion) and on the moving object (the car) are tracked. On the second row the same frames where the 2D points have been grouped according to their motion are shown. Figure adapted from [8].

When the subject of the segmentation is not a still image but a video sequence, new information becomes available: *the motion*. In motion segmentation a meaningful component is a group of pixels that follow a similar motion. Segmentation of moving objects in video sequences plays an important role in image processing and computer vision. In fact, once the moving objects are extracted, they can serve a wide variety of purposes.

1.2 Motivation

In many computer vision tasks the decomposition of the video into moving objects and background is the first fundamental step. It is an essential building block for robotics, inspection, metrology, video surveillance, video indexing, traffic monitoring and many other applications. An example of motion segmentation is shown in Figure 1.1. In the figure it is possible to see three frames of a sequence in which there is a moving car. Note that the background is also moving (the trees on the left of the first frame do not appear in the last frame) due to the camera motion. The aim of motion segmentation is to provide a

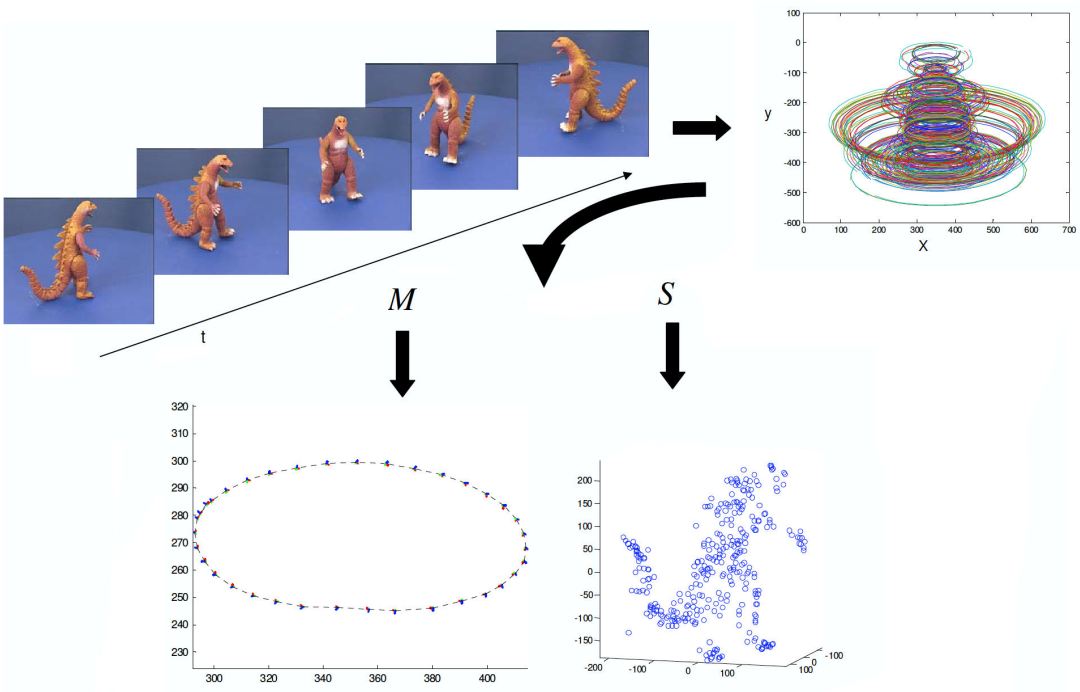


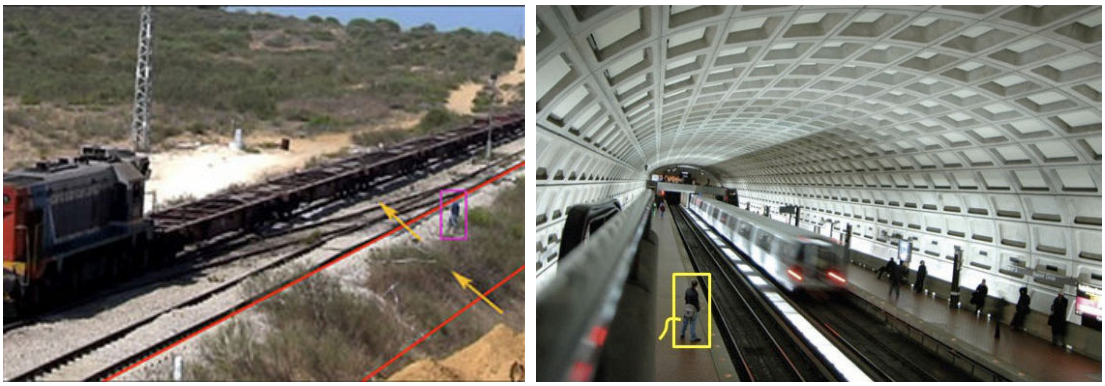
Figure 1.2: General concept of SfM. Some 2D points on the projected surface of a moving object (a dinosaur) are tracked and the image trajectories that they generate are shown on the top right plot. Using these trajectories SfM algorithms are able to recover the apparent motion `M` of the camera and the 3D structure `S` of the object. Figure adapted from [9].

classification of which 2D points follow the same motion. In the example of Figure 1.1 the points tracked throughout the sequence (the red dots on the images of the first row) are correctly segmented as shown in the second row, where the points that belong to the car are all represented with green dots, while the points in the background are all represented with blue dots.

One of the applications that has played an important role in the development of motion segmentation algorithms is *Structure from Motion* (SfM). SfM algorithms can recover the 3D structure of a moving object and a description of the motion in each frame, as shown in Figure 1.2. The first requirement of SfM algorithms is to track a group of 2D points on the projected surface of a moving object. By doing so a set of image trajectories are obtained. In the example of Figure 1.2 the little dinosaur is on a rotating platform and the image trajectories of the points tracked during the rotation of the platform are shown in the plot on the top right of the image. By using only the information of these image trajectories,

SfM algorithms are able to recover the apparent motion of the camera in each of the frames, as shown in the plot on the bottom left of the image. In addition, SfM algorithms recover also the 3D structure of the moving object, as shown in the plot on the bottom right of the image. Ever since Tomasi and Kanade [10] introduced a technique for recovering structure and motion of a moving object, the field of SfM has become very popular and the amount of SfM techniques has rocketed. While a lot of effort has been spent on reconstruction of a single object that moves throughout the video sequence (single-body SfM), very little has been done in the case of multiple objects (multi-body SfM). Therefore, most of SfM techniques rely on the ability of a motion segmentation algorithm to decompose the video into independently moving objects, so that the SfM algorithm can be fed with one object at a time.

Another important application of motion segmentation is in a very active research area at the moment: *action recognition and interpretation* [11, 12]. Recognition of unusual or dangerous actions is important for public safety, and the existence of a high number of CCTV cameras makes automated activity recognition a natural extension to human operated surveillance [13]. Automatic classification of human behaviour involves: the understanding of bodily motion, gestures and signs, analysis of facial expressions, and interpretation of affective signals [13]. Examples of action recognition are shown in Figure 1.3. In Figure 1.3(a) an automatic system is able to identify a man that is walking dangerously towards the railway, or in Figure 1.3(b) a person that is walking too closely to the metro line. Figure 1.3(c) shows another example where a person has climbed a fence and, therefore, the system recognises the anomaly and keeps track of the subject. Finally, in Figure 1.3(d) different anomalous behaviours are depicted, some of them like face or graffiti detection are not related with motion, but others like a person falling or loitering require a motion analysis of the scene. In all of these situations the anomalous behaviour is automatically detected and the area connected to the anomaly is highlighted so that a human operator can be alerted and, if it is required, he can raise an alarm. In order for action recognition to take place one common problem has to be solved: inside one video



(a) www.integoo.cz (b) www.security-technologynews.com

(c) cctv.boschsecuritynews.us (d) iomniscient.com

Figure 1.3: Examples of action recognition. Images adapted from the cited websites (accessed in February 2011).

sequence there is a vast amount of information (scalability problem) and there can be a variety of subjects. To be able to deal efficiently with these situations the information contained in the video must be filtered: each of the subjects should be treated as a separate entity in order to analyse, through pattern recognition techniques, its behaviour and its interaction with other entities. Therefore, the ability of an algorithm to segment different entities in a video sequence and extract relevant information about them, like their motion, is a fundamental step for the success of these applications.

A last example of the importance of motion segmentation is *visual communication*. The importance of video compression and streaming in visual communication has been increasing tremendously. Current standards of compression and streaming, like the extended profile of MPEG-4, from the Moving Pictures Experts Group (MPEG), heavily relies on the ability of an algorithm to perform motion estimation and segmentation. In fact, in

video compression the basic idea, in order to reduce the load of data, is to store (or transmit in the case of video streaming) only the pixels, or block of pixels like in MPEG-1 and 2, that have changed from one frame to the previous, rather than the whole new frame [14]. This way of coding a video is known as block-based coding. Figure 1.4(a) shows an example of three frames in which one car is moving from right to left and a person is moving from left to right. The idea on which MPEG-1 and 2 relies, consists of dividing the image into block units and perform a block association from one frame to the previous in order to be able to estimate the motion occurred in each of the units. This concept is shown in Figure 1.4(b). Once this step is accomplished only the blocks that have changed will be compressed and transmitted. However, this technique suffers from different problems, among which, the fact that videos become affected by artifacts and that the association of the blocks can be ambiguous, as shown in Figure 1.4(c). Better results could be reached by pushing this concept further and not considering any more pixels, or block of pixels, but objects. The idea is that each video scene (let us focus only on the visual components and avoid other components like the audio) can be decomposed in Video Object Planes (VOPs), which are planes that contain only one object of interest [15]. Typically, each object of interest is an object that has its own motion, different than other VOPs. The encoder and the decoder should be able to decompose and recompose the scene by using a hierarchical relation between VOPs. This way of coding a video is known as object-based coding. An example of how an MPEG-4 scene is decomposed is shown in Figure 1.4(d), while an example of object-based coding generation can be found in [16]. Once this set up is built not only a better compression can be reached, but also a series of features can be added to the streaming service. Thanks to the advances in the field of visual communication and to the high velocities of the Internet, applications such as 3D video conferencing [17, 18] are becoming a reality. These services are beyond the scope of this thesis, however, it is evident that an efficient motion segmentation algorithm is an essential key for the success of these applications.

SfM, action recognition and video communication are only some of the applications of

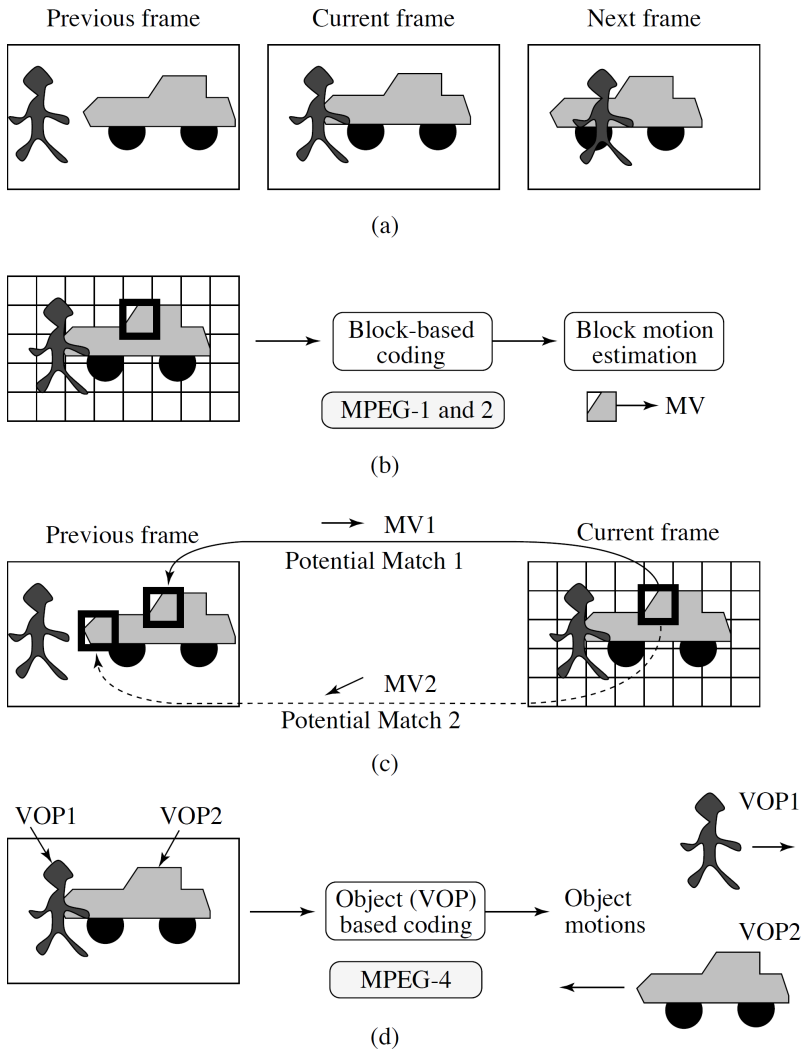


Figure 1.4: Difference between block-based and object-based coding. Figure taken from [19].

motion segmentation. The importance of this field of research is evident by reviewing its vast literature. However, algorithm performances still fall far behind human perception. This thesis scope is to provide a contribution towards filling the gap between motion segmentation algorithms and human ability to differentiate objects with different motion.

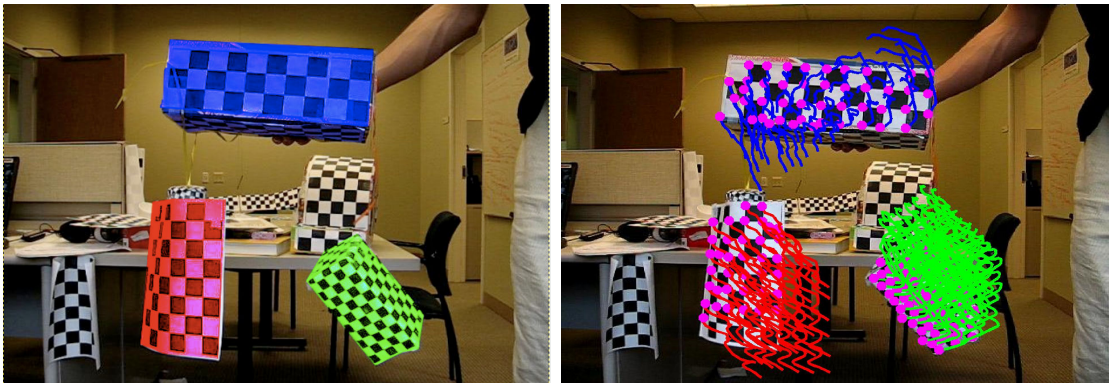
1.3 Context

This thesis was carried out in the VICOROB laboratory at the University of Girona. One of the research interests of VICOROB is 3D perception (DPI2007-66796-C03-02). As already

explained, one group of techniques for 3D reconstruction, the SfM group, is mainly able to deal with the presence of one moving object. From this, it is easy to understand the necessity of developing a motion segmentation algorithm that could feed a single-body SfM approach with the trajectories of one object at a time. Nevertheless, motion segmentation is a low level generic step that can find applications in a variety of contexts. For example, among the interests of the VICOROB group it is possible to find also the construction of multi-modal maps (CTM2010-15216) and the development of autonomous underwater vehicles (robots) for multi-purpose intervention missions (DPI2008-06548-C03-03 and 7PM-STREP). Both these tasks require a system able to fuse together images of the explored area in order to build a large map. Once a map of the area of interest is built, higher level tasks can take place, like: identifying changes in the marine environment, or monitoring human crafted structures that may need maintenance. However, during the exploration of the target area other moving objects, which do not belong to the sea-bed or the structure under analysis, may be encountered. For example fish, other underwater vehicles, humans or weeds may enter into the field of view of the cameras carried by the robot. All these moving objects have to be removed so that the final map does not contain any other element than the one under analysis. For the accomplishment of this removal step a motion segmentation algorithm is again required.

1.4 Problem definition

In this section a more precise and formal definition of the motion segmentation problem is presented. The aim of this definition is to provide the reader with clear information about what motion segmentation is, what the input and the output of the motion segmentation algorithms developed in this thesis are, and which typical assumptions motion segmentation techniques make. The definition and the assumptions explained in this section are not common to every motion segmentation algorithm, they refer to the specific category to which the algorithms proposed in this thesis belong: the motion segmentation from feature trajectories.



(a) Dense-based segmentation (b) Feature-based segmentation

Figure 1.5: Example of the same frame sequence (sequence *articulated* from the Hopkins155 database [22]) segmented by using a dense-based approach and by using a feature-based approach.

There are two categories of motion segmentation algorithms [20, 21]: *dense-based* segmentation and *feature-based* segmentation. Dense-based motion segmentation algorithms provide a classification (i.e. a segmentation) for each individual pixel, whereas feature-based motion segmentation algorithms focus on the classification of selected features that are tracked throughout a video sequence. An example of dense-based and feature-based segmentation results are shown in Figure 1.5, where each motion group is colour labelled. The frame presented in the figure is taken from the Hopkins155 database [22], more information about this database can be found in Section 2.5. While dense-based approaches classify all the pixels of the moving objects, feature-based approaches classify only the selected features (shown as pink dots). For more details about dense-based and feature-based motion segmentation the reader is referred to Chapter 2. The algorithms developed in this thesis belong to the feature-based group.

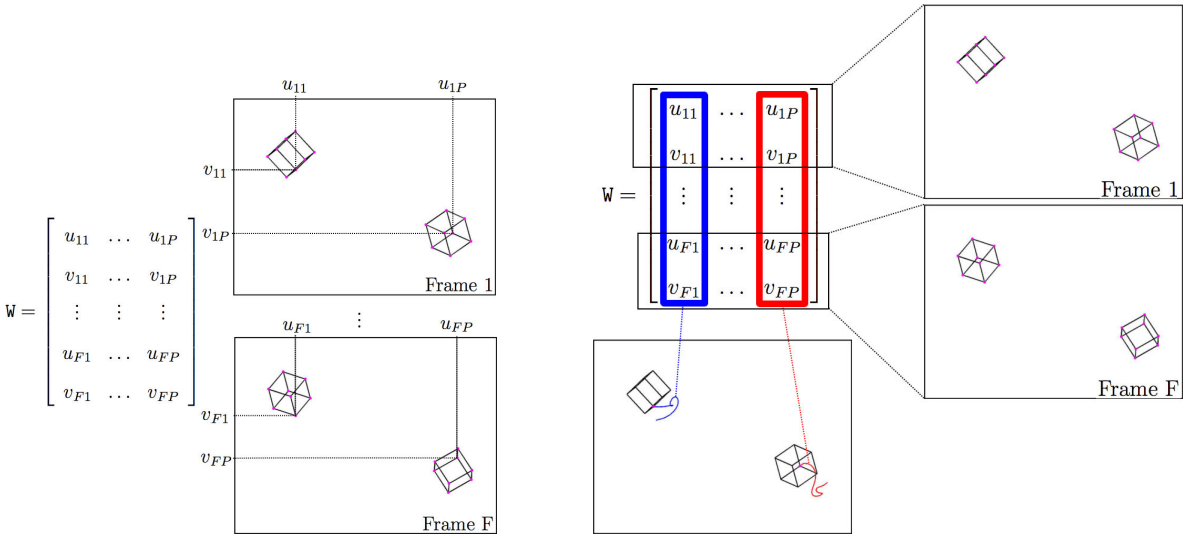
Feature-based motion segmentation algorithms assume that a pre-processing step, the tracking, has already been performed. The field of feature selection and association (tracking) has experienced a very active research period during the last decades. One of the first famous feature trackers is the Kanade-Lucas-Tomasi (KLT) [23] algorithm. Since then many more tracking algorithms based on features have been proposed, among which are algorithms based on the Scale-Invariant Feature Transform (SIFT) [24] and the Speeded

Up Robust Features (SURF) [25]. Nowadays, feature trackers have become very robust and tasks like motion segmentation can greatly benefit from them. In fact, one of the problems of motion segmentation has always been the heavy computation required. However, the possibility to deal with only a "small", but representative, amount of feature points reduces dramatically the amount of data to be processed. In summary, feature-based motion segmentation algorithms do not deal with feature selection and tracking. Such a problem is assumed to be solved by a pre-processing step. The input of a generic feature-based motion segmentation algorithm is the position u (x-axis) and v (y-axis) in each frame of all of the tracked points. Given F frames and P points the typical structure used to represent the input is called *trajectory matrix* which is a $2F \times P$ matrix:

$$\mathtt{W} = \begin{bmatrix} u_{11} & \dots & u_{1P} \\ v_{11} & \dots & v_{1P} \\ \vdots & \vdots & \vdots \\ u_{F1} & \dots & u_{FP} \\ v_{F1} & \dots & v_{FP} \end{bmatrix}. \tag{1.1}$$

In Figure 1.6(a) a pictorial representation of the trajectory matrix is presented. If one column p of $\mathtt{W}$ is analysed, it is possible to identify for each frame the position of the point p, therefore, each column of $\mathtt{W}$ defines a trajectory. On the other hand, if a couple of rows f and $f+1$ (f being any odd index from 1 to $2F$) are taken into account, the positions of all of the tracked points inside the frame f are defined. These two concepts are illustrated in Figure 1.6(b).

Another common assumption of most feature-based motion segmentation algorithms concerns the camera model. Often, an *orthographic* camera model is assumed. The choice of assuming an orthographic camera model is rather popular also in the Structure from Motion field as it eases the mathematical tractability of the problem. The orthographic camera model is a simplified model of the real perspective camera [26]. The simplification relies on the fact that the rays captured by the camera are supposed to hit the image plane



(a) The position of each tracked point is stored in the trajectory matrix.

(b) If one column of the trajectory matrix is analysed the trajectory generated by one point can be drawn. If two rows of the trajectory matrix are analysed all the points in a given frame can be drawn.

Figure 1.6: Pictorial representation of the trajectory matrix.

perpendicularly. For the orthographic camera to be a realistic model the focal length has to be long enough that the depth of the objects do not create any perspective distortion. Most of the time, especially for segmentation problems, the assumption of this simple model is sufficiently respected. Given the simplicity of the orthographic camera model, and given the applications targeted in this thesis, the proposed algorithms assume an orthographic camera model.

Finally, assuming an orthographic camera model, and given a trajectory matrix $\mathtt{W}$, the goal of the feature-based motion segmentation algorithms presented in this thesis is to group together features (i.e. trajectories) that follow a similar motion.

1.5 Objectives

The aim of this thesis is to

> *propose new motion segmentation algorithms that overcome some typical limitations of current approaches, and that do not require tuning steps nor prior knowledge. Moreover, a first step towards the unification of the motion segmentation and structure from motion problems should be investigated.*

In more detail this thesis should:

- provide a complete review of the state of the art of motion segmentation;
- identify in the state of the art which approach seems the most suitable for further investigation;
- propose new motion segmentation algorithms that do not require tedious tuning processes and whose performances must be comparable with or exceed the state of art;
- study the relation between the problems of motion segmentation and structure from motion with the intent to jointly solve them and provide a first step towards merging the two research fields;
- point out future directions connected to this work.

1.6 Structure of the thesis

The remainder of the thesis is structured as follows.

- In Chapter 2 the state of the art of motion segmentation is reviewed and a classification is suggested.

- In Chapter 3 new algorithms for motion segmentation are presented. The first is called Enhanced Model Selection+ (EMS+) and relies on a new and robust estimation of the rank of the trajectory matrix. The second is called Adaptive Subspace Affinity (ASA). ASA is composed of two main algorithms: the Principal Angle Clusterization (PAC) and the Sum of Clusterization-based Affinity (SCbA). PAC is a new way of estimating and interpreting the rank of the trajectory matrix. The second step, SCbA, is a new way of computing the affinity between two subspaces. SCbA tackles some of the issues that so far have been neglected and is a geometrically correct measure. In the same chapter an algorithm for the estimation of the number of motions is also presented. The estimation is based on a dynamic analysis of the eigenvalue spectrum of a matrix involved in the clustering step. The combination of EMS+ and of ASA with the automatic estimation of the number of motions creates two new automatic algorithms called respectively: Enhanced Local Subspace Affinity (ELSA) and Automatic-Adaptive Subspace Affinity (A-ASA).

- In Chapter 4 the problem of motion segmentation is compared with the problem of structure from motion. With the aim of merging those two problems, a novel framework, which is able to correct an initial erroneous segmentation and to provide the 3D structure of the moving objects, is described. The new framework is called Joint Estimation of Segmentation and Structure from Motion (JESS). JESS exploits some constraints that are largely used in SfM, but that have never been adopted to solve the segmentation problem.

- To complete the thesis, in Chapter 5 a summary is presented, conclusions are drawn and future directions are discussed.

2

State of the Art

In this chapter a complete state of the art review of motion segmentation methods is presented. The main problems and the most important attributes of motion segmentation algorithms are described in Section 2.1. Afterwards, a possible classification of motion segmentation algorithms is proposed in Section 2.2. Some of the most recent and important techniques for each class are reviewed in Section 2.3, and considerations are discussed in Section 2.4. Finally, the databases used throughout the whole thesis for the evaluation of the proposed algorithms are presented in Section 2.5

2.1 Problems and attributes

In this section the common problems and the most important attributes of motion segmentation algorithms are analysed. Attributes describe in a compact way the assumptions on which algorithms are based, as well as their limitations and strengths. These attributes will then be used to summarise each of the analysed techniques.

One of the first choices that has to be taken when developing a motion segmentation algorithm is the *representation* of the motions: there are *feature-based* and *dense-based* approaches. An example of both groups is shown in Figure 1.5. In feature-based methods, objects are represented by a limited number of salient points, like KLT [23], SIFT [24]

or SURF [25] features, previously extracted and tracked throughout the video sequence. Features represent only part of an object, hence, the object can be tracked even in cases of partial occlusions. In opposition to feature-based methods there are dense-based methods, which do not use sparse points but compute a pixel-wise motion. The result is a more precise segmentation of object contours, but the occlusion problem becomes harder to solve [27]. Moreover, the main drawback of dense-based methods is the heavy computational time required to process such a big amount of data.

Motion segmentation algorithms usually exploit temporal continuity: each point, that belongs to a moving object, changes its position smoothly from one frame to the following. However, when using only temporal clues a rather big piece of the available information is discarded and this lack of information can easily lead to problems. This is the reason why some techniques exploit also *spatial continuity*. In these cases each pixel is not considered as a single point but the information provided by its neighbours (in terms of spacial proximity) is taken into account. For example, one of the problems that is sometimes caused by the use of temporal information only, is the difficulty to deal with *temporary stopping*. In fact, many techniques lose the segmentation when the objects stop moving even if for a limited amount of time.

Another common issue that has to be tackled by motion segmentation algorithms is the fact that moving objects may not be always visible. The ability to deal with *missing data* is one of the most difficult problems. Missing data can be caused by many factors: small random variation in image brightness (noise) that may induce the tracker to loose some feature points, occlusions, or points (either pixels or features) that are not in the scene for the whole length of the sequence.

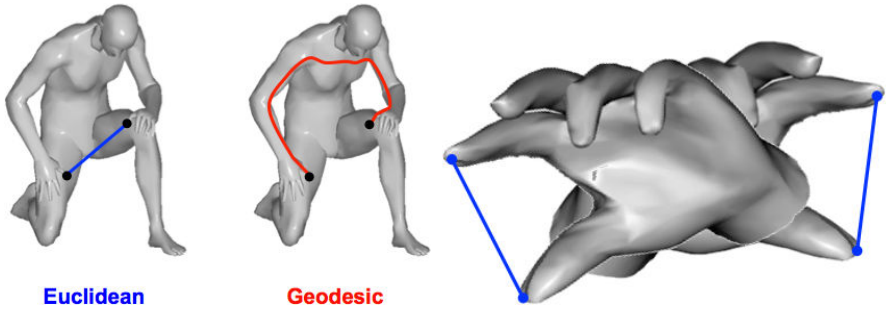
Missing points due to image noise is the most extreme consequence of the presence of noise in tracking systems. Nevertheless, even when the points are not lost, noise can affect the accuracy of the positions of the tracked features. Hence, *robustness* of the algorithms against noise (from now on when talking about noise it will be always referred to as the perturbations of the tracked feature positions) is as an essential factor to take into account.

However, robustness is not affected only by the presence of noise. A robust algorithm should be able to deal also with the presence of outliers, i.e. points erroneously tracked (as in wrong associations made by the tracker algorithm). Finally, for iterative algorithms that require an initialisation, robustness is related to the ability to converge to the right solution even when the initialisation is not close to the final solution.

Another important attribute that has to be analysed is the ability to deal with different types of motion. There is a bit of confusion in the literature when it comes to "types of motion" as people tend to use different adjectives to describe the same property or the same adjective to describe different properties. Hence, it is important to clarify which is the exact meaning that is given to each adjective in this work. A motion can be described in terms of: *dependency* and *kind*.

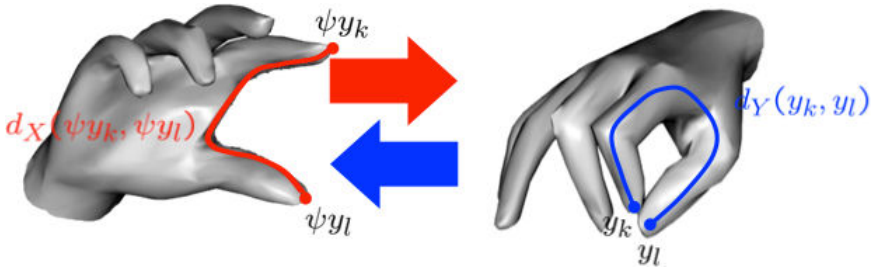
The dependency is an attribute that describes the relationship between a pair of motions and is not a property of one single motion. Motions can be *independent* or *dependent*. Two motions are dependent if, somehow, they have some degree of similarity. For example, two cars that proceed in parallel with a similar speed (but not equal, otherwise they would be the same motion) are an example of dependent motions. Vice versa, a car that turns at ninety degrees and another car that comes from the opposite direction and continues straight are an example of independent motions.

The kindness of motion is related to the nature of the motion, which can be: *rigid*, *non-rigid* and *articulated*. To present the different kinds of motion the concepts of extrinsic and intrinsic properties defined in [28] are used. The extrinsic property refers to how the object is laid out in the space, while the intrinsic property refers to the metric structure of an object and it is invariant to the object deformations. If a shape is regarded to as a metric space, its extrinsic properties are described by the 3D Euclidean metric, while intrinsic properties are described by geodesic metric, which measures distances between points as the lengths of the shortest path on the 3D shape. In Figure 2.1(a) it is possible to find an example of extrinsic and intrinsic properties. Shape transformations that preserve the metric are called isometries; extrinsic isometries are rigid motions and intrinsic isometries are inelastic



(a) Examples of an extrinsic property (Euclidean distance) and of an intrinsic property (geodesic distance)

(b) Extrinsic isometry



(c) Intrinsic isometry

Figure 2.1: Extrinsic and intrinsic properties. Images adapted from [28].

deformations, i.e. non-rigid and articulated motions. In Figure 2.1(b) an object (a hand) underwent a rigid transformation and it is possible to see that the Euclidean distance (an extrinsic property) between the same pair of points remains the same. On the other hand, in Figure 2.1(c) the object underwent a non-rigid transformation. In this latter case the Euclidean distance between the same pair of points is very different, however, the geodesic distance (an intrinsic property) remains similar.

The fact that a motion is rigid or non-rigid is an attribute of the single motion, while an articulated motion is actually a specific class of dependent motions. A motion is rigid if the moving object is not deformable. In terms of shape transformation, the different views of the object in the sequence can be described by extrinsic isometries. For instance, the motion of a car is a rigid motion. The mathematical definition of a rigid motion is relatively simple. Given an initial set of point p_0 that describes an object, its evolution in time can be modelled as a rotation matrix $\mathtt{R}(t) \in SO(3)$ and a translation vector $\mathtt{t} \in \mathcal{R}^3$, such that $p(t) = \mathtt{R}(t)p_0 + \mathtt{t}(t)$ [29].

Figure 2.2: Two examples of non-rigid motions. On the left the face of a person who smiles. On the right a pillow that is being compressed. Images adapted from [30].

A non-rigid motion is the motion of an object that can be deformed, compressed or extended. Therefore, in terms of shape transformation, the different views of the object could be extrinsically dissimilar but, under mild topological assumptions, they can be described by intrinsic isometries. For example, the face of a person who is talking and is moving his or her head from left to right is a non-rigid motion. In fact, all his or her facial points are moving from left to right, but at the same time the relative Euclidean distance between some of the points (like the corners of the lips, the corners of the eyes, or the nose) changes because they are influenced also by another motion. Two examples of non-rigid motions are shown in Figure 2.2. While the mathematical definition of a rigid motion is relatively simple, the definition of non-rigid motion is more challenging. Intuitively, a non-rigid motion can be seen as the composition of an overall motion and a deformation. Following this intuitive definition, a deformation f can be described as the composition of a rigid motion and a deformation function $h(\cdot, t)$, so that $p(g) = h(\mathtt{R}(t)p_0 + \mathtt{t}(t), t)$. However, it is always possible to find infinitely many different choices $\tilde{h}(\cdot, t), \tilde{\mathtt{R}}(t), \tilde{\mathtt{t}}(t)$ that give rise to the same overall deformation f. Therefore, by this intuitive definition, the motion, which is unique to the observed scene, would have an ambiguous description [29]. In order to provide a unique description of a non-rigid motion it is necessary to couple the concept of motion with the notion of "shape average" as explained in [29]. The shape average is

defined as the shape that minimises the object deformations observed. Therefore, a non-rigid motion at time t can be modelled as the rigid motion $g(\cdot, t)$ performed by the shape average μ, on top of which a local deformation $h(\cdot, t)$ is applied: $p(t) = h(\mu, t) \circ g(\mu, t)$. The reader is referred to [29] for the details of the formalisation of non-rigid motions.

In terms of segmentation all of the points that belong to a non-rigid motion should be considered as one motion. It has to be noted that the ability to deal with non-rigid motions (i.e. the ability to consider points that belong to a non-rigid body as points that belong to one non-rigid motion) is constrained to the cases when, besides local deformations, it is possible to extract a shape average whose motion is rigid. When this is not the case, for example if the person who is talking is keeping his or her head completely still, each group of points that undergo a different motion cannot be distinguished from the case when there are different objects that move independently.

Finally, there are articulated motions. Articulated motions can be considered as a specific case of non-rigid motions. In fact, like non-rigid motions, also articulated motions are intrinsic isometries. Moreover, also an articulated motion is composed of more than one dependent motion, like for the non-rigid case, but with the constraint that the motions are usually rigid. In addition, the different part of an articulated motion are linked together by joints. Thanks to this specification, as far as motion segmentation is concerned, each segment (this is how each part that creates an articulated motion is called) has to be segmented separately. Articulated motions are highly dependent motions as they are characterised by the dominant motion of the whole body and by segments that follow their own motion. In Figure 2.3 an example of an articulated motion is shown. On the left of the image an excavator is moving forward without moving its arm, hence, its motion is correctly classified as one simple (rigid) motion. On the right, another excavator is rotating and at the same time is moving its arm, hence, this is an articulated motion correctly segmented in its two (rigid) motions. In Section 2.3.6 a more formal algebraic definition of motion dependency and kind is provided.

These are all the attributes, related with motion, that will be taken into account in

Figure 2.3: On the left an example of a rigid motion: an excavator is moving forward without moving its arm, hence, its motion is correctly classified as one motion (blue dots). On the right another excavator is rotating and at the same time is moving its arm, hence, this is an articulated motion correctly segmented in its two motion components (yellow and red dots). Image adapted from [31].

the summary presented in Table 2.1 and 2.2. However, authors do not always clearly state under which conditions their algorithms work, therefore, the table is filled to the best of our knowledge given the information provided in the cited papers.

Furthermore, if the aim is to develop a generic algorithm able to deal in many unpredictable situations, there are some features that may be considered as drawbacks. For instance, one important aspect is the amount of *prior knowledge* required. In particular, prior knowledge that is commonly required includes: the number of moving objects, and the dimension of the generated subspaces.

For the sake of completeness a word about *transparent motions* should be spent here. Transparent motions are motion generated by transparent objects. This class of motions is particularly difficult to segment for all those methods that use image distance between points as discriminant feature for the segmentation. However, the small presence of interesting transparent motions in real life, and the difficulty that any tracking algorithm would face when dealing in such a situation, make this class of motions of secondary importance. Nevertheless, the algorithms developed in this thesis would have no problems to segment

transparent motions (given that the tracking of the features is provided) as they are not based on image distances.

In this section an overview of the main attributes and problems that should be taken into account when developing a motion segmentation algorithm were presented. In the next section a classification of the most important algorithms is proposed.

2.2 Classification

As motion segmentation has been a hot topic for many years its literature is particularly voluminous. In order to make the overview easier to read and to create order, the approaches will be divided into categories which represent the main principle underlying each algorithm. For each category some articles, among the most representative and recent, are provided. The division is not meant to be restrictive, in fact, some of the algorithms could be placed in more than one category. The classes identified are:

- *Image Difference*
- *Statistics*, further divided into:
  - Maximum A posteriori Probability (MAP)
  - Particle Filter (PF)
  - Maximum Likelihood (ML)
- *Optical Flow* (OF)
- *Wavelets*
- *Layers*
- *Manifolds Clustering*, further divided into:
  - Iterative
  - Statistics

- Sparse representation
- Factorisation
- Subspaces estimation

The main techniques described in the next section are summarised in Tables 2.1 and 2.2, which offer a compact at-a-glance overview with respect to the proposed classification and the attributes described in the previous section. Table 2.1 summaries all the techniques based on: image difference, statistics, wavelets, optical flow, and layers. Table 2.2 summaries all of the manifold-based techniques.

Image	**Differ**	*Cavallaro et al. 2005* [32]	F/D	✓	✓	✓	X−−	RN	✓	
		Cheng et al. 2006 [33]	D	✓	✓		✓−−	RN	✓	X
		Li et al. 2007 [34]	D		✓	✓	✓−−	RN	✓	X
		Colombari et al. 2007 [35]	D	✓	✓	✓	✓−−	RN	✓	
Statistics	MAP	*Rasmussen et al. 2001* [36]	D	✓	✓	✓	X−−	✓	✓	CX
		Cremers et al. 2005 [37]	D	✓	✓	✓	X−−	✓	RA	X
		Shen et al. 2007 [38]	D	✓	✓	✓	✓−−	✓	✓	CX
	PF	*Rathi et al. 2007* [39]	D	✓	✓	✓	X−−	✓	RN	X
	ML	*Stolkin et al. 2008* [40]	D	✓	✓	✓	✓−−	✓	R	CX
		Thakoor et al. 2010 [41]	F		✓	✓	✓−−	✓	R	X
Wavelets		*Wiskott 1997* [42]	F		✓		X−−	I	R	
		Kong et al. 1998 [43]	F	✓	✓		✓−−	I	R	X
OF		*Trucco et al. 2005* [44]	F/D	✓			X−−	✓	RA	
		Zhang et al. 2007 [45]	F		✓		X−−	✓	RA	C
		Li Xu et al. 2008 [46]	D	✓	✓	✓	X−−	✓	✓	
		Klappstein et al. 2009 [47]	F	✓	✓		✓−−	✓	RA	X
		Bugeau et al. 2009 [48]	F/D	✓	✓		✓−−	I	R	
		Ommer et al. 2009 [49]	F	✓	✓		X−−	I	R	
		Brox et al. 2010 [50]	F	✓	✓	✓	✓−−	I	RA	
Layers		*Kumar et al. 2008* [27]	F/D	✓	✓	✓	✓−✓	✓	✓	X
		Min et al. 2008 [51]	D	✓	✓	✓	✓−X	✓	✓	X
		Nordberg et al. 2010 [52]	F		✓	✓	✓−X	I	R	CX
		Zografos et al. 2010 [53]	F		✓	✓	✓−X	✓	RA	CX
		Xu et. al. 2011 [54]	F/D	✓	✓	✓	✓−−	I	R	X
		Wang et al. 2011 [55]	F/D		✓	✓	X✓−	I	R	X

Features (F) / Dense (D)
Occlusion or Missing Data
Spatial Continuity
Temporary Stopping
Robustness (Noise, Outliers, Initialisation: ✓yes, Xno, - not related)
Dependency (Independent, Dependent, ✓all)
Kind (Rigid, Non-rigid, Articulated, ✓all)
Prior knowledge (C Number of clusters, D Subspace dimensions, X Other)

Table 2.1: Summary of the techniques not based on manifold clustering with respect to the attributes described in Section 2.1.

Manifold Clustering	Iter	*Ho et al. 2003* [56]	F			✓	✓✓X	✓	✓	CD
		da Silva et al. 2008 [57]	F			✓	✓✓X	✓	✓	CD
		da Silva et al. 2009 [58]	F			✓	✓✓X	✓	✓	CD
	Stat	*Fishler et al. 1981* [59]	F			✓	✓✓–	I	RA	C
		Kanatani et al. 2002 [60]	F			✓	✓✓✓	I	R	
		Sugaya et al. 2004 [61]	F			✓	✓✓X	I	R	C
		Gruber et al. 2004 [62]	F	✓		✓	✓✓X	I	R	X
		Gruber et al. 2006 [63]	F	✓	✓	✓	✓✓X	I	R	X
		Sugaya et al. 2010 [64]	F			✓	✓✓✓	I	R	C
	Sparse	*Rao et al. 2008/2010* [65, 66]	F	✓		✓	X✓–	I	R	
		Elhamifar et al. 2009 [67]	F	✓		✓	✓✓–	✓	✓	DX
	Fact	*Costeira et al. 1998* [68]	F			✓	XX–	I	R	
		Ichimura et al. 2000 [69]	F			✓	XX–	I	R	
		Zelnik-Manor et al. 2003 [70]	F			✓	✓X–	✓	RA	CD
	Subspaces	*Vidal et al. 2004* [71]	F	✓		✓	XX–	✓	R	C
		Yan et al. 2006/08 [31, 72]	F			✓	X✓–	✓	✓	CDX
		Goh et al. 2007 [73]	F			✓	X✓–	✓	R	CD
		Julià et al. 2008 [74]	F	✓		✓	X✓–	I	R	D
		Vidal et al. 2008 [75]	F	✓		✓	X✓–	✓	R	C
		Goh et al. 2008 [76]	F			✓	X✓–	✓	R	CD
		Chen et al. 2009 [77, 78]	F			✓	✓✓–	✓	✓	CD
		Kim et al. 2009 [79]	F			✓	✓✓–	✓	R	C
		Yang et al. 2009 [80]	D			✓	✓✓–	I	R	CD
		Lauren et al. 2009 [81]	F			✓	✓✓–	✓	✓	CD

Features (F) / Dense (D)
Occlusion or Missing Data
Spatial Continuity
Temporary Stopping
Robustness (Noise, Outliers, Initialisation: ✓yes, Xno, - not related)
Dependency (Independent, Dependent, ✓all)
Kind (Rigid, Non-rigid, Articulated, ✓all)
Prior knowledge (C Number of clusters, D Subspace dimensions, X Other)

Table 2.2: Summary of the manifold clustering-based techniques examined with respect to the attributes described in Section 2.1.

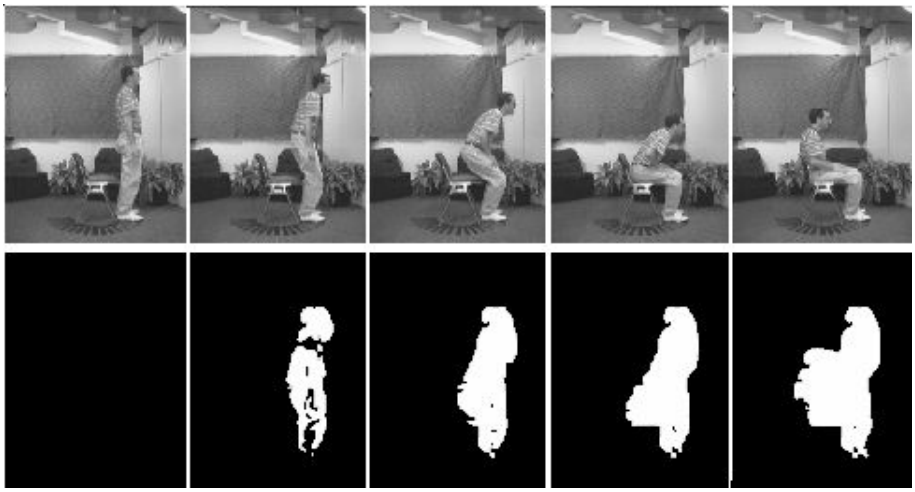


Figure 2.4: Example of an image sequence and its image difference result. Sequence taken from [82].

2.3 Main techniques

2.3.1 Image difference

Image difference techniques are some of the simplest and most used for detecting changes. They consist of thresholding the pixel-wise intensity difference of two consecutive frames. The result is a coarse map of temporal changes. An example of an image sequence and the image difference result is shown in Figure 2.4. A brief presentation of some of the recent techniques, which improved the basic concept of image difference, follows.

Cavallaro et al. (2005) [32] reinforce the motion difference using a probability-based test in order to change the threshold locally. As previously explained, this first step allows a coarse map of the moving objects. Each blob is then decomposed into non-overlapping regions. From each region spatial and temporal features are extracted. The spatial features used are: colour (in the CIE lab space), textures and variance. The temporal features are the displacement vectors from the optical flow computed via block matching. The idea is that spatial features are more uncertain near edges, whereas temporal features are more uncertain on uniform areas, however, the union of the two should guarantee a more robust behaviour. The tracking is performed by minimisation of the distance between feature

descriptors, and the same criteria is also responsible for the merge and split decision. This technique is capable of dealing with multiple objects, occlusions and non-rigid objects. Unfortunately, the region segmentation stage is based on an iterative process which makes the algorithm time consuming. Another limitation is due to the initialisation performed when a group of objects enter in the scene, in such cases the algorithm assigns them a unique label rather than treating them as separated objects.

Cheng and Chen (2006) [33] perform image difference on the low frequency sub-image of the third level of a the discrete wavelet transform. On the extracted blobs they perform some morphological operations and extract the colour and spatial information. In this way each blob is associated with a descriptor that is used to track the objects throughout the sequence. In this approach the wavelet transform is only exploited for noise reduction. The authors focus on segmenting and tracking human beings, hence, they pay particular attention to selecting features that could help to solve this task; specifically, they introduce some prior information about human being silhouettes. By doing so, they can successfully track humans, but of course the method loses its generality. Furthermore, no motion compensation or statistical background is built which makes the method not suitable for moving camera applications.

Li et al. (2007) [34] use image difference in order to localise moving objects. The noise problem is attenuated by decomposing the image in non-overlapping blocks and working on its average intensity value. They also use an inertia compensation to avoid loosing tracked objects when the objects temporarily stop. Simultaneously, the watershed algorithm [2] is performed in order to extract the gradient of the image. Finally, the temporal and the spatial information are fused together so that the coarse map is refined using an anisotropic morphological dilation that follows the gradient of the image. This technique deals successfully with the temporary stopping problem, but its main drawbacks are the high number of parameters that require tuning and the inability to deal with moving cameras.

Colombari et al. (2007) [35] propose a robust statistic to model the background. For each frame a mosaic of the background is back-warped onto the frame. A binary image that indicates for each pixel whether it belongs to a moving object or not is obtained. Then, the binary image is cleaned and regions are merged and define blobs. By exploiting temporal coherence the blobs are tracked throughout the sequence. This technique is able to deal with occlusions, appearing and disappearing objects. The background mosaic is done off-line and in order to recover the motion of the camera it is necessary to extract many features in the non-moving area.

As can be seen from Table 2.1, image difference is mainly based on dense representation of objects. It combines simplicity and good overall results because it is able to deal with occlusions, multiple objects, independent motions, non-rigid objects. Image difference alone cannot perform segmentation, it is able to detect motions. Spatial considerations need to be done in order to obtain the segmentation. The main problem of this group of techniques is the difficulty they have in dealing with temporary stopping and moving cameras. In order to be successful in these situations a history model of the background needs to be built. Furthermore, image difference algorithms are still very sensitive to noise and to light changes, hence, they cannot be considered an ideal choice in the case of a cluttered background. Finally, these techniques cannot usually distinguish among the different components of an articulated object when the whole object is also influenced by an overall rigid motion.

2.3.2 Statistics

Statistical theory is widely used in the motion segmentation field. In fact, a first level of motion segmentation can be seen as a classification problem where each pixel has to be classified as background or foreground. Statistical approaches can be further divided depending on the framework used. Common frameworks are Maximum A posteriori Probability (MAP), Particle Filter (PF) and Maximum Likelihood (ML). Statistical approaches provide a general tool that can be used in a very different way depending on the specific

technique.

Maximum A posteriori Probability

MAP is based on the Bayes rule:

$$P(c_j|x) = \frac{p(x|c_j)P(c_j)}{\sum_{i=1}^{C} p(x|c_i)P(c_i)}, \tag{2.1}$$

where x is the object to be classified (usually the pixel), $c_1 \dots c_C$ are the C classes (usually background or foreground), $P(c_j|x)$ is the "a posteriori probability", $p(x|c_j)$ is the conditional density, $P(c_j)$ is the "a priori probability" and $\sum_{i=1}^{C} p(x|c_i)P(c_i)$ is the "density function". MAP classifies x as belonging to the class $\hat{c}$ that maximises the "a posteriori probability":

$$\hat{c} = \arg\max_{c_j} P(c_j|x). \tag{2.2}$$

Rasmussen and Hager (2001) [36] use a MAP framework, namely they use the Kalman Filter [83] (and the Probabilistic Data Association Filter) to predict the most likely location of a known target in order to initialise the segmentation process. They also discuss an extension for tracking multiple objects but without estimating the number of them. Depending on the noise level the authors use different clues for tracking, such as colours, shapes, textures and edges. Unfortunately, the choice of which information should be used is not automatised.

Cremers and Soatto (2005) [37] use level sets [6] in order to incorporate motion information. The idea is to avoid the computation of the motion field, which is usually inaccurate at boundaries. Instead, they jointly estimate the segmentation and the motion by minimising a functional that depends on parametric motion models of each of the segments, and on the boundaries that separate the segments. The algorithm is based on a geometric model of the motion, therefore, if the motion of the objects deviates from the model hypothesis the segmentation gradually degrades. The main limitation of this model is that it is based on the assumption that objects do not change their brightness throughout time.

This assumption is often violated especially in cluttered backgrounds. This method is able to deal with multiple objects but it requires knowledge of the maximum number of objects, and it has problems with new objects that enter the scene.

Shen et al. (2007) [38] use the MAP framework to combine and exploit the inter-dependence between motion estimation, segmentation and super resolution. The authors observe that when the scene contains multiple independently moving objects the estimated motion vectors are prone to be inaccurate around boundaries and occlusion regions, thus the reconstructed high-resolution image contains artifacts. On the other hand, a sub-pixel accuracy would facilitate an accurate motion field estimation and, hence, a better segmentation. They propose a MAP formulation to iteratively update the motion fields and the segmentation fields along with the high-resolution image. The formulation is solved by a cyclic coordinate descent process that treats motion, segmentation and the high-resolution image as unknowns, and estimates them jointly using the available data. The presented results show that the algorithm is able to segment multiple objects and construct a high-resolution image. Though theoretically the algorithm should be able to deal with non-rigid objects and moving cameras, in the paper there are no experiments that show results in these situations. The algorithm has quite a few parameters, six, to be tuned and one of them is the number of iterations to be performed, unfortunately there is not other termination criteria. Probably, the main drawback of the method is the required prior knowledge of the number of motions that compose the image. The fact that the number of moving objects is a required prior information is a common assumption in motion segmentation algorithms. The authors of this paper state that the estimation of the number of objects is still an open and challenging problem.

Particle Filter

Another widely used statistical method is the PF. Whenever the posterior probability can be approximated by a gaussian function, methods like Kalman Filter are optimal choices. However, when the posterior probability has multi-modal or heavily skewed probability

density function, PF becomes a better choice. The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. It has been shown that as the number of samples approaches infinity, the sample set converges to the true posterior [84].

Let $\{c^i_{0:k}, w^i_k\}^M_{i=1}$ denote a random measure that characterises the posterior probability $p(c_{0:k}|x_{1:k})$, where $\{c^i_{0:k}, 1 = 1, \ldots, M\}$ is a set of support points with associated weights $\{w^i_k, i = 1, \ldots, M\}$, $c_{0:k} = \{c_j, j = 1, \ldots, k\}$ is the set of all states up to time k, $x_{1:k}$ is the set of measurements up to time k. The weights are normalised such that $\sum^M_{i=1} w^i_k = 1$. Then, the posterior density at time k can be approximated as:

$$p(c_{0:k}|x_{1:k}) \approx \sum_{i=1}^{M} w^i_k \delta(c_{0:k} - c^i_{0:k}). \tag{2.3}$$

Key points of PF are the choices of the weights and of the samples. These choices are driven by the principle of *importance sampling* [85]. Ideally, the particles should be sampled from the posterior probability, which is unknown. Hence, the samples has to be taken from the prior estimate of the posterior probability, which is known. Then, the weight of each of the samples is updated according to the observations. At this point the particles are re-sampled in order to provide a finer sampling near particles with higher weight, while particles with small weights are eliminated [86]. At each iteration the prediction of the variable c can be chosen as the one corresponding to the maximum weight (MAP-like choice) or the one corresponding to the weighted mean.

Rathi et al. (2007) [39] unify some well known algorithms for object segmentation that use spatial information, such as geometric active contours [4] and level sets, with the PF framework. The PF is used to estimate the conditional probability distribution of a group of motions and the contour at each time. The algorithm requires the knowledge of the object shapes in order to deal with major occlusions. By exploiting a fast level set implementation [87] the authors claim that the algorithm can be used in near real-time speeds (no further indications are provided).

Maximum Likelihood

ML is also a frequently exploited tool especially when in presence of missing data. In ML the aim is to estimate the model parameter(s) c to which the observed data x is most likely to belong:

$$\hat{c} = \arg\max_{c} p(x|c). \tag{2.4}$$

ML differs from MAP in that MAP assumes that the estimated parameter c is also a random variable which has a prior distribution $P(c)$. ML is often obtained by using the Expectation Maximization (EM) algorithm. Each iteration of the EM algorithm consists of an E-step and an M-step. In the E-step, by using the conditional expectation, the missing data is estimated. In the M-step the likelihood function is maximised. Convergence is assured since the algorithm is guaranteed to increase the likelihood at each iteration [88]

Stolkin et al. (2008) [40] present a new algorithm which uses EM and Extended-Markov Random Field (E-MRF). The authors developed this method for poor visibility environments, specifically for underwater. The algorithm merges the observed data (the current image) with the prediction derived from prior knowledge about the object being viewed, in order to track the camera trajectory. The merging step is driven by the E-MRFs within a statistical framework. The importance of the observed image rather than the predicted model is decided by means of two weights. In this way it is possible to have an ad hoc behaviour depending on the degree of noise: if the visibility is good it is desirable to rely on observed data, while if the visibility is bad it is necessary to make greater use of predicted information. This latter feature is an interesting ability to adapt to different conditions but in this implementation parameters are selected once and they do not change dynamically if conditions change. The ability to work in environments with very poor visibility requires the use of some prior information like: the knowledge of the model of the object being viewed and the parameters of the camera. The authors admit that the algorithm is computationally expensive.

Thakoor et al. (2010) [41] present a framework for two-view segmentation. They gen-

erate hypotheses for the estimation of the fundamental matrices by local sampling. Once the hypotheses are generated a combinatorial optimisation model is created and the optimisation is performed by the branch-and-bound technique (B&B). The B&B optimisation searches for the solution that minimises the cost of the model given the current set of hypotheses. The problem of the estimation of the number of clusters is tackled by using the Bayesian information criterion (BIC) defined in [89]. As the authors of the paper say, the outcome of the method heavily depends on the initial set of hypotheses. In fact, the B&B algorithm guaranties optimality over the current set of hypotheses, however, if the set does not explore and represent the solution space correctly the final segmentation is negatively affected.

Statistical approaches mainly use dense-based representation. They work well with multiple objects and can deal with occlusions and temporary stopping. In general they are robust as long as the model reflects the actual situation, but they degrade quickly as the model fails to represent reality. Moreover, most of the statistic approaches require some kind of prior knowledge.

2.3.3 Wavelets

Another group of motion segmentation algorithms is the one based on the wavelet transform. These methods exploit the ability of wavelets to perform analysis of the different frequency components of images, and then study each component with a resolution matched to its scale. Usually wavelet multi-scale decomposition is used as a pre-processing step in order to reduce the noise. These approaches rely on other techniques, such as optical flow, to perform the actual segmentation. However, some examples of wavelet-based algorithms used to solve the segmentation problem can be found.

Wiskott (1997) [42] combines both the Gabor and the Mallat wavelet transforms to overcome the aperture problem. The former transform is used to estimate the motion field and roughly cluster the image, while the latter is used to refine the clustering. The main limitation of this model is that it assumes that objects only translate in front of the camera.

A different approach is presented by *Kong et al. (1998)* [43] where the motion segmentation algorithm is based on Galilean wavelets. These wavelets behave as matched filters and perform minimum mean-squared error estimations of velocity, orientation, scale and spatio-temporal position. These pieces of information are finally used for tracking and segmenting objects. The authors claim that the algorithm is robust, it can deal with temporary occlusions and by tuning a threshold it can estimate the number of moving objects in the scene.

Wavelet-based solutions seem to provide good overall results but are limited to simple cases (such as translation in front of the camera). Wavelets were in fashion during the 90s, nowadays the research interest seems to be less active, at least in relation to motion segmentation.

2.3.4 Optical Flow

Optical flow (OF) is the distribution of apparent velocities estimated by the analysis of the brightness changes of an image sequence. An example of OF is shown in Figure 2.5. Like image difference, OF is an old concept greatly exploited in computer vision. It was first formalised and computed for image sequences by Horn and Schunck in 1980 [90]. However, the idea of using discontinuities in the OF in order to segment moving objects is even older, in [90] there is a list of older methods based on this idea but they all assume the OF is already known. Since the work of Horn and Schunck, many other approaches have been proposed. In the past, the main limitations of such methods were high sensitivity to noise and expensive computational cost. Nowadays, thanks to the high process speed of computers and to improvements made by research, OF is widely used. As an example, *Trucco et al. (2005)* [44] merge image difference with OF. Specifically, they propose to estimate the OF only at pixels where motion is significant. The image difference is responsible for identification of areas where significant motion occurs. The map produced by the image difference is called in the paper "disturbance field" and is computed as the difference between the current frame and an exponentially-weighted average of the past

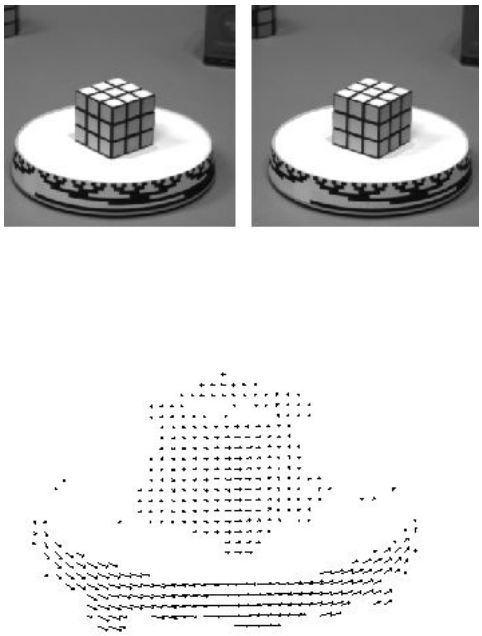


Figure 2.5: Example of OF, the platform under the cube rotates and in the bottom image the vector field that describes such a motion is shown. Image adapted from [91].

frames. The OF is based on a recursive-filter formulation. This technique assumes that flow is smooth and changes slowly over time. Thanks to the fact that the OF is computed only when it is necessary the algorithm is faster than most of OF-based solutions.

Zhang et al. (2007) [45] propose a method to segment multiple rigid-body motions using Line OF [92]. In contrast to classical OF algorithms (which are based on points), the line OF algorithm can work also when the moving object has a homogeneous surface, provided that the object edges can be identified and used as pieces of straight lines. The limitation of this method is that it is not able to deal with non-rigid motions because it requires straight lines in order to compute the OF. This approach uses K-means in order to build the final clusters, hence, it also assumes that the number of moving objects (i.e. the number K of clusters) is known a priori.

Xu et al. (2008) [46] propose a variational formulation of OF combined with colour segmentation obtained by the Mean-Shift [93] algorithm. Moreover, in order to deal with non-rigidity a confidence map is built to measure the confidence of whether the segmentation and the corresponding motion model suit the flow estimation or not. This technique

is also able to deal with partial occlusions.

Klappstein et al. (2009) [47] exploit OF in order to build a robust obstacle detection for driver assistance purposes. The work is done both with monocular (exploiting some motion constraints) and stereo (using Extended Kalman Filter) approaches. The idea is to compute a map of likelihood of each tracked feature to belong to the background or to a moving object. The tracked features are then used as seed points for a dense segmentation. The main limitation of this technique is that it assumes a specific type of motion (camera moving forward) and a particular environment (for example they try to localise the street in order to exploit some additional constraints). Moreover, it is only able to distinguish between background (static objects) and foreground (moving objects) and not among different motion directions.

Bugeau and Pérez (2009) [48] address the segmentation problem by combining motion information, spatial continuity and photometric information. The first step of the algorithm consists of extracting a rough map of moving areas (foreground), this is done by an estimation of the camera motion. The points considered as foreground are then selected in order to have an evenly sparse grid, and each of them is characterised by position, motion (vector flow computed by tracking the point with KLT) and photometric parameters (texture for grey scale images, YUV for coloured images). Mean-Shift algorithm is used to create clusters from those points. At the end they use a graph-cut minimisation algorithm in order to perform a dense segmentation of the whole image (again using position, motion and colour). The algorithm assumes prior knowledge about the number of clusters and that each moving object has at least one representative among the created clusters. One of the problems of this technique is that the segmentation is computed every two or three frames, hence if during this amount of time the motion is not completely different, it does not help to distinguish between dependent motions.

Ommer et al. (2009) [49] present an algorithm for segmentation, tracking and object recognition. The segmentation and tracking parts are done by OF (using salient features and KLT tracking). The algorithm is based on grouping together salient features that follow

a proximity criteria. By KLT, the features are tracked and the mean flow is computed. The position of the group of features is predicted using the previous mean flow in order to constrain the tracking area. At every iteration the mean flow is updated taking into account the old flows with an exponential decay over time. In order to extend the segmentation from features to dense representation the authors build an assignment matrix that associates every pixel to one of the moving groups. The assignment matrix is built by an EM algorithm which also estimates the transformation matrix by minimising the difference between the estimated transformation and the mean flow computed by OF. The authors say that the algorithm is competitive for situations where background subtraction works. In fact, this technique has problems with noisy images and light changes and it can only segment rigid independent motions.

Brox and Malik (2010) [50] propose to compute a dense OF field and extract measures of similarity between each pair of trajectories. The similarity involves Euclidean distances between trajectories in given windows of time and the average flow variation of the local field (as a negative contribution to the similarity). The authors also propose a spectral clustering method that includes a spatial regularity constraint. By exploiting such a regularity constraints the algorithm is able to estimate the number of clusters in the scene by model selection. The results presented in this work show accurate segmentations. However, the fact that the algorithm relies on 2D image spatial distances may create some problems when dealing with non-rigid motions. Furthermore, as a consequence of the use of 2D image distances, it seems that the algorithm requires long sequences in order to be able to perform the segmentation accurately. In fact, if two object are close and they undergo two different, but not long, motions they are likely to be given a high similarity and, therefore, be clustered as the same object.

OF, theoretically, can provide useful information to segment motions. However, OF alone cannot deal with occlusions or temporary stopping. Moreover, in its most simple version it is very sensitive to noise and light changes. Often, statistical techniques or spatial analysis (like colour or texture) are required to increase OF robustness.

2.3.5 Layers

One of the first layer techniques was proposed by *Wang and Adelson* (1993) in [94]. The key idea of layer-based techniques is to divide the image into layers with uniform motion, and establish some geometrical relations between them. Furthermore, the most refined techniques, associate each layer with a depth level and a "transparency" level that determines the behaviour of the layers in case of overlap. This approach is often used in stereo vision as it is easier to compute depth distances. However, even without computing the exact depth it is possible to estimate which objects move on similar planes. This is extremely useful as it helps to solve the occlusion problem. Another of the earliest techniques based on layers was proposed by *Criminisi et al.* (2006) in [95]. In their proposal the authors probabilistically fuse together motion, colour and contrast cues with spatial and temporal priors to infer layers accurately. They use an efficient motion vs non-motion trained classifier to operate directly and jointly on intensity-change and contrast. Its output is then fused with colour information. The prior on segmentation is represented by a second order, temporal, Hidden Markov Model, together with a spatial Markov Random Field favouring coherence except where contrast is high. Finally, accurate layer segmentation and explicit occlusion detection are achieved by binary graph cut. This technique works well when the final aim is to segment foreground from background, but there is no evidence that the same accuracy could be maintained in presence of more than two motions or with a moving camera.

Kumar et al. (2008) [27] propose a method for learning a layered representation of the scene. They initialise the method by first finding coarse moving components between every pair of frames. They divide the image into patches and find the rigid transformation that moves the patch from frame j to frame $j+1$. The initial estimate is then refined using $\alpha\beta$-swap and α-expansion algorithms [96]. Figure 2.6 gives an example of how a sequence of frames can be used to learn layers, and the segments (objects or part of an object) that lie in a specific layer. The method performs very well in terms of quality of the segmentation

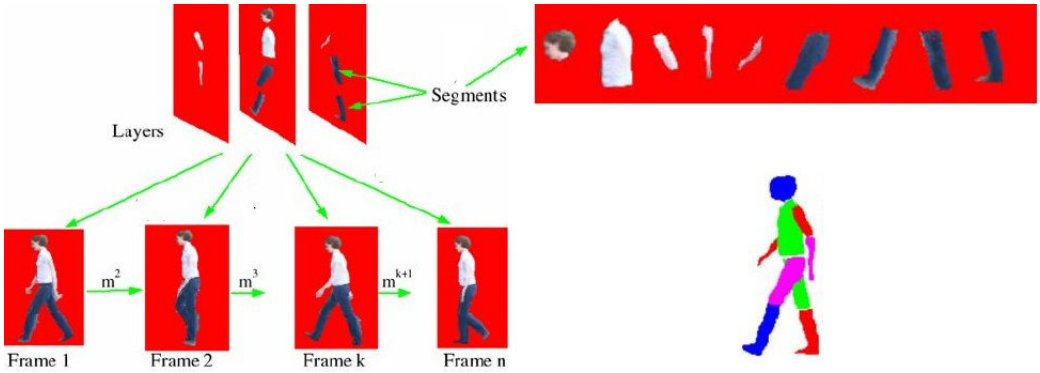


Figure 2.6: Example of layers representation. Image taken from [27].

and is able to deal with occlusions and non-rigid motion. The authors also reported one sequence with moving (translating) camera. Unfortunately, there is no additional information regarding the performance of the algorithm with moving cameras and non static background. The main drawback of this method is its complexity. Furthermore the accuracy of the model degrades if the parameters are not tuned properly.

Min and Medioni (2008) [51] present a spatio-temporal approach to produce motion segmentation and dense temporal trajectories. The first step of the algorithm consists of finding matches simply by cross correlation. Then, for all the matches a 5D token is generated with the information of the 2D position in the frame, time and 2D velocity. The 5D token can be seen as extension of a 3D fibre bundle. Each token, and its neighbours in terms of spatial vicinity (spatial smoothness is also assumed), are used to estimate the generated layer. The layer is a 3D variety described by three tangents and two normals which can be extracted by the 5D tensor built by the tokens. Next step is an outlier rejection and a "densification" process (in order to fill missing data), this is again done by tensor voting. In order to produce more accurate results the last two steps are constrained within pre-segmented areas based on colour segmentation (performed by Mean-Shift). The final step consists of segmenting the layers depending on their normals and on the velocities of the points at the border of the layers. The authors explain that even though, theoretically, this technique should work with any kind of motion, in some circumstances the cross-correlation match may fail, leading to a wrong segmentation. The algorithm seems to have

a very good performance but has similar problems to the previously explained technique. As an example this algorithm has seven parameters that have to be tuned depending on the type of sequence. Another weak point is the strong dependency on the initial colour segmentation: in order to work the colour segmentation has to over-segment the image so that in each segment there is only one motion.

Nordberg and Zografos (2010) [52] present a method based on the geometry of the 6 points [97]. The method finds initial cluster seeds in the spatial domain, and then classifies points as belonging to the cluster that minimises a motion consistency score. The score is based on a geometric matching error measured in the image, implicitly describing how consistent the motion trajectories of 6 points are relative to a rigid 3D motion. Finally, the resulting clusters are merged by agglomerative clustering using a similarity criterion. The method is extended by *Zografos et al. (2010)* [53] to adopt a new matching score and a modified classification algorithm. These changes improve the performance and allow the algorithm to produce one of the lowest misclassification rate in the state of the art of motion segmentation when tested on the Hopkins155 database. However, in order to obtain such a result the parameters of the algorithm have to be tuned per each sequence and a good initialisation remains crucial. Furthermore, the number of motions is a required prior knowledge.

Xu et al. (2011) [54] present a layer-based technique that merges temporal and spatial information. The authors estimate 2D motion features by SIFT matches and simultaneously perform, on each frame, colour segmentation (tuned to over-segment) by graph-cut [5]. After this step an iterative routine merges the over-segmented areas according to the analysis of the 2D motion features that are contained in each segment. Assuming that there are points tracked also in the background, and that the camera is not moving, they estimate the camera parameters in order to compute the 3D shape of the objects. Finally, they use a Hidden Markov Model-based algorithm [98] in order to refine the segmentation. The results presented in the paper show the high quality of the segmentation obtained, however, almost all of the results contain only one moving object per sequence. As per most of the

layer-based techniques, this one also requires the tuning of many parameters. Moreover, as it is presented in the paper, the algorithm can only cope with rigid motions and assumes that the camera does not move.

Wang et al. (2011) [55] propose an approach to extract motion layers from a pair of images with large disparity motion. First, motion models are established by SIFT matches and by a topological clustering algorithm described in the paper. Then, the motion of each cluster, with no less than three matches, is modelled by an affine transformation. With the obtained motion models, a graph cut-based algorithm is employed to segment the scene into several motion layers. This technique seems to have more of a problem than other layer-based techniques to handle occlusions and is sensitive to change in the illumination of the scene.

Layers solutions are very interesting. It is probably the most natural solution for occlusions: human beings also use depth concept to solve this problem. The main drawback is the level of complexity of these algorithms and the number of parameters that have to be tuned manually.

2.3.6 Manifold Clustering

Manifold clustering aims to define a low-dimensional embedding of the data points (trajectories in motion segmentation) that preserves some properties of the high-dimensional data set, such as geodesic distance or local relationships. The manifolds are then clustered in the projected subspace. The projection is usually performed to reduce the amount of computation and to exploits properties of the manifolds that become more evident in the low-dimensional embedding.

Nowadays, manifold clustering is a popular technique applied in many fields. Motion segmentation seems one of the most natural applications. In fact, from structure from motion theory, it is known that the column vectors of the trajectory matrix form an algebraic subspace (here called *global subspace*) whose dimension is equal to the rank of the trajectory matrix. Within this global subspace, the trajectories that belong to the

same motion span a low-dimensional linear subspace (called *local subspace*). Specifically, the trajectories generated by the points of a rigid object form a linear subspace of no more than 4 dimensions [10]. While the trajectories generated by the points of a non-rigid object can be approximated by a combination of k weighted key basis shapes, and they form a linear subspace of no more than $3k + 1$ dimensions [99, 100].

Finally, motions can be articulated. There are two types of articulated motions depending on their joint. When two objects (in an articulated motion also called segments) are linked by a two or three degree of freedom *universal* joint, the position of one body with respect to the other is constrained but their rotations are independent. In these cases (assuming the two segments are rigid) two 4D subspaces have a 1D intersection, therefore, $rank(\mathtt{W}) = 7$ [101]. When segments are linked by a *hinge*, their relative orientation is also constrained since their coordinate frames have a common axis that is parallel to the axis of rotation. In this case, all points on the rotation axis satisfy both motions such that the subspaces have a 2D intersection and $rank(\mathtt{W}) = 6$ [101].

Given this definition of rigid, non-rigid and articulated motions, it is possible to define also more formally independent and dependent motions. From an algebraic point of view, two motions are independent if the pairwise intersection of the two subspaces generated by their trajectory vectors is empty. On the other hand, motions are dependent if the pairwise intersection of the subspaces is not the zero vector. The dependency can be partial (which means that the subspaces intersect in some points) or complete (which means that one subspace is completely inside the other) [65].

At this point it is also possible to introduce one more attribute that, for the sake of completeness is described here, but that is not considered in the table because very few authors clearly explain this aspect. In this review it is called "degeneracy". Many authors use it when they refer to dependent motions or non-rigid and articulated motions, but it is used here with a different meaning. Degeneracy is another aspect of a single motion. *Non-degenerate motion* is a motion whose subspace dimension is maximum (i.e. 4 for a rigid motion, $3k + 1$ for a non-rigid motion, etc.). Whereas, *degenerate motion* is a motion

whose subspace has a dimension which is lower than its theoretical maximum. To provide an intuitive example of a degenerate motion one could think of a car moving on a straight street. The motion of the car is described only by a translation with no rotation. Therefore, the generated subspace has size 1 instead of being size 4 (rigid motion). A formal discussion of degenerate deformations can be found in [102].

Techniques that belong to the manifold clustering category, in general, try to estimate the subspaces generated by each motion in order to obtain the segmentation. Manifold clustering comprises a large number of different techniques, and a further classification can help in giving some order. Manifold clustering can be divided into, iterative solutions, statistical solutions (solutions that fall inside this category could be placed in the previous statistics group) sparse-based algorithms, factorisation solutions and subspace estimation solutions.

Iterative

Iterative algorithms work by starting from an initial solution and iteratively refining the clusters in order to fit a model which describes the manifolds.

Ho et al. (2003) [56] present an algorithm called "K-Subspace Clustering" for face clustering. However, the same idea could be adopted to solve the motion segmentation problem. K-Subspace can be seen as a variant of K-means. K-Subspace iteratively assigns points to the nearest subspace, which, after having gained a new member, is updated by computing its new basis. The basis is estimated such that the sum of the square distances from the basis to all the points of that cluster is minimum. The algorithm ends after a finite number of iterations. The main drawback of this algorithm is that it needs an initialisation of the clusters. Moreover, depending on the quality of the initialisation, the algorithm may converge to a local solution which could be far from the optimal solution.

A different strategy is presented by *da Silva and Costeira (2008)* [57]. The authors of this work propose a subspace segmentation algorithm based on a Grassmannian minimisation approach. The technique consists of estimating the subspace with the maximum

consensus (GMC): maximum number of data that are inside the subspace. Then the algorithm is recursively applied to the data inside the subspace in order to look for smaller subspaces embedded in it. This algorithm works with any kind of motion and it is very robust against outliers thanks to the way the subspaces are constructed. Moreover, it does not require any rank estimation of the trajectory matrix, however, the local subspace size generated by each object is required. The main drawbacks are: the parameters that have to be tuned in the presence of noise, and the non convergence to the optimal solution in the cases where the initialisation is not good. The same authors further extended the algorithm by adopting a new similarity measure between subspaces [58]: the Normalized Subspace Inclusion (NSI). By exploiting NSI the authors reduce the sensitivity of GMC to the estimation of the local subspace size dimension. In fact, now only an underestimation of the local subspace size is required. Then, embedded subspaces with high NSI similarity can be merged. In this paper they also explain that common similarity measures adopted for subspace clustering are ambiguous or geometrically incorrect. Both of the algorithms assume that the number of motions is known.

Iterative approaches are in general robust to noise and outliers, and they provide good solutions if the number of clusters and the dimensions of the local subspaces are known. This prior knowledge can be clearly seen as their limitation as this information is not always available. Moreover, they require an initial estimation and are not robust against bad initialisations and hence, are not guaranteed to converge.

Statistics

Statistical theory can be used also in the case of manifold clustering. In this section a few techniques that employ statical tools are presented.

Fishler and Bolles (1981) [59] present the RANdom SAmple Consensus (RANSAC) algorithm. RANSAC tries to fit a model to the data randomly sampling n points, then it computes the residual of each point to the model and those points whose residual is below a threshold are considered inliers. The procedure is repeated until the number of

inliers is above a threshold, or enough samples have been drawn [22]. This algorithm can be applied to motion segmentation by assuming that dimension of subspaces is $d = 4$ and using PCA for basis estimation of the subspaces. Hence, RANSAC is able to handle a relatively high presence of outliers. On the other hand, the fact that it uses a fixed dimension makes RANSAC not suitable to deal with degenerate, non-rigid or articulated motions. Moreover, when the number of clusters increases, the probability of sampling n points of the same motion decreases and so does the performance.

Kanatani and Matsunaga (2002) [60] use a statistical framework for detecting degeneracies of a geometric model. They use the geometric Akaike information criterion (AIC), defined in [89], in order to evaluate whether two clouds of points should be merged or not. Doing so they can segment without any estimation about the number of moving objects. However, rigid independent motions are assumed. Moreover, this technique works well only when it is free of noise.

Sugaya and Kanatani (2004) [61] analyse the geometric structure of dependent motions, and suggest a multi-stage unsupervised learning scheme. They use EM by progressively assuming motion models from particular to general. Once the tested motion fits the motion under analysis the segmentation does not change. This technique works with rigid motions and it assumes that the number of motions is known. Moreover, a good initialisation is the key to obtain a good performance. The same authors extended this proposal in [64] (2010) where they substitute the previous initialisation model, which was based on a heuristic scheme, with an analytic computation based on the Generalized Principal Component Analysis [103] (GPCA). GPCA is used to fit 2D affine spaces in 3D by the method of Taubin [104]. This method with such an improved initialisation outperforms the previous proposal.

Gruber and Weiss (2004) [62] extend the EM algorithm already proposed in [105] for the single object case in order to deal with multiple objects and missing data. In [63] (2006) they further extend the method incorporating non-motion clues (such as spatial coherence) into the M step of the algorithm. This technique can successfully deal with

noise and missing data and has the advantage of not requiring assumptions on the rank of the trajectory matrix. However, the minimisation problem used is convex only in the case of one or two moving objects, while for more objects it becomes non convex and the results depend on the initialisation.

Statistical solutions have more or less the same strengths and weaknesses of iterative techniques. They are robust against noise whenever the statistical model is built taking the noise explicitly into account. However, when noise is not considered, or is not modelled properly, their performances degenerate rapidly. As previously stated for general statistical approaches: they are robust as long as the model reflects the actual situation.

Sparse representation

Lately, some techniques that take advantage of the compressed sensing [106] field have been proposed. The general idea is to use the smallest amount of information in order to represent the subspaces generated by each motion.

Rao et al. (2008/2010) [65, 66] propose a framework for motion segmentation of trajectories in the presence of corrupted trajectories and missing data. The framework uses the Agglomerative Lossy Compression algorithm [107] (ALC). This algorithm consists of minimising a cost function by grouping together trajectories. Roughly speaking, the cost function is given by the amount of information required to represent each manifold (it could be for example the number of bits), summed over all of the manifolds. This technique has very good performance, however, its greedy implementation does not guarantee to find the global maximum. Another problem is the need to tune a parameter, which depends on the noise level of the input sequence and on the number of clusters of the video. Although, the tuning can be automated by trying many different values and choosing at the end the solution with the lowest cost, this process is highly time-consuming. Finally, ALC performance seems to degrade in the presence of more than 3 motions, as shown in [108].

Elhamifar and Vidal (2009) [67] propose a new way for describing subspaces called Sparse Subspace Clustering (SSC). The authors exploit the fact that each point (in the

global subspace) can be described by a sparse representation that uses the information of the remaining points. By using ℓ_1 optimisation, and under mild assumptions, they estimate the subspaces and they build a similarity matrix which is used to obtain the final segmentation by spectral clustering. SSC has a very good performance (one of the best on the Hopkins155 database) and it is able to deal with a small amount of missing data. In order to deal with missing data SSC requires that, once the rows and columns of the interrupted trajectories are removed, there is enough information left in order to perform the segmentation and then to fill the missing data entries. Theoretically, SSC is able to segment also without knowing the number of motions, however, in [67] there is no experiment under this condition. The main drawback of SSC is the presence of some parameters that should be tuned for each sequence, especially the parameter that controls the degree of sparsity in the ℓ_1 optimisation. This parameter seems to play an important role. The results presented in [67] cannot be reached if the tuning process is not performed for every sequence.

Techniques based on sparse representation provide a connection between coding theory and motion segmentation. They perform extremely well with a variety of motions, however, they tend to depend on parameters that have to be tuned per each sequence according to the number of motions and the amount of noise.

Factorisation methods

Since *Tomasi and Kanade (1992)* [10] introduced a factorisation technique to recover structure and motion (structure from motion problem) using features tracked throughout a sequence of images, factorisation methods have become very popular especially thanks to their simplicity. The idea is to factorise the trajectory matrix `W` into two matrices, motion `M` and structure `S`, under the assumption that the object is rigid and the camera is orthographic. Such a factorisation is shown in Figure 2.7. Before the factorisation the origin of the world coordinate system is moved at the centroid of all the feature points. This corresponds to compute the translation of the object (or of the camera if the object is stationary

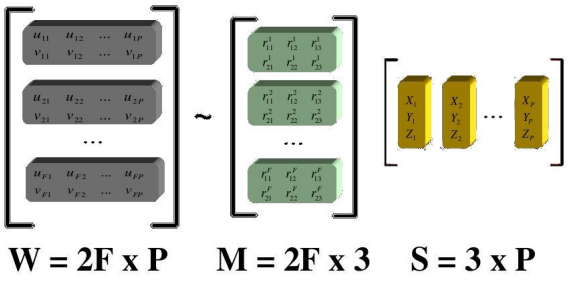


Figure 2.7: Basic idea of structure from motion: factorise matrix W into matrices M and S.

and the camera is moving) and remove its influence. Therefore, after the factorisation the motion matrix M is composed by two rows for each frame f, for $f = 1, \ldots, F$, that define the first two rows of a rotation matrix $\mathtt{R}_f$ of size 2×3. $\mathtt{R}_f$ describe the rotation of the object from an initial reference framework to the to position in the frame f. The matrix S contains the metric coordinates of the 3-D points. Therefore, in the absence of noise, the trajectory matrix is at most rank $r = 3$. By exploiting this constraint, W can be decomposed and truncated using the singular value decomposition (SVD), as shown in Figure 2.8. Because

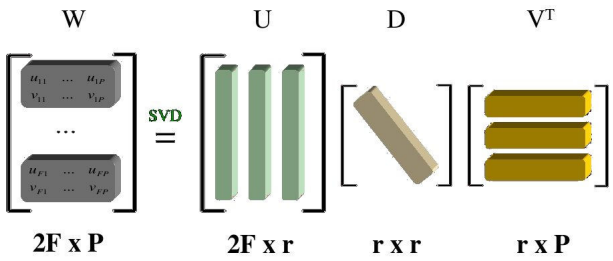


Figure 2.8: W can be decomposed and truncated using SVD and exploiting its rank deficiency.

the rows of the rotation matrices R_f are orthonormal, matrix D can be identified using the orthogonality constraints: $\mathtt{R}_f\mathtt{R}_f^T = \mathtt{I}_2$, for $f = 1, \ldots, F$, $\mathtt{I}_2$ being the identity matrix of size 2×2. Finally, W can be decomposed, up to a scale factor, as shown in figure 2.9. This algorithm works for one static object viewed from a moving camera which is modelled with the simplest affine camera model: the orthographic projection. Despite the fact that this method gives the 3D structure of the object and the motion of the camera, it has evident limits: it cannot really segment (it assumes that the features belong to the same object),

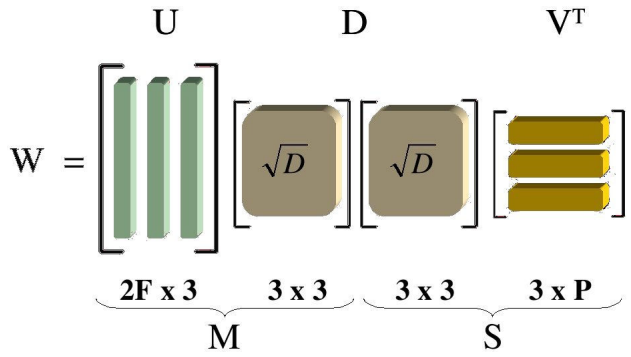


Figure 2.9: Exploiting orthogonality constraints of the motion matrix and assuming rank $r = 3$, $\mathtt{W}$ can be decomposed into motion and structure, up to a scale factor.

it can deal only with a single rigid object, it is very sensitive to noise and it is not able to deal with missing data and outliers. However, it was the first method of this family and the solution is mathematically elegant. From this initial structure from motion approach, and following the same idea of forcing the rank constraint, more approaches were proposed with the aim to improve the original technique [109–111]. Others pointed out some weaknesses of the original method. *Anandan and Irani (2002)* [112] point out that SVD is powerful at finding the global solution of the associated least-square-error minimisation problem only when the errors in the position of the tracked features are uncorrelated and identically distributed. This is rarely the case in real images. Hence, they propose a method that is based on transforming the raw-data into a covariance-weighted data space, where the components of noise in the different directions are uncorrelated and identically distributed.

Okatani and Deguchi (2007) [113] present another factorisation approach. They use the Gauss-Newton method, originally proposed by Wiberg [114], to factorise the trajectory matrix. The idea is to separate the variables of the problem into two sets and estimate them alternatively, similarly to what is done in the EM algorithm. *Hartley and Schaffalizky (2003)* [115] introduce the PowerFactorization algorithm. PowerFactorization is an iterative factorisation algorithm which is guaranteed to converge from any random initialisation. It may be applied to a variety of 3D reconstruction problems, but most importantly it can cope also with missing data by filling missing entries.

Even if none of the previous methods can be directly used for segmentation, they

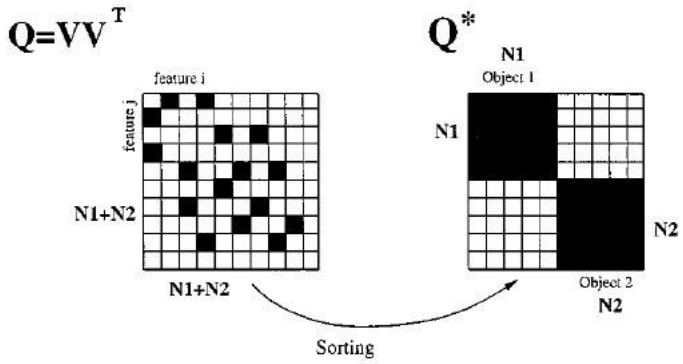


Figure 2.10: [68] computes the interaction matrix Q and finds the permutation of rows and columns that gives a block diagonal matrix. Image taken from [68].

started a new field of research: structure from motion based on factorisation. This new field eventually led to solutions also for the segmentation problem. In the following a review on factorisation methods that can be used to solve the motion segmentation problem is presented. A comprehensive survey on factorisation methods, with more detail of some of these techniques, can be found in [9].

Costeira and Kanade (1998) [68], use the same factorisation technique presented in [10] and then they compute the shape interaction matrix $\mathtt{Q} = \mathtt{V}\mathtt{V}^T$, where V is the right singular vectors matrix of size $P \times r$, P being the total number of tracked features and r the rank of the trajectory matrix. The shape interaction matrix, among other properties, has zero entries if the two indices represent features that belong to different objects, non-zero otherwise. Hence, the algorithm focuses on finding the permutation of the interaction matrix that gives a block diagonal matrix structure, as shown in Figure 2.10. Unfortunately, this process is time consuming and in the presence of noise the interaction matrix may have non-zero values, even when it should have zero entries. Moreover, this technique works only if the subspaces to which each features belong do not intersect (i.e. the subspaces are independent).

Ichimura and Tomita (2000) [69] also exploit the shape interaction matrix, however, Q is built differently than in [68]. Their idea is that if in the shape interaction matrix there are all of the P tracked features the optimisation problem becomes non-linear. Their ob-

servation is that, if the rank of the trajectory matrix is r then there are only r independent features, therefore, only those r features should be selected to build the shape interaction matrix. For example if there are N rigid objects only 4 features per object should be used. The problem is how to select features from different objects without knowing the segmentation. First, the rank r of the trajectory matrix is estimated by SVD, which provides also the right singular vectors $\mathtt{V}$. Then QR decomposition [116] is used so that $\mathtt{V}^T\Pi = \mathtt{Q}_{QR}\mathtt{R}_{QR}$, where Π is a permutation matrix. At this point the matrix $\mathtt{Q}_{QR}$ (note that this is not the shape interaction matrix but it is the matrix obtained by the QR decomposition) contains the basis vectors of the "shape space". By using the first r vectors of $\mathtt{Q}_{QR}$ it is possible to know which are the r independent features that should be used in order to compute the shape interaction matrix $\mathtt{Q}$. Note that the mapping from the shape space to the vectors of $\mathtt{V}$ is kept by the permutation matrix. At this point the algorithm becomes similar to [68] and the segmentation of the first r features is obtained by permutation of $\mathtt{Q}$. Then the existing segmentation can be exploited to estimate the basis of the subspace generated by each of the motions, and any subspace classification method can be used in order to classify the remaining $P - r$ features. This technique is very sensitive to noise, besides, rank estimation is known to be a very challenging problem.

All of the previous techniques assume that objects follow independent motions. *Zelnik-Manor and Irani (2003)* [70] study the degeneracy in cases of dependent motions. They propose a factorisation method that consists of building an affinity matrix by using only the dominant eigenvector of the trajectory matrix and estimating the rank r of the trajectory matrix by studying the ratio between the eigenvalues. The results presented in the paper are obtained from forcing the number of objects and the local subspace dimensions.

The following technique, *Zelnik-Manor and Irani (2004)* [117], cannot be considered as a motion segmentation algorithm, however, it is worth to mention it as it introduces a dual approach to the usual factorisation strategies. Traditionally, factorisation approaches provide spatial clustering by grouping together points that move with consistent motions. In [117] the authors propose a temporal clustering by grouping together frames to capture

consistent shapes. The advantages are the smaller quantity of tracked points required and the smaller dimensionality of the data. They also show that any of the existing algorithms for column factorisation can be transformed in row factorisation.

Factorisation techniques are based on a very simple and elegant framework. However, factorisation methods are particularly sensitive to noise and cannot deal very well with outliers and missing data. Moreover, most of these techniques assume rigid and independent motions.

Subspace estimation

Subspace estimation approaches exploit algebraic tools for the estimation of the subspace bases generated by each motion.

Vidal and Hartley (2004) [71] present an algebraic geometric approach that uses PowerFactorization [115] and GPCA [103]. First, by exploiting the fact that trajectories of a rigid and independent motion generate subspaces at most of dimension 4, they project the trajectories onto a 5 dimensional space using PowerFactorization. Then, GPCA is used to fit a polynomial of degree N, where N is the number of subspaces (i.e. the number of motions), through the data and estimate the bases of the subspaces using the derivatives of the polynomial. Then common measure of distance between subspaces, such as principal angles [118] (refer to Section 3.1.1 for a formal definition of principal angles), are used to compute the similarity between the generated subspaces. This algorithm is able to deal with missing data. As explained by the authors, the main drawback of GPCA is that the performance degrades when the number of objects and the dimension of the largest underlying subspace increase. More recently, *Vidal et al. (2008)* [75] extend the previous explained framework using RANSAC in order to be able to deal with outliers. Theoretically, these two techniques can estimate the number of motions in the scene and the size of the subspaces, however, these estimations are not robust in the presence of noise. In general, a weakness of GPCA-based techniques is that, the required number of sample points per each motion grows exponentially with the total number of motions. Specifically,

given N motions and d being size of the largest generated subspace, the number of trajectories required in order to estimate correctly the subspaces is $O((d+1)^N)$. In practise, the number of trajectories can hardly satisfy GPCA's requirement for it to handle more than 3 rigid motions [31].

Yan and Pollefeys (2006/2008) [31, 72] present an algorithm known as Local Subspace Affinity (LSA) for segmentation of different types of motion: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. The key idea is to estimate and compare the local subspaces in which each trajectory lies. Thus, the subspaces generated by each trajectory, and its nearest neighbours, are estimated. An affinity matrix that compares the subspaces is built by using principal angles as a measure of distance between subspaces. The final segmentation is obtained by computing the spectral clustering of the affinity matrix. This algorithm seems promising, though there are a few issues that need to be solved. Usually the knowledge of the number of moving objects is required, moreover, the dimensions of the global subspace (i.e. the rank of the trajectory matrix) and of each local subspace are crucial information for the algorithm, but their estimation is a particularly difficult task. *Tron and Vidal* in [22] present a benchmark that compares GPCA [71], a RANSAC-based approach [119] and LSA [31]. They conclude that LSA is the best performing algorithm in the case of non-missing data, however, they also state that tuning the parameter that controls the rank estimation was a difficult task and the number of moving objects was known a priori.

Goh and Vidal (2007) [73], by starting from the Locally Linear Embedding algorithm [120], propose the Locally Linear Manifold Clustering algorithm (LLMC). With LLMC the authors try to deal with linear and non-linear manifolds. The same authors extend this idea to Riemannian manifolds (2008) [76]; they project the data from the Euclidean space to a Riemannian space and reduce the clustering to a central clustering problem. The main problem of these techniques is what is known as the "curse of dimensionality" (i.e. the higher the number of motions, the more data is required to be able to perform the segmentation, and at the same time the more the data to process, the longer

the time required for completion). Moreover, similarly to LSA, the number of objects and the dimension of the subspaces can be theoretically estimated, however, the estimation is not very robust in the presence of noise.

Julià et el. (2008) [74] propose a method similar to LSA in order to perform the segmentation in the case of missing data. The idea is to fill the missing data using a frequency spectra representation for the matrix estimation. When a full trajectory matrix is obtained, a clustering algorithm based on normalized cuts is applied in order to provide the segmentation. Even though the method can estimate also the number of moving objects, it assumes only the presence of rigid and non-degenerate motions.

Chen and Lerman (2009) [77, 78] present the Spectral Curvature Clustering (SCC), which is a generalisation of Yan and Pollefeys' method (LSA). SCC differs from LSA for two main reasons. The first is related to the way the subspaces are compared, SCC uses polar curvature while LSA uses principal angles. The second reason is how they select which trajectories have to be combined together in order to estimate the local subspaces. SCC uses an iterative solution based on random sampling, while LSA uses a Nearest Neighbours (NN) solution. The iterative solution proposed in SCC seems to be more robust than LSA in the case of dependent motions. In fact, in such a case a NN selection may choose trajectories that belong to different manifolds, while SCC should be able to refine its choice and eliminate outliers within a few iterations.

Kim and Agapito (2009) [79] propose to compute different shape interaction matrices [68] that are built assuming an increasing dimension of the global subspace. The interaction matrices are then merged together by the Hadamard product (the entry-wise product) to compose the final affinity matrix that is clustered by K-means. This algorithm is simple and very effective, as proved by the results on the Hopkins155 database. The limitation of this technique is in the lack of ability to deal with articulated motions, as the authors explain in the paper. Moreover, the number of motions is an assumed prior knowledge.

Yang et al. (2009) [80] exploit the same idea of LSA but instead of 2D flow fields they use 1D pixel intensities as the input measurements. Specifically, they introduce the notion of the "pixel intensity trajectory", a vector that represents the intensity changes of a specific pixel over multiple frames. Like 2D feature trajectories, intensity trajectories of pixels associated with the same motion span a low-dimensional linear subspace. They formulate the problem of motion segmentation as that of clustering local subspaces. This technique has the advantage over LSA of not requiring a tracking step and of being less sensitive to noise. However, it suffers from all of the other weak points of LSA. Moreover, in [80] only rigid motion experiments are presented.

Lauren and Schnörr [81] use the same idea of LSA but with a few changes. Firstly, the authors show that the dimension of the global subspace is a critical choice, and that the dimensions chosen in prior works [65, 103, 121] do not offer sufficient separability between subspaces. Therefore, they suggest lower and upper bounds of this dimension. Then, they compute the affinity between subspaces using angular information (more efficiently than LSA as they do not compute PA between local subspaces, but normal dot products between points). Finally, they use spectral clustering of the affinity matrix to obtain the final segmentation. This technique is faster than LSA as it does not require estimation of each local subspace, however, it does require knowledge of the dimensions of the expected local subspaces and the number of motions.

Subspace estimation techniques can deal with intersection of subspaces and generally they do not need any initialisation. However, all of these techniques suffer from common problems: curse of dimensionality, weak estimation of the number of motions and of subspace dimensions. The curse of dimensionality is mainly solved in two ways: projection into smaller subspaces or random sampling. Whereas the number of motions and the subspace dimension estimations remain two open issues.

2.4 Analysis of the state of the art

In this section a brief summary of each of the classes analysed is presented.

Techniques that rely on image difference typically employ a dense-based representation of objects. This class of algorithms is based on a simple idea but it provides good overall results. Image difference can deal with occlusions, multiple objects and non-rigid objects. The issues of this class are related with the difficulty to deal with non-still cameras and with temporary stopping. Moreover, high sensitivity to noise and light changes are also problems that need to be taken into account. A common solution to some of these issues requires the use of history models of the background.

Another class that mainly uses a dense-based representation is the statistics. Techniques that belong to this class can deal with multiple objects, occlusions and temporary stopping. When the model on which they rely is a good approximation of the actual situation, these algorithms are very robust. However, when the model does not reflect the actual situation their performances degrade quickly. In order to build a realistic model, typically, some sort of prior knowledge is required.

At present, wavelets techniques are mainly used for noise reduction. However, it is possible to find some approaches that use wavelets as a tool for segmentation. Techniques that employ wavelets for segmentation provide good results, but in limited scenarios and with simple motions. Even if, recently, the research community of motion segmentation seems to have lost interest in wavelets, their ability to perform multi resolution analysis could be exploited to extract information about the different depth planes.

The optical flow class of techniques suffers from similar problems that affect the image difference class. OF has always been considered a very important clue for motion segmentation. Nevertheless, OF, alone, is still very sensitive to noise and light changes. Other approaches, such as statistical tools or spatial analysis, need to be merged with OF in order to increase its robustness.

Layer-based solutions seem the most natural way for solving the occlusion problem. In addition, Table 2.1 shows that they are among the most complete techniques. However, the price to pay for being this thorough is a high complexity of the algorithms, which often involve a high number of parameters that have to be tuned for each sequence.

Manifold clustering can easily tackle temporary stopping and provides good overall results. This class of solutions is usually based on feature points, and therefore, the algorithms can be easily extended to recover the 3D structure of moving objects. The intense work done on manifold clustering for motion segmentation has led to satisfactory performances, which make these solutions appealing. However, more work has to be done in order to have a motion segmentation algorithm that is completely automatic and independent from prior knowledge. A common drawback to all these techniques is that they can deal very well when the assumptions of rigid, independent and non degenerate motions, are respected, but if one of these assumptions fails, then problems start to arise as the properties of motions have to be explicitly taken into account. Within the manifold clustering group five sub-classes were identified: iterative, statistics, sparse, factorisation and subspace.

When information about number of motions and dimension of the subspace generated by each motion is known, iterative solutions are very robust to noise and outliers. However, when these pieces of information, or when a good initialisation, are not available their performances tend to degrade rapidly.

For statistical solutions the same conclusions drawn for general statistical approaches hold. They can deal well with any situation that was properly modelled, however, when this is not the case their solutions become unreliable.

Recently, a new class has emerged: the sparse-based approaches. This class takes advantage of the knowledge gained in the compression and sparse representation communities. Their main drawback is that they suffer from the curse of dimensionality problem and, typically, their results are influenced by the presence of parameters that require tuning.

Factorisation approaches rely on a simple and elegant mathematical framework. Their limitation is their high sensitivity to noise and to the presence of outliers and missing data.

Finally, in recent years, the subspace estimation class has been one of the most prolific. These algorithms can deal with different kinds of motion, provided that local and global subspace dimensions are known. Unfortunately, this estimation is very difficult and most of the time the dimensions are assumed to be known. Another drawback is the assumed

Techniques		Pros		Cons	
Image Diff.		- Simple - Occlusions	- Dependency - Kind of motion	- Sensitive to noise - Temporary stopping	- Moving camera
Statistical		- Occlusions - Temporary stopping		- Sensitive to model - Prior knowledge	- Dependency
Wavelets		- Depth estimation	- Dependency	- Multiple motions	- Kind of motion
O.F.		- Simple		- Dependency - Non-rigid motions	- Sensitive to noise
Layers		- Occlusions		- Complexity	- Many Parameters
Manifold Clustering	Iter	- Extension to SfM - Temporary stopping		- Prior knowledge - Sensitive to initialisation - Kind of motion	- Occlusions - Dependency
	Stat	- Extension to SfM - Temporary stopping		- Prior knowledge - Dependency - Kind of motion	- Occlusions - Sensitive to model
	Sparse	- Extension to SfM - Temporary stopping	- Occlusions - Misclassification	- Time consuming - Dependency	- No justification - Curse of dimensionality
	Fact	- Extension to SfM - Temporary stopping	- Elegant	- Prior knowledge - Dependency - Kind of motion	- Occlusions - Sensitive to noise
	Sub	- Extension to SfM - Temporary stopping	- Misclassification	- Prior knowledge - Curse of dimensionality	- Occlusions

Table 2.3: Summary and generalisation of pros and cons of each class of techniques.

knowledge about the number of motions. As stated also by Shen et al. in [38] the estimation of the number of motions is a practical open issue.

Table 2.3 summarises and generalises the advantages and disadvantages of each class of techniques. This review should have given an idea of how vast the motion segmentation literature is. The fact that research in this field is still active (most of the papers presented here were published after 2005) is a sign of the importance of this problem. On the other hand, effervescent research activity signifies also that many problems have still to be solved and there is not yet an outstanding solution.

Manifold clustering algorithms, specifically the sub-category of subspace estimation, seem to have room for improvement despite that the performance of some of the algorithms is already good. A quick glance at Table 2.2 may draw attention to the fact that the price to pay in order to be able to deal with different kinds of motion is a higher prior knowledge (in particular about the dimension of the generated subspaces). Specifically, the estimation of global and local subspace dimensions is a crucial step as pointed out by many authors [22, 81, 108, 122, 123]. Moreover, another weak point of subspace-based segmentation algorithms is the use of an appropriate similarity measure, in fact, as pointed

out in [58, 124] most of the similarity measures used so far are ambiguous or geometrically incorrect. Another observation is that, with the exception of [63], spatial continuity is not exploited. This may suggest that the use of this information could help to improve the performance of motion segmentation algorithms, especially in terms of robustness and ability to deal with occlusions and missing data.

Considering the aspects that have emerged in this review, a manifold clustering approach could be a good starting point for proposing a novel segmentation technique. Among the manifold clustering techniques reviewed, the Local Subspace Affinity method, proposed by *Yan and Pollefeys* [31, 72], seems a promising approach. LSA is able to segment different types of motion (independent, dependent, articulated, rigid, non-rigid, degenerate and non-degenerate), and it is also able to deal with a small amount of outliers produced by a wrong association of the tracked features. Differently from other techniques, like GPCA that requires $O(d+1)^N$ trajectories (N being the number of motions and d the dimension of the largest underlying subspace), LSA is able to compute the segmentation with just $O(d \times N)$ trajectories. This is the reason why LSA can handle, not only a higher number of motions, but also motions that generate subspaces of a higher dimension. Furthermore, it is based on a relatively simple framework that does not require tuning of a high number of parameters. As already concluded by Tron and Vidal in [22] LSA is the best performing algorithm among LSA, GPCA and a RANSAC-based solution, in the case of non-missing data. Nevertheless, it does present some weaknesses that could be further investigated and strengthened. Moreover, the LSA framework is built in such a way that it can be extended to include more information.

The LSA algorithm is the starting point of this thesis. In the next chapter LSA is explained in detail and its strengths and weaknesses are pointed out.

2.5 Databases

The algorithms proposed in this thesis will be tested and compared with the best state of the art techniques: GPCA [103], LSA [72], ALC [65, 66], GMC+NSI [57, 58], and SSC [67].

The comparisons will be performed on two databases that are now presented.

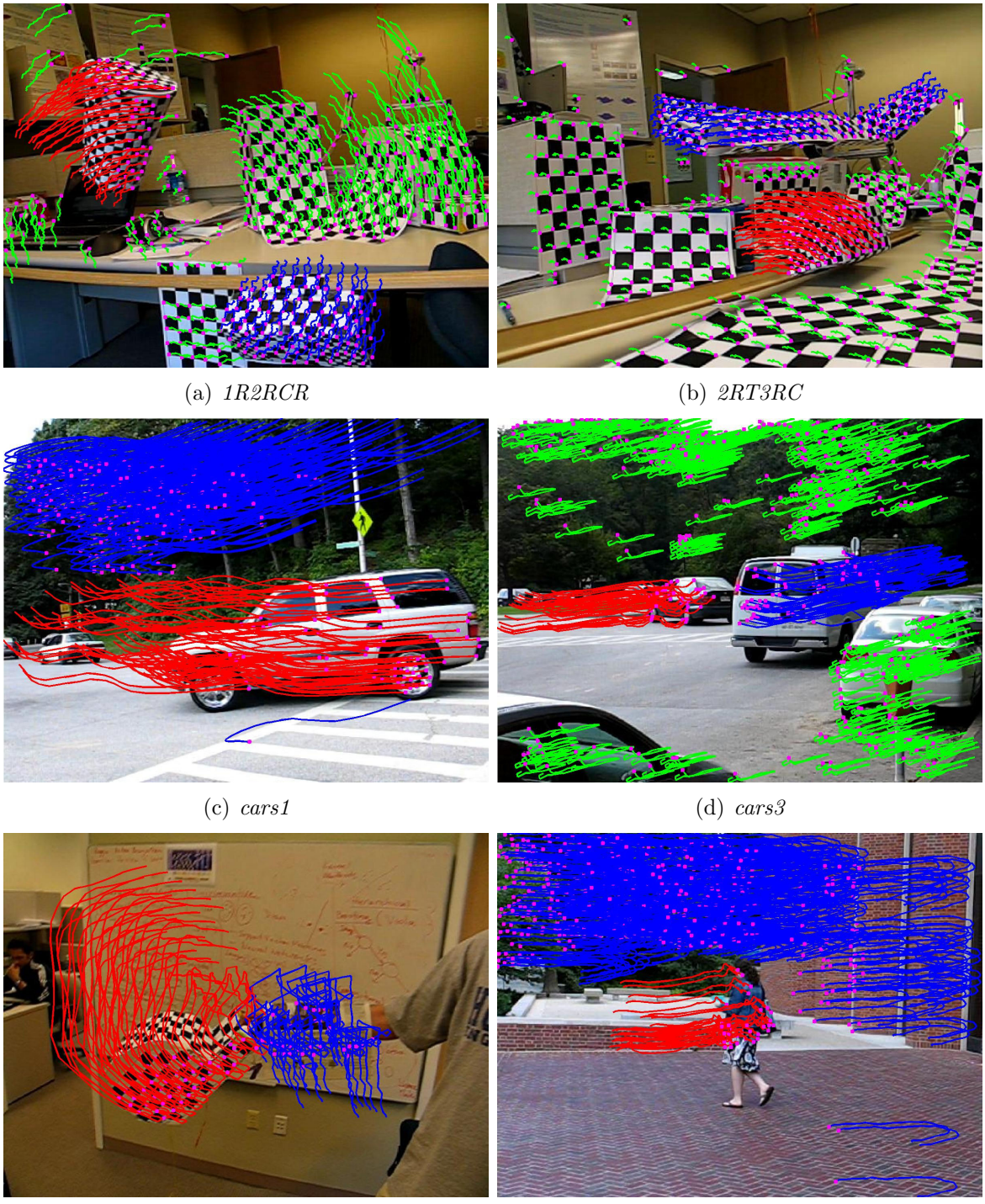
2.5.1 The Hopkins155 database

The main database used in this thesis is the reference benchmark database for motion segmentation algorithms based on feature trajectories. It was proposed by Tron and Vidal in 2007 [22]. The database contains 155 real video sequences: 120 with 2 motions and 35 with 3 motions. There are different types of sequences: Checkerboards, Traffic and Articulated/non-rigid, as shown in Table 2.4. In Figure 2.11 some frames of each group of sequence are shown.

The following description is adapted from the original paper [22] that introduced the database. The Checkerboard group consists of 104 sequences of indoor scenes taken with a hand-held camera under controlled conditions. The checkerboard pattern on the objects, Figure 2.11(a) and 2.11(b), is used to assure a large number of tracked points. Sequences *1R2RC–2T3RTCR* contain three motions: two objects (identified by the numbers 1 and 2, or 2 and 3) and the camera itself (identified by the letter C). The type of motion of each object is indicated by a letter: R for rotation, T for translation and RT for both rotation and translation. If there is no letter after the C, this signifies that the camera is fixed. For example, if a sequence is called *1R2TC* it means that the first object rotates, the second translates and the camera is fixed. Sequence *three-cars* is taken from [125] and contains three motions of two toy cars and a box moving on a table taken by a fixed camera [22].

The Traffic group consists of 38 sequences of outdoor traffic scenes taken by a moving hand-held camera. Sequences *carsX–truckX* have vehicles moving on a street. Sequences *kanatani1* and *kanatani2* are taken from [61] and present a car moving in a car park. Most scenes contain degenerate motions, particularly linear and planar motions [22].

The Articulated/non-rigid group contains 13 sequences that display motions constrained by joints, head and face motions, and people walking. Sequences *arm* and *articulated* contain checkerboard objects connected by arm articulations and by strings, respectively. Sequences *people1* and *people2* display people walking, thus one of the two



(a) *1R2RCR* (b) *2RT3RC*

(c) *cars1* (d) *cars3*

(e) *arm* (f) *poeple1*

Figure 2.11: Examples of the Hopkins155 sequences. The first two frames are taken from the Checkerboards group, then second two frames are taken from the Traffic group and the last two frames are taken from the Articulated/non-rigid group.

	2 motions			3 motions		
	Seq.	Points	Frames	Seq.	Points	Frames
Checkerboards	78	291	28	26	437	28
Traffic	31	241	30	7	332	31
Articulated	11	155	40	2	122	31
All	120	266	30	35	398	29

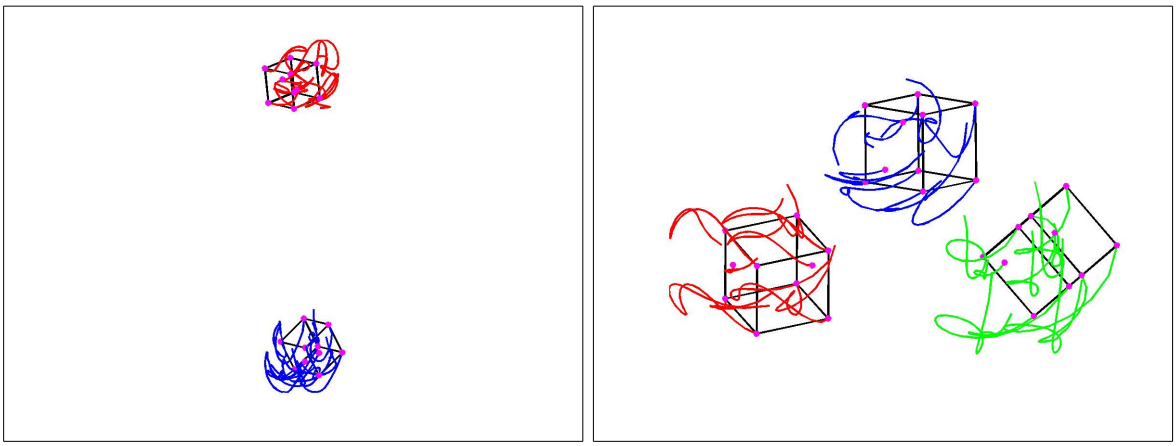
Table 2.4: Summary of the Hopkins155 database. Data taken from [22].

motions (the person walking) is partially non-rigid. Sequence *kanatani3* is taken from [61] and contains a moving camera tracking a person who is moving his head. Sequences *head* and *two_cranes* are taken from [31] and contain two and three articulated objects, respectively [22].

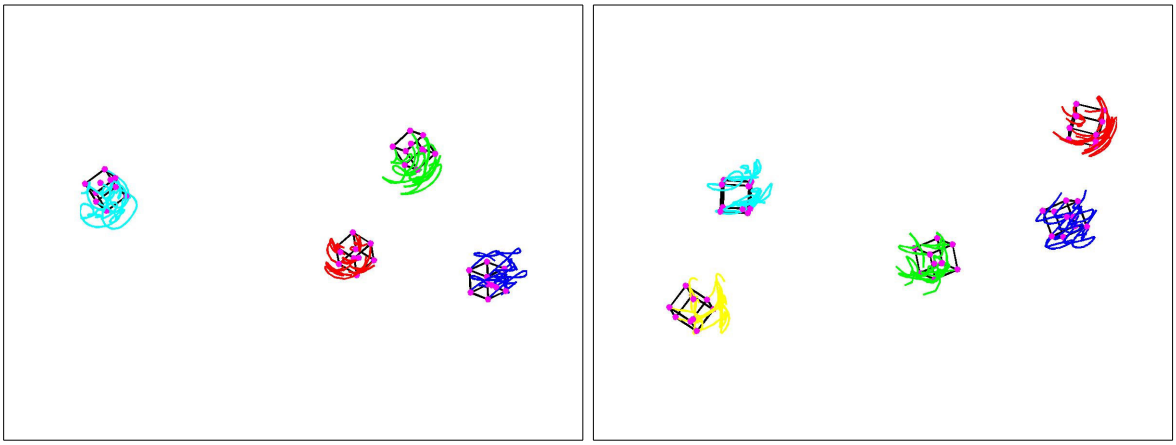
For the sequences taken from [31, 61, 125], the point trajectories were provided in the respective datasets. For all the remaining sequences, the authors of [22] used a tracking algorithm implemented in OpenCV. The ground-truth segmentation was obtained in a semi-automatic manner. First, the tool was used to extract feature points in the first frame and to track them in the following frames. Then an operator removed obviously wrong trajectories (e.g., points disappearing in the middle of the sequence due to occlusion by another object) and manually assigned each point to its corresponding cluster [22]. The number of points per sequence ranges from 39 to 556, and the number of frames from 15 to 100. All the trajectories of the Hopkins155 database are complete, hence, no missing data cases exist. Details of the Hopkins155 database are shown in Table 2.4.

Synthetic database

Besides the Hopkins155 database, also a synthetic database was built and used. Specifically, the database contains synthetic sequences composed of 50 frames, with rotating and translating cubes. Each cube has 56 tracked features. The synthetic database contains 10 sequences with 2 moving cubes, 10 sequences with 3 moving cubes, and so forth up to 5 moving cubes. Examples of synthetic frames (for plotting reasons with just a few tracked features) is shown in Figure 2.12. In order to make the synthetic database a more chal-



(a) *2 motions* (b) *3 motions*



(c) *4 motions* (d) *5 motions*

Figure 2.12: Examples of synthetic sequences with different numbers of motion.

lenging test 4 databases were derived by adding noise with different standard deviations (from 0.5 to 2 pixels) to the original database for a total of 200 synthetic sequences. As for the Hopkins155 database, also the synthetic database does not contain sequences with missing data.

3

Motion Segmentation

In this chapter the LSA algorithm [31, 72] is analysed in more detail with the aim of finding its weaknesses, Section 3.1. To follow this, some of the weaknesses of LSA are strengthened, leading to new robust and efficient algorithms whose performances are comparable to, or exceed, the best state of the art techniques. Specifically, two algorithms proposed in this chapter are related to the rank estimation of the trajectory matrix: Section 3.2 and Section 3.3. An issue regarding the affinity measure will be tackled in Section 3.3.4. Finally, a technique to estimate the number of motions will be presented in Section 3.4. For each of the proposed algorithms results that show the robustness and the improvements of the proposals are presented.

3.1 Local Subspace Affinity (LSA)

The analysis of the motion segmentation state of the art revealed that this problem has been tackled using a variety of techniques. Among all of the studied algorithms the LSA emerged as a very elegant and promising solution. The elegance and the efficiency of LSA, together with the possibility to work on its open issues, make this algorithm very interesting and appealing.

LSA, as well as all of the motion segmentation algorithms based on feature trajectories,

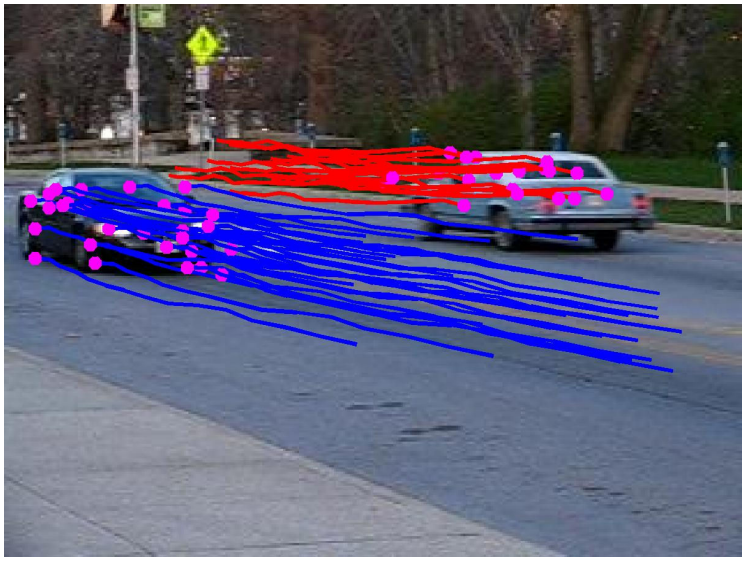


Figure 3.1: Example of an LSA result applied to a video sequence. Blue and red represent different labels, and therefore, different motions. The pink dots are the tracked features extracted from the first video frame. Sequence *cars2_06_g12* of the Hopkins155 database.

assumes that a tracking step has already been performed. Therefore, the input data of LSA is the position (u, v) of each tracked feature in each frame. Usually, this data is stored in a *trajectory matrix* $\mathtt{W}_{2F\times P}$, where F is the number of frames of the input sequence and P is the number of tracked features. Note that $\mathtt{W}$ has to be full, this means that no missing data is allowed. Given $\mathtt{W}$, LSA performs its segmentation and provides as output a motion label for each trajectory $p = 1 \dots P$, where trajectories with the same label should belong to the same motion. A typical output of the LSA algorithm is shown in Figure 3.1

The general idea of LSA is that different motion trajectories, which mathematically are vectors in $\mathbb{R}^{2F}$, when projected onto an opportune lower dimension, generate different algebraic subspaces. This is the common idea of manifold clustering algorithms and it was first introduced by Tomasi and Kanade in [10] (refer to Section 2.3.6 for a detailed explanation). Ideally, trajectories of two independent motions should lie on two orthogonal subspaces, while trajectories of the same motion should lie on the same subspace. Thus, the segmentation can be obtained by studying which trajectory lies on which subspace. Following this idea, LSA estimates the subspace generated by each trajectory and builds an affinity matrix. The affinity measure between two trajectories is inversely proportional

to the distance between the two generated subspaces. The final segmentation is obtained by clustering together trajectories with high affinity.

3.1.1 The algorithm

Given P feature points tracked throughout a video sequence of F frames LSA can be summarised in 5 fundamental steps. At the end of this section the algorithm flow is presented in Figure 3.4.

Rank estimation

The first step of the algorithm consists of estimating the rank of `W`. Theoretically the number of singular values different from zero gives the rank of a matrix. Often the analysis is slightly more complex. In fact, due to the presence of noise, singular values that were supposed to be zero may assume small values. Therefore, the rank could be estimated as the number of singular value above a certain threshold. In the case of the trajectory matrix the estimation is even more challenging. Due to the presence of partially dependent motions, singular values that were supposed to be high become smaller. The combination of these two phenomena makes the singular value spectrum smooth, and the selection of a threshold becomes not obvious.

In LSA such a task is performed by using the Model Selection (MS) technique proposed in [126]. The rank is estimated as shown in equation (3.1):

$$r = \underset{r}{\operatorname{argmin}} \left(\frac{\lambda_{r+1}^2}{\sum_{i=1}^{r} \lambda_i^2} + kr \right), \tag{3.1}$$

λ_i being the i^{th} singular value of `W`, and k a parameter that depends on the noise of the tracked point positions: the higher the noise level is, the larger k should be [72]. For simplicity, from now on this rank estimation will be referred to as MS.

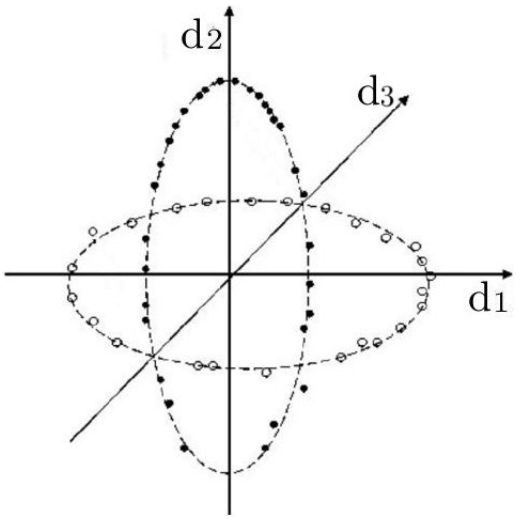


Figure 3.2: Example of trajectories that belong to two different subspaces of dimension 2 projected onto a $\mathbb{R}^3$ unit sphere. The two big circles are the 2 underlying local subspaces. Each of the smaller circles (white and black) represents a trajectory in the new reduced global subspace. White circles represent trajectories that belong to one motion, while black circles represent trajectories that belong to another motion. Image adapted from [31].

Data transformation: projection onto the global subspace

Given the trajectory matrix $\mathtt{W}$ and its estimated rank r it is possible to perform a data transformation. The idea is to consider each of the P columns of $\mathtt{W}$ as a vector in $\mathbb{R}^{2F}$ and to project them onto an $\mathbb{R}^r$ unit hypersphere. This data transformation provides a dimensionality reduction, a data normalisation and a preparation for next step: the local subspace estimations. Figure 3.2 shows an example of trajectories that belong to two different local subspaces of dimension 2 embedded in a global subspace of dimension 3. Note that in the projected subspace each trajectory is represented by one single point (small circle). White circles represent trajectories that belong to one motion, while the black circles represent trajectories that belong to another motion. As it can be seen the small circles that represent trajectories of the same motion lie on the same (or very close to) subspace (one of the two bigger circles). In this example the two subspaces are orthogonal, therefore, the two motions are completely independent.

Such a projection can be performed by singular value decomposition (SVD) and truncation to the first r components of the right singular vectors. Specifically, SVD is applied

to the matrix $\mathtt{W}$, which is decomposed as follows: $\mathtt{W} = \mathtt{U}_{2F\times 2F}\mathtt{D}_{2F\times P}\mathtt{V}^T_{P\times P}$. Dimensionality reduction is achieved by considering only the first r columns of $\mathtt{V}$. Hence, after truncation, each row of the matrix $\mathtt{V}_{P\times r}$ is normalised. An example of this truncation is shown in Figure 3.6(b) on page 78.

Local subspace estimations

In the projected global subspace most trajectories (of the same motion) and their closest neighbours lie on the same subspace. The underlying subspace of a trajectory j can be estimated by local samples from the trajectory j and its n nearest neighbours (NNs), being $n+1 \geq r_h$, where r_h is the highest dimension of the linear subspace generated by the trajectories.

This result can be achieved by SVD. The estimation of the local subspace dimension r_h is required again in order to truncate the SVD result; such an estimation can be done by MS as illustrated in Equation (3.1). The basis of the local subspace generated by each trajectory j are estimated by computing the SVD of the matrix $\mathtt{V}^T_{n+1\times r}$, that is composed of only the trajectory j and its n NNs. The first r_h left singular vectors are the estimated basis of the subspace generated by j.

If the kind of motion is known, n can be tuned knowing that, for example, for rigid motion $r_h \leq 4$, for an articulated motion with one joint $r_h \leq 7$ and so on. If the motion is not known the highest dimension (i.e. $r_h = 7$, those $n \geq 6$) should be considered [31].

Affinity matrix

The affinity between two generic trajectories j and l depends on the distance between their local subspaces, which in LSA is computed through the principal angles (PAs). Let us recall that PAs between two subspaces S_j and S_l are defined recursively as a series of angles $0 \leq \theta_1 \leq, \ldots, \leq \theta_M \leq \pi/2$, where $M = \min\{rank(S_j), rank(S_l)\}$ [118]:

$$\cos(\theta_1) = \max_{u\in S_j, v\in S_l} u^T v = u_1^T v_1, \qquad (3.2)$$

while

$$\begin{aligned} \cos(\theta_k) &= \max_{u \in S_j, v \in S_l} u^T v = u_k^T v_k, && \forall k = 2, \ldots, M, \\ s.t.: &\quad \|u\| = \|v\| = 1,\ u^T u_q = 0,\ v^T v_q = 0 && \forall q = 1, \ldots, k-1. \end{aligned} \tag{3.3}$$

The vectors $u_1, \ldots, u_k$ and $v_1, \ldots, v_k$ are called principal vectors (u and v being two generic principal vectors). Intuitively, the first pair of principal vectors corresponds to the most similar modes of variation of the two subspaces. Every next pair corresponds to the most similar modes orthogonal to all of the previous ones.

Once PAs have been computed an affinity matrix can be built. The affinity measure used in LSA is the following:

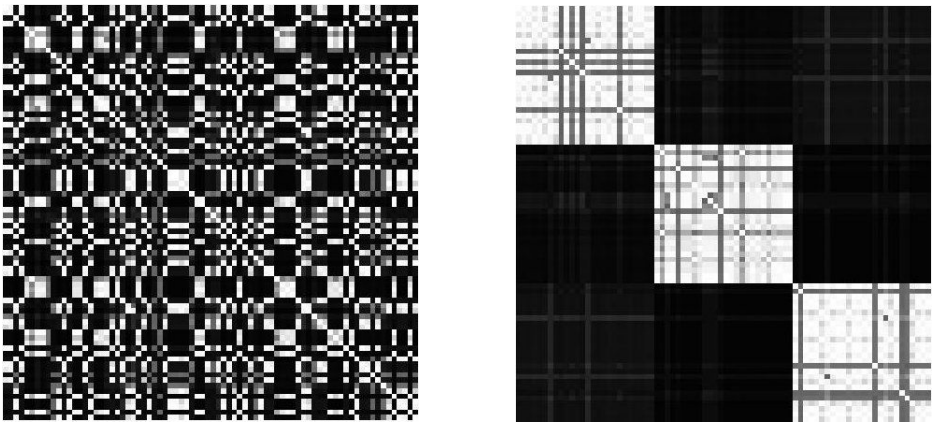
$$\mathtt{A}(r)_{(S_j, S_l)} = e^{-\sum_{i=1}^{M} \sin^2(\theta_i)} \tag{3.4}$$

where, r is the estimated rank of $\mathtt{W}$, S_j and S_l are two generic subspaces, M is the minimum dimension between the size of S_j and S_l and θ_i is the i^{th} principal angle between the subspaces S_j and S_l. From this definition it can be deduced that $\mathtt{A}$ is a symmetric matrix and its entries take positive values from almost 0 to 1, The closer to 1 is an entry the more similar the local subspaces identified by the entry indices are. Figure 3.3(a) shows the affinity matrix of trajectories that belong to three different motions; the diagonal is white as all the values are equal to 1 (being any subspace identical to itself) while black spots represent subspace pairs with low similarity.

Clustering

Now that the affinity matrix $\mathtt{A}$ has been computed the idea is to group together subspaces with high degree of similarity. Any clustering algorithm can be applied, Yan and Pollefeys suggested to use the recursive 2-way spectral clustering technique (Normalized Cuts, or simply N-cuts) proposed by Shi and Malik [5].

Figure 3.3(b) shows the same affinity matrix shown in Figure 3.3(a) rearranged after the spectral clustering algorithm. In the rearranged affinity matrix it is easy to see the



(a) Pre-clustering (b) Post-clustering

Figure 3.3: Example of affinity matrix (sequence with three motions).

block diagonal structure where each of the three clusters is represented by a bright square which signifies high affinity between the trajectories clustered together.

In Figure 3.4 the LSA algorithm is summarised. Starting from a trajectory matrix `W` LSA steps are:

1. perform rank estimation r of `W`
2. project the data onto a unit hypersphere;
3. estimate the local subspace bases by local sampling of n NNs of each trajectory;
4. compute the PAs and build an affinity matrix;
5. perform spectral clustering on the affinity matrix in order to obtain the final segmentation.

3.1.2 Problems

LSA provides an elegant solution for the manifold clustering problem, however, it is possible to identify some weaknesses of the algorithm.

The first weakness is that LSA is able to deal only with full trajectories, no missing data is allowed. This is a strong assumption because it means that no occlusions are allowed

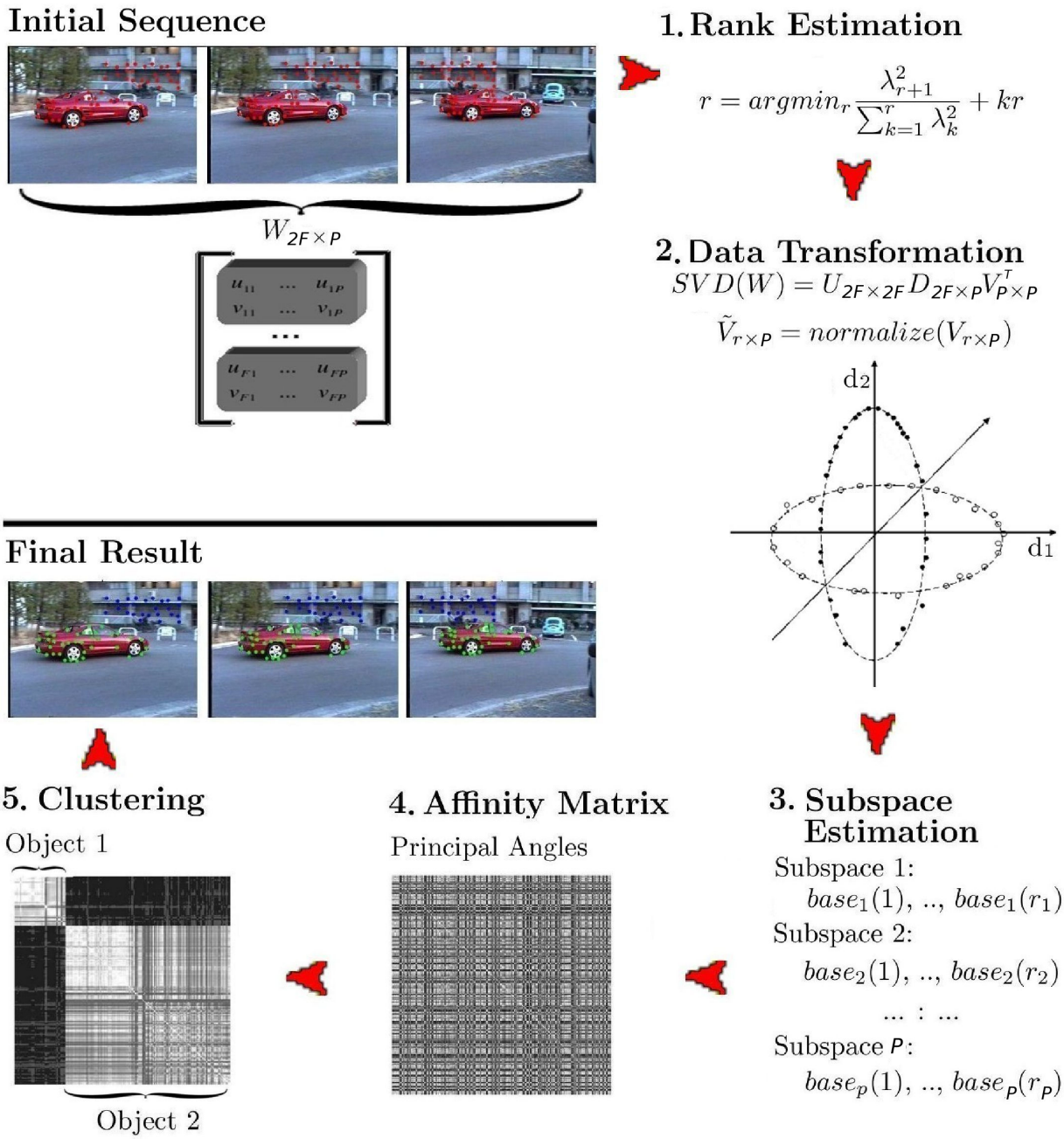


Figure 3.4: LSA flow, starting from a sequence of tracked features (red dots) via the five steps that end with the final result where the features that belong to the two motions are segmented (blue and green dots). Image adapted from [127].

and that the tracker must be very robust. However, some authors have already pointed out possible solutions [121, 128].

A second issue is related with the rank estimation of the trajectory matrix. In order to accomplish this task LSA uses a MS technique, Equation (3.1), that relies on a parameter k which has to be tuned depending on the noise level and on the number of motions (steps 1 and 3). Hence, MS requires previous knowledge about the amount of noise and the number of motions. Moreover, MS is very sensitive to the parameter k, to the extent that Tron and Vidal, in their implementation of LSA [22], claim that they had problems in finding a k value that was good for all of the sequences of the Hopkins155 database. Therefore, in [22] the authors avoided rank estimations and preferred to fix the global and the local space sizes. They chose to fix the global space dimension to $4N$ (step 1), where N is the number of motions. They also fixed the dimension of the local subspaces generated by each trajectory to 4 (step 3). The estimation of the local subspace size is not required to be particularly precise, moreover, the variance of the local subspace size is limited. On the other hand, the global space size requires a much more precise estimation, as shown in Section 3.2, and fixing it to a specific value greatly reduces the ability of LSA to deal with different types of motion. In fact, only fully independent, rigid, and non-degenerate motions ensure a rank equal to $4N$ [10]. Besides, in order to fix the global subspace size ($4N$) the amount of motions (N) is also required, but such information is not always available.

In step 4 of the LSA algorithm there is also another potential source of problems. The affinity formula suggested by the authors is a well known and used measure, however, when applied to the principal angles between subspaces it may not be the most suitable measure. Some of the problems of the sin^2-based measures have been already pointed out in [58, 124]. In Section 3.3.4 more details, and some, so far neglected issues about this measure are highlighted.

A final problem of LSA consists of the fact that Normalized Cuts (as well as other clustering techniques like K-means) does not provide a reliable instrument for estimating

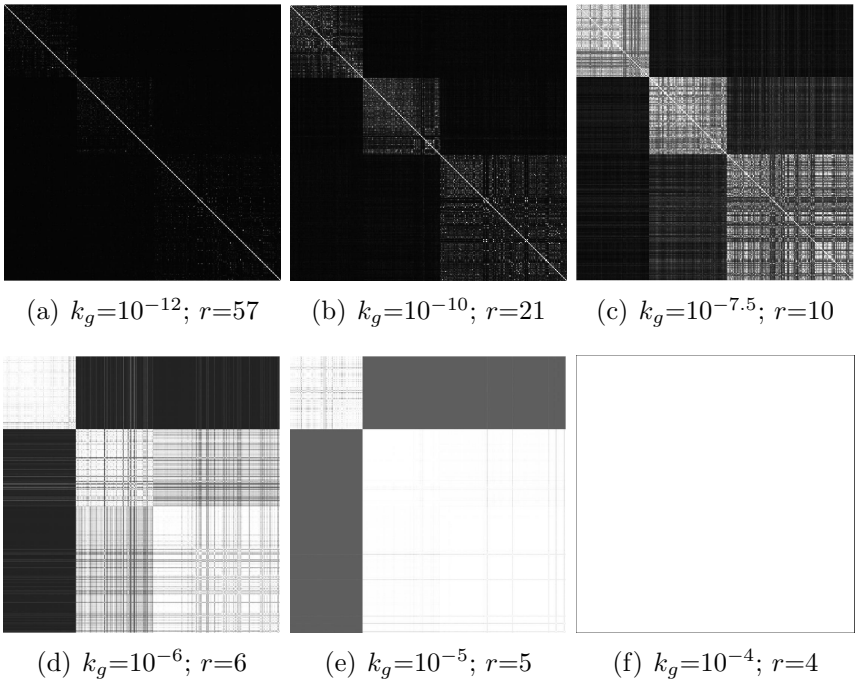
the number of clusters (in the case of motion segmentation, a cluster is equivalent to a motion). Shi and Malik in [5] suggest to use the Cheeger constant [129] or the cost of the last cut as an indication of whether or not it is worth to continue the cutting process. However, these two constants are highly sensitive to noise and they require the use of thresholds that may change depending on the input sequence. This is the reason why most of the available implementations of Normalized Cuts assume the final number of clusters as known data.

The elegance of the framework and the good performance provided by LSA, together with the possibility to work on some of the weaknesses just discussed make this algorithm very interesting and appealing for further studies. Of all of its problems one of the most recognised is surely the rank estimation step [22,108]. A better rank estimation will be the aim of two of the algorithms proposed in this thesis: the Enhanced Model Selection (EMS), Section 3.2 and the Principal Angles Clusterization (PAC), Section 3.3.3. A new affinity measure between subspaces, based on principal angles, is presented in Section 3.3.4. Finally, an automatic tool for the estimation of the number of motions is proposed in Section 3.4.

3.2 Enhanced Model Selection (EMS)

The EMS technique is an improved version of the MS technique. As explained in Section 3.1.2 MS is very sensitive to the parameter k. EMS automatically finds a good k value for Equation (3.1), so that the amount of noise and the number of motions are not required as prior knowledge. As Equation (3.1) is used in step 1 and step 3 of LSA, the estimated k for the global space (step 1) is called k_g, while the k for the local subspaces (step 3) is called k_s.

EMS is first applied to step 1 for the rank estimation of the matrix `W`. For the moment, the dimension of the local subspace generated by each trajectory is fixed to 4, the extension of EMS to step 3 is explained in Section 3.2.4. The key idea of EMS lies in the discovered relationship between the rank of `W` estimated by MS (Equation 3.1), and the computed affinity matrix `A` (step 4). A more rigorous explanation of such a relationship will be



(a) k_g=10^{-12}; r=57 (b) k_g=10^{-10}; r=21 (c) k_g=$10^{-7.5}$; r=10

(d) k_g=10^{-6}; r=6 (e) k_g=10^{-5}; r=5 (f) k_g=10^{-4}; r=4

Figure 3.5: Affinity matrices computed with different k_g values; real sequence *1R2RC* from the Hopkins155 database, theoretical maximum rank of $\mathtt{W}$ is 12; black is minimum affinity, white is maximum affinity and r is the estimated rank. Note that the block diagonal structure of the affinity matrices is due to the fact that the trajectories had been previously ordered according to the ground truth of the segmentation.

presented in Section 3.2.1. To offer a pictorial understanding an example is shown in Figure 3.5(a) to 3.5(f), where the affinity matrices, obtained after estimating the rank with different k_g values, are shown. Note that, for plotting convenience, the trajectories had been previously ordered so that trajectories of the same motion would appear next to each other. This explains the block structure of the affinity matrices, however, such a structure could not always be guaranteed before the final clustering step. The sequence used in this example has 3 rigid motions (from structure from motion theory it is known that the trajectory matrix composed of 3 rigid motions is at most 12 [10], 4 dimensions for each motion). When the rank of $\mathtt{W}_{2F\times P}$ is estimated using an inappropriate k_g value, the affinity matrix does not provide any useful information, as in Figure 3.5(a) and 3.5(b) (rank overestimated) and Figure 3.5(d) to 3.5(f) (rank underestimated). Qualitatively speaking, the "best affinity matrix" for a successful clustering step, which is the affinity matrix where the values between similar and different subspaces are well distinguished, is obtained with

$k_g = 10^{-7.5}$. When such a value for k_g is used the estimated rank is 10, Figure 3.5(c). The example of Figure 3.5 is also useful to understand how difficult the tuning process of the k_g parameter is. The variations of the k_g values are minimal but they still produce very different rank estimations. This is due to the extremely smooth singular value spectrum of the trajectory matrix, as explained in Section 3.1.1.

The relationship between the chosen k_g and the computed affinity matrix clarifies why a wrong choice of k_g can greatly affect the final segmentation. It is crucial for LSA to have a good rank estimation of the trajectory matrix in order to perform a correct segmentation. On the other hand, this relationship gives additional information: by looking at the affinity matrix it is possible to assess the accuracy of the rank estimation. In Section 3.2.1 a deep analysis of such a relationship is presented, and in Section 3.2.2 it is explained that, without knowing the number of motions and without assuming any order of the tracked features, the entropy can be used as a measure of the "reliability" of the affinity matrix, and hence of the estimated rank. Moreover, in Section 3.2.3 a speed up algorithm is proposed in order to quickly obtain an affinity matrix with high entropy, avoiding an exhaustive search among a big range of k_g values. Finally, the extension of EMS for the estimation of the size of the local subspaces is presented in Section 3.2.4.

3.2.1 Affinity matrix as a function of the estimated rank

In order to exploit the pattern shown in Figure 3.5 it is necessary to understand its nature and be sure that such a pattern exists independently from the specific input sequence analysed. Before getting into details about how the relationship between the affinity matrix and the k_g value (and hence the estimated rank r) could be used, let us analyse more deeply the behaviour of the affinity matrix in relation to the estimated rank r.

As there are many factors that influence the final affinity matrix, it is easier to start with a simplified problem and extend it later to the real case. Assume for the moment that there is no noise in the tracked feature positions, and that the ground truth set of NNs of each trajectory is known. Also, assume that the local subspaces generated by trajectories

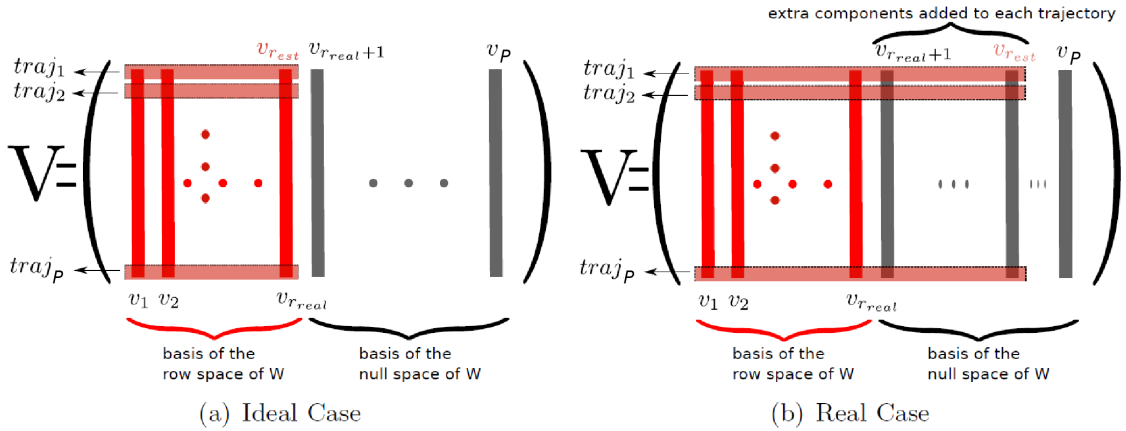
of different motions are orthogonal (ideal case).

The affinity between two generic trajectories j and l depends on the distance between their local subspaces, which in LSA is computed via PAs. In the ideal case and with a correct rank estimation, all the PAs θ_i (for $i = 1, \ldots, M$) between two local subspaces generated by trajectories of the same motion should be 0. On the other hand, when j and l belong to different motions, all the θ_i should be close to $\pi/2$. Let us now discuss the behavioural trend of PAs as a function of r (the rank of the global subspace) in the cases of under and overestimation.

The analysis in the case of underestimation is quite easy to develop. In fact, in this case the components of the lost dimensions are projected onto the remaining dimensions, artificially forcing two local subspaces towards each other, to the point when they collapse onto exactly the same local subspace. As a consequence, two local subspaces tend to become closer as r decreases.

On the other hand, the behaviour of PAs when r is an overestimation is less intuitive. In fact, in [130] it is explained that the problem of computing principal angles and principal vectors when the rank of the global space is overestimated is an ill-posed problem. To these authors' knowledge, the most helpful mathematical result for the case under analysis is presented in [131]. The authors study the probability density function of the largest principal angle between two subspaces chosen from a uniform distribution on the Grassmann manifold of t-planes embedded in $\mathbb{R}^n$. They show that, when n is appreciably bigger than t (precisely $n > 2t - 1$), the probability density function is close to zero for small angles and rapidly increases to reach a global maximum in $\pi/2$.

The resemblance between this abstract mathematical situation and the practical case under analysis is represented by the fact that, when the rank is overestimated, the extra components added to the trajectory vectors (in the global space) are sampled from the basis vectors of the null space of $\mathtt{W}$. In fact, the projection onto the global space is done as follows. The matrix $\mathtt{W}$ is decomposed by SVD as: $\mathtt{W} = \mathtt{U}\mathtt{D}\mathtt{V}^T$. Hence, if the rank of $\mathtt{W}$ is r_{real}, the first r_{real} columns of $\mathtt{V}$ correspond to the basis of the row space of $\mathtt{W}$, whereas the



(a) Ideal Case (b) Real Case

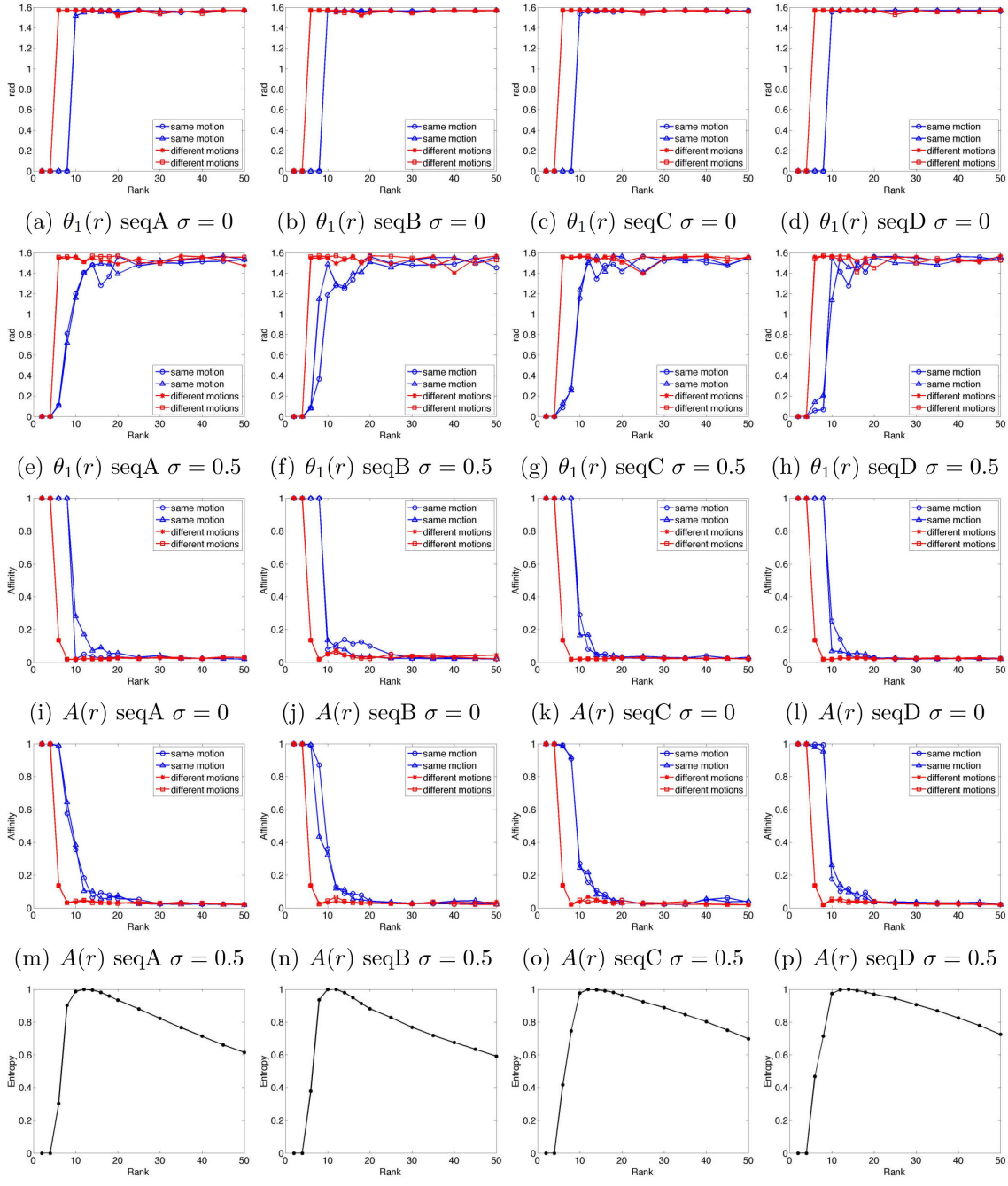
Figure 3.6: Pictorial description of what happens when the rank of the global space is estimated. $traj_i$ stands for the i^{th} trajectory.

remaining $P - r_{real}$ columns of $\mathtt{V}$ correspond to the basis of the null space of $\mathtt{W}$. In this new global subspace the first r_{real} components of each row i of $\mathtt{V}$ (for $i = 1, \ldots P$) represent the trajectory i. In Figure 3.6(a) the meaning of each column and row of $\mathtt{V}$ is summarised in the case that the estimated rank of $\mathtt{W}$ is $r_{est} = r_{real}$. However, if the estimated rank of $\mathtt{W}$ is $r_{est} > r_{real}$, each trajectory i will be represented by its true r_{real} components (taken from the first r_{real} components of row i of matrix $\mathtt{V}$) plus $r_{est} - r_{real}$ extra components that are taken from row i of the $r_{est} - r_{real}$ basis of the null space of $\mathtt{W}$. These extra components are unrelated to the trajectory i, hence they are random with respect to that trajectory. This second case is summarised in Figure 3.6(b). Note that the projection of the trajectories onto the global space is *oblique*, for this reason the components that exceed the real rank are not eliminated and they act as random values.

Finally, note that all of the reasoning holds true even in the case of no noise, and it does not involve the selection of the NNs. Hence, when r_{est} is sufficiently big the work of [131] applies to the case under analysis. Therefore, the higher the overestimation, the closer the resemblance to a uniform distribution.

From the result presented in [131], it can be inferred that PAs, as a function of the estimated rank, have an *increasing trend* [1]. To support this inference, in Figure 3.7(a)

[1]To these authors' knowledge, no information is known about the precise analytical behaviour of such functions.



(a) $\theta_1(r)$ seqA $\sigma = 0$ (b) $\theta_1(r)$ seqB $\sigma = 0$ (c) $\theta_1(r)$ seqC $\sigma = 0$ (d) $\theta_1(r)$ seqD $\sigma = 0$

(e) $\theta_1(r)$ seqA $\sigma = 0.5$ (f) $\theta_1(r)$ seqB $\sigma = 0.5$ (g) $\theta_1(r)$ seqC $\sigma = 0.5$ (h) $\theta_1(r)$ seqD $\sigma = 0.5$

(i) $A(r)$ seqA $\sigma = 0$ (j) $A(r)$ seqB $\sigma = 0$ (k) $A(r)$ seqC $\sigma = 0$ (l) $A(r)$ seqD $\sigma = 0$

(m) $A(r)$ seqA $\sigma = 0.5$ (n) $A(r)$ seqB $\sigma = 0.5$ (o) $A(r)$ seqC $\sigma = 0.5$ (p) $A(r)$ seqD $\sigma = 0.5$

(q) $E(\mathtt{A}(r))$ seq A (r) $E(\mathtt{A}(r))$ seq B (s) $E(\mathtt{A}(r))$ seq C (t) $E(\mathtt{A}(r))$ seq D

Figure 3.7: Trend of the largest principal angle (Figure 3.7(a) to 3.7(h)), and of the affinity (Figure 3.7(i) to 3.7(p)), between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The trajectories are randomly taken from synthetic sequences (first and third rows are sequences with no noise, second and fourth rows are same sequences with Gaussian noise $\sigma = 0.5$) with two rigid motions, hence the maximum rank is 8 (NNs are estimated). Last row shows the entropy trend of $\mathtt{A}(r)$ related to the sequences with Gaussian noise $\sigma = 0.5$.

to 3.7(d) the trend of the largest PA of synthetic sequences with 2 rigid motions (hence the maximum rank is 8) and no noise is presented. As the test is performed using the first part of the LSA algorithm the NNs are estimated as explained in step 3 of the LSA summary (Section 3.1). For each sequence four angles are compared: two angles between trajectories of the same motion (blue lines) and two angles between trajectories of different motions (red lines). These results are just a few samples of a large number of experiments performed on the whole synthetic and the Hopkins155 databases. In Appendix A more trends could be found with synthetic sequences and a noise level up to $\sigma = 3.0$ pixels, and some of the Hopkins155 sequences. All the experiments show the same pattern, i.e. when the rank is very small all PAs tend towards zero, while when the rank is heavily overestimated all of the angles tend towards $\pi/2$. For simplicity, only the largest PA is shown in the examples. However, smaller angles also follow the same trend. These experiments confirm the increasing trend inferred from [131]. Moreover, from Figure 3.7(a) to 3.7(d) it is possible to appreciate that when the estimation of the rank is close to the correct rank, then PAs between local subspaces generated by trajectories of different motions are higher than those between local subspaces generated by trajectories of the same motion.

From now on, the behaviour of the PA θ_i with respect to rank r will be referred to as the function $\theta_i(r)$ or simply θ_i^r. Let us now analyse how this behaviour is reflected in the affinity value between two generic trajectories j and l, where the affinity is defined as in Equation (3.4) on page 70. In the ideal case, and with a correct rank estimation, the affinity between trajectories of the same motion is maximum (i.e. 1), whereas the affinity between trajectories of different motions is minimum (i.e. close to 0). Similarly to what was done for PAs, it is interesting to understand the global behaviour of the function $A(r)$. In order to do so, the first derivative of $A(r)$ is studied:

$$\frac{dA(r)}{dr} = -e^{-\sum_{i=1}^{M} \sin^2(\theta_i^r))} \sum_{i=1}^{M} 2\sin(\theta_i^r)\cos(\theta_i^r)\theta_i'^r. \qquad (3.5)$$

All of the functions appearing in the derivative (3.5) are non negative for all values of r,

except for $\theta_i'^r$ (the first derivative of θ_i^r). However, it has been shown that θ_i^r displays an overall increase, so $\theta_i'^r \geq 0$ for the majority of the values of r. The presence of the minus sign implies that $\frac{dA(r)}{dr} \leq 0$ for the majority of the values of r, i.e. $A(r)$ has a *decreasing* trend. Specifically, when r is an underestimation all of the affinities tend towards the maximum value, whereas when r is an overestimation they tend towards the minimum value. Figures 3.7(i) to 3.7(l) show the affinity values of the same pairs of trajectories used in Figures 3.7(a) to 3.7(d), confirming that the affinity function has a decreasing trend as concluded in the analysis just performed.

So far, the case without noise and with perfectly orthogonal local subspaces was considered . The effect of the presence of noise and the estimation of the NNs is that there may be oscillations in the functions of the PAs (as in Figure 3.7(e) to 3.7(h)), and so, potentially, also in the affinity functions. However, it will be shown later in this section why this, in turn, does not lead to big oscillations of the affinity values (as confirmed in Figure 3.7(m) to 3.7(p)).

So far, in this theoretical discussion the local subspaces were assumed to be orthogonal. In real sequences, however, local subspaces are not always perfectly orthogonal. The effect of non perfect orthogonality is that, even in the case of overestimation, some pairs of trajectories may have low, but not exactly zero, affinity. The effects of non perfect orthogonality and of the estimation of the NNs can be seen in Figure 3.7(a) to 3.7(d): despite the fact that there is no noise, it is possible to notice small oscillations in the θ_i^r.

The main consequence of moving from an ideal to a real situation is, therefore, that the θ_i^r may have wider oscillations. However, such oscillations rarely lead to oscillations of the affinity values. In fact, the oscillations in one of the M PAs between two subspaces may be compensated by the remaining $M-1$ PAs (especially if the oscillation affects one of the smallest angles). To this end, note that with the affinity measure used, Equation (3.4), all the $\sin(\theta_i)^2$ are summed together. Moreover, the highly non linear behaviour of the decreasing exponential used to define the affinity tends to smooth the effect of small changes. Even in the worst case scenario, it is very unlikely that all of the affinity values oscillate

in the same manner. Hence, it is highly probable that the trend of $\mathtt{A}(r)$ remains overall decreasing. Let us stress that Figures 3.7(m) to 3.7(p) testify that, even when the presence of noise induces considerable oscillations in the θ_i^r, the affinities are not dissimilar to those of the case without noise (Figure 3.7(i) to 3.7(l)).

In summary, even without the assumptions made at the beginning (i.e. no noise, perfect NNs estimation and perfect orthogonality between the subspaces), the affinity between every pair of trajectories is maximum when the rank of the global subspace is highly underestimated and is minimum when the rank is highly overestimated. In between, the affinities have a decreasing trend. Due to particularly big oscillations in the PAs, specific pairs may present oscillations in their affinity, but the majority of the affinities remain, overall, decreasing functions of r. This analysis justifies the pattern initially shown in Figure 3.5 and it is now possible to expect a similar behaviour for any input sequence.

3.2.2 How to choose a good affinity matrix

Now that the relationship between the affinity function and the estimated rank has been clarified, it is possible to exploit it. Intuitively: many affinity matrices could be built using different rank estimations and "the most reliable" affinity matrix could be selected for the clustering step. However, in order to do so, it is necessary to find a measure of the "reliability" of the affinity matrix. Let us recall here that the affinity matrix $\mathtt{A}(r)_{P\times P}$ is a symmetric matrix that contains the affinity values between every pair of subspaces (generated by each of the P trajectories) computed assuming a trajectory matrix of rank r (i.e. a global subspace of size r). It is reminded to the reader that the block structure of the affinity matrices shown in Figure 3.5 is due to the fact that, for plotting convenience, the trajectories had been previously ordered so that trajectories of the same motion would appear next to each other. However, such a structure is usually the result of the clustering step and, therefore, cannot be used during the decision process of which matrix should be selected.

The ideal criterion for the selection of the affinity would be to choose the rank r that

leads to an affinity matrix that results in the minimum misclassification rate. However, the ground truth of the segmentation is not always known. In real cases a convenient criterion could be to choose a high contrasted affinity matrix, because the higher the contrast the higher the quantity of information that can be used to compare the trajectories. A well known measure of contrast, and of quantity of information, is the entropy [132]:

$$E(\mathtt{A}(r)) = -\sum_{i=0}^{1} h_{\mathtt{A}(r)}(i)\ \log_2(h_{\mathtt{A}(r)}(i)), \tag{3.6}$$

where $h_{\mathtt{A}(r)}(i)$ is the histogram count in the bin i (in all of the experiments 256 bins were used). As stated in the previous section, when r is an underestimation, all of the affinities tend to be clustered around 1, leading to a uniform $\mathtt{A}$ and, therefore, to a very low entropy value. As r approaches the correct rank, the affinities between trajectories of the same motion tend to diverge from those between trajectories of different motions, hence, $\mathtt{A}$ starts to contain heterogeneous values and as a consequence its entropy increases. Note that, theoretically, it is not possible to exclude that the entropy, as a function of r, can have oscillations, but regardless at this stage it is more interesting to focus on its overall behaviour. The more r is increased, becoming an overestimation, the more the affinities tend to converge around 0, consequently $\mathtt{A}$ becomes more uniform and its entropy decreases again. Figure 3.7(q) to 3.7(t) show the trend of the entropy for the same synthetic sequences used to show θ_i^r and $\mathtt{A}(r)$ in the presence of noise (Figure 3.7(e) to 3.7(p)). Similarly, Figure 3.8 shows the entropy trend of the real sequence *1R2RCR* (Hopkins155 database) used to compute the affinity matrices shown in Figure 3.5.

Note that the entropy would not be maximum in the case of a perfect affinity matrix with only two values (maximum and minimum). In Figure 3.9 there is an example where it is possible to see that a perfect affinity matrix (Figure 3.9(b)) would have a smaller entropy than the other two suboptimal cases (Figure 3.9(a) and 3.9(c)). However, such a situation is extremely rare in real sequences. Also in synthetic sequences with no noise a perfect affinity matrix is obtained only when the motions are completely independent.

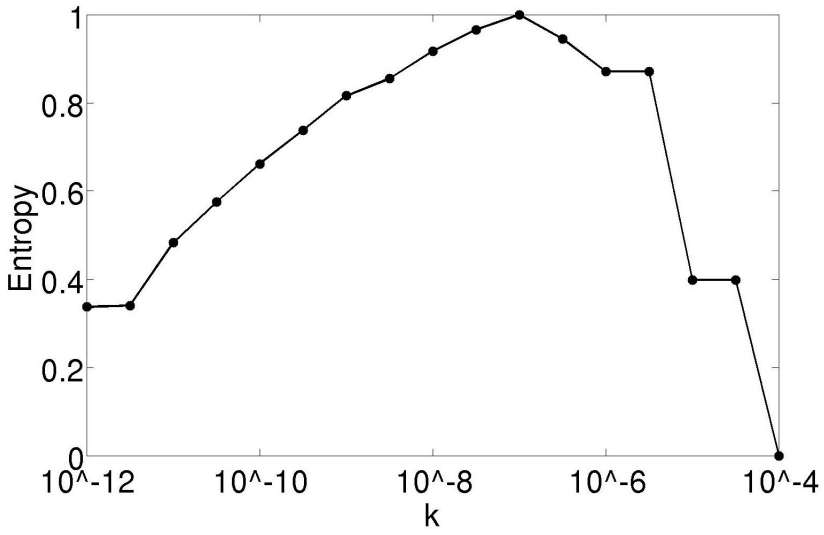
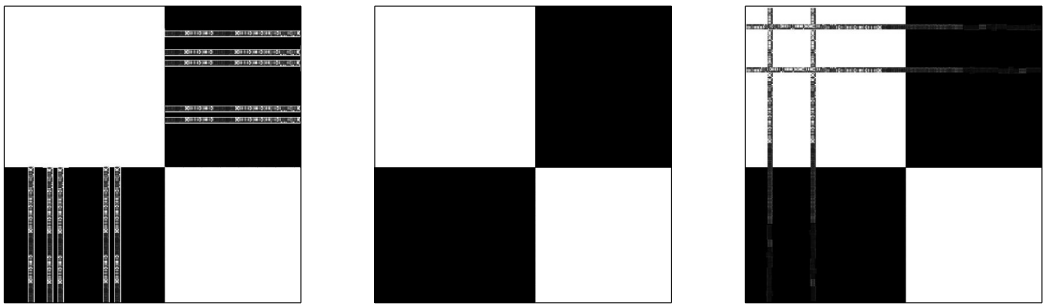


Figure 3.8: Example of the entropy trend experienced with all the sequences used in the experiments. This specific example refers to the same sequence used to show the affinity matrices in Figure 3.5.

In practical cases, without having a clear indication about which affinity matrix is the most suitable, the choice of the affinity matrix with the highest entropy has the interesting property of discarding uniform (and thus useless) matrices. In the examples shown in Figure 3.7(q) to 3.7(t) the highest entropy always corresponds to a rank value where the affinities between trajectories of the same motion are higher than those between trajectories of different motions (Figure 3.7(m) to 3.7(p)). In the example of Figure 3.8 the maximum corresponds to the affinity matrix of Figure 3.5(c), which is the one where the 3 blocks corresponding to the 3 motions are more clearly distinguished. Moreover, the entropy does not depend on the position of the affinity values inside the matrix. This independence is fundamental, as there is no control of the evaluation order of the subspaces. This new way of estimating the rank of the trajectory matrix by using the maximum entropy criteria, is called the Enhanced Model Selection.

As stated before, it is not possible to ensure that the affinity matrix with the maximum entropy corresponds to a correct estimation of the rank. Although it is expected to provide a good estimation of the rank, and it contains enough information for a successful clustering step, the rank could be slightly over- or underestimated. Unfortunately, it is also not



(a) Rank overestimated, Entropy: 1.9034

(b) Correct rank estimation, Entropy: 1.0360

(c) Rank underestimated, Entropy: 2.1470

Figure 3.9: Three examples of affinity matrices. The affinity matrix built using the correct rank estimation has a smaller entropy value than the other two suboptimal cases. This example shows that the entropy is not an optimal measure, however, such a situation is extremely rare in real sequences.

possible to assume that the maximum entropy has always a fixed bias. In fact the distance between the correct rank and the rank estimated with the maximum entropy observation depends on the trajectories of each specific video sequence. Clearly the correct estimation is the most desirable outcome, however, even a small overestimation is acceptable. The least desirable outcome would be an underestimation as this corresponds to cutting important information. In Figure 3.7(i) to 3.7(p) it can be appreciated how all the affinities go to 1 very quickly when the rank is underestimated.

In order to prevent a possible underestimation, or reduce its effects, it may be safer to increase the estimated rank by a small amount. Our experiments have shown that there is a correlation between the number of motions (or the amount of noise) and the position of the maximum entropy: the greater the number of motions (or the higher the noise) the lower the estimation of the rank. In Figure 3.10 the error of the EMS rank estimation is shown in relation to the number of motions and the noise level. These results are computed on the synthetic database described in Section 2.5. Each point in the plot corresponds to the average error over 10 synthetic sequences for each number of motions and each noise level. As can be seen, when the number of motions increases EMS becomes biased and tends to have a negative error (i.e. tends to underestimate the rank). Similarly, the higher the noise the lower the rank estimation. As a preliminary study, a possible way to correct the

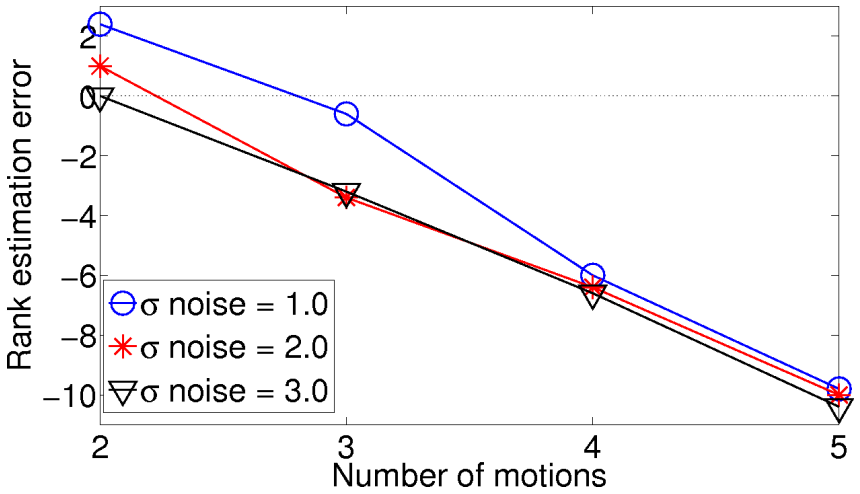


Figure 3.10: Error of EMS rank estimation with different number of motions and noise levels. Each value is obtained as an average of the error over 10 different sequences.

estimated rank could be to add 1 for each motion to the previous EMS estimated rank. By adding only 1 unit for each motion, even if the maximum entropy already corresponds to an overestimation or a correct estimate, the correction does not introduce too much random information, whereas if the maximum is an underestimation some important information is added.

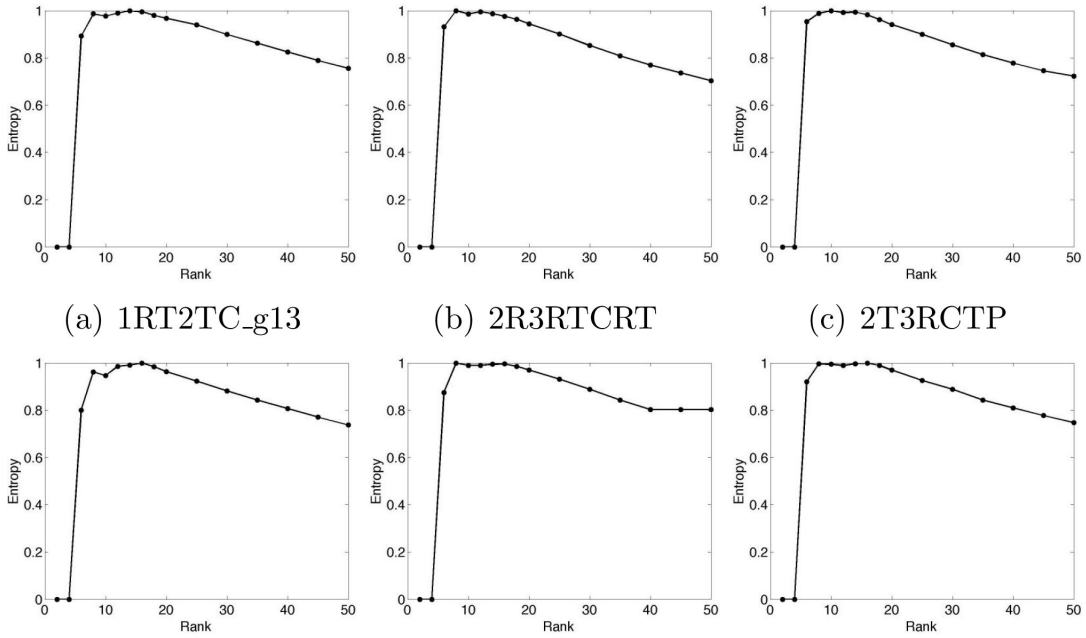
EMS with correction is called EMS+, in Section 3.2.5 the performances between EMS and EMS+ are compared.

3.2.3 How to speed up the choice

In the previous section it was shown that a good choice for the affinity matrix corresponds to the matrix with the highest entropy value. However, this may require to build a large number of affinity matrices and, therefore, the algorithm would be computationally expensive. A speed up technique that can be used in order to quickly find an affinity matrix with a high entropy is here proposed. In order to have a fast but good estimation of the rank, the concavity of the overall entropy behaviour is exploited. As explained in the previous section, the entropy may have oscillations. However, by taking opportune safety measures it is possible to quickly find an affinity matrix with an entropy very close to the highest

value. One can always renounce to speed in favour of a better estimation, however in Section 3.2.5 it is shown that even with this approximated (but faster) choice good results are obtained.

Assuming that there may be small oscillations in the entropy function, in order to avoid to select a local maximum it is sufficient to use a large sampling step (in all of the experiments the step is $\Delta k_g = 10$). Moreover, to establish the gradient of the entropy 3 samples are considered in preference to only 2. When a minimum is encountered the sampling can be extended towards the two extremes until a choice can be made. Once the gradient is established, the search is shifted towards the increasing gradient repeating the sampling process until the maximum is found. It should be remarked that in all of the tests performed, when the entropy was sampled in the way just explained, no fluctuations in the entropy trend were encountered. However, when the entropy is computed with a finer sampling step, in some sequences a very small oscillation may be found close to the maximum values. On the whole Hopkins155 database an oscillation can be found only in the following sequences: *1RT2TC_g13*, *2R3RTCRT*, *2T3RCTP*, *articulated*, *cars1* and *cars8*. As shown in Figure 3.11 the oscillations are very small and occur very close to the maximum entropy anyway. Hence, even if in these few cases EMS had selected a local maximum, the built affinity matrices would have had a very high entropy. One can always argue that there may be a particularly unlucky situation where the combination of bad estimation of NNs and noise generates a very big oscillation, to the point that even with the kind of sampling just described, EMS would choose a local maximum. However, the oscillations are more likely to be around the correct rank, hence there is a chance that the selected affinity matrix can provide enough information anyway. Moreover, in such extreme conditions the assumptions of the LSA framework would probably be violated leading to bad segmentation even if the rank is correctly estimated.



(a) 1RT2TC_g13 (b) 2R3RTCRT (c) 2T3RCTP

(d) articulated (e) cars1 (f) cars8

Figure 3.11: The only entropy trends with oscillations (always only one oscillation in correspondence to the maximum of the function) found on the whole Hopkins155 database. In these plots the rank was sampled from 2 to 20 with a step of 2, and from 20 to 50 with a step of 5.

3.2.4 Size estimation of the local subspaces

So far it was assumed that the size of the local subspaces (step 3) was fixed to 4. Once, the value k_g has been found equal to 10^x, k_s can be set to $10^{x/2}$. This corresponds to the choice made by the authors of LSA. In fact, Yan and Pollefeys explain in [72] that when estimating the rank of the local subspaces, due to a small number of samples, the noise level may play a more relevant role, so it is desirable that $k_s > k_g$. Therefore, after all of the trajectories have been projected onto the normalised hypersphere of size r_g each local subspace size is estimated using the MS formula and the k_s parameter equal to $10^{x/2}$. This should provide a more precise estimation of the basis of the local subspaces. When all the local subspace bases have been estimated the PAs between each pair of subspaces are computed and the affinity matrix can be built. The last remaining step is just an application of any spectral clustering algorithm that will provide the final segmentation.

Algorithm 3.1 EMS algorithm

1: Build a trajectory matrix $\mathtt{W}$;
2: $k_{g1} = 10^{-7.0}$, $k_{g2} = 10^{-7.5}$ and $k_{g3} = 10^{-8.0}$ {this initial range values were suggested in [31]};
3: **repeat**
4: **for** $i = 1 \dots 3$ **do**
5: compute global r_{gi} rank by MS using k_{gi};
6: project every trajectory, which can be seen as a vector in $\mathbb{R}^{2F}$, onto an $\mathbb{R}^{r}_{gi}$ unit sphere by SVD and truncation to the first r_{gi} components of the right singular vectors;
7: exploit the fact that in the new space (global subspaces) most of the points and their closest neighbours lie in the same subspace, to compute by SVD the local subspaces generated by each trajectory;
8: compute PAs between all of the subspaces assuming local subspace size 4;
9: compute affinity matrix $\mathtt{A}_i$;
10: compute entropy E of $\mathtt{A}_i$;
11: **end for**
12: shift $k_g i$ values towards the positive gradient as in Section 3.2.3
13: **until** maximum entropy found
14: compute local subspace sizes using MS with $k_s(k_g)$;
15: re-estimate the subspaces and compute final affinity matrix $\mathtt{A}_{opt}$;
16: cluster $\mathtt{A}_{opt}$ using any spectral clustering technique;

In summary, EMS exploits the relationship between the rank estimation of the trajectory matrix and the affinity matrix to automatically tune the value of k for the global and local dimension estimation without requiring knowledge about the number of motions nor the amount of noise. Better rank estimations result in a better motion segmentation and, as EMS does not make any assumption regarding the type of motions, it can be used under any condition. EMS+ is a small extension of EMS that should compensate the biased estimation of EMS at the cost of knowing the number of motions of the scene. In Section 3.4 a technique for the estimation of the number of motions will be discussed so that also EMS+ will not required any prior knowledge. A pseudo-code of the EMS algorithm is presented in Algorithm 3.1.

3.2.5 Experiments

In Section 3.2 it was explained how EMS and EMS+ are able to automatically adjust the parameter k, Equation (3.1), in accordance to different noise conditions and different

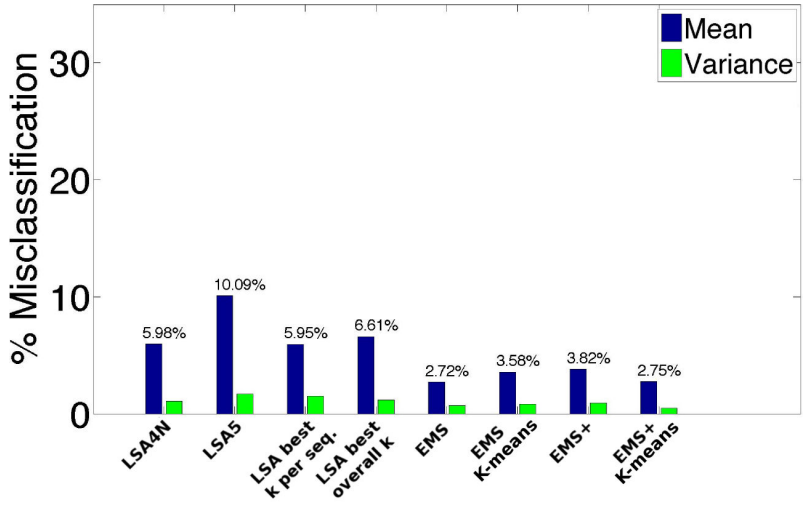
number of motions in the sequence. In this section the results of the misclassification of EMS and EMS+ on the Hopkins155 database are shown in comparison with other LSA-based techniques.

In these first sets of experiments the knowledge about the number of motions is assumed so that it is possible to assess the model selection techniques (MS, EMS and EMS+) independently from the accuracy of the estimation of the number of motions.

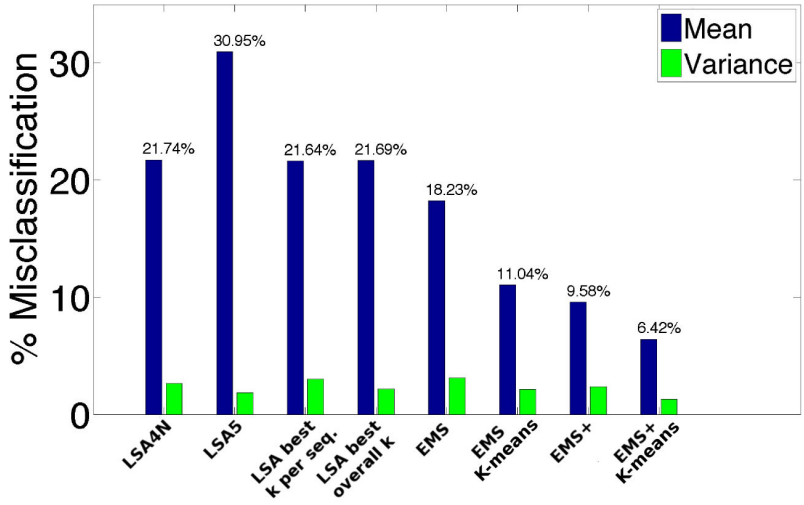
To evaluate the model selection the results of LSA with EMS and LSA with EMS+ are compared with the results obtained with: LSA fixing the global subspace size to 5 and $4N$ (where N is the number of motions), LSA estimating the global subspace size with MS using the best k per each sequence (i.e. the k that provided the lowest misclassification rate per each sequence of the Hopkins155 database), and LSA with MS using the overall best k (i.e. the k value, common for the whole database, that provided the lowest mean misclassification rate on the Hopkins155 database: $k = 10^{-7.5}$). Notice that for LSA with MS, the best overall k and the best k per sequence were previously computed by estimating both the global and the local subspace size (k for the local subspace sizes were determined as explained in Section 3.2.4 for EMS) and not by fixing the local subspace sizes to 4. This may seem a little detail but later it will be shown how this detail will play a relevant role in terms of misclassification. For all of the algorithms the Recursive Two-Way N-cut [5] is used for the final clustering step, as originally proposed by the authors of LSA. For EMS and EMS+ the misclassification rate obtained by clustering the affinity matrix by K-means is also presented (results obtained with 200 random initialisations). Two sets of experiments were performed: in the first set the global subspace size was estimated and the local subspace sizes were fixed to 4, while in the second set both global and local subspace sizes were estimated.

Local subspace sizes fixed and number of motions known

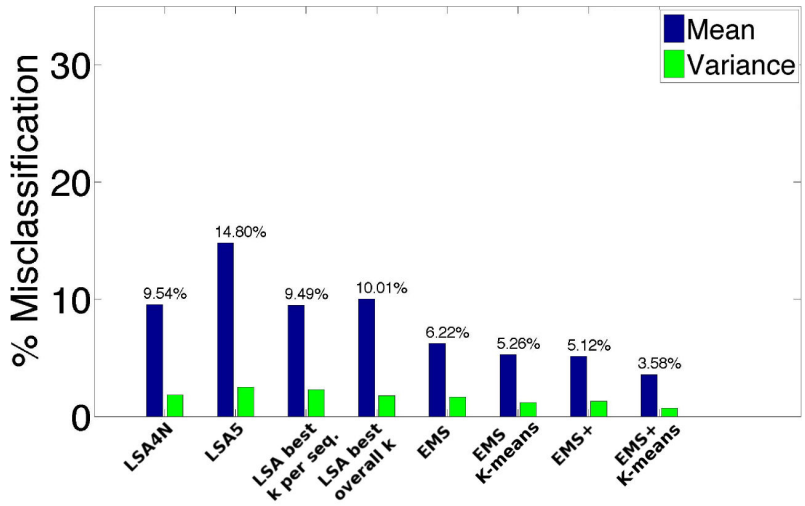
In Figure 3.12 the results on the Hopkins155 database where the local subspace sizes were fixed to 4 are shown in terms of mean and variance of the misclassification rate. The highest



(a) 2 Motions



(b) 3 Motions



(c) 2 and 3 Motions

Figure 3.12: Mean and variance of the misclassification rate on the Hopkins155 database (number of motions known and local subspace size fixed to 4).

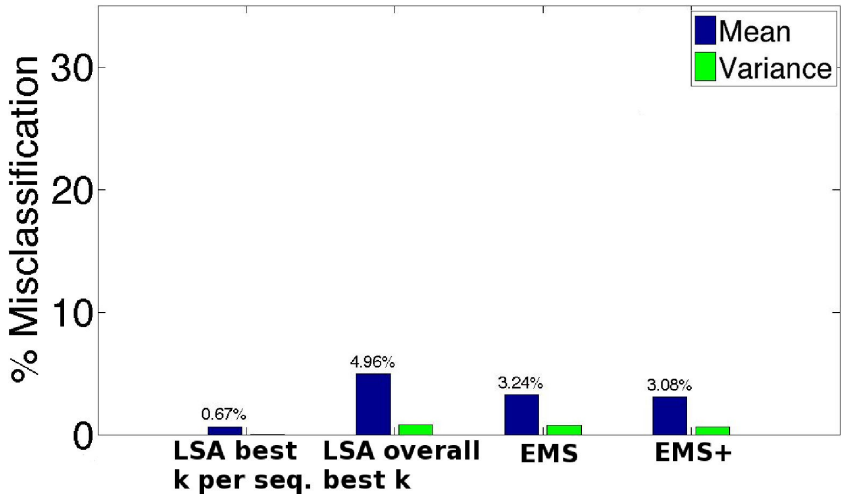
error rate is obtained when the global subspace size is fixed to 5 (10.09% with 2 motions and 30.95% with 3 motions), this happens because a dimension of 5 corresponds, for most of the global subspaces, to a considerable underestimation of their size. The performance of LSA with best overall k, best k per sequence, and where the global subspace sizes were fixed to $4N$ are very similar to each other (around 6% with 2 motions and 21% with 3 motions). Both EMS and EMS+, independently from the clustering algorithm, perform much better than any other technique, with EMS+ proving to be more solid than EMS when the number of motions increases. EMS and EMS+ perform even better than MS with best k per sequence. This may seem counter-intuitive as one would expect that the best k per sequence leads to the best misclassification rate. However, it has to be remembered that the best k values were computed when also the local subspace sizes were estimated and not fixed to 4. This small difference clearly changes which are the best k values that have to be used, and shows once again how unstable the MS is (the value of best k changes even if only the local subspace size is changed). Moreover, EMS and EMS+ perform better than LSA $4N$ because when the global subspace size is fixed to $4N$ the motions are considered rigid and fully independent even when this assumption is not verified. Finally, the difference between the performances of EMS and EMS+ using N-cuts or K-means is little. For 2 motions EMS with N-cuts has a misclassification rate of 2.72% while EMS with K-means has a misclassification rate of 3.58%, while EMS+ with N-cuts has a misclassification rate of 3.82% and EMS+ with K-means has a misclassification rate of 2.75%. In the case of sequences with 3 motions EMS with N-cuts has an error rate of 18.23% and EMS with K-means has a misclassification rate of 11.04% (biggest difference between N-cuts and K-means). Finally, EMS+ with N-cuts scored a 9.58% of misclassification rate while EMS+ with K-means scored a 6.42%. In general, K-means seems to be slightly better than N-cuts. Nevertheless, considering that EMS/EMS+ are being compared against the original MS used in LSA (whose suggested clustering technique is N-cuts), in the remainder of this section the results of EMS and EMS+ will be presented using N-cuts, as suggested by the authors of LSA.

Overall, these experiments show that a good model selection leads to better results than when the global subspace size is fixed. Moreover, MS proved to be very sensitive to the chosen dimension of the local subspaces, in addition to noise and changes in the number of motions. MS requires manual tuning in order to cope successfully with a variety of cases and in unknown scenarios. To conclude the comments on this set of experiments, it should be noted how much the performances degrade when the number of motions is increased from 2 to 3 showing one common weak point of all of these motion segmentation algorithms.

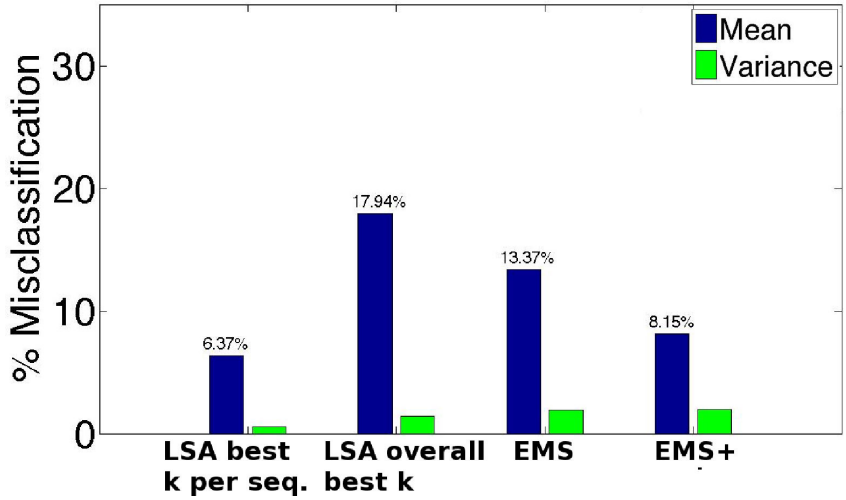
Local subspace size estimated and number of motions known

The following set of experiments is obtained when the local subspace sizes were also estimated instead of fixed to 4. As LSA 5 and LSA $4N$ do not use any instrument to estimate the local subspace size they are not included in this set of experiments. The results are shown in Figure 3.13.

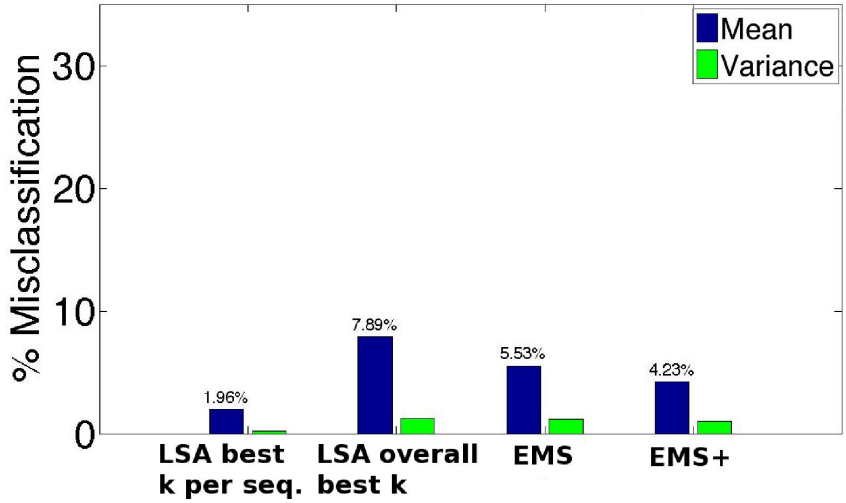
As expected, the lowest misclassification rate is obtained when k is manually tuned for each sequence, the error rate in this case is of 0.67% with sequences that contain 2 motions and 6.37% with sequences that contain 3 motions. This result sets the lower bound of the misclassification when using LSA-based techniques. Naturally, both EMS and EMS+ perform worse than MS with the best k per sequence, their misclassification is respectively: 3.24% and 3.08% with 2 motions, and 13.37% and 8.15% with 3 motions. However, EMS and EMS+ do better than MS with the best overall k: 4.96% with 2 motions and 17.94% with 3 motions. As in the previous set of experiments, the misclassification rate of EMS+ is lower than that of EMS. With 2 motions the performances are very similar. However, with 3 motions the compensation strategy plays an important role. In fact, the performances of EMS+ are quite close to the one of LSA with best k per sequence. If the plots of Figure 3.12 and Figure 3.13 are compared, it is possible to notice that, given a technique, the misclassification when the local subspace sizes are estimated are almost always better than when the local subspace sizes are fixed to 4. Such a result shows that



(a) 2 Motions



(b) 3 Motions



(c) 2 and 3 Motions

Figure 3.13: Mean and variance of the misclassification rate on the Hopkins155 database (number of motions known and local subspace size estimated).

Computational time (in seconds)	ALC	EMS+
Total with 2 motions	55625	2807
Avg. with 2 motions	464	23
Total with 3 motions	33205	1902
Avg. with 3 motions	949	56
Total overall	88831	4709
Avg. overall	573	31

Table 3.1: Computational time comparison between ALC and EMS+ on the Hopkins155 database. The two algorithms were implemented in Matlab and ran on an Intel Core2 Duo CPU @ 2.66GHz with 16 GB RAM.

a correct estimation of the size of the local subspaces also plays a role, albeit a minor one, in providing a better segmentation. For the sake of completeness, the misclassification of EMS with K-means (instead of N-cuts) is of 3.15% with 2 motions and 10.68% with 3 motions, whereas the misclassification of EMS+ with K-means is of 3.11% with 2 motions and 6.14% with 3 motions. Once again this shows that K-means has a slightly better performance than N-cuts.

It was shown that EMS/EMS+ outperform all practical LSA-based techniques (trying all possible values of k and taking the one with the lowest misclassification rate is only useful for theoretical reasoning but it is not a practical solution). At the time when EMS and EMS+ were developed the best performing algorithm on the Hopkins155 database was the Agglomerative Lossy Compression (ALC) [66]. Its average misclassification rate with 2 motions is 2.40% and with 3 motions is 6.69%. The difference between ALC and EMS+ performances is only of 0.68% with 2 motions and 1.46% with 3 motions in favour of ALC. However, the price that ALC has to pay in order to gain this little advantage on the misclassification side is a much heavier computational time. Table 3.1 shows the computational time required by the two algorithms. When the two algorithms, implemented in Matlab, were run on the same computer, EMS+ processed the whole Hopkins155 database more than 18 times faster than ALC.

In summary, these results show that EMS and EMS+ are able to provide a good estimation of the rank of the trajectory matrix in an automatic fashion. They do not

require any prior knowledge and are able to deal successfully with different noise levels. As it is not necessary for EMS and EMS+ to know in advance any subspace dimension, they are able to deal with different types of motion, which is not possible when the subspace size is fixed. Moreover, despite the fact that EMS already provides very good results, the simple correction strategy of EMS+ allows an even better performance to be reached. EMS+ has a very similar performance to one of the best state of the art algorithms, the ALC, however, ALC is around 18 times slower than EMS+. In all of these experiments the only information provided for EMS/EMS+ and ALC was the total number of motions in the scene. In Section 3.4.1 EMS+ and ALC are tested without the need for any information at all, and more details about their performances are explained. Furthermore, the results of EMS+ and ALC on the synthetic database are also presented.

3.3 Adaptive Subspace Affinity

In the previous section it was shown how the MS proposed in LSA can be greatly improved by EMS/EMS+. EMS takes advantage of the relationship between the estimated rank of the trajectory matrix and the entropy of the affinity matrix and, without any tuning process, is able to automatically discard homogeneous, and therefore not useful, affinity matrices. The experiments show that EMS and EMS+ have a much better performance than LSA with the classic MS. Nevertheless, EMS is not optimal as it would discard a perfectly binary affinity matrix (as shown in the examples of Figure 3.9) and it tends to underestimate the rank (as explained in Section 3.2.2). Moreover, the results in terms of misclassification rate, can be dramatically different when the estimated rank changes even from only r to $r \pm 1$. As the real exact rank of the trajectory matrix is not known and it is necessary to rely on estimations, it would be desirable to have a more stable behaviour (i.e. small changes in the estimated rank should correspond to small changes in the misclassification rate). Finally, as explained in Section 3.1.2, the affinity measure used in LSA is not ideal and even if EMS has increased the performances of LSA, it is still using the same sin^2-based measure.

In this section three main contributions are provided: one issue related with the use of PAs, which partially solves the instability problem, is identify and tackled, a new interpretation of the global subspace size estimation is proposed, and finally a new similarity measure between subspaces is presented. The new subspace size estimation does not depend on any sensitive parameter, and it is able to select the dimension of the global subspace where the distribution of the PAs is the most suitable for the clustering step. Moreover, the new similarity measure is able to dynamically adapt to the distribution of the PAs.

3.3.1 Notation

Let us recall the notation that was used until now and that will be used also in this section. Given a collection of N subspaces, the PAs between two subspaces S_j and S_l, for $j, l = 1, \ldots, N$, are defined recursively as a series of angles, as shown in Equation (3.2) and (3.3). The i^{th} PA between the subspaces S_j and S_l, computed when the estimated size of the global subspace is r, is denoted as:

$$\theta_i^r(S_j, S_l). \tag{3.7}$$

As j and l vary the set of PAs are defined as:

$$\Theta_i^r = \{\theta_i^r(S_j, S_l),\ j, l = 1, \ldots, P\}. \tag{3.8}$$

Finally, the collection of all the Θ_i^r, for a given dimension i is defined as:

$$\Theta_i = \bigcup_{r=1}^{r_{\max}} \Theta_i^r, \tag{3.9}$$

where $r_{\max}$ is the upper bound of the global subspace size. For an at-a-glance overview of the notation refer to Figure 3.14.

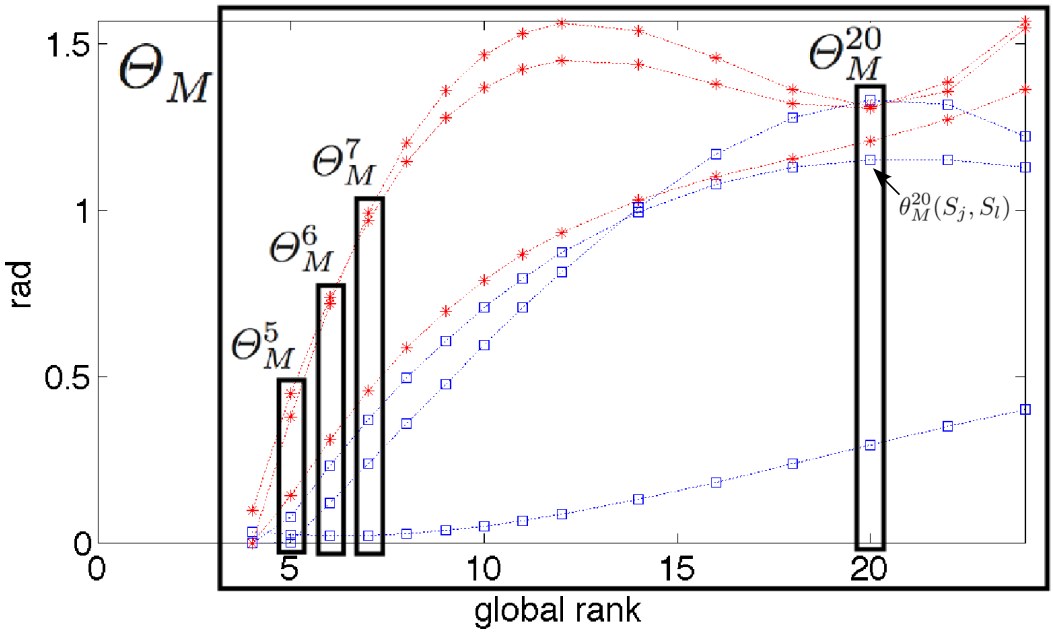
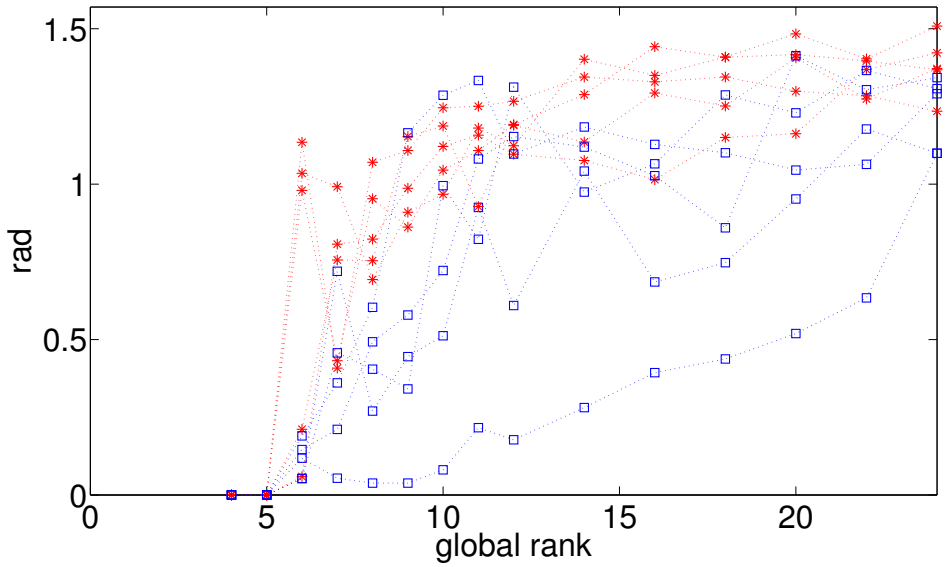


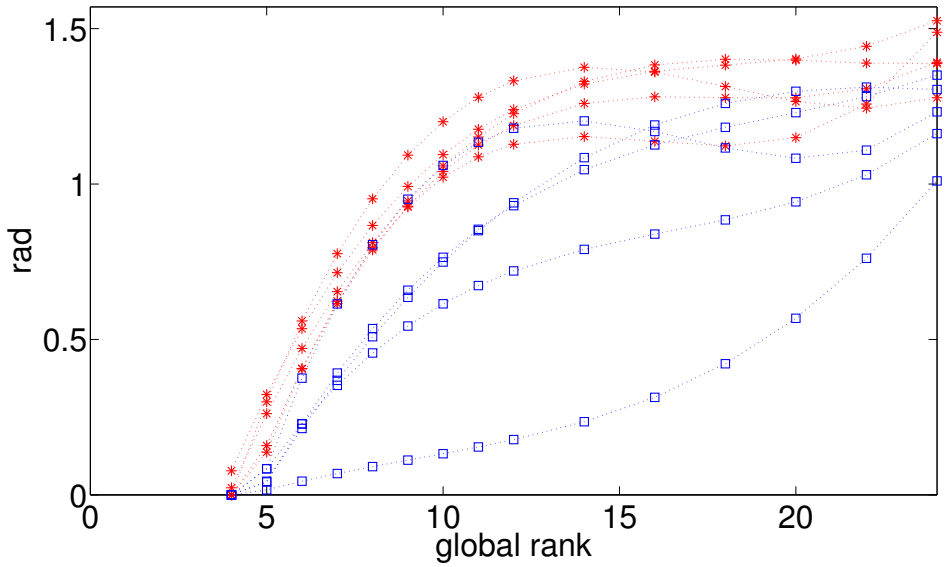
Figure 3.14: Overview of the notation. Random subset of the PAs of Θ_M (largest PAs) of a generic input sequence. PA between similar subspaces are represented with blue squares, PAs between different subspaces are represented with red asterisks.

3.3.2 Issues regarding the behaviour of principal angles

PAs between two subspaces are an efficient measure of orthogonality when the exact subspace bases are known. However, when the bases are estimated there are some issues that should be taken into account, especially when the exact size of the global subspace is unknown. In Section 3.2.1 the behaviour of PAs (computed following the LSA algorithm) as functions of r (the estimated size of the global subspace) was studied. It was shown that the overall trend of PAs, going from an underestimation to an overestimation of r, increases, typically starting from 0 radians and ending in $\pi/2$ radians, as shown in Figure 3.15. It was also explained that despite the increasing trend, PAs may have oscillations, as shown in Figure 3.15(a), due to the fact that when the rank is underestimated the bases are not well defined, while when the rank is overestimated the extra components introduced act like noise. Such an unstable trend of PAs results in a very different final segmentation when using the set of PAs Θ_i^r rather than $\Theta_i^{r\pm1}$, independently of which technique is used for the rest of the algorithm.



(a) Original PAs



(b) Interpolated PAs

Figure 3.15: Small random subset of the PAs of Θ_M (largest PAs) of the sequence *1R2RCT_A* taken from the Hopkins155 database. PAs between similar subspaces are represented with blue squares, PAs between different subspaces are represented with red asterisks.

In order to reduce the influence of these oscillations a *polynomial interpolation* of the PAs across the different ranks is here proposed. The trivially useless interpolation of order 1 can be discarded. The interpolation of order 2 is decreasing after its maximum, this does not fit with the increasing behaviour of the PAs. The interpolation of order 3 is able to smoothly follow the PAs trend, as shown in Figure 3.15(b). Interpolation of higher degrees would adhere too much to the data making the interpolation not effective. Different tests conducted on synthetic and real sequences confirmed PAs behaviour and the reliability of the interpolation of order 3.

This simple "pre-processing" step greatly reduces the oscillations of the PAs, as shown in the example of Figure 3.15(b). The choice about which rank to use becomes less critical now that the trend of each PA is smoother. In fact, now the differences between the sets Θ_i^r and $\Theta_i^{r\pm1}$ are not as big as they were in the original configuration. From now on, all the plots of the PAs will show the post-processed angles.

3.3.3 Rank selection via Principal Angles Clusterization (PAC)

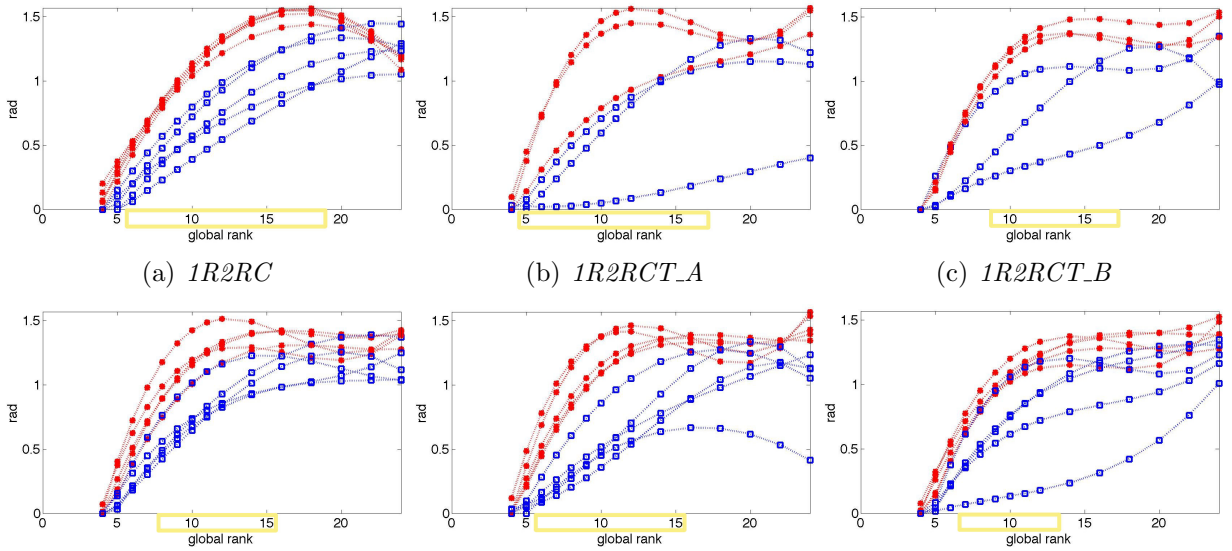
Let us recall once again that one of the most recognised weaknesses of LSA is the lack of robustness of the MS procedure for the estimation of the rank r, Equation (3.1), page 67. Equation (3.1), is extremely sensitive to changes of the parameter k. On the other hand, k is necessary in order to deal with sequences with different amounts of noise and number of motions. In order to solve this problem an algorithm named EMS/EMS+ was presented. EMS+ consists of computing different affinity matrices, by using different k values with the MS formula, and selecting the affinity matrix with the maximum entropy. This technique allows homogeneous affinity matrices (which correspond to over- or underestimation of the rank) to be discarded, and to use an affinity matrix with the highest content of information. Nevertheless, it was shown that EMS+ tends to underestimate the rank and it fails in the ideal case when the affinity matrix is binary.

Typically, if one wants to avoid the use of complex and unstable algorithms for rank estimation, the simplest way is to study the singular value spectrum of the matrix under

analysis, as explained in Section 3.1.1. Theoretically the number of singular values different from zero gives the rank of the matrix. Often the analysis is slightly more complex as the singular values may not be exactly equal to zero but they may assume small values. Therefore, the rank is given by the number of singular value above a certain threshold. However, the problem of the trajectory matrix rank estimation in real cases, with noise and dependent motions, is very challenging because the singular value spectrum of `W` tends to become smooth and the selection of a threshold becomes a difficult task.

Taking into account all of these problems and that the considerable amount of work already done for the rank estimation of the trajectory matrix has not led to satisfactory results, it was decided to renounce to the computation of the rank by the study of the singular value spectrum. Instead, the distribution of PAs in each set Θ_i (all the i^{th} PAs computed using different rank values) was studied. The fundamental idea on which this proposal is based is that *the rank r should be selected, for each dimension i, as the one that maximises the clusterization level of the PAs in the set Θ_i.* By clusterization is meant that the angles between similar and different local subspaces have to be well separated. In the ideal case (no noise and perfectly orthogonal local subspaces) PAs would cluster around 0 and $\pi/2$. In real cases PAs are not perfectly clustered, however, it is possible to evaluate the clusterization level for each Θ_i^r and select the one with the highest clusterization level for each i. For example, in Figure 3.16 some set of Θ_M are shown for different sequences. Note that PAs are plotted in different colours according to the classes of the angles (blue for angles between similar subspaces and red for angles between different subspaces). Clearly, the classification between "red" and "blue" angles is not known but it helps to illustrate the idea. In each set Θ_M it is possible to point out a small range of rank values, highlighted in yellow, where there is a good separation between "big" angles (red) and "small" angles (blue). Note that within this range, the rank with the maximum separation between big and small angles is actually always compatible with the theoretical maximum. Such separation is what is here called clusterization.

The measure of clusterization of each Θ_i^r could be performed by using a function in-



(a) *1R2RC* (b) *1R2RCT_A* (c) *1R2RCT_B*

(d) *1R2TCR* (e) *1R2TCRT* (f) *1R2RCR*

Figure 3.16: Small random subset of the PAs of Θ_M (largest PAs) of some Hopkins155 sequences. PA between similar subspaces are represented with blue squares, PAs between different subspaces are represented with red asterisks. Note that in all of the cases the ranks highlighted by the yellow square are characterised by the presence of a gap between blue and red angles.

spired by the Linear Discriminant Analysis, such a function is here called Principal Angles Clusterization (PAC):

$$\mathrm{PAC}(\Theta_i^r) = \frac{(\mu_a - \mu_{\mathrm{PAC}})^2 + (\mu_b - \mu_{\mathrm{PAC}})^2}{\sigma_a^{\gamma(\sigma_a)} + \sigma_b^{\gamma(\sigma_b)}}, \tag{3.10}$$

where μ_{PAC} is the centre of Θ_i^r computed as the mean of the $\mathcal{P}$ largest and smallest angles, μ_a, σ_a and μ_b, σ_b are the arithmetic means and the standard deviations of the PAs that are above and below μ_{PAC}, respectively. Tests on a random subset of the Hopkins155 database (70 sequences) have shown that $\mathcal{P} = 25\%$ of the Θ_i^r, gives a μ_{PAC} that is robust with respect to the presence of outliers (due to the few remaining oscillations of the PAs). Therefore, the belonging of each angle θ_i^r to the class of "small" or "big" angles is determined by the position of the angle with respect to μ_{PAC}. Note that μ_{PAC} is not computed as the mean of all the PAs to avoid biases due to the unbalanced number of representatives of one or the other class (the number of pair combinations between trajectories of different motions

is higher than the number of pair combinations between trajectories of the same motion). In the experiments r ranged from 2 to $8N$ (such a range guarantees to include the rank with the maximum PAs separation as $8N$ is above any theoretical maximum rank). The aim is to find the Θ_i^r with the maximum PAC value. In fact maximising the PAC value requires a maximisation of the numerator of Equation (3.10), which means to maximise the distance between the centre of the two classes, and minimising the denominator which implies minimising the spread of the angles in the two classes.

An important component of the formula is the functional exponent $\gamma(\sigma)$. If $\gamma(\sigma) \equiv 2$ was used, as in the LDA formulation, the maximum of the PAC function would always be in the extremes of its domain. In fact, it was explained in Section 3.3.2 that when $r \simeq 2$ the PAs tend to cluster around 0, hence the tiny values of the σ's that appear in the denominator of Equation (3.10) would boost the PAC value, despite the fact that the μ's are very close to each other. At the other extreme, when $r \simeq r_{\max}$, the μ's increase and become well separated even though the two classes partially overlap. However, as the σ's remain smaller than 1, the global effect would be a magnification of the already big numerator, boosting again the PAC value. Hence, it is necessary to use a variable exponent that takes small values at the extremes while approaching to 2 for middle values. As one of the aims of this work was to avoid algorithms with any manual tuning process, a functional variable $\gamma(\sigma)$ that can accomplish this task automatically was built.

A simple function that complies with these requirements is the following:

$$\gamma(\sigma) = \begin{cases} a_1\sigma^2 + a_2\sigma & \text{if } \sigma \leq \pi/8, \\ 0.1 & \text{if } \sigma > \pi/8. \end{cases} \tag{3.11}$$

The numerical coefficients a_1 and a_2 are not chosen after a tuning procedure but are determined through the following reasoning. Assuming an average case with PAs uniformly distributed, $\mu_{\text{PAC}} = \pi/4$, $\mu_a = 3\pi/8$ while $\mu_b = \pi/8$. Therefore, the upper bound of $\sigma_a, \sigma_b < \pi/8$. The numerical coefficients $a_1 = -50.63$ and $a_2 = 20.13$ define a function that fulfils the previous request by making the parabola pass through the points $A \equiv (0, 0)$,

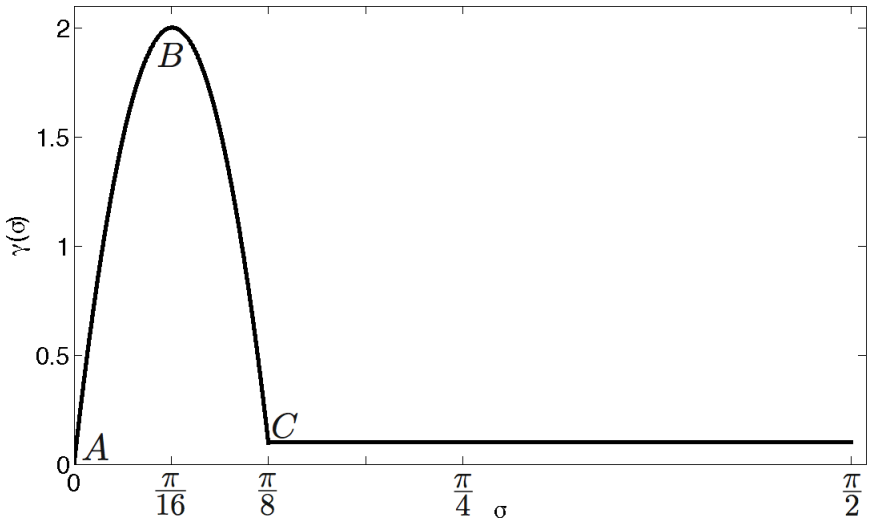


Figure 3.17: $\gamma(\sigma)$ function used in the PAC formula.

$B \equiv (\pi/16, 2)$ and $C \equiv (\pi/8, 0.1)$, as shown in Figure 3.17. When σs $> \pi/8$ the angles are excessively spread and the two classes are likely to overlap. For this reason, after $\pi/8$ the function becomes constant: $\gamma(\sigma) \equiv 0.1$. By adopting the $\gamma(\sigma)$ defined in Equation (3.11) small PAC values are ensured when the angles in the Θ_i are overlapping, either due to underestimation of the rank (cases in which σs are very small) or to overestimation of the rank (cases in which μs and σs are very big). Note that in the ideal case of PAs perfectly divided (which would lead to perfect affinity matrices like the one in Figure 3.9(b)) $\sigma_a = \sigma_b = 0$, therefore, the PAC value would be infinitely big. Hence, the selection of the rank via the PAC function does not suffer from the problem of not being able to exploit the ideal case, as previously described for the EMS/EMS+ technique.

To summarise, instead of selecting a dimension of the global space (i.e. estimating the rank of the trajectory matrix) through a model selection technique, the set of angles Θ_i^r with the highest PAC value for each i is chosen. Note that the selected rank may be different for each set Θ_i. This is a new interpretation of the size of the global subspace: the "most expressive" dimension for each set Θ_i in terms of clusterization level is identified, without estimating the rank of `W`. In Figure 3.18 is possible to see an example of PAC selection applied to Θ_M of the sequence *1R2RCT_A* taken from the Hopkins155 database. The

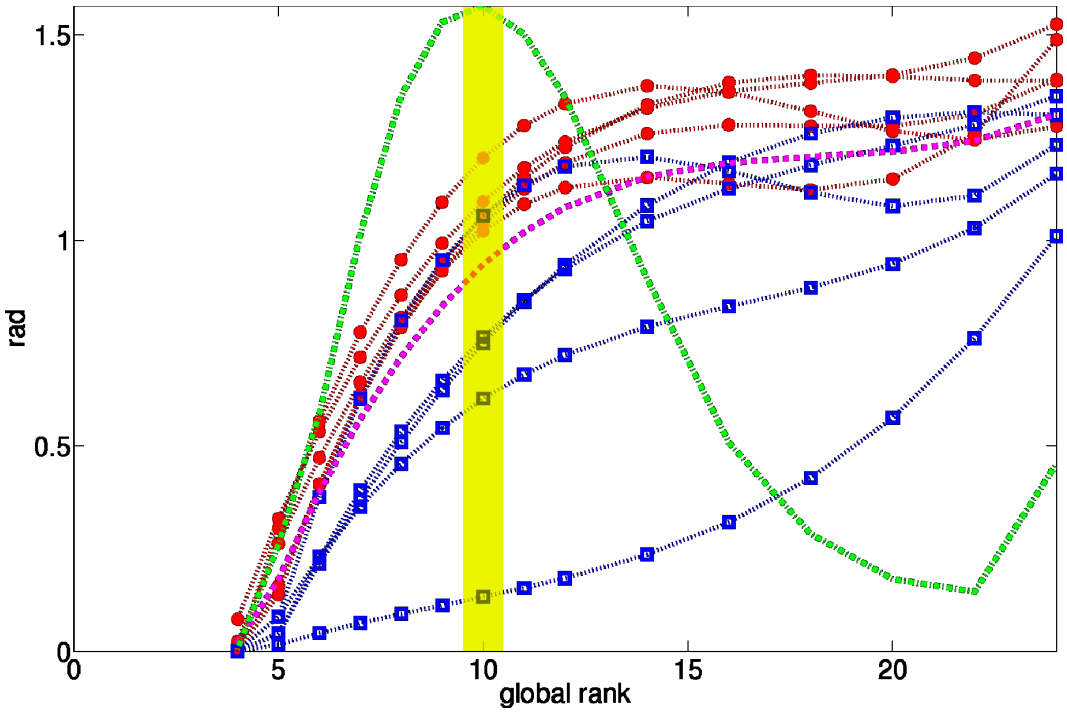


Figure 3.18: Example of PAC selection. In green the PAC function, the yellow column highlights the set of PAs that should be used according to the PAC selection.

PAC function, green line, has low values at positions that correspond to the ranks where there is a considerable overlap between the two classes of angles. The maximum of the PAC function occurs for rank 10 where there is a good separation between the two classes. This choice is also compatible with the theoretical maximum rank, in fact, the sequence analysed has 3 rigid motions, therefore, the maximum rank allowed is 12.

3.3.4 Sum of Clusterization-based Affinity (SCbA)

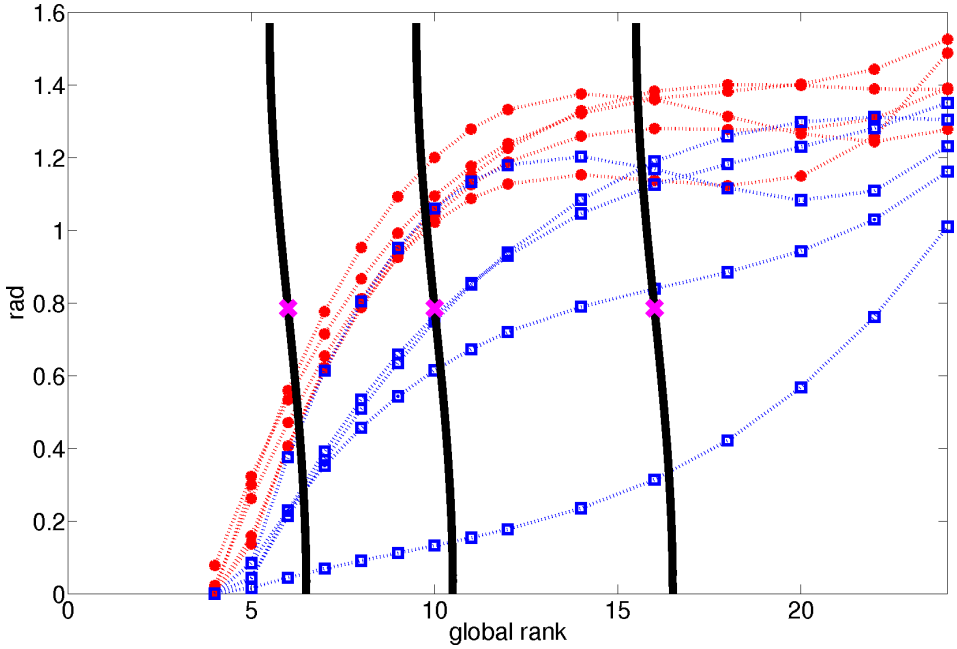
In the analysis of the state of the art, Chapter 2, it was explained that manifold clustering based algorithms often rely on the ability to compare the subspaces in order to assess the similarity between them. In the literature it is possible to find many affinity measures with different characteristics. A discussion of different affinity measures can be found in [58]. For many years accepted affinity measures have been based on the *sin* or the *cos* functions of the angles between subspaces. Even the LSA algorithm [72] uses an affinity measure based on the sin^2 function, Equation (3.4).

While in [58] some general weaknesses of sin^2-based affinity measures have been already pointed out, to our knowledge, no one has ever taken into account the specific issues of such an affinity measure in relation to the nature and the behaviour of the PAs. All affinity measures applied to PAs share a common assumption: the angles between similar subspaces are always close to zero, and the angles between different subspaces are always close to $\pi/2$. None of them takes into account that the recursive definition of the PAs tends to force the angle between two subspaces to increase when moving from Θ_i^r to Θ_{i+1}^r. Moreover, none of them take into account that the angles tend to increase also when moving from Θ_i^r to Θ_i^{r+1}, as explained in Section 3.3.2.

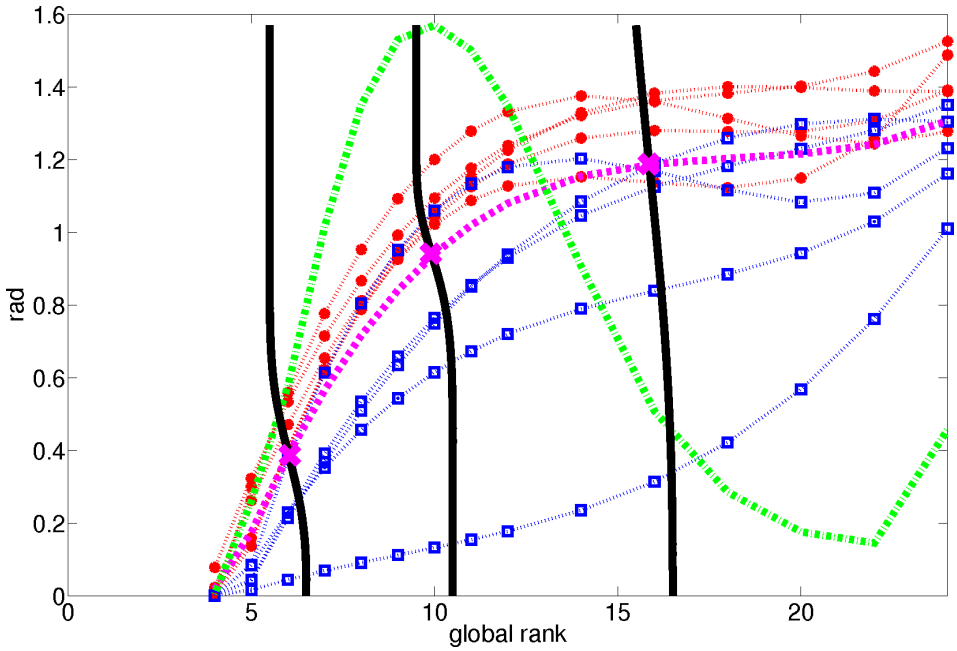
To illustrate this concept a simple example can be used. In Figure 3.19(a) PAs of Θ_M of the sequence *1R2RCR* (Hopkins155 database) were randomly plotted. In black the cos^2 function is plotted swapping the x with the y axis (note that the shape of the $-sin^2$ or the cos^2 functions are exactly the same, the two functions are only shifted on the y-axis). The cos^2 function always has the same shape, and the inflection point (magenta cross) is always in the same position, regardless of the rank to which it is applied. As a consequence of this *rigidity*, if the estimated rank is $r = 6$ all of the PAs have an affinity value that falls prior to the inflection point. Opposite cases are when $r = 10$ and $r = 16$, in which most of the PAs have an affinity value after the inflection point. Therefore, the cos^2 function, as well as any other rigid function, is very sensitive to the rank estimation.

An ideal, and better suited, affinity measure applied to PAs should be *flexible* and change its shape according to the rank to which it is applied or, generally, to the data to which it is applied. The affinity measure that is here proposed is able to adapt itself to the distribution of the PAs in any set Θ_i^r, so that it minimises the negative effects of a wrong rank estimation and it emphasises the difference between similar and different subspaces. The not normalised Clustering-based Affinity ($\overline{\text{CbA}}$) between two generic subspaces S_j, S_l, for $j, l = 1, \ldots, P$, for a given Θ_i^r is defined as the function $\overline{\text{CbA}} : \Theta_i^r \to \mathbb{R}^+$,

$$\overline{\text{CbA}}(\theta_i^r(S_j, S_l)) = \exp\left(-\frac{\beta - 1}{\beta}\left(\frac{\theta_i^r(S_j, S_l)}{\alpha}\right)^{\beta}\right), \qquad (3.12)$$



(a) cos^2 functions (black)



(b) CbA functions (black)

Figure 3.19: Comparison between cos^2 and CbA functions on a random subset of the PAs of Θ_M (largest PAs, 3 rigid independent motions, hence maximum rank 12). PAs between similar subspaces are represented with blue squares, PAs between different subspaces are represented with red asterisks. The affinity functions computed at the rank $r = 6, 10, 16$, appear in black. In Figure 3.19(a) the inflection point of the function is denoted with a magenta cross. In Figure 3.19(b) the magenta dotted line is μ_{PAC} (which for every r is also the inflection point of the CbA function), the green line-dot-line is the value of the PAC function.

where θ_i^r is the i^{th} principal angle computed at rank r. α and β are the two positive parameters ($\alpha > 0$, $\beta \geq 2$) that allow the function to change in relation to the distribution of the PAs.

The arrangement of the parameters of Equation (3.12) has been chosen so that $\overline{CbA}$ has a negative first derivative over all its domain, while its second derivative is negative for $\theta < \alpha$, positive for $\theta > \alpha$ and equal to zero for $\theta = \alpha$. Once again, one of the objectives of the thesis is to avoid parameters that the user has to manually tune. In fact, a solution is to set $\alpha = \mu_{\text{PAC}}$, used also in Equation (3.10), so that the inflexion point occurs at the estimated centre of the distribution. In this way the function is always stretched or compressed in order to fit the distribution of PAs. The β parameter is used in order to emphasise the differences between similar and different subspaces in an automatic fashion. In fact, β controls the slope of the function: the higher the β the steeper the slope. Ideally, an affinity function should have a steep slope when PAs are well clustered and a more gentle slope when the clusterization is not clear. As the PAC function provides a measure of the clusterization level, β should be proportional to the PAC value. Specifically, $\beta = \mathcal{F} \times \text{PAC}(\Theta_i^r)$, where $\mathcal{F}$ is a constant, a boosting factor, used in order to give more or less importance to β. In all of the experiments $\mathcal{F} = 5$, which has been empirically shown (on a random subset of 70 sequences of the Hopkins155 database) to be a suitable factor.

It is now possible to define the normalised Clustering-based Affinity (CbA) as follows:

$$\text{CbA}(\theta_i^r(S_j, S_l)) = \frac{\overline{\text{CbA}}(\theta_i^r(S_j, S_l)) - \min(\overline{\text{CbA}})}{\max(\overline{\text{CbA}}) - \min(\overline{\text{CbA}})}. \tag{3.13}$$

This simple normalisation ensures that the affinity values of CbA are defined between 0 and 1. In Figure 3.19(b) three CbA functions were plotted (as for the cos^2 function the x and the y axes were swapped) applied to different ranks r within the set Θ_M. In this picture it is possible to appreciate that, thanks to the parameter α, the inflection point (magenta cross) changes so that it always corresponds to the μ_{PAC} value (magenta line), hence minimising the effect of possible errors in the choice of the rank r. Moreover, thanks

to the parameter β the slope of CbA changes depending on how well the small angles are separated from the large angles: for example it is possible to appreciate that if the selected rank is 6 or 10 the two classes of angles are well separated and the slope is steep, on the other hand, when the selected rank is 16 the two classes tend to overlap and there is not a clear division, therefore, the slope is more gentle. A gentle slope is desirable when the two classes of angles are not well separated because giving an average affinity value does not compromise the final classification. The use of a gentle slope postpones the classification of the angles until after the analysis of the remaining $M-1$ angles (remember that for each pair of subspaces there are M angles, but so far the analysis was focused only on one generic angle i).

The affinity between two subspaces has to take into account all of the M angles. Therefore, the final affinity is defined as the normalised weighted Sum of CbA (SCbA):

$$\mathrm{SCbA}(S_j, S_l) = \frac{\sum_{i=1}^{M} \mathrm{CbA}(\theta_i^r(S_j, S_l))\mathrm{PAC}(\Theta_i^r)}{\sum_{i=1}^{M} \mathrm{PAC}(\Theta_i^r)}, \tag{3.14}$$

M being the minimum size between subspaces S_j and S_l. Note that by weighting the CbA values by the PAC function more importance is given to the set Θ_i^r where the angles between similar and different subspaces are better separated.

In the case of the EMS algorithm also the size M of the local subspaces was estimated. In this case there is no information to perform such estimation, hence, M was fixed to 4. This is not a weak choice. In fact, subspaces with a real dimension bigger than 4 are not very common and even if a pair of subspaces have more than 4 dimensions the method is taking their 4 most representative ones (note that both of the subspaces have to be bigger than 4 in order to really loose some data). More often it may happen that the smallest of the two subspaces is actually smaller than 4, therefore, some extra information may be included.

However, the affinity of the first larger angles and the presence of the weighting system (that gives more emphasis to the set of Θ_i where angles are better separated) should

compensate for the extra random information included.

Finally, let us analyse some theoretical properties of the SCbA measure. SCbA respects the axioms of an affinity function proposed in [58]:

- **basis independent**: even if principal vectors are not uniquely defined, PAs always are [130];
- **symmetry**: from Equation (3.14) it is possible to see that $\text{SCbA}(S_j, S_l) = \text{SCbA}(S_l, S_j)$;
- **normalised**: from Equation (3.12) and (3.14) it is possible to see that both CbA and SCbA are normalised, therefore, $0 \leq \text{SCbA} \leq 1$;
- **orthogonality consistency**: given that

$$S_j \perp S_l \iff \theta_i^r(S_j, S_l) = \pi/2, \tag{3.15}$$

  $\forall i = 1, \ldots, M$, from Equation (3.12), (3.13) and (3.14) it follows that:

$$\text{SCbA}(S_j, S_l) = 0; \tag{3.16}$$

- **inclusion consistency**: given that

$$S_j \subseteq S_l \iff \theta_i^r(S_j, S_l) = 0, \tag{3.17}$$

  $\forall i = 1, \ldots, M$, from Equation (3.12), (3.13) $\text{CbA}(0) = 1$ and from Equation (3.14) it follows that:

$$\text{SCbA}(S_j, S_l) = 1. \tag{3.18}$$

In the remainder of the thesis the use of PAC and SCbA together will be referred to as the Adaptive Subspace Affinity (ASA) algorithm. The ASA algorithm is summarised in Algorithm 3.2. ASA does not have any sensitive parameter that should be manually tuned.

All the necessary parameters are designed to be automatically tuned by the algorithm itself. The few other parameters, namely $\mathcal{P}$ (the percentage of PAs used to compute the μ_{PAC} for the PAC function) and $\mathcal{F}$ (the boosting factor of the auto tuned β parameter in the SCbA measure) that are not automatically tuned are fixed to the provided constants that were empirically found using a subset of 70 random sequences of the Hopkins155 database and are not changed in any of the experiments. When the term ASA is used without specifying any clustering algorithm it will be intended as ASA using K-means, nevertheless, in the experiments also the N-cuts algorithm is tested.

Algorithm 3.2 ASA algorithm

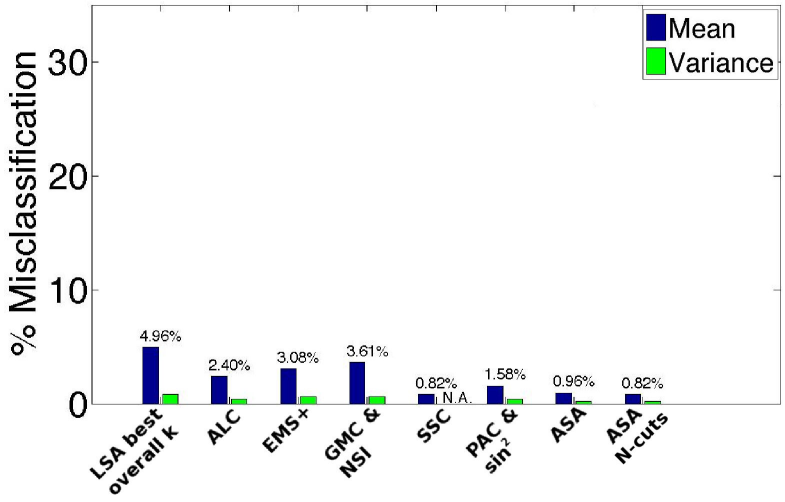
1: Build a trajectory matrix `W`;
2: **for** $r = 2$ to $r_{\max}$ {in our tests $r_{\max} = 8N$} **do**
3: project every trajectory, which can be seen as a vector in $\mathbb{R}^{2F}$, onto an $\mathbb{R}^r$ unit sphere by singular value decomposition (SVD) and truncate to the first r components of the right singular vectors;
4: exploit the fact that in the new space (global subspace) most of the points and their closest neighbours lie in the same subspace, to compute by SVD the local subspaces generated by each trajectory and its nearest neighbours (NNs);
5: compute PAs between all of the subspaces;
6: **end for**
7: smooth the PAs;
8: apply PAC to find the best r for each Θ_i $(i = 1 \dots M)$;
9: apply SCbA to build the affinity matrix `A`;
10: cluster `A` by any spectral clustering technique;

3.3.5 Experiments

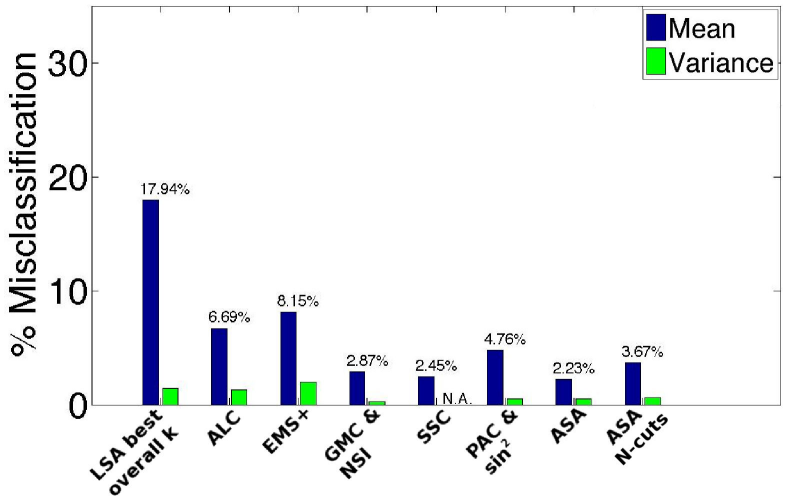
The ASA algorithm was tested on the same databases described in Section 2.5. As for the EMS+ case, the main state of the art techniques were compared: LSA best k ($k = 10^{-7.5}$), EMS+ and ALC (results taken from [58]). In addition, new algorithms had been proposed during the development of the ASA technique, therefore, the most important and successful were also compared: the Grassmannian Maximum Consensus (GMC) [57] combined with the Normalized Subspace Inclusion (GMS+NSI, results taken from [58]), and the Sparse Subspace Clustering technique (SSC) (results taken from [67]).

In this section ASA is compared using N-cuts and K-means. Moreover, in order to evaluate separately the PAC function (for the rank selection) and the SCbA function (for the affinity estimation) PAC was initially tested using the old affinity function (sin^2-based) described in Equation (3.4).

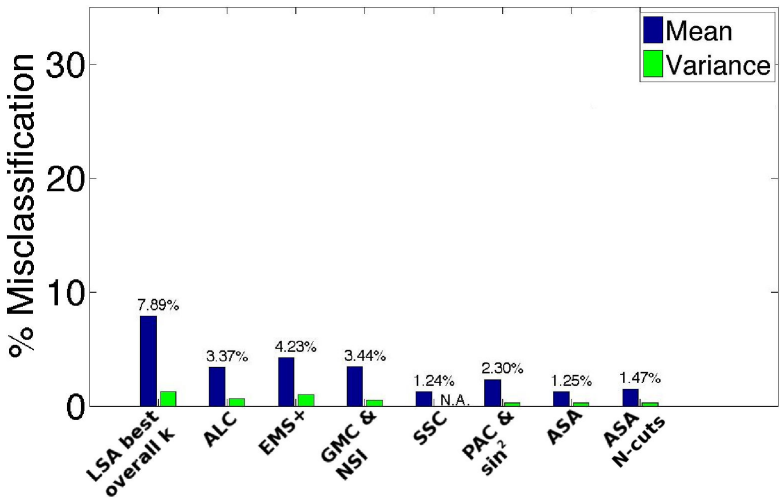
Figure 3.20 shows the average misclassification rates and the variance of each method, while Table 3.2 shows the details of the misclassification rates for each type of video sequence (checkerboards, articulated and traffic). Firstly, it is possible to see that ASA proposal outperforms every LSA-based technique (LSA and EMS+), proving that ASA further improved the already good performances of EMS+. Moreover, if with EMS+ there was a considerable gap between the performances with 2 and 3 motions, ASA has closed this gap to a difference of only about 1%. When also the other techniques are taken into account, ASA, together with SSC, has the lowest misclassification rates both with 2 and 3 motions. Let us take a deeper look at these results. The ability of PAC to estimate the global subspace size is proved by the fact that the results of EMS+ are improved (the misclassification rates were halved) simply by using PAC together with the old sin^2-based affinity function. On the other hand, when PAC was used together with the new affinity function SCbA (in the plots this combination is called ASA) the misclassification rates were further halved leading to one of the lowest error rates in the state of the art of motion segmentation: 0.96% with 2 motions and 2.23% with 3 motions. This result testifies that, when the subspaces generated by each motion are almost orthogonal (i.e. the motions are independent), then the assumption of the classical affinity functions (angles are well separated) is verified and good performance can be obtained. For example, Table 3.2 shows that PAC with sin^2-based affinity function has a very low misclassification rate in the checkerboard sequences (2 motions): 0.85% against 1.00% of ASA. However, when the input sequence become more challenging and the motions become partially dependent, like in the articulated case or when more than 2 motions are present, then the sin^2-based affinity function is not so effective anymore. In fact, the misclassification rate of PAC with sin^2 on the articulated sequences is 5.48% with 2 motions and 21.81% with 3 motions, whereas



(a) 2 Motions



(b) 3 Motions



(c) 2 and 3 Motions

Figure 3.20: Mean and variance of the misclassification rate on the Hopkins155 database (number of motions known).

2 Motions	Checkerboards(78)		Articulated(11)		Traffic(31)		All types(120)	
Method	% Avg	% Var	% Avg	% Var	% Avg	% Var	% Avg	% Var
LSA best k	5.15	0.93	3.65	0.18	4.95	0.75	4.96	0.80
ALC	1.49	0.21	10.70	2.25	1.75	0.03	2.40	0.40
EMS+	2.20	0.52	2.32	0.15	5.58	1.19	3.08	0.68
GMC&NSI	3.75	0.62	8.05	0.72	1.69	0.49	3.61	0.62
SSC	1.12	NA	0.62	NA	0.02	NA	0.82	NA
PAC&sin^2	0.85	0.20	5.48	1.53	2.02	0.39	1.58	0.37
ASA	**1.00**	0.324	**1.75**	0.10	**0.57**	0.016	**0.96**	0.22
ASA N-cuts	0.69	0.164	2.47	0.28	0.57	0.02	0.82	0.13
3 Motions	Checkerboards(26)		Articulated(2)		Traffic(7)		All types(35)	
Method	% Avg	% Var	% Avg	% Var	% Avg	% Var	% Avg	% Var
LSA best k	19.09	1.70	9.57	1.83	16.06	0.33	17.94	1.42
ALC	5.00	0.84	21.08	8.34	8.86	1.73	6.69	1.32
EMS+	8.76	2.30	6.38	0.82	6.354	1.53	8.15	2.00
GMC&NSI	2.29	0.33	6.38	0.82	1.67	0.02	2.87	0.28
SSC	2.97	NA	1.42	NA	0.58	NA	2.45	NA
PAC&sin^2	4.47	1.26	21.81	9.51	0.96	0.01	4.76	1.41
ASA	**2.41**	0.65	**3.72**	0.28	**1.11**	0.04	**2.23**	0.50
ASA N-cuts	4.27	0.81	4.79	0.46	1.11	0.03	3.67	0.63

Table 3.2: State of the art performance comparison. Misclassification rates on the Hopkins155 database. In brackets the number of sequences for each type of video. NA stands for value not available.

ASA is able to reach rates of 1.75% and 3.72% respectively. Only SSC is comparable to ASA scoring a misclassification rate of 0.82% with 2 motions and 2.45% with 3 motions. However, it should be remarked that for ASA the only two free parameters, $\mathcal{P}$ and $\mathcal{F}$, were fixed for the whole database whereas the results of SSC were obtained by tuning some of its parameters per each sequence of the database. GMC with NSI also preforms well, however, with 2 motions its misclassification rate is higher than ASA and SSC: 3.61%. Finally, the misclassification rate of ASA using N-cuts is very similar to ASA (which uses K-means), however, K-means tends to be slightly faster than N-cuts. For this reasons from now on ASA will be used only with K-means as final spectral clustering algorithm.

In Figure 3.21 the histogram of the misclassification rates of ASA is presented. The majority of the 155 sequences, 134, has a misclassification rate between 0% and 1%, and the total number of sequences with a misclassification rate below 5% is 145. The median

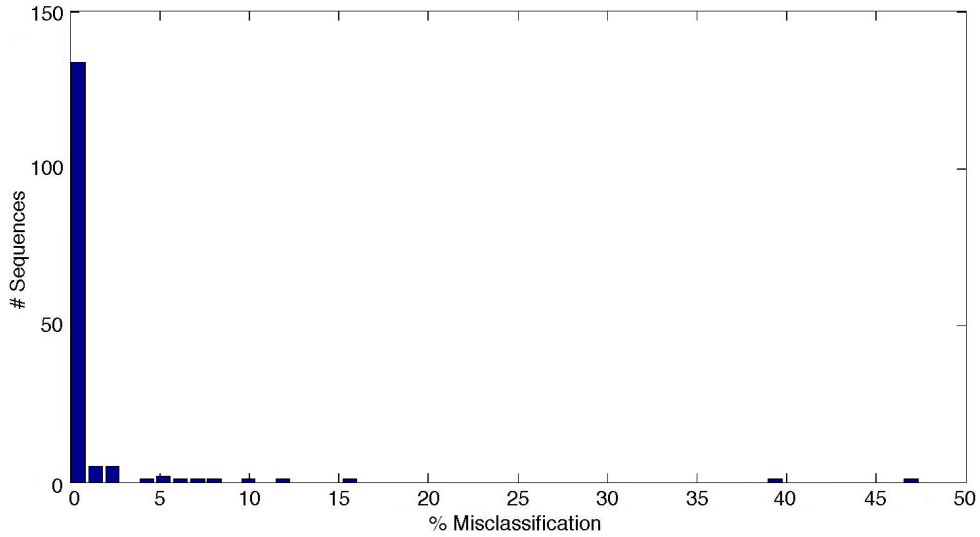


Figure 3.21: Histogram of the misclassification rate of ASA on the Hopkins155 database; each bin has a width of 1%.

Computational time (in seconds)	ALC	EMS+	ASA
Total with 2 motions	55625	2807	59066
Avg. with 2 motions	464	23	492
Total with 3 motions	33205	1902	35557
Avg. with 3 motions	949	56	1016
Total overall	88831	4709	94623
Avg. overall	573	31	610

Table 3.3: Computational time comparison between ALC, EMS+ and ASA on the Hopkins155 database. The three algorithms were implemented in Matlab and ran on an Intel Core2 Duo CPU @ 2.66GHz with 16 GB RAM.

misclassification of every group is always 0% with the exception of the articulated with 3 motions group where the median is equal to the mean (due to the presence in this group of only 2 sequences).

Table 3.3 extends the previously shown Table 3.1 with the computational time of ASA. Clearly, the computational time is the main drawback of ASA which performs (in terms of time) similarly to ALC. Despite the fact that ASA has a much better misclassification rate than both ALC and EMS+, such a long computational time requires further analysis. In Figure 3.22 a pie chart shows the average time (in percentage) spent by ASA on the whole Hopkins155 database. The different steps included in the plot are: the time required to

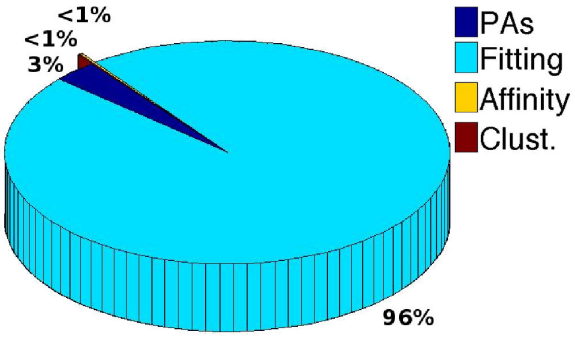


Figure 3.22: Pie chart that shows the distribution of the time spent by each of the ASA steps.

compute all the PAs for all the ranks, the time required to perform the polynomial fitting, the time required to compute the affinity and the time required to perform the spectral clustering. As the pie chart shows 96% of the total time is spent on the polynomial fitting routine, only 3% is spent on the computation of the PAs and less than 1% is spent on both the affinity computation (PAC and SCbA functions) and on the clustering step. This plot explains why ASA is so demanding in terms of computational power. Nevertheless, this is also a clear indication that the bottle neck is not in the method itself (PAC or SCbA), therefore, the computational time could easily be reduced by adopting a faster and simpler Median Filter or Savitzky-Golay filter [133] instead of using a polynomial fitting.

Synthetic database: extension to 4 and 5 motions

In order to verify how ASA performs on a different database, when the number of motions and noise increase, further tests on synthetic sequences with 2, 3, 4 and 5 motions (10 different sequences for each number of motions) and an increasing noise level were performed (from 0 to 2 pixels of standard deviation). In total 200 synthetic sequences were used. The misclassification rates are shown in Table 3.4. All the misclassification rates are smaller than 1%. For a given number of motions the misclassification rate remains rather stable even when the noise level increases. Moreover, the behaviour of ASA even with 4 and 5 motions (more than the motions in the Hopkins155 database) is very satisfactory.

By pointing out and tackling the instability of the PAs, introducing a new way of

Motions	2(10)	3(10)	4(10)	5(10)
Our Proposal	% Avg	% Avg	% Avg	% Avg
$\sigma_{\text{noise}} = 0$	0.00	0.30	0.36	0.68
$\sigma_{\text{noise}} = 0.5$	0.09	0.12	0.28	0.75
$\sigma_{\text{noise}} = 1$	0.09	0.24	0.31	0.64
$\sigma_{\text{noise}} = 1.5$	0.37	0.42	0.40	0.79
$\sigma_{\text{noise}} = 2$	0.63	0.24	0.49	0.89

Table 3.4: ASA misclassification rates on synthetic sequences with 2, 3, 4 and 5 motions and increasing noise level. In brackets the number of sequences for each type of video. Variance was omitted as always very close to zero.

selecting the set of PAs to use (PAC function), and presenting a new adaptive affinity measure (the SCbA), ASA has reached one of the best results on the Hopkins155 database without requiring any tuning process or any prior information about rank or noise. The synthetic results have confirmed the good performance of ASA which has proved to be robust also in presence of 4 and 5 motions. The only prior information that remains necessary is about the number of moving objects in the scene. In the next section a technique for estimating this data is presented and in Section 3.4.1 the results of ALC, EMS+ and ASA are compared when dealing with no priors at all.

3.4 Estimation of the number of motions

In the literature review presented in Chapter 2 it has emerged that very few of the existing algorithms are able to estimate how many objects are moving in a scene. Such an estimation is actually very challenging, as stated in [38], to the extent that the number of moving objects is usually assumed to be known information. However, this is a strong assumption that greatly limits the use of motion segmentation algorithms in real applications, where the number of objects is typically an unknown variable.

Theoretically, LSA does not require this information. However, the performance of the algorithm when the number of motions is not known degrade rapidly. This information is vital for the final step of the algorithm: the spectral clustering of the affinity matrix. Yan and Pollefeys in [72] suggest to use Normalized Cuts [5]. Normalized Cuts, however, is

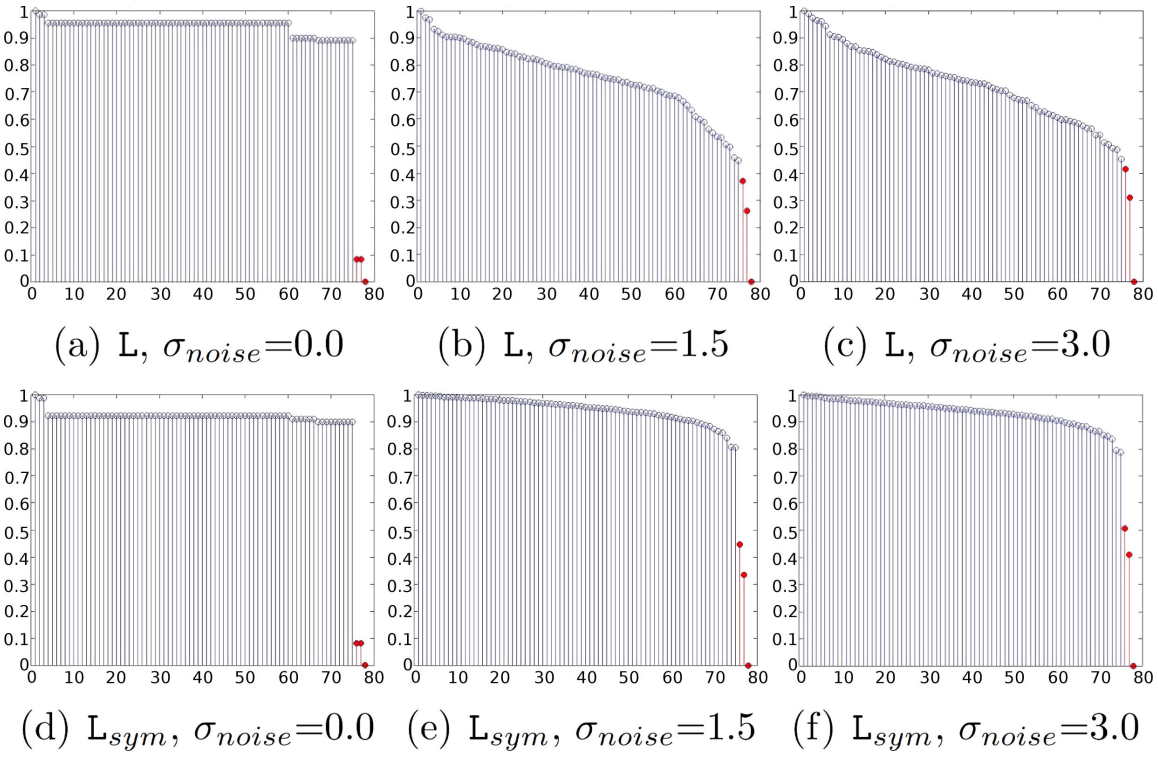


Figure 3.23: Eigenvalues spectrum of L (first row) and of L_{sym} (second row) for a synthetic sequence with 3 rotating and translating cubes and increasing Gaussian noise level.

only a good solution when the number of motions is known. When this information is not available, at every iteration the decision whether to terminate the process or not has to be taken. The authors of Normalized Cuts suggest to use the Cheeger constant [129] or the cost of the last cut in order to take this decision. The Cheeger constant and the cost of the last cut are both clues of how difficult is to split the graph by removing a specific edge. When, after finding the minimum cut, one of these two values is "high" it becomes worthless to split the graph any further and the process should stop. Therefore, these indicators need a threshold to decide when the value is "high enough". The problem is that such a threshold is strongly influenced by the noise level and the number of motions. This explains why most of the Normalized Cuts implementations require advance knowledge of the number of motions.

Having tested the difficulty of using the Cheeger constant or the cost of the last cut, a different way was taken by exploiting some spectral graph theory theorems. Specifically

the following proposition.

Proposition (Number of connected components) *Let G be an undirected graph with non-negative weights. Then, the multiplicity n of the eigenvalue 0 of the Laplacian matrix equals the number of connected components in the graph [134].*

In [5] it is shown that finding the minimum cut for splitting the graph, is equivalent to thresholding the values of the second smallest eigenvector of the Laplacian matrix $\mathtt{L}$:

$$\mathtt{L} = \mathtt{D} - \mathtt{A}, \tag{3.19}$$

where $\mathtt{A}$ is the adjacency matrix (specifically, in our case it corresponds to the affinity matrix), and $\mathtt{D}$ is a $P \times P$ diagonal matrix, P being the number of tracked features. Every entry $\mathtt{D}(i,i)$ contains the sum of the weights that connect node i to all of the others. Hence, matrix $\mathtt{L}$ and the proposition could be used in order to estimate the number of connected components, which is also the number of motions. The proposition refers to an ideal case where the eigenvalues that correspond to the connected components are exactly equal to 0 (which means no noise and fully independent motions). However, perturbation theory says that, in practical cases, when there is not an ideal situation, the last n eigenvalues may not be equal to 0, nevertheless, they should be very close to those of the ideal case [134]. Naturally, in motion segmentation, especially with real sequences, the ideal situation is not expected, but theoretically it should be possible to identify the threshold between the eigenvalues that correspond to the connected components and the remaining eigenvalues.

The proposition holds true also when using the eigenvalue spectrum of the Symmetric Normalised Laplacian matrix $\mathtt{L}_{sym}$ [134, 135] instead of the Laplacian matrix $\mathtt{L}$:

$$\mathtt{L}_{sym} = \mathtt{D}^{-1/2}\mathtt{L}\mathtt{D}^{-1/2}. \tag{3.20}$$

Figure 3.23 shows the eigenvalues of $\mathtt{L}$ (first row) and of $\mathtt{L}_{sym}$ (second row) of a synthetic

sequence with 3 motions and with an increasing noise level. The last 3 eigenvalues, which should suggest the number of motions, are plotted in red. In the plots all the eigenvalues are normalised in order to allow an easier comparison between $\mathtt{L}$ and $\mathtt{L}_{sym}$. As can be seen, when the noise increases the difference between the red eigenvalues and the others decreases. However, the difference between the fourth to last and the third to last eigenvalues remains rather large in the $\mathtt{L}_{sym}$ spectrum, while it becomes really small in the $\mathtt{L}$ spectrum. In the next section the experimental results of the estimation of the number of motions using $\mathtt{L}$ and $\mathtt{L}_{sym}$ are compared.

As was previously stated, when the Cheeger constant or the cost of the last cut are used, the main problem is the choice of a robust threshold. The same problem is present when using the spectrum of the eigenvalues, regardless of the choice of $\mathtt{L}$ or $\mathtt{L}_{sym}$. However, with the Cheeger constant or the cost of the last cut there is not much information that can be used in order to take such a decision, while with the eigenvalues the information of the whole spectrum could be exploited in order to gain more knowledge about that specific video sequence. Nevertheless, the threshold cannot be fixed as the noise greatly influences the differences between eigenvalues, as shown in Figure 3.23. In order to dynamically find a threshold for every case, different techniques were implemented and tested.

In general, estimating the number of motions using the eigenvalue spectrum can be seen as a two class classification problem: class 1 is the class of the eigenvalues above the threshold, while class 2 is the class of the eigenvalues below the threshold. The number of eigenvalues inside class 2 provides the estimation of the number of motions. The first tested technique is the Fuzzy c-means clustering (FCM) [136]. This technique returns a probability of belonging to class 1 and to class 2 for every element of the set. The second technique is the Otsu's method [137] which chooses the threshold that minimises the intra-class variance. As in this particular case it seems that the inter-class variance is also playing an important role, a trade-off between the intra and the inter-class variance was sought. Similarly to the case of the PAC function (Section 3.3.3), inspiration was again taken from the Linear Discriminant Analysis (LDA), which minimises the intra-class variance while

maximising the inter-class variance. With LDA the chosen threshold t is given as:

$$t = \underset{t}{\text{argmax}} \frac{Q_1(\mu_1(t) - \mu_{all})^2 + (1 - Q_1)(\mu_2(t) - \mu_{all})^2}{Q_1\sigma_1^2(t) + (1 - Q_1)\sigma_2^2(t)}, \tag{3.21}$$

where μ_1 and σ_1 are the mean and the variance of class 1 given a certain threshold t, μ_2 and σ_2 are the mean and the variance of class 2, and μ_{all} is the mean of all the eigenvalues spectrum. In the original formulation of LDA, Q_1 is the probability of belonging to class 1. However, in this context this probability is unknown (knowing the probability means knowing already the number of motions). At the same time, not providing any weight for the two classes would mean that both classes are equally likely even though this is not true (the number of eigenvalues of class 1 should be much bigger than the number of eigenvalues of class 2). Therefore, Q_1 should be seen as a weight that has to favour class 1 over class 2.

The nominator of Equation (3.21) measures the inter-class dissimilarity, whereas the denominator measures the intra-class dissimilarity. Therefore, choosing the threshold that maximises this ratio is like choosing the threshold that maximises the inter-class dissimilarity and minimises the intra-class dissimilarity.

In the next section, one of the experiments presented is about the estimation of the number of motions using the eigenvalue spectrum of $\mathtt{L}$ and of $\mathtt{L}_{sym}$, and thresholding them with FCM, OTSU and LDA.

3.4.1 Experiments

As explained in Section 3.4, different thresholding techniques were tested in order to perform the estimation by exploiting the eigenvalues spectrum of either the Laplacian matrix $\mathtt{L}$ or the Symmetric Normalised Laplacian $\mathtt{L}_{sym}$. In both cases the Laplacian matrices are built after EMS has been used in order to estimate the dimension of the global space. Concerning the setting of the thresholding algorithms, different values were tried and here the sets that obtain the best results on a random subset of the Hopkins155 database (70

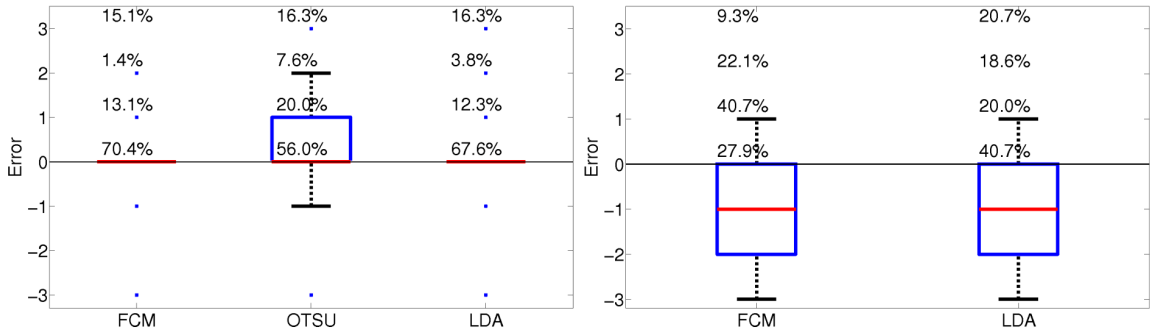
|Error|		FCM	OTSU	LDA
L	μ	0.6	0.8	0.7
	σ	0.6	0.9	0.9
L_{sym}	μ	0.23	0.47	0.37
	σ	0.19	0.48	0.42

Table 3.5: Mean and variance of the absolute value error of the estimation of the number of motions on the Hopkins155 database.

sequences) are presented. For the estimation using FCM, the best results are obtained by counting the eigenvalues with a probability of belonging to class 2 equal to or greater than 0.9. For OTSU the best results are obtained using only the last 20 eigenvalues. For LDA the best results are obtained with $Q_1 = 0.8$.

A first qualitative study suggests that $\mathtt{L}_{sym}$ is more robust against noise, as previously shown in Figure 3.23. Quantitative tests were also performed estimating the number of motions on the Hopkins155 database using both $\mathtt{L}$ and $\mathtt{L}_{sym}$ with the thresholding techniques explained in the previous section. Table 3.5 shows mean and standard deviation of the error of the estimated number of motions for the tested thresholding techniques (the absolute value of the error is considered). As expected, independently of the technique, mean and standard deviation are always considerably smaller when using $\mathtt{L}_{sym}$ than when using $\mathtt{L}$. As the Symmetric Normalised Laplacian spectrum seems to be more robust against noise, $\mathtt{L}_{sym}$ was chosen in order to estimate the number of motions.

From the results of Table 3.5, OTSU seems to be the weakest measure, while FCM and LDA have very similar performances. In order to perform a deeper test, extra noise was added to the Hopkins155 sequences. Another six databases derived from the Hopkins155 were created by adding random Gaussian noise with standard deviations of 0.5, 1, 1.5, 2, 2.5 and 3 pixels, to the tracked point positions. The original database plus the six derived from it composes a bigger database with 1085 video sequences. The estimation of the number of motions performed on all of the six databases confirmed the weakness of OTSU, as shown in Figure 3.24. The numbers on each boxplot correspond to the percentage of sequences where the error in the number of cluster estimation was (from bottom to top)



(a) Hopkins155 with extra noise (b) 4 and 5 synthetic motions with noise

Figure 3.24: Boxplots of the error of the estimation of the number of motions.

0, ± 1, ± 2 or greater than 2 (in absolute value). From these boxplots it is possible to see that FCM and LDA both have very high percentage of correct estimation and their first and second quartile collapse on the median.

FCM and LDA have similar performances. A deeper analysis reveals that FCM is particularly good with 2 motions: on the original database it has a percentage of correct estimation equal to 84.2% against 75.0% of LDA. However, when the number of motions increases, FCM appears to be less robust than LDA: with 3 motions the percentage of correct estimation of FCM is equal to 54.3% against 57.1% of LDA. The fact that the Hopkins155 database has more sequences with 2 motions tends to favour FCM. A similar conclusion can be drawn when the noise level increases, in fact the difference between the correct estimation of FCM and LDA drops from 6.4%, in the original Hopkins155 (considering all the sequences), to only 1.3%, in the database with 3 pixels of noise level, despite that the sequences with 2 motions are still over-represented in the Hopkins155 database.

In order to verify the reliability of these clues, an experiment with synthetic sequences with 4 and 5 motions, and different noise levels, was performed comparing FCM and LDA. The results of this experiment, shown in Figure 3.24(b), confirmed the conclusions drawn from the results on the Hopkins155 database and show that LDA outperforms FCM in terms of percentage of correct estimation (40.7% against 27.9%). If one wants to focus only on the Hopkins155 database, which has a majority of sequences with 2 motions, on

average FCM would have a slightly better performance than LDA. However, given the results just presented, if one wants to develop a motion segmentation algorithm for generic purposes (and not specialised on the Hopkins155 database) it would be better to use LDA. As the aim of this work is not to find the best tuning for the Hopkins155 but is to have a generally good motion segmentation algorithm, LDA will be used as the thresholding technique of the $\mathtt{L}_{sym}$ eigenvalue spectrum.

EMS+ combined with the LDA estimation of the number of motions is called Enhanced Local Subspace Affinity (ELSA), while ASA combined with the LDA estimation of the number of motions is called Automatic-ASA (A-ASA). In the following section ELSA, A-ASA and ALC [66] are compared when applied to the Hopkins155 database and to the synthetic database without requiring any other data than the trajectory matrix.

Segmentation without any prior information

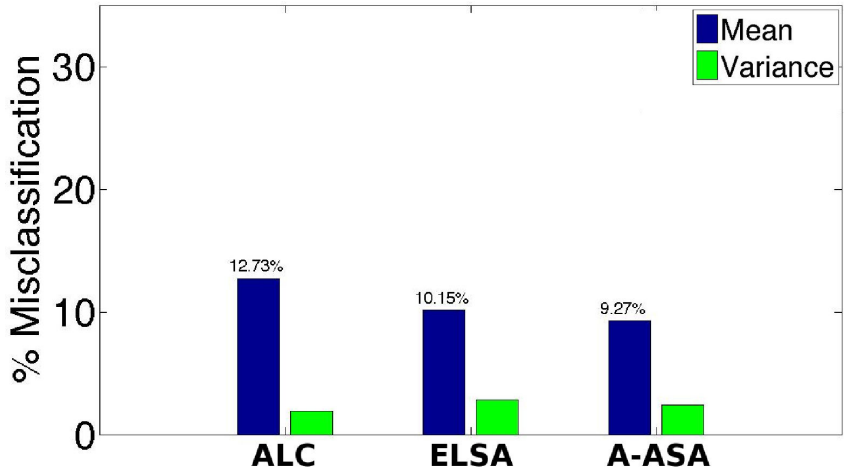
Finally, the misclassification rate when segmenting without any prior knowledge is presented. To the best of our knowledge, besides ELSA and A-ASA, the only technique which is able to provide satisfactory results without prior knowledge is the ALC [66]. Theoretically, also SSC [67] should be able to estimate the number of motions, however, no results were provided under this scenario. Moreover, SSC should be tuned for each sequence in order to obtain good performances, while ALC, ELSA and A-ASA can be run with the same set of parameters on the whole database.

In Section 3.2 it was shown that EMS+ performs better than the simpler EMS. However, the dimension of the global space (i.e. the final rank used) and the spectrum of the eigenvalues of $\mathtt{L}_{sym}$ have a mutual influence on each other. Thus, after the first estimation of the rank performed with EMS an iterative procedure is used. In an alternating fashion the number of motions and the rank are estimated. This alternation is iterated until one of the following conditions is verified: the estimated number of motions does not change, a loop is detected, or a maximum number of iterations is reached (for the experiments the maximum number of iterations is set to 5, however this condition is never verified

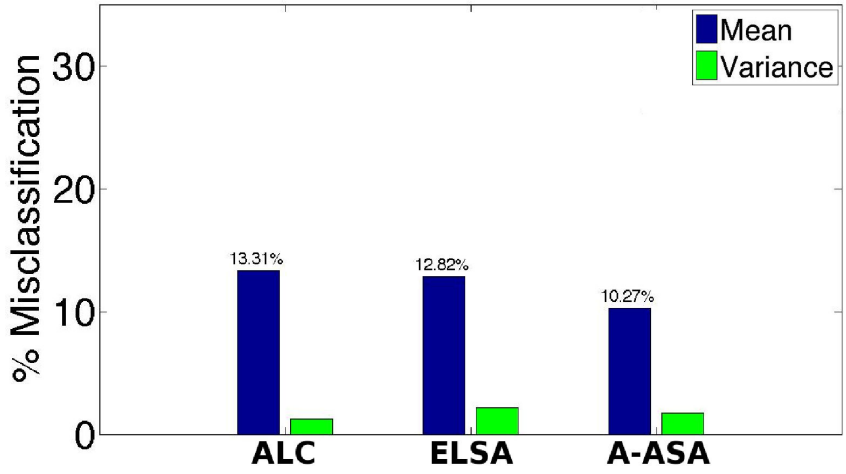
in our tests). Summarising, the results presented in this section for ELSA are obtained using: EMS+ for the model selection, and the eigenvalue spectrum of L_{sym} thresholded dynamically by LDA for the estimation of the number of motions. As far as A-ASA is concerned, the number estimation can simply be performed by thresholding with LDA the eigenvalue spectrum of the matrix $\mathtt{L}_{sym}$, built using the affinity matrix computed after the SCbA step. As for ALC, it can be forced to select the segmentation with the smallest coding length, in this way the number of motions is not required.

In order to compare the three techniques an upper bound in the number of motions estimated equal to 5 was imposed. It should be noted that computing the misclassification rate when the number of estimated motions does not match with the real number could be done in different ways. In fact the error due to overestimation of the number of motions could be somehow corrected in a post-processing step with a merging strategy. On the other hand, an underestimation usually means that points from at least two different motions are considered as the same one, so it can be seen as a more crucial error. However, also in this case one could argue that for each cluster a deeper analysis could be performed in order to correct the segmentation with a splitting approach rather than a merging one. In these experiments the misclassification rate when the number of motions is wrongly estimated is as follows. All the possible one-to-one associations between the estimated groups of trajectories and the ground truth groups of trajectories are taken into account (clearly, some groups will not be associated). The association that minimises the error, computed by summing the error of the associated groups and the number of features that belong to non-associated groups, is selected.

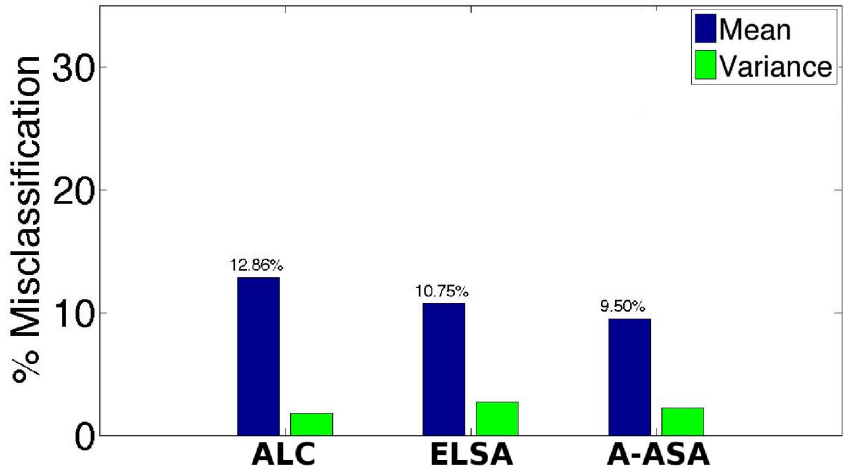
Figure 3.25 shows mean and variance of ALC, ELSA and A-ASA when using no prior information about the number of motions in the scene. Table 3.6 shows mean and variance of the misclassification rate divided by number of motions and per type of sequence. The overall mean misclassification rate is similar for the three algorithms: 12.86% for ALC, 10.75% for ELSA, and 9.75% for A-ASA. However, if the median is taken into account the differences among the algorithms become more remarkable: the median of ALC is



(a) 2 Motions



(b) 3 Motions



(c) 2 and 3 Motions

Figure 3.25: Mean and variance of the misclassification rate on the Hopkins155 database without any prior knowledge.

2 Motions	Checkerboards(78)		Articulated(11)		Traffic(31)		All types(120)	
Method	% Avg	% Var	% Avg	% Var	% Avg	% Var	% Avg	% Var
ALC	14.15	1.86	5.27	1.67	12.03	2.06	12.73	1.93
ELSA	8.90	2.48	10.36	2.41	12.82	3.91	10.15	2.86
A-ASA	11.08	2.69	6.80	1.85	5.60	1.76	9.27	2.40
3 Motions	Checkerboards(26)		Articulated(2)		Traffic(7)		All types(35)	
Method	% Avg	% Var	% Avg	% Var	% Avg	% Var	% Avg	% Var
ALC	12.51	1.04	6.72	0.73	17.46	2.03	13.31	1.25
ELSA	10.71	1.53	11.17	2.50	19.84	4.29	12.82	2.19
A-ASA	11.37	1.92	3.72	0.28	8.04	1.44	10.27	1.72

Table 3.6: State of the art comparison. Misclassification rates on the Hopkins155 database without prior information. In brackets the number of sequences for each type of video.

10.23%, the median of ELSA is only 1.31%, whereas the median of A-ASA is strikingly low: 0.27% . This suggests that ELSA, and especially A-ASA, perform very well in most of the sequences and they fail in only a few of them. This is confirmed also by the results shown in Figure 3.26 where the histogram of the misclassification is presented. From the histogram it is possible to appreciate that the number of sequences with a misclassification rate between 0 and 1% are 97 for A-ASA, 77 for ELSA and around 45 for ALC.

The reason why A-ASA and ELSA are generally better than ALC, but when they fail the misclassification is higher than ALC, could be explained by Figure 3.7(i) to 3.7(p), page 79. From these plots it is possible to notice that if the rank is not well estimated (especially if it is underestimated) the affinity values quickly collapse to 1 (in the case of underestimation) or to 0 (in the case of overestimation). When this happens the clustering process becomes very noisy, hence the segmentation result is greatly affected.

To conclude, in this chapter the performance of these three algorithms, in terms of number estimation, is analysed. Table 3.7 shows the mean error in absolute value and the percentage of correct estimation. Both these pieces of information are important, as a small mean error is desirable in terms of good estimation of the number of motions, however, in terms of misclassification rate when even a small mistake is made in the estimation the segmentation could be dramatically affected. Therefore, a high percentage of correct

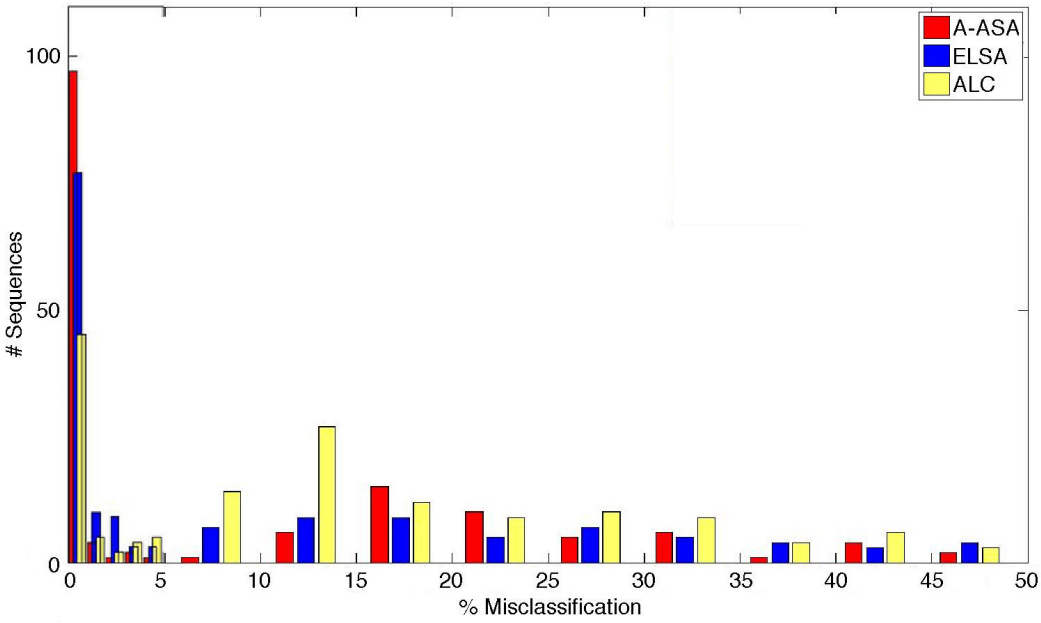


Figure 3.26: Histogram of the misclassification rate of ALC and ELSA on the Hopkins155 database; misclassifications from 0% to 5% are sub-sampled with bins of 1%, misclassifications greater than 5% are sub-sampled with bins of 5%.

estimation rate is even more important than a low mean error (although the majority of the times these two values are connected). The table shows that ALC estimation is not very efficient, on the whole database only 25.81% of the time it is able to estimate correctly the number of motions, with an average error of 1.16. On the other hand, ELSA and A-ASA perform much better with ELSA being slightly more precise: on the whole database ELSA correct estimation happens 70.97% of the time, with an average error of 0.37, while A-ASA correct estimation happens 67.10% of the time with an average error of 0.42. This small difference between ELSA and A-ASA may be due to the iterative refined estimation of ELSA while A-ASA performs the estimation only one time without any refining process. Nevertheless, A-ASA has a better performance in terms of misclassification rates, proving once again the robustness of the algorithm.

Synthetic database: extension to 4 and 5 motions

As the Hopkins155 database contains a maximum of 3 motions, some synthetic experiments were also done in order to have an idea of the behaviour of the three algorithms when the

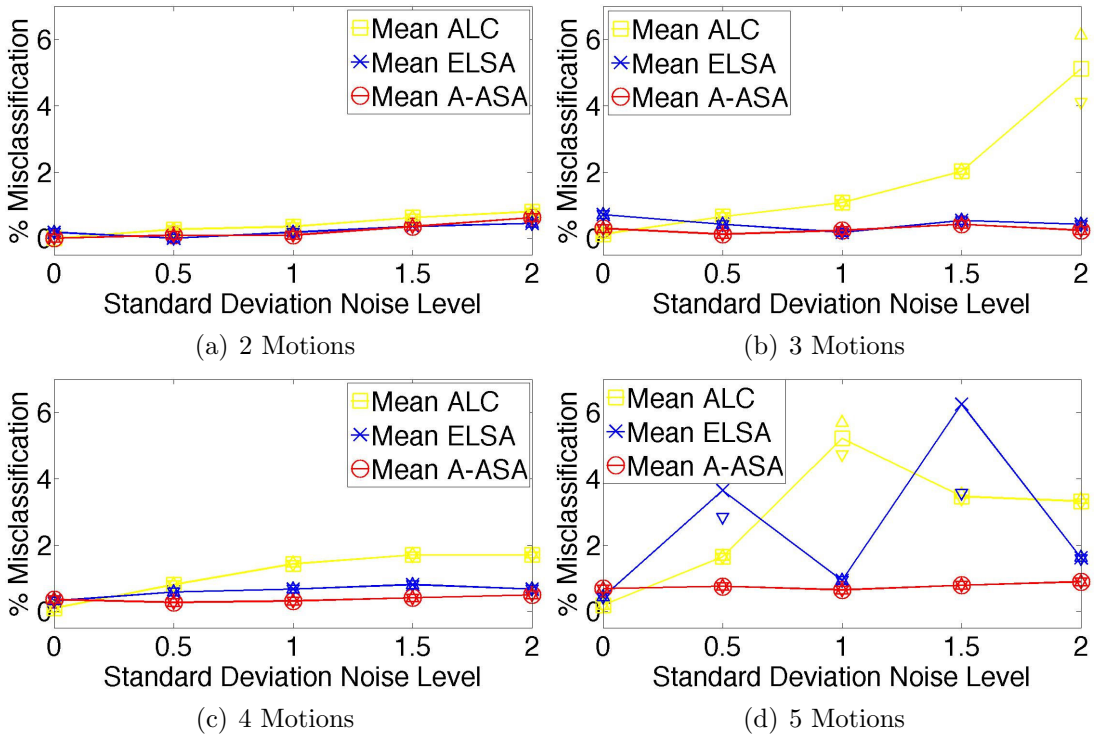
2 Motions	$\mu(\lvert error \rvert)$	% correct estimation
ALC	1.13	30.00
ELSA	0.33	75.00
A-ASA	0.39	70.00
3 Motions	$\mu(\lvert error \rvert)$	% correct estimation
ALC	1.25	11.43
ELSA	0.49	57.14
A-ASA	0.51	57.14
Whole DB	$\mu(\lvert error \rvert)$	% correct estimation
ALC	1.16	25.81
ELSA	0.37	70.97
A-ASA	0.42	67.10

Table 3.7: Comparison of the estimation of the number of motions on the Hopkins155 database.

number of motions increases. Figure 3.27 shows the mean of misclassification rates for any given number of motions. In all of the cases the three algorithms have very low misclassification rates (always below 6.5%). With 2 motions the three algorithms have very similar performances and have a misclassification rate below 1%, independently from the noise level. When the number of motions becomes higher, with 3 and 4 moving objects, ELSA and A-ASA are more stable than ALC. In the challenging case of 5 motions ELSA and ALC performances degrade faster than the ones of A-ASA, and their misclassification rates change depending on the noise level. Note that the oscillations of the misclassification rate are rather big for ELSA and ALC, nevertheless, in the plot the y-axis goes only from 0% to 6% giving the illusion of an even bigger change. Despite of the scale, A-ASA shows a very robust behaviour keeping its misclassification rate always below 1%.

In terms of estimation of the number of motions the three algorithms perform almost perfectly: ELSA and A-ASA have 100% of correct estimation while ALC scores a correct estimation rate of 99.5% (in one sequence with 3 motions and 2 pixels of standard deviation noise it estimates only 2 motions, this can be seen in Figure 3.27(b) where ALC has for this case a high misclassification rate).

The behaviour of the three algorithms in these synthetic experiments is coherent with



(a) 2 Motions (b) 3 Motions

(c) 4 Motions (d) 5 Motions

Figure 3.27: Mean misclassification rate and variance versus noise level for synthetic experiments.

that on the Hopkins155 database and confirm the robustness and the efficiency of the A-ASA algorithm.

3.5 Conclusion

In this chapter new algorithms for the motion segmentation problem were proposed. From the analysis of the state of the art performed in Chapter 2 the LSA algorithm [22] emerged as one of the best performing techniques and also one of the most promising, as LSA is able to deal with a variety of types of motions. Therefore, this chapter was opened with a more detailed explanation of the LSA. From the study of LSA some weaknesses emerged, in particular the difficult and unstable rank estimation step, the assumptions implied by the use of the sin^2-based affinity function and the required knowledge about the number of motions.

In Section 3.2 a much more robust rank estimation was proposed: the Enhanced Model

Selection+ (EMS+). EMS+ is based on the fact that principal angles follow a specific pattern when the rank of the trajectory matrix is increased. EMS+ exploits such a pattern in order to identify which rank leads to the affinity matrix with the highest information content (i.e. the highest entropy). Results showed that EMS+ outperforms every LSA-based technique without requiring any prior information, besides the number of motions, and without any tuning of its parameters. Moreover, EMS+ performs similarly to the Agglomerative Lossy Compression algorithm (ALC) [66], but EMS+ is more than 18 times faster than ALC.

Despite the very good performance, EMS+ still has few drawbacks that were pointed out at the beginning of Section 3.3. While the first of its weaknesses is mainly theoretical, in that EMS+ is suboptimal because it would never choose a perfectly binary affinity matrix (which is an ideal and not existing case, anyway), the second weakness is much more relevant: due to the behaviour of the principal angles the choice of using a rank size equal to r rather than $r \pm 1$ can lead to very different misclassification rates. Given that the real rank of the trajectory matrix is unknown such a behaviour is quite undesirable. Firstly, in order to obtain a more robust behaviour the principal angles trends were smoothed. Then, a new way of selecting the rank based on the analysis of the principal angles clusterization (PAC) was proposed. PAC selects the rank where the principal angles between similar and different subspaces are best separated. PAC not only performs a robust estimation (as shown by the results), but it also introduces a new way of interpreting the selection of the rank: in fact the selected rank was allowed to change for each principal axes of the local subspaces. Finally, PAC would have a maximum value in the ideal case, solving also the theoretical issue emerged with EMS+. The results showed that PAC has better performance than EMS+. However, the new proposed algorithm, Adaptive Subspace Affinity (ASA), does not include only the smoothing of the principal angles and the PAC function. ASA contains also a completely new affinity measure that is automatically able to adapt itself in order to fit the distribution of the principal angles. The major achievement of this measure is that it can deal with every distribution of principal angles minimising the effect of an

erroneous rank estimation of the trajectory matrix while maximising the distance between similar and different local subspaces. Results of the experiments show that, even without changing the value of the only two free parameters of ASA, its misclassification rates are among the lowest in the literature.

Finally, in Section 3.4 a technique for estimating the number of motions in the scene was presented. The estimation of the number of motion is a challenging task and it is considered one of the open issues of motion segmentation [38]. The estimation of the number of motions presented in this thesis is based on the analysis of the eigenvalue spectrum of the Symmetric Normalised Laplacian matrix. The final number of motions is automatically estimated by finding a threshold dynamically computed using an LDA inspired approach. The combination of EMS+ and the estimation of the number of motions is called Enhanced Local Subspace Affinity (ELSA). The combination between ASA and the estimation of the number of motions is called Automatic ASA (A-ASA). ELSA and A-ASA were compared with one of the best performing techniques for motion segmentation without prior knowledge: the ALC. Results showed that both ELSA and A-ASA have better performances than ALC, with A-ASA scoring the lowest misclassification rate both with 2 and with 3 motions.

Future work should aim to reduce the computational time of ASA by adopting other ways for reducing the principal angles oscillations (this task took 96% of the total computational time of ASA). For example, instead of performing a time-expensive polynomial fitting, a simple and faster Median Filter or the Savitzky-Golay filter [133] could be enough to reduce the oscillations of the principal angles. Moreover, instead of selecting one specific rank it may be possible to use the whole set of ranks exploiting also the information contained in the derivative of the principal angles trend. In fact, angles between different subspaces tend to increase faster and earlier than angles between similar subspaces. The use of the whole trend could provide the extra information necessary to solve ambiguous cases, whilst simultaneously removing the rank estimation (and all the connected problems).

4

Joint Estimation of Segmentation and Structure from Motion

In this chapter a generic framework that can be applied to correct the result of any motion segmentation algorithm is proposed. The proposed solution not only assigns the trajectories to the correct motion (motion segmentation) but it also solves the 3D location of multi-body shape and fills the missing entries in the measurement matrix. Such a solution is based on two fundamental principles widely used in SfM but never applied, so far, to motion segmentation: the multi-body motion is subject to a set of metric constraints given by the specific camera model, and the shape matrix describing the 3D shape is generally sparse.

The chapter is organised as follows. In Section 4.1 there is an introduction, followed in Section 4.2 by a brief overview about multi-body SfM. In Section 4.3 an explanation of the single and multi-body SfM problem with missing data is offered. In Section 4.4 one theoretical approach and its limits are presented, while the final proposed algorithm is described in Section 4.5. The experiments on synthetic and real data sets, which validate the proposed algorithm, are shown in Section 4.6. In Section 4.7 conclusions are drawn and future work is discussed.

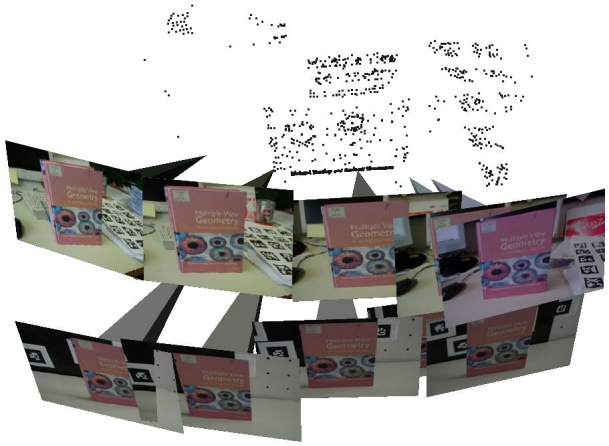


Figure 4.1: An example of SfM: different views of the same object are taken and a SfM algorithm can recover the 3D structure of the object and the position of the camera for each view. Image taken from [138].

4.1 Introduction

SfM is a problem closely connected to motion segmentation. SfM algorithms can recover the 3D structure of a moving object and the description of the motion for each frame. Figure 4.1 provides an example of SfM: given different views of the same book, a SfM algorithm can recover the 3D structure of the book and the position of the camera for each of the views. The input of SfM algorithms is a trajectory matrix where, very often, it is assumed that only the trajectories of one object are stored. However, most of the time a simple tracker cannot focus on one single object, therefore, it is more likely that the tracked points belong to different objects with different motions. Hence, in order to apply canonical SfM algorithms a motion segmentation algorithm should first group together trajectories that follow the same motion (i.e. trajectories that belong to the same object). The use of motion segmentation algorithms as a pre-processing step for SfM seems a straightforward application. In Figure 4.2 an example of how motion segmentation can be used as a pre-processing step of SfM is shown. If the segmentation is correct SfM can recover the 3D structure and the motion of the objects. However, SfM requires that no outliers are present in the trajectories of the segmented objects. This means that the motion segmentation algorithm has to provide a perfect segmentation. In Figure 4.3 one of

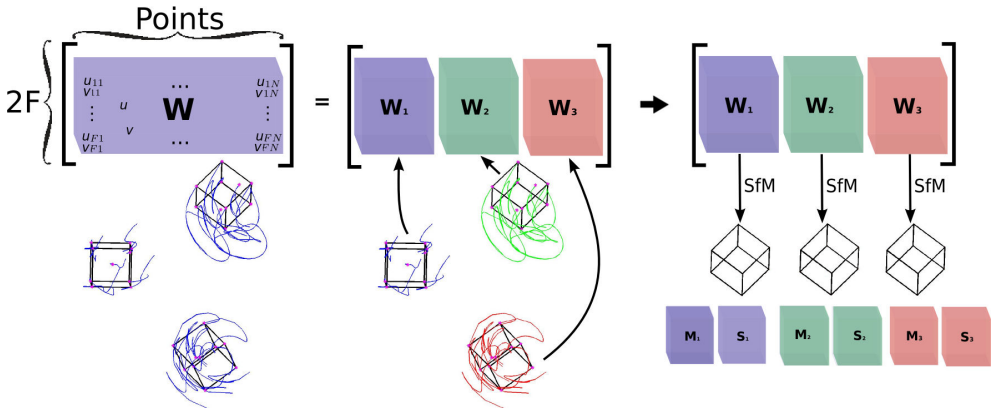


Figure 4.2: An example of motion segmentation as a pre-processing step of SfM.

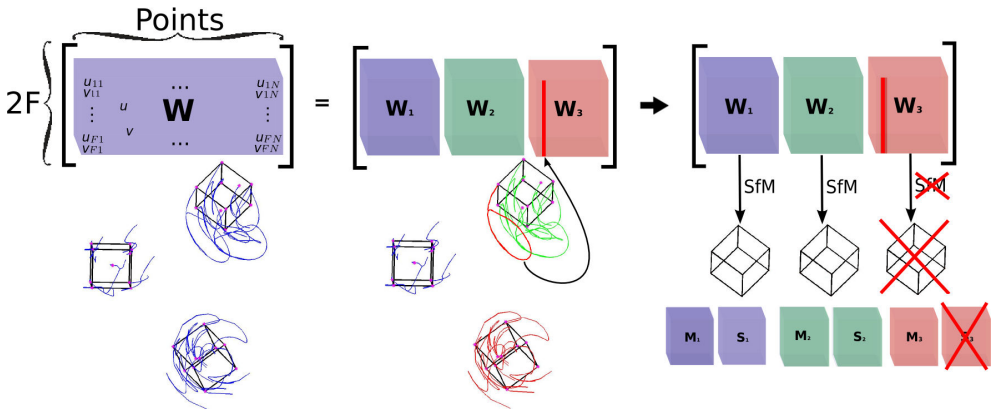


Figure 4.3: When the segmentation result has even only one single error, like the trajectory of $\mathtt{W}_2$ wrongly associated to $\mathtt{W}_3$, the 3D reconstruction $\mathtt{S}_3$ may not be correct. On the other hand, the motion description $\mathtt{M}_3$ is still reliable as it is influenced by all of the points of $\mathtt{W}_3$ and only one single error plays a minor role.

the trajectories of $\mathtt{W}_2$ is wrongly classified as belonging to $\mathtt{W}_3$, this mistake leads to a possibly wrong reconstruction of the 3D shape generated by $\mathtt{W}_3$. The motion description of $\mathtt{W}_3$ should be reliable in spite of the error, in fact the motion is influenced by the all of the points and if the majority is correctly classified the influence of one error plays a minor role [139]. In the previous chapters the motion segmentation problem was deeply investigated. The final algorithm proposed, ASA/A-ASA, solves part of the common problems of motion segmentation algorithms and proved to lead to very competitive results. Nevertheless, for some video sequences the final segmentation is still not perfect.

In this chapter, rather than trying to propose a further algorithm or a further improve-

ment of ASA, a deeper connection between motion segmentation and SfM is drawn. A generic framework that can be applied to correct the result of any motion segmentation algorithm is proposed. It is a novel optimisation framework for the estimation of the multi-body motion segmentation and 3D reconstruction of a set of image trajectories in the presence of missing data. The proposed solution not only assigns the trajectories to the correct motion (motion segmentation) but it also solves the 3D location of multi-body shape and fills the missing entries in the measurement matrix.

The problem of missing entries it is popular not only in motion segmentation or SfM. In many practical problems of interest, one would like to recover a matrix from a sampling of its entries. In general, this is impossible without some additional information. In many instances, however, the matrix we wish to recover is known to be structured in the sense that it is low-rank or approximately low-rank [140] (as in the case of the trajectory matrix). An example of a problem in which the missing entries has to be recovered could be the Netflix problem [141]. Users (rows of the data matrix) are asked to rate movies (columns of the data matrix), but users typically rate only very few movies. Yet one would like to complete this matrix so that the vendor (here Netflix) might recommend titles that any particular user is likely to be willing to order. In this case, the data matrix of all user-ratings may be approximately low-rank because it is commonly believed that only a few factors contribute to an individual's tastes or preferences [140]. Triangulation from incomplete data is another case in which given a scattered matrix it is necessary to recover the missing entries. Despite its popularity, the problem of finding a matrix of minimum rank that satisfies a set of affine constraints is NP-hard [142]. Nevertheless, if the problem is opportunely relaxed solutions can be found.

Thanks to the way the JESS algorithm is built, the missing entries of the trajectory matrix can be recovered while computing the segmentation and the 3D reconstruction. Such a solution is based on two fundamental principles widely used in SfM but never applied, so far, to motion segmentation: the multi-body motion is subject to a set of *metric constraints* given by the specific camera model, and the shape matrix that describes the 3D

shape is generally *sparse*. Such constraints are jointly included in a unique optimisation framework which iteratively enforces these constraints in three stages. First, given an initial (almost correct) segmentation, metric constraints are used to estimate the 3D metric shape and to fill the missing entries according to an orthographic camera model. Then, wrong assignment of the trajectories to each motion are detected using sparse optimisation of the shape matrix. A final reclassification strategy assigns the detected points to the right motion or discards them as outliers. Experiments which show consistent improvements to previous approaches both on synthetic and real data are provided.

4.2 Single versus multi-body SfM

In recent years several works have presented solutions to the 2D trajectories motion segmentation problem of a video sequence [58,67,71,81,108,124]. Most of the motion segmentation methods assume that the complete feature trajectories are visible throughout the sequence and do not deal with outliers introduced by a wrong association of the tracker. Recently, some approaches [65, 67, 75, 143] have also tackled these issues providing promising results.

Once a segmentation of the image trajectories is available other higher level tasks such as 3D reconstruction can take place. For instance, SfM methods may be applied to the 2D tracked features with the aim of recovering both the 3D coordinates of the points and the motion of the whole structure for each frame. Numerous techniques have proposed extensions to Tomasi and Kanade's SfM algorithm (refer to Chapter 2, Section 2.3.6 for an explanation of the Tomasi and Kanade SfM technique) in order to deal with rigid, articulated and also non-rigid objects [10, 144, 145]. Furthermore, different methods deal with the SfM problem in the case of missing data in the original 2D feature trajectories [146, 147]. However, all of these techniques share the common assumption that there is a single object moving in the scene.

Several attempts have been made to directly solve the multi-body SfM problem. Most of them tried to compute the motion segmentation intrinsically, exploiting epipolar geometry and mixing algebraic and statistical tools [148–151]. The main limitation of these methods

is the sensitivity to noise and outliers. Moreover, to the authors knowledge, the work presented in [152] is the only multi-body reconstruction approach that is able to deal with missing data. The main idea of [152] is to enforce two-view constraints between consecutive frames and to use a model selection strategy to perform the segmentation. However, this strategy can lead to under and over segmentation results if the model selection is not properly fed with the right candidates. As stated in [151] a practical multi-body SfM algorithm which can handle realistic sequences is still missing.

The large amount of successful single-body SfM algorithms and the weaknesses of the multi-body SfM algorithms mean that the link between motion segmentation and SfM is essential and requires strengthening. If the multi-body SfM problem is to be solved, a successful motion segmentation algorithm has to be applied as a pre-processing step in order to feed the single-body SfM algorithms with the trajectories of one object at a time. So far, this is how motion segmentation and SfM have been related to each other. However, SfM theory provides some constraints that motion segmentation algorithms have never exploited. Specifically, the *metric constraints* that have to be satisfied by each estimated motion, and the fact that the shape matrix that describe the 3D shape of the multi-body case is *sparse*. The aim of the work described in this chapter is to bring the motion segmentation and SfM problems closer by trying to use SfM constraints in order to solve the motion segmentation problem. If this would be possible, then motion segmentation and SfM could be solved simultaneously by the same algorithm.

Unfortunately, as will be described in Section 4.4, the constraints provided by SfM are useful but do not provide a unique solution for the multi-body case. Nevertheless, it was possible to develop an iterative bilinear optimisation strategy that, using the SfM constraints, corrects an initial (and possibly erroneous) solution given by any motion segmentation algorithm. Furthermore, the algorithm achieves a 3D multi-body reconstruction and it fills the missing entries according to an orthographic camera model. These constraints are particularly effective in the presence of missing data, since metric constraints are the key to obtain effective matrix completion of the 2D trajectories as demonstrated

in [147]. Hence, an initial segmentation is exploited to solve the multi-body SfM problem, which, in turn, provides unexploited constraints to correct the segmentation. Once a stop condition is verified a reclassification strategy can take place in order to reclassify the removed points. The proposed approach, Joint Estimation of Segmentation and Structure from motion (JESS), is a generic framework that can be applied to correct the result of any motion segmentation algorithm and it contributes towards the challenging direction of merging the problems of motion segmentation and SfM.

In the following section the theory for the single and multi-body SfM cases are developed. During this analysis the two constraints used, the metric constraints and the sparsity of the multi-body shape matrix, will be highlighted.

4.3 Structure from motion with missing data

First the bilinear SfM problem with missing data is introduced for the single object shape case. The solution to this problem provides the metric constraints given by an orthographic camera model. This formulation will then be extended to the multi-body case showing explicitly the sparsity constraint.

4.3.1 Single shape SfM with missing data

Consider a set of P_n image point trajectories extracted from an object n that rigidly moves in F frames. By stacking each image trajectory in a single matrix $\mathtt{W}_n$ of size $2F \times P_n$, it is possible to express the global motion and the 3D shape matrices of the single object n in bilinear form as:

$$\mathtt{W}_n = \mathtt{M}_n \mathtt{S}_n = \left[\begin{array}{c|c} \mathtt{R}_{1n} & \mathbf{t}_{1n} \\ \vdots & \vdots \\ \mathtt{R}_{Fn} & \mathbf{t}_{Fn} \end{array}\right] \mathtt{S}_n, \tag{4.1}$$

where $\mathtt{M}_n$ is a $2F \times 4$ motion matrix and $\mathtt{S}_n$ is a $4 \times P_n$ shape matrix in homogeneous coordinates. Each frame-wise element $\mathtt{R}_{fn}$, for $f = 1, \ldots, F$, is a 2×3 orthographic camera

matrix that has to satisfy the metric constraints of the model (i.e. $\mathtt{R}_{fn}\mathtt{R}_{fn}^T = \mathtt{I}_{2\times 2}$). The 2-vector $\mathbf{t}_{fn}$ represents the 2D translation of the rigid object. Here also the registered $\bar{\mathtt{W}}_n$ measurement matrix is introduced, such that $\bar{\mathtt{W}}_n = \mathtt{W}_n - \mathbf{t}\mathbf{1}_{P_n}^T$ where $\mathbf{1}_{P_n}^T$ is a vector of P_n ones and $\mathbf{t} = [\mathbf{t}_1^T, \ldots, \mathbf{t}_F^T]^T$. Thus, one of the bilinear factors includes a set of non-linear constraints given by the camera matrix, i.e. $\mathtt{M}_n$ resides in a specific *motion manifold* [147].

In the case of missing data given by occlusions or interrupted image tracks, the binary mask matrix $\mathtt{G}_n$ of size $2F \times P_n$ can be introduced such that a 1 represents a known entry and a 0 a missing one. In order to solve the bilinear components, and thus the SfM problem, the equivalent optimisation problem [146] can be defined as:

$$\begin{aligned} &\text{minimize} \quad \|\mathtt{G}_n \odot (\mathtt{W}_n - \mathtt{M}_n\mathtt{S}_n)\|^2 \\ &\text{subject to} \quad \mathtt{R}_{fn}\mathtt{R}_{fn}^T = \mathtt{I}_{2\times 2}, \quad f = 1, \ldots, F. \end{aligned} \tag{4.2}$$

This problem requires not only the estimation of the camera motion $\mathtt{M}_n$ and the 3D shape $\mathtt{S}_n$, but also the imputation (filling) of the missing entries in $\mathtt{W}_n$. The reader is referred to Buchanan and Fitzgibbon's work [146] for an extensive review of the approaches able to solve the missing data SfM problem.

4.3.2 Multi-body SfM with missing data

If the 2D image tracks belong to a set of N independently moving objects, it is still possible to formalise the problem in bilinear form. For the moment, the segmentation of each image trajectory is considered as given. Thus, by grouping the measurement in a single $\mathtt{W}$ it is possible to write:

$$\mathtt{W} = [\mathtt{W}_1|\mathtt{W}_2|\ldots|\mathtt{W}_N], \tag{4.3}$$

where $\mathtt{W}_n \in \mathbb{R}^{2F\times P_n}$, for $n = 1, \ldots, N$, is the trajectory matrix that contains only the P_n points of motion n (i.e. $P = \sum_{n=1}^{N} P_n$). Consequently, the $2F \times 4N$ aggregate motion

matrix $\mathtt{M}$ and the $4N \times P$ aggregate shape matrix $\mathtt{S}$ are written as:

$$\mathtt{M} = [\mathtt{M}_1|\mathtt{M}_2|\dots|\mathtt{M}_{N-1}|\mathtt{M}_N] \quad \text{and} \quad \mathtt{S} = \begin{bmatrix} \mathtt{S}_1 & \mathtt{0} & \dots & & \mathtt{0} \\ \mathtt{0} & \mathtt{S}_2 & \mathtt{0} & \dots & \mathtt{0} \\ \vdots & \mathtt{0} & \ddots & \mathtt{0} & \vdots \\ \mathtt{0} & & \mathtt{0} & \mathtt{S}_{N-1} & \mathtt{0} \\ \mathtt{0} & & \dots & \mathtt{0} & \mathtt{S}_N \end{bmatrix}, \tag{4.4}$$

so that:

$$\mathtt{W} = \mathtt{MS}. \tag{4.5}$$

It is now possible to note that the aggregate shape matrix $\mathtt{S}$ is remarkably sparse.

Finally, the optimisation problem with missing data is defined as:

$$\begin{aligned} &\text{minimize} && \|\mathtt{G} \odot (\mathtt{W} - \mathtt{MS})\|^2 \\ &\text{subject to} && \mathtt{R}_{fn}\mathtt{R}_{fn}^T = \mathtt{I}_{2\times 2}, \quad f = 1, \dots, F, \\ &&& \qquad\qquad\quad n = 1, \dots, N, \end{aligned} \tag{4.6}$$

where the matrix $\mathtt{G}$ of size $2F \times P$ defines the overall missing entries mask matrix. Solving this problem not only requires the estimation of the bilinear components, but also the classification of each 2D point to the correct moving body.

4.4 Motion segmentation and SfM: the missing constraint

In this section the attempt to solve the motion segmentation and SfM problems simultaneously is described. The missing constraint that would allow the complete merging of the two problems is highlighted.

In principle, each strategy that decreases the reprojection error of Equation (4.6) is appropriate to the task. However, the bilinear formulation $\mathtt{W} = \mathtt{MS}$ has a wide set of solutions, which greatly increases the chance to fall into a local minima. In order to give a qualitative evaluation of the solution space, let us consider $\hat{\mathtt{M}}$ and $\hat{\mathtt{S}}$ solutions that

correspond to a minimum of Equation (4.6). In this case, if the metric constraints are not considered, any non-singular matrix $\mathtt{Q}_{4N\times 4N}$ could be interposed between the factors, such that:

$$\mathtt{W} = \hat{\mathtt{M}}\mathtt{Q}\mathtt{Q}^{-1}\hat{\mathtt{S}} = \tilde{\mathtt{M}}\tilde{\mathtt{S}}, \tag{4.7}$$

would provide another valid solution. This happens in the case of non missing data. When some image trajectory points are missing the problem becomes even more complicated.

From now on, let us focus on the reconstruction, without considering the translation. This means that from now on only the registered trajectory matrix will be used and each matrix $\mathtt{M}_{fn} = \mathtt{R}_{fn}$. This simplification corresponds with the assumption that the objects are rigid. Therefore, for the object n, matrix $\mathtt{M}_n$ can be written as:

$$\mathtt{M}_n = \begin{bmatrix} \mathtt{R}_{1n} \\ \vdots \\ \mathtt{R}_{Fn} \end{bmatrix} = \begin{bmatrix} \mathtt{M}_{1n} \\ \vdots \\ \mathtt{M}_{Fn} \end{bmatrix}. \tag{4.8}$$

In the single-body case, an affine reconstruction $\tilde{\mathtt{M}}_1$ of the motion matrix, and an affine reconstruction $\tilde{\mathtt{S}}_1$ of the shape matrix can be obtained by SVD:

$$\mathrm{SVD}(\mathtt{W}) = \mathtt{U}\mathtt{D}\mathtt{V}^T = \tilde{\mathtt{M}}_1\tilde{\mathtt{S}}_1. \tag{4.9}$$

Then, in order to obtain the final valid motion and structure of the object, metric constraints of Equation (4.2) are imposed, so that matrix $\mathtt{Q}_{3\times 3}$ can be uniquely identified:

$$\begin{aligned} &\tilde{\mathtt{M}}_1\mathtt{Q}\mathtt{Q}^{-1}\tilde{\mathtt{S}}_1 = \mathtt{M}_1\mathtt{S}_1 \\ &\text{subject to} \qquad \tilde{\mathtt{M}_{f1}}\mathtt{Q}\mathtt{Q}^T\tilde{\mathtt{M}_{f1}}^T = \mathtt{I}_2 \;\; \forall f = 1, \ldots, F. \end{aligned} \tag{4.10}$$

If a similar approach is followed also for the multi-body case, it becomes necessary to deal with N matrices of the form $\mathtt{Q}_{3\mathtt{N}\times 3}$ (one for each moving object). A solution could be

found by solving N times the problem:

$$\begin{aligned} & \mathtt{M}_n = \tilde{\mathtt{M}}\mathtt{Q}_n \\ & \text{subject to} \quad \mathtt{M}_{fn}\mathtt{M}_{fn}^T = \mathtt{I}_2 \qquad \forall f = 1, \ldots, F, \\ & \qquad\qquad\qquad\qquad\quad \forall n = 1, \ldots, N. \end{aligned} \tag{4.11}$$

Note that $\tilde{\mathtt{M}}$ is the whole affine matrix of size $2F \times 3N$, which means that it contains the motions of all of the objects. Equation (4.11) can be solved by a non-linear least-square algorithm but only for one object. However, without any additional constraint, at each iteration n the same components used to recover the first object could be reused in order to recover the remaining $N-1$ objects. The optimisation would constantly fall inside the same minima unless an additional constraint is found, or the motion of the reconstructed object is "removed" from $\tilde{\mathtt{M}}$.

4.4.1 Projections of subspaces

Finding the additional missing constraint on the motions is more challenging that anticipated. Hence, a different direction was initially followed: once an object n is reconstructed by a non-linear least-square optimisation, the idea was to "subtract" $\mathtt{M}_n$ from $\tilde{\mathtt{M}}$. Note that it is not possible to perform a simple matrix subtraction because $\tilde{\mathtt{M}}$ is affine, therefore, $\mathtt{M}_n$ is spread throughout the whole $\tilde{\mathtt{M}}$ matrix. Besides, the dimensions of the two matrices do not match.

Nevertheless, *projectors* may help to perform such a subtraction. For the sake of clarity a case with only 2 moving objects is considered. Specifically, $\mathtt{Q}_1$ such that $\mathtt{M}_1 = \tilde{\mathtt{M}}\mathtt{Q}_1$ can be obtained by solving Equation (4.11). If there is a way to "extract" $\mathtt{M}_1$ from $\tilde{\mathtt{M}}$, then it should be possible to apply the same procedure in order to obtain $\mathtt{M}_2$ (and so on recursively in case of more motions) without the issue that the non-linear optimisation used for Equation (4.11) may give back always $\mathtt{M}_1$ (as $\mathtt{M}_1$ has been "deleted").

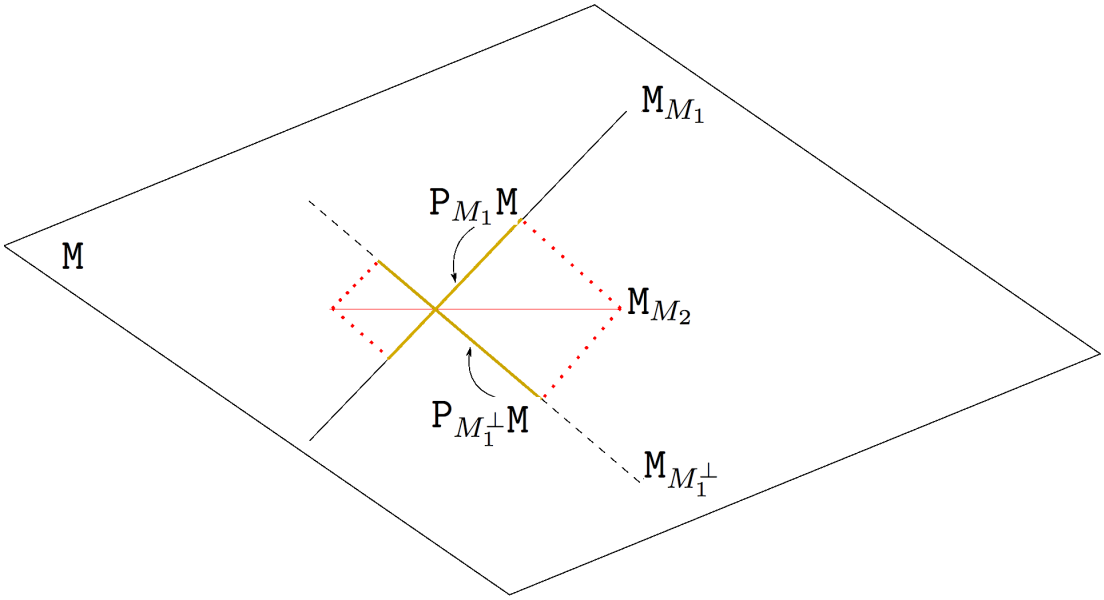


Figure 4.4: An example of two non orthogonal subspaces $\mathtt{M}_{M_1}$ and $\mathtt{M}_{M_2}$. When $\mathtt{M}_{M_2}$ is projected onto $\mathtt{M}_{M_1^\perp}$ it "looses" some of its components. Specifically, during the projection $\mathtt{P}_{M_1}\mathtt{M}$ is lost. However, $\mathtt{P}_{M_1}\mathtt{M}$ does not contains only $\mathtt{M}_{M_1}$ but also part of $\mathtt{M}_{M_2}$.

Let us define the orthogonal projector operator $\mathtt{P}_{M_1} : \tilde{\mathtt{M}} \rightarrow \tilde{\mathtt{M}}_{M_1}$ [153] as:

$$\mathtt{P}_{M_1} = \mathtt{M}_1(\mathtt{M}_1{}^T\mathtt{M}_1)^{-1}\mathtt{M}_1{}^T, \tag{4.12}$$

where $\tilde{\mathtt{M}}_{M_1}$ represents motion 1 within the subspace generated by $\tilde{\mathtt{M}}$. Therefore, the orthogonal complement of $\mathtt{P}_{M_1}$ [153] is another projector that projects from the subspace $\tilde{\mathtt{M}}$ to the subspace $\tilde{\mathtt{M}}_{M_1^\perp}$ that should contain all of the remaining motions (in this example the motion 2) except motion 1:

$$\mathtt{P}_{M_1^\perp} = \mathtt{I} - \mathtt{P}_{M_1}. \tag{4.13}$$

If the subspaces spanned by the columns of $\mathtt{M}_1$ and $\mathtt{M}_2$ (let us call them $\mathtt{M}_{M_1}$ and $\mathtt{M}_{M_2}$) were orthogonal, then the non-linear least-square optimisation could be applied directly to $\tilde{\mathtt{M}}_{M_1^\perp}$ in order to get $\mathtt{M}_2$. In fact, in such a space the components of $\mathtt{M}_1$ are completely suppressed.

Unfortunately, the subspaces are rarely completely orthogonal hence the components of $\mathtt{M}_2$ are spread between $\tilde{\mathtt{M}}_{M_1^\perp}$ and $\tilde{\mathtt{M}}_{M_1}$ as shown in Figure 4.4.

In order to reconstruct correctly $\mathtt{M}_2$ it is not sufficient to consider $\mathtt{M}_{M_1^\perp}$, instead it is

necessary to take into account also the subspace $\mathtt{M}_{M_1}$. Given this, motion 2 has to be written as a composition between the components projected onto $\mathtt{M}_{M_1^\perp}$ and onto $\mathtt{M}_{M_1}$:

$$\mathtt{M}_2 = \overbrace{\tilde{\mathtt{M}}_{M_1}\mathtt{Q}_{2M_1}}^{\text{how much of } M_2 \text{ is on } M_1} + \underbrace{\tilde{\mathtt{M}}_{M_1^\perp}\mathtt{Q}_{2M_1^\perp}}_{\text{how much of } M_2 \text{ is on } M_1^\perp} \tag{4.14}$$

Note that the ambiguity is not yet solved. In fact, while $\mathtt{Q}_{2M_1^\perp}$ is now completely unrelated to the first motion, $\mathtt{Q}_{2M_1}$ is operating within the subspace of $\mathtt{M}_1$. This means that there is a matrix $\mathtt{Q}_{1M_1}$ that can recover $\mathtt{M}_1$ from $\tilde{\mathtt{M}}_{M_1}$. Hence, it is still necessary to avoid the solution $\mathtt{Q}_{2M_1} = \mathtt{Q}_{1M_1}$ that would give $\mathtt{M}_1 = \tilde{\mathtt{M}}_{M_1}\mathtt{Q}_{2M_1}$. The problem can be written as:

$$\mathtt{M}_2 = \tilde{\mathtt{M}}_{M_1}\mathtt{Q}_{2M_1} + \tilde{\mathtt{M}}_{M_1^\perp}\mathtt{Q}_{2M_1^\perp} \tag{4.15}$$

subject to

$$\mathtt{M}_{f2}\mathtt{M}_{f2}^T = \mathtt{I}_2 \quad \forall f = 1, \ldots, F. \tag{4.16}$$

$$\mathtt{Q}_{2M_1}{\mathtt{Q}_{2M_1^\perp}}^T \neq \mathtt{0} \tag{4.17}$$

The constraint of Equation (4.16) guarantees a metric solution, while the constraint of Equation (4.17) guarantees that the two matrices are not zero. This automatically avoids the solution $\mathtt{Q}_{2M_1} = \mathtt{Q}_{1M_1}$. In fact, note that if $\mathtt{Q}_{2M_1} = \mathtt{Q}_{1M_1}$ then $\mathtt{M}_1 = \tilde{\mathtt{M}}_{M_1}\mathtt{Q}_{2M_1}$, hence, in order to satisfy the constraint of Equation (4.16): $\mathtt{M}_1 = \tilde{\mathtt{M}}_{M_1}\mathtt{Q}_{2M_1} + \tilde{\mathtt{M}}_{M_1^\perp}\mathtt{0}$. What is obtained with this double projection is to constrain the issue only to the components of $\mathtt{M}_2$ that are shared with $\mathtt{M}_1$, nevertheless the problem is still open.

The projector idea would work in an ideal case with perfectly orthogonal subspaces, but in real situations it does not solve the problem. One additional constraint, which would lead to a simultaneous solution of the motion segmentation and the SfM problems, is still missing, and at present it remains an open issue. In the next section a different approach is presented. The idea is to compensate for the missing constraint by relaxing the problem. Hence, instead of starting from a completely unknown segmentation, an initialisation provided by a motion segmentation algorithm can be adopted. The initial

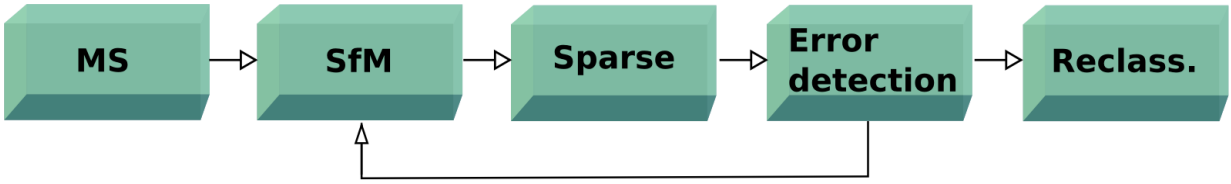


Figure 4.5: Summary of the JESS algorithm.

segmentation will be corrected by forcing *metric* and *sparse* constraints, moreover, the 3D reconstruction of the objects will be provided.

4.5 The JESS algorithm

In the previous chapters it was shown that, at present, state of the art motion segmentation algorithms provide very good results with very low misclassification rates (but yet not perfectly correct). Furthermore, in the previous sections of this chapter two constraints that have never been used in motion segmentation were highlighted. In this section, an optimisation procedure able to correct an initial segmentation is presented. The flow of the proposed algorithm is shown in Figure 4.5. Such a procedure starts from an initial solution to the multi-body motion segmentation problem (MS block in Figure 4.5), then exploits the constraints of multi-body SfM (SfM block in Figure 4.5) to iteratively correct the segmentation (Error detection block in Figure 4.5). The key idea is that misclassified points tend to disobey the motion model to which they are assigned and, therefore, contribute to the final reprojection error. The question is how to define an approach that can select the points that contribute the most to the error and remove them. Optionally, the algorithm could then be extended so that, once the segmentation is corrected, the removed points may be assigned to the proper group or discarded in the case that they are outliers (Reclass. block in Figure 4.5).

In the next section, two constraints that appear in the multi-body SfM problem are presented: metric constraints and sparsity of the shape matrix $\mathtt{S}$. After enforcing these constraints, the algorithm detects wrongly classified points and reclassifies them into the

correct motion group.

4.5.1 Multi-body metric constraints

Each motion is subject to the respective constraints given by the chosen camera model, as already shown in Equation (4.1). Specifically, the matrix $\mathtt{M}$ cannot assume arbitrary values but it lies on a particular *motion manifold* [147]. When missing data affect the measurements, this constraint can be used both to design specific optimisation algorithms, and to reduce the chance of computing solutions that correctly minimise the reprojection error but that lead to inaccurate 3D reconstructions [147]. In the context of this approach, a SfM problem has to be solved for each of the N registered measurement matrices $\bar{\mathtt{W}} = [\bar{\mathtt{W}}_1 | \dots | \bar{\mathtt{W}}_N]$, possibly with missing data. Note that this step, not only finds $\mathtt{S}$ and $\mathtt{M}$ by enforcing the metric constraints, but it also fills in the missing entries of $\bar{\mathtt{W}}$ by exploiting the motion description and the 3D shape of the moving objects. This is a new way of filling missing entries. In fact, other approaches, like [67], perform this step by removing rows and columns that correspond to trajectories with missing data from $\bar{\mathtt{W}}$ (and, therefore, they renounce to the information provided by these trajectories). Following this, they estimate the segmentation, and only at this point they fill the missing data. Even in the cases where there is no removal, like in [71, 128], only the 2D data is considered, while in [147] it is shown that missing data are filled more effectively using 3D information.

For this SfM step the Bilinear Augmented Lagrangian Multipliers (BALM) [154] method was used. BALM has the property of enforcing exact metric constraints and the ability to deal with a high ratio of missing data. Besides, it is a generic bilinear optimiser, thus it could be used also for non-rigid and articulated SfM.

Therefore, an initial segmentation result is used to divide the trajectory matrix in N trajectory matrices. For each trajectory matrix n the BALM algorithm is applied providing the motion matrix $\mathtt{M}_n$ and the structure matrix $\mathtt{S}_n$. At the end of this step the aggregate matrices described in Equation (4.4) can be created.

Clearly, as the initial segmentation may contain errors this result needs to be corrected.

The correction is performed by exploiting the sparse constraint of the aggregate shape matrix $\mathtt{S}$, as explained in the following section.

4.5.2 Sparsity of the matrix $\mathtt{S}$

The sparse pattern of the aggregate shape matrix can be used in order to estimate the matrix $\mathtt{S}$ which best satisfies such a constraint. In order to make this problem tractable, the ℓ_1 norm can be used as a surrogate for sparsity. In such terms, the optimisation problem becomes solving, for each point p in $\mathtt{S}$, a *basis pursuit denoising* problem [155]:

$$\min_{\bar{\mathbf{s}}_p} \frac{1}{2}||\bar{\mathbf{w}}_p - \mathtt{M}\ \bar{\mathbf{s}}_p||_2^2 + \tau||\bar{\mathbf{s}}_p||_1, \tag{4.18}$$

where $\tau \in \mathbb{R}^+$ is a *regularisation parameter*, $\bar{\mathbf{w}}_p$ is the registered trajectory vector p, and the $3N \times P$ matrix $\bar{\mathbf{S}} = [\bar{\mathbf{s}}_1 \dots |\bar{\mathbf{s}}_P]$ represents the collection of non-homogeneous 3D coordinates. Accordingly, the matrix $\mathtt{M}$ of size $2F \times 3N$ is the motion matrix minus the translation vector at each frame. Each $3N$-vector $\bar{\mathbf{s}}_p$ contains $3(N-1)$ zeros, thus the image trajectory $\bar{\mathbf{w}}_p$ can be described by a (small) subset of the 3D points. The sparse optimisation is initialised so that $\bar{\mathbf{S}} = \mathtt{S}$, where $\mathtt{S}$ is the aggregate shape matrix obtained at the end of the previous SfM step. Note that the resulting $\bar{\mathbf{S}}$ from the sparse optimisation may not satisfy the 3D metric structure of the objects. However, the result of the minimisation can be used to detect points which do not comply with the previously estimated metric constraints.

For the sparse minimisation step the Sparse Reconstruction by Separable Approximation (SpaRSA) algorithm [156] was used. SpaRSA is able to solve large-scale optimisation problems efficiently and it requires only the tuning of the regularisation term τ (Equation (4.18)). The parameter τ was empirically tuned on 50% of the synthetic database. During this tuning process a correlation between the value of τ and the number of motions in the sequence was noted. Specifically, an increase in the number of motions requires an increase in values of τ. Such a correlation can be explained by the fact that in the presence of many motions the overall reprojection error tends to be higher, therefore, the

sparse contribution has to be weighted accordingly. In all of the experiments the following τ values were used: for 2 motions $\tau = 0.9$, for 3 motions $\tau = 1.6$, for 4 motions $\tau = 2.5$ and for 5 motions $\tau = 3.2$.

At the end of this step it is possible to define two different reconstructed trajectory matrices, the one obtained after imposing the metric constraints:

$$\tilde{\mathtt{W}} = [\mathtt{M}_1\mathtt{S}_1|\dots|\mathtt{M}_N\mathtt{S}_N], \tag{4.19}$$

and the one obtained by optimising the reprojection error while imposing the sparsity constraint:

$$\hat{\mathtt{W}} = [\mathtt{M}_1\bar{\mathtt{S}}_1|\dots|\mathtt{M}_N\bar{\mathtt{S}}_N]. \tag{4.20}$$

4.5.3 Identifying candidate errors

In order to identify the candidate errors the trajectory matrices $\tilde{\mathtt{W}}$, obtained after the SfM stage, and $\hat{\mathtt{W}}$, obtained after the sparse minimisation stage, are compared. Since SfM and sparse optimisation perform arbitrary normalisations on $\tilde{\mathtt{W}}$ and $\hat{\mathtt{W}}$, aligning the matrices so that the image centroids are at the respective origins, and the mean distance of all of the points from the origin is $\sqrt{2}$, is a necessary step.

After registration, the 2D distance between $\tilde{\mathtt{W}}$ and $\hat{\mathtt{W}}$, for each point p and for each frame f is computed. Two measures to identify the candidates are used: a) the point p_a with the highest 2D reprojection difference for any of the F frames, b) the point p_b with the highest mean 2D reprojection difference over all the F frames. Therefore, at each iteration the points p_a and p_b are removed from $\bar{\mathtt{W}}$.

4.5.4 Stop condition

Once the candidate errors are removed the algorithm can iterate again from the beginning until a stop condition is verified. Note that if JESS is applied to an algorithm that is known to have an average misclassification rate of $x\%$ a valid stop condition would be to let JESS

iterate for $\mathcal{M} = Px\%$ (P being the total number of points, hence, $\mathcal{M}$ is the expected number of misclassified points). If one wants to be cautious more than one attempt could be given for any expected error. Therefore, the number of iterations would become $\mathcal{AM}$, where $\mathcal{A}$ is the number of attempts for each error. Note also that, if desired, the last step of the algorithm can reclassify all the removed points that were considered candidate errors, so that no information is lost.

Nevertheless, JESS is provided also with another stop condition for those cases when there is no prior information about the expected misclassification rate. The most intuitive condition is to use the reprojection error of Equation (4.6): when the error decreases below a threshold, then the algorithm should stop. However, tests showed that such an error could have a non increasing behaviour when the number of errors in the segmentation increases. On the other hand, it was also noted that when the segmentation is correct the reprojection error tends to become stable. Hence, one useful condition is that the difference in the reprojection error between one iteration and the following has to be less than a threshold for a fixed amount of iterations. Specifically, it was empirically chosen that the reprojection error has to be smaller than 5×10^{-7} for at least 3 consecutive JESS iterations. Moreover, this first condition was associated with a second that is: the 2D reprojection difference of the candidate point p_a (or p_b) has to be smaller than 0.5. When both conditions are satisfied the algorithm can terminate. The conditions are purposely very strict as it is better to perform more iterations and remove as many errors as possible rather than to stop the algorithm too early and leave some errors in the segmentation. However, to avoid the algorithm running indefinitely a maximum number of iterations can also be defined.

4.5.5 Reclassification

Once the stop condition is satisfied all of the removed points can be reclassified to the correct motion. For this task, the NSI algorithm [58] is used. NSI was chosen because it provides a fast and correct measure of similarity. Moreover, such a measure can also be

used to detect outliers which, by definition, do not belong to any of the motions.

A summary of JESS is shown in Algorithm 4.1. Starting from an initial motion segmentation solution, the main building blocks of JESS are the computation of SfM with missing data, which enforces the metric constraints, the sparse minimisation, which detects candidate errors, and the reclassification step, which enables the reassignment of the detected misclassified points to the correct motion.

Algorithm 4.1 JESS

1: Compute an initial mot. segm., arrange $\mathtt{W}$ as in Equation (4.3) and build $\bar{\mathtt{W}}$.
2: **repeat**
3: $\forall$ motion $n = 1, \ldots, N$ compute SfM: $\tilde{\mathtt{W}}_n = \mathtt{M}_n\mathtt{S}_n$.
4: Perform sparse minimisation, Equation (4.18), and obtain $\bar{\mathtt{S}}$.
5: Compute $\tilde{\mathtt{W}}$, Equation (4.19), and $\hat{\mathtt{W}}$, Equation (4.20), and register them.
6: $\forall$ points $p = 1, \ldots, P$ and $\forall$ frames $f = 1, \ldots, F$ compute 2D distance $Dist(p, f)$ between $\tilde{\mathtt{W}}$ and $\hat{\mathtt{W}}$.
7: Find $p_a = \max_p(Dist(f, p))\ \forall f = 1 \ldots F$.
8: Find $p_b = \max_p(\sum_{f=1}^{F} Dist(f, p)/F)$.
9: Remove p_a and p_b (if $p_b \neq p_a$) from $\bar{\mathtt{W}}$.
10: **until** stop condition satisfied
11: Reclassify removed points

4.6 Experiments

Different features of the algorithm were evaluated: the validity of JESS independently from the stop condition, the results using the stop condition, the ability of the algorithm to deal with missing data (using the stop condition), performances of the reclassification strategy and quality of the final 3D reconstruction. For simplicity, whenever JESS is stopped before the reclassification strategy it will be called JESS-R, whereas the name JESS will be referred to the complete algorithm (including the reclassification step). Experiments were performed on four different datasets: the synthetic and the Hopkins155 databases, already described in Section 2.5, a new database composed of 12 sequences published in [103] by the same authors of the Hopkins155 database (for simplicity these additional set is called Hopkins12 database), and a new sequence composed of two known videos used in SfM.

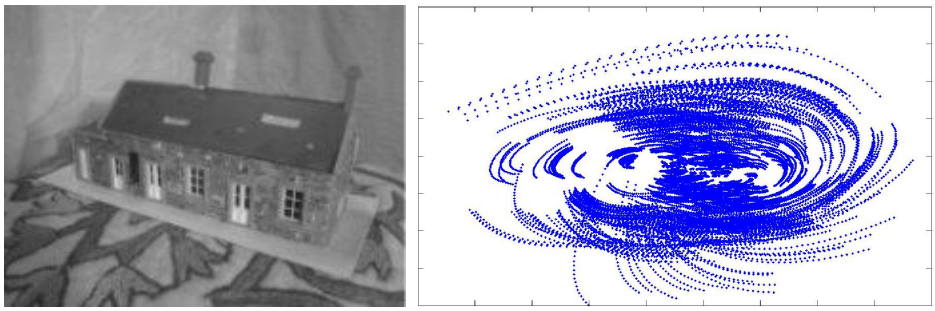
The databases are now described briefly as a reminder. The proposed synthetic sequences contain a set of moving cubes, with 56 tracked features each, that randomly rotate and translate. The database includes different sequences of 50 frames each with a varying number of independent motions and amount of noise. Specifically, 10 randomly generated motions were tested with 2, 3, 4 and 5 independently moving objects (cubes) giving a total of 40 sequences. Gaussian noise with standard deviation of 0, 0.5, 1, 1.5 and 2.0 pixels was added to each sequence (for a total of 200 sequences).

The Hopkins155 database contains 104 checkerboard sequences, 38 traffic sequences and 13 other sequences (among which are sequences with articulated motions). As JESS assumes independence of the motions, the Hopkins155 database is a very challenging test.

The Hopkins12 is a similar database with 12 checkerboard sequences in the presence of missing data due to occlusions. The database contains 3 sequences with 3 motions and 9 sequences with 2 motions. The average number of feature points is 418, while the average number of frames is 35. The average amount of missing data is 8.33%.

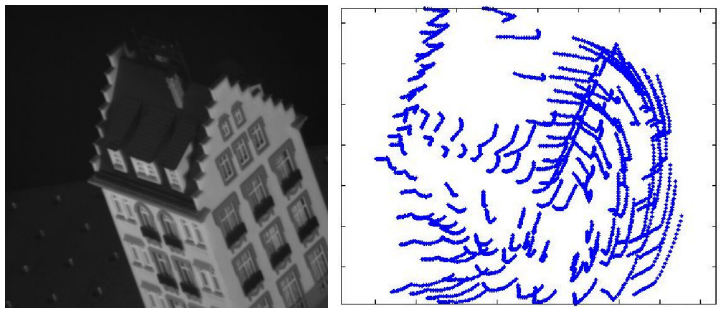
Different tests were performed with an increasing numbers of misclassified points (randomly selected). In order to check algorithmic convergence the simplest case with only 1 misclassified point per sequence was tested. Further tests were performed with higher numbers of misclassified points: 1%, 2%, 3%, 4%, 5% and 10% of the total number of points in each sequence. In the previous chapter it was shown that current motion segmentation algorithms have an average misclassification rate smaller than 10% so these tests go beyond the expected real scenario. Moreover, in order to simulate occlusions in one of the tests 10% of the data of each sequence was randomly removed.

The Hopkins155 and Hopkins12 databases are well known benchmarks among the MS community, however, as JESS involves SfM constraints these databases are not the best test sets. In fact not all of the sequences contain enough motion to satisfy SfM requirements. For this reason JESS was also tested on a sequence, called House and Hotel, obtained by merging two known sequences, used in SfM, in one single video with two moving objects. One frame of the House sequence and its trajectories are shown in Figure 4.6, while one



(a) First frame (b) Trajectories

Figure 4.6: Example of the House sequence.



(a) First frame (b) Trajectories

Figure 4.7: Example of the Hotel sequence.

frame of the Hotel sequence and its trajectories are shown in Figure 4.7. The two sets of trajectories were unified in a unique sequence and translated apart so that the whole set of points could be seen as a new real sequence that contains two moving objects. The new sequence is composed of 30 frames, the House object contains 672 points while the Hotel object contains 133 points for a total of 805 trajectories.

4.6.1 Fixed number of iterations

The first set of experiments evaluated the ability of JESS-R to converge to the correct segmentation. Accordingly, the algorithm was allowed to run for a fixed number of iterations without imposing a stop condition. The number of iterations was $3 \times \mathcal{M}$, where $\mathcal{M}$ was the number of initially misclassified points.

In Figure 4.8 it is possible to observe the percentage of identified errors (i.e. how many

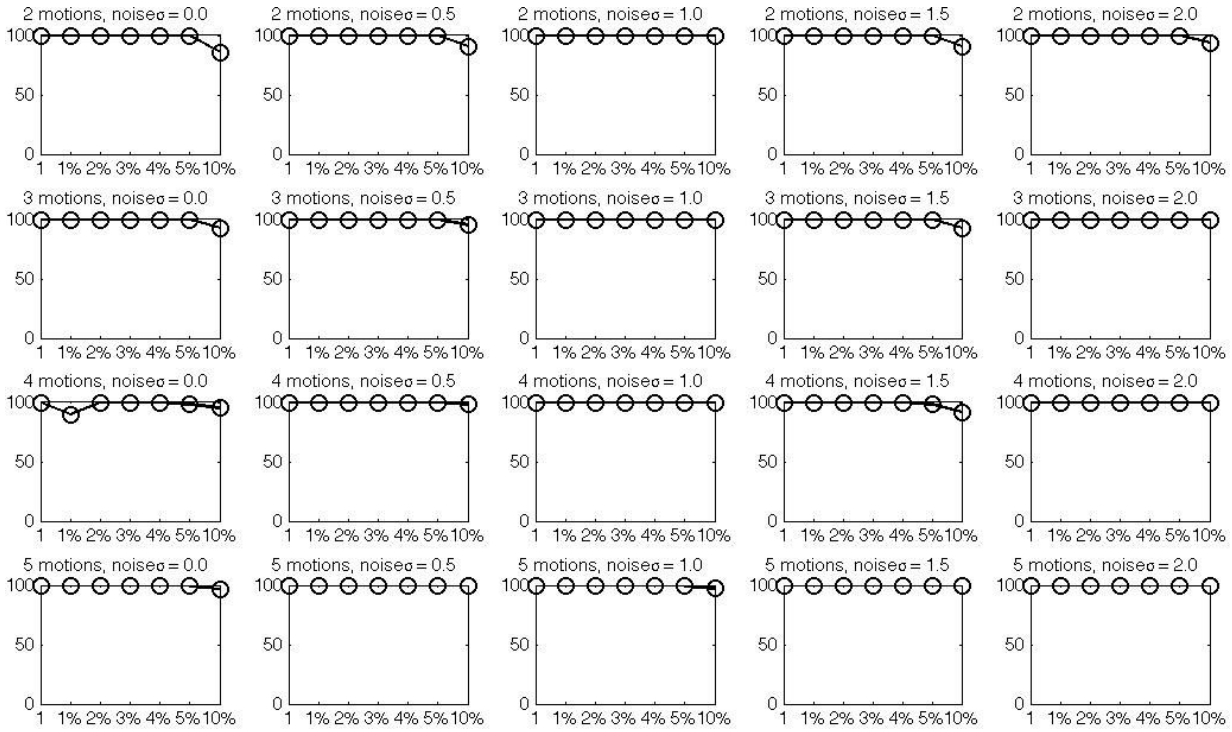


Figure 4.8: Average results of JESS-R with a fixed amount of iterations applied to the synthetic database that contains different numbers of motion (from 2 to 5). On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors is shown. ◯ is the percentage of the removed misclassified points over all the misclassified points.

of the initially misclassified points were identified as a percentage of all of the original misclassified points, a line with ◯ symbol) for each different number of motions and level of noise. The results show a very robust behaviour of JESS-R against different numbers of motion. JESS-R is robust also in the presence of an increasing amount of noise and initial errors: the percentage of error detection is 100% for almost all of the cases. Also, it is worth to stress the importance of the tests with only 1 misclassified point, in fact these tests showed that even in the presence of only 1 error, JESS-R was always able to detect it (among all of the $56 \times N$ points of each sequence) within only 3 iterations, proving the validity of the algorithm.

The same experiment was repeated on the real sequences of the Hopkins155 database where initial misclassified points were randomly selected. Averaged results are shown in Figure 4.9. Similarly to the results on the synthetic database, the real test demonstrates

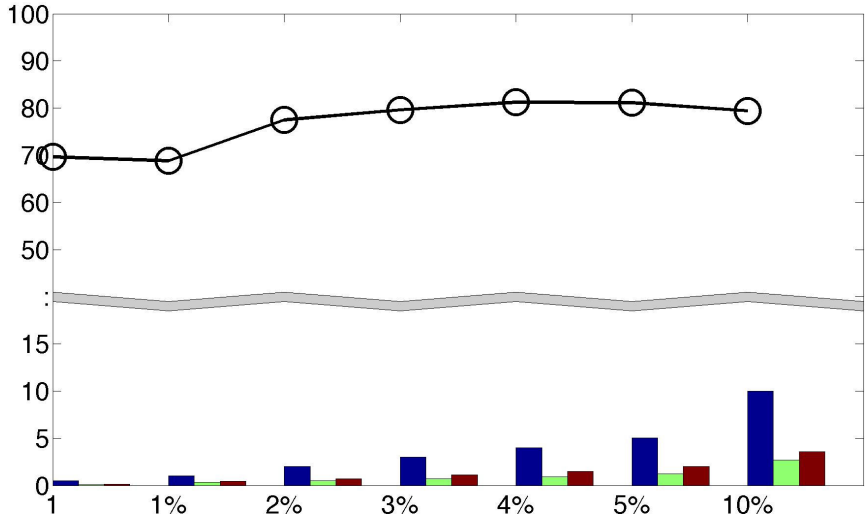


Figure 4.9: Average results of JESS with fixed number of iterations applied to the Hopkins155 database. On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors over all the misclassified points is shown (JESS-R). On the bottom of the plots the bars show from left to right and for each percentage of misclassified points: initial misclassification, misclassification after removal of errors by JESS-R, misclassification after removal and reclassification by JESS.

the stability of the behaviour of JESS-R against the initial misclassification rate: the percentage of error detection is stable between 70 and 80%.

This first set of tests shows that JESS-R is able to greatly reduce the misclassification rate of the input sequences. On the synthetic database correction is almost perfect while on the challenging sequences of the Hopkins155 database any of the initial misclassification rates is reduced, at least, by 70%.

4.6.2 Stop condition

The aim of the second set of experiments was to verify the proposed stop condition (Section 4.5.4). The maximum number of iterations was set to be $(x+3)\%$ of the points of the sequence, x being the percentage of initial misclassification.

Results on the synthetic database are shown in Figure 4.10. The detection of the errors is very similar to the case with a fix amount of iterations. In this set of experiments also the amount of false positives, as a percentage of all of the points of the sequence, are

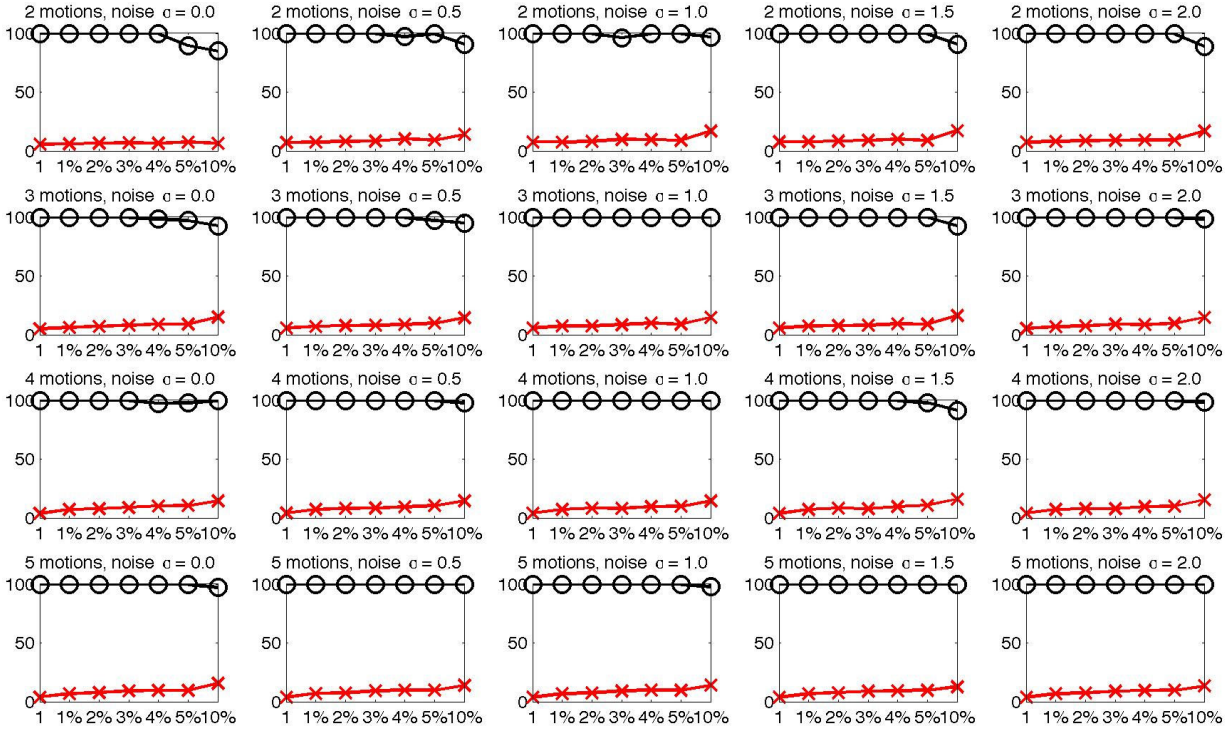


Figure 4.10: Average results of JESS-R with stop condition applied to the synthetic database that contains different numbers of motion (from 2 to 5). On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors is shown. ◯ is the percentage of the removed misclassified points over all the misclassified points. False positives as a percentage of the total amount of points of each sequence are shown in a red ×.

reported (a line with × symbol). False positives are points that were removed by JESS-R even if they were correctly classified. As shown in Figure 4.10 the amount of false positives ranges between 5% to 15%. An almost perfect error detection behaviour testifies that the condition is not usually satisfied until all of the errors have been removed. On the other hand, a small amount of false positives indicates that the algorithm terminates the loop not too long after detection of the last error. All of the removed points, including the false positives, will be reassigned to the correct motion by the reclassification strategy.

Results on the Hopkins155 database are shown in Figure 4.11. The results are very stable and show only a slight decreasing trend when the misclassification rate increases. This may suggest that the stop condition may be too strict in the presence of high misclassification rates and not completely independent motions. Nevertheless, even when a tuning

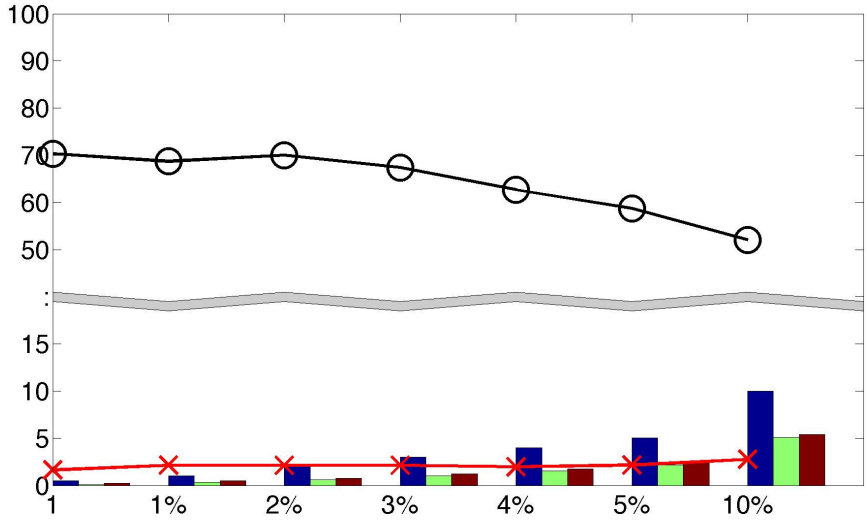


Figure 4.11: Average results of JESS with stop condition applied to the Hopkins155 database. On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors over all the misclassified points is shown (JESS-R). False positives as a percentage of the total amount of points of each sequence are shown in a red $\times$. On the bottom of the plots the bars show from left to right and for each percentage of misclassified points: initial misclassification, misclassification after removal of errors by JESS-R, misclassification after removal and reclassification by JESS.

process for the stop condition is avoided, JESS-R is able to reduce the initial misclassification rate by 70%, with 1% up to 3% of initial error rate. With a higher amount of initially misclassified points, the error detection is reduced of about 60% for misclassification rates of 4% and 5%, and to above 50% with a misclassification rate of 10%. Similarly to the results on the synthetic database, the amount of false positives is stable, in the case of the Hopkins155 database is around 4%.

4.6.3 Missing data

This set of experiments was performed on synthetic and real sequences with missing data in order to demonstrate the algorithm performance on such challenging cases. Tests on the synthetic and the Hopkins155 databases were performed with 10% of missing data (randomly selected) and using the proposed stop condition. Figure 4.12 shows the results of JESS-R on the synthetic database. The behaviour of JESS-R is very robust and the percentage of error detection is almost always 100% and never less than 80%. On the

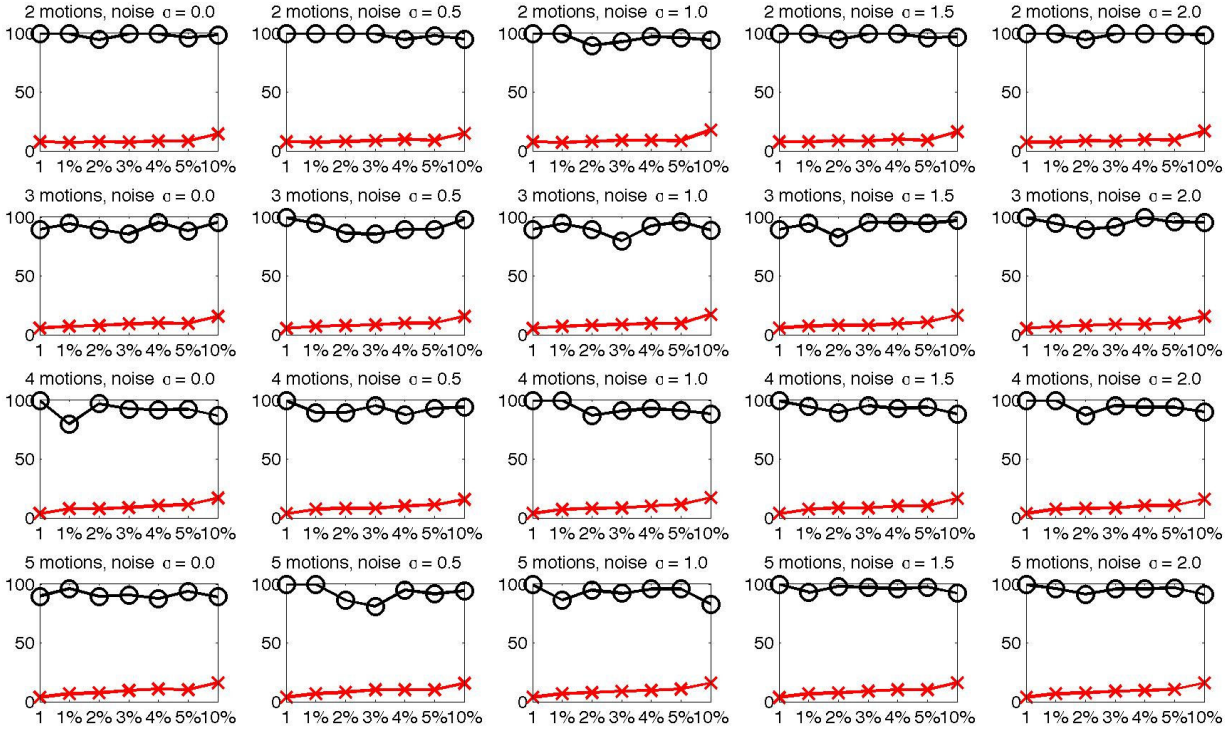


Figure 4.12: Average results of JESS-R with stop condition and 10% of missing data applied to the synthetic database that contains different numbers of motion (from 2 to 5). On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors is shown. ◯ is the percentage of the removed misclassified points over all the misclassified points. False positives as a percentage of the total amount of points of each sequence are shown in a red ×.

Hopkins155 database JESS-R is able to keep an error detection rate of approximately 60% for all of the amounts of initial misclassification tested, as shown in Figure 4.13.

A further test on real sequences was performed on the Hopkins12 database. In this case the missing data were not simulated but were due to occlusions, while the misclassified points were given by the errors of the GPCA [103] and SSC [67] segmentation algorithms[1]. The results are summarised in Table 4.1. Amount of missing data varied from 0.96% to 22.20% of the total points, with an average rate of 8.33%, while initial misclassification was between 0% and 48.6%. In this test the imposed maximum number of iterations (in case the stop condition is not satisfied before) was equal to 10% of the points of the sequence (clearly much less than the initial misclassification of some of the sequences). Considering

[1]GPCA and SSC implementations available at `vision.jhu.edu`; SSC parameters used were: $\tau = 0.01$, subspace size equal to 4, cluster step performed by Random Walks [157].

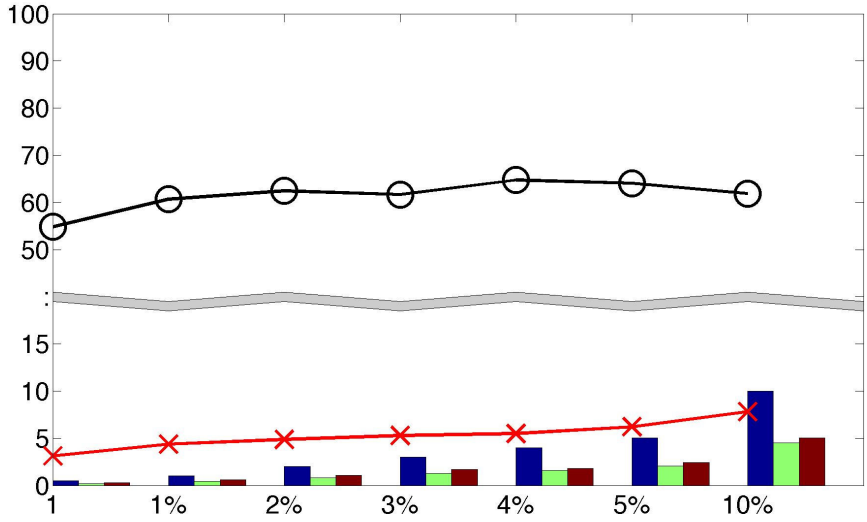


Figure 4.13: Average results of JESS with stop condition and 10% of missing data applied to the Hopkins155 database. On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors over all the misclassified points is shown (JESS-R). False positives as a percentage of the total amount of points of each sequence are shown in a red $\times$. On the bottom of the plots the bars show from left to right and for each percentage of misclassified points: initial misclassification, misclassification after removal of errors by JESS-R, misclassification after removal and reclassification by JESS.

all of the sequences with an initial error rate not higher than 10%, JESS-R is able to detect the errors and decrease the initial misclassification in the majority of the cases.

When the initial misclassification level is above 10% results are not significant as JESS-R assumes an almost correct initial segmentation, therefore, those cases are not considered in the following discussion. The misclassification rate of JESS-R applied to GPCA did not improve the initial rate (excluding of course the cases when the initial misclassification was already 0%) only in three sequences (*oc1R2RC_g23* whose misclassification remained constant, *oc2R3RCRT_g13* whose misclassification became worse by 0.32%, and *oc2R3RCRT_g23* whose misclassification became worse by 1.07%). In all of the three cases this is due to the fact that the stop condition was prematurely verified. Similarly, when JESS-R is applied to the results of SSC, only in one case JESS-R did not improve the initial segmentation (*oc1R2RC_g13*, whose misclassification became worse by 0.01%).

Overall, the results of these tests showed that JESS-R can deal successfully also with

Name	MD	GPCA	**JESS-R**	**JESS**	SSC	**JESS-R**	**JESS**
oc1R2RC	4.8%	34.30%	**35.76%**	**34.45%**	0.46%	**0.00%**	**0.00%**
oc1R2RCT	4.5%	12.36%	**9.67%**	**11.82%**	2.36%	**0.22%**	**1.27%**
oc1R2RCT_g12	10.1%	4.33%	**0.00%**	**2.16%**	0.00%	**0.00%**	**0.00%**
oc1R2RCT_g13	5.2%	3.52%	**1.02%**	**1.64%**	3.05%	**1.28%**	**1.41%**
oc1R2RCT_g23	1.0%	3.16%	**0.00%**	**0.00%**	0.00%	**0.00%**	**0.23%**
oc1R2RC_g12	10.0%	0.00%	**0.00%**	**0.00%**	0.00%	**0.00%**	**0.00%**
oc1R2RC_g13	6.0%	2.24%	**0.00%**	**0.00%**	0.41%	**0.42%**	**0.41%**
oc1R2RC_g23	0.6%	0.78%	**0.78%**	**0.78%**	0.00%	**0.00%**	**0.00%**
oc2R3RCRT	13.1%	38.33%	**35.71%**	**33.62%**	48.61%	**50.00%**	**49.68%**
oc2R3RCRT_g12	22.2%	4.94%	**4.29%**	**3.70%**	0.00%	**0.00%**	**0.00%**
oc2R3RCRT_g13	9.7%	5.37%	**5.69%**	**5.37%**	39.13%	**39.05%**	**38.62%**
oc2R3RCRT_g23	12.7%	9.97%	**11.04%**	**9.71%**	44.09%	**47.32%**	**41.73%**

Table 4.1: Average results of JESS applied on the results of the GPCA and SSC algorithms on the Hopkins12 database. MD: Missing Data; GPCA/SSC: misclassification of GPCA or SSC algorithm; JESS-R: misclassification of the JESS algorithm before reclassification; JESS: misclassification of the complete JESS algorithm.

sequences that contain missing data. Also when JESS-R was tested on the Hopkins12 and Hopkins155, which are composed by sequences that are not ideal for the application of SfM constraints, JESS-R was able to reduce the initial misclassification rate in most of the cases. Moreover, in those few cases where the misclassification was not improved, the error introduced by JESS-R was only marginal.

4.6.4 Reclassification strategy

All of the results discussed until here have concerned the detection and removal of segmentation errors. If it is required, once the segmentation has been improved, the removed points can be reintroduced using a reclassification strategy, as explained in Section 4.5.5. The reclassification strategy on the synthetic database with a fixed number of iterations shows a success rate, on average, of 99.99%. When the stop condition is used, and more points are removed, the success rate remains very high: 99.95%. The same result was confirmed also with the missing data, with a success rate of 99.97%.

Results of the reclassification strategy on the Hopkins155 database are shown on the bottom of the plots of Figures 4.9, 4.11 and 4.13. The first bar shows the initial misclassi-

fication while the second presents the misclassification after the removal of the points and the third gives the misclassification after the reclassification strategy. Often the misclassification before and after the reclassification remains the same (i.e. the reclassification works perfectly in most of the cases), only in a few occasions the misclassification after the reclassification is slightly increased. This small increment that happens in some cases also testifies that the SfM constraints used by JESS, and never exploited before, can solve some of the cases where rules (like NSI) used in classical motion segmentation algorithms would fail.

Overall, these results confirm that if the segmentation is mostly correct the reclassification strategy is able to reclassify the removed points (both errors and false positives) correctly. The same test was also applied in the case of missing data on the Hopkins12 database and the results are shown in the JESS column of Table 4.1. Even in this case, it is possible to appreciate that misclassification rates before and after the reclassification are very similar. Moreover, in none of the relevant cases JESS had a worse misclassification than GPCA and only in one case (*oc1R2RCT_23*) JESS had a worse misclassification than SSC by only 0.23%. In all of the remaining cases (when the initial misclassification was not already equal to 0%) the final misclassification is always equal or smaller than the one provided by the tested MS algorithms.

As far as the computational time is concerned, on the whole Hopkins155 database with 1 error per sequence JESS (including reclassification) required on average approximately 20 seconds per sequence (Matlab implementation on Quad-Core @ 2.4GHz, with 16 GB RAM). Note that the stage of the algorithm that was most time consuming was the sparse optimisation. This time could be shortened by adopting high performance implementations of sparse optimisation on Graphic Processing Units [158].

4.6.5 ASA and JESS

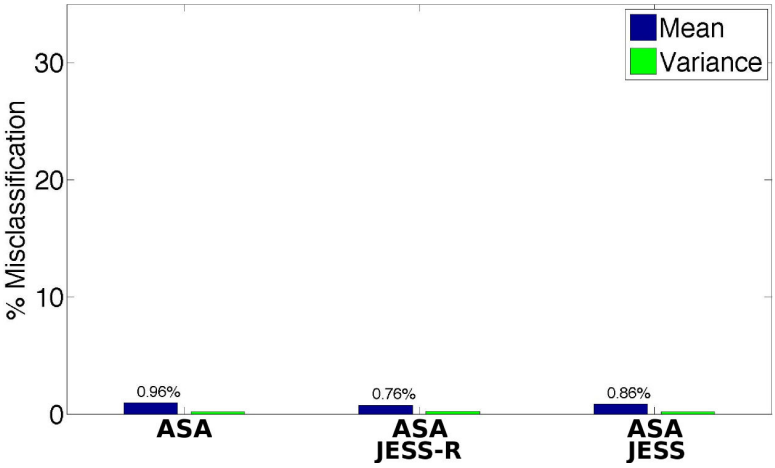
In this section the results presented in Section 3.3.5 of ASA applied to the Hopkins155 database are used as an input for JESS (using the stop condition with a maximum limit

of iterations equal to 10% of the total number of points per each sequence). The results are presented before the reclassification step (JESS-R), which is an optional step, and after the reclassification step (JESS).

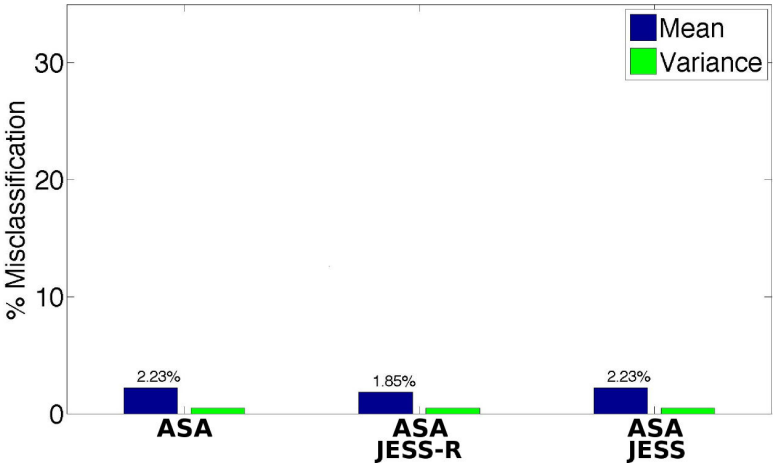
In Figure 4.14 the misclassification rate of ASA, JESS-R and JESS are shown. Note that JESS-R improves the performance of ASA both with 2 (from 0.96% to 0.76%) and 3 motions (from 2.23% to 1.85%). Also JESS improves the misclassification rate of ASA, however, some of the detected errors are wrongly reintroduced by the reclassification strategy, this leads to a slightly higher error rate than JESS-R. Nevertheless, the error rate of JESS is never worse than the one of ASA (from 0.96 to 0.86 with 2 motions, while for 3 motions the error remains constant).

In general the improvement may seem small (or absent in the case of SSC with 3 motions), however, the initial misclassification rates were already small, hence, it is not possible to note big improvements. Moreover, the aim of JESS-R/JESS is mainly to eliminate errors left in an already good segmentation. It is difficult to appreciate this result in terms of final average misclassification rate, because those few sequences that have a high misclassification rate, and therefore that contribute heavily to the final rate, do not satisfy the assumption of JESS-R/JESS. While the sequences that JESS-R/JESS can correct are those whose misclassification rate is small but not zero. Table 4.2 offers a detail of the misclassification rates. As already stated, the improvement is small, sometime there is no improvement in the misclassification rate. Particularly interesting is that the checkerboard sequences are those where the correction has been less effective, whereas articulated and traffic sequences show a better correction rate. The reason for this difference relies probably on the fact that checkerboard sequences contain objects that perform very small motions, while in the other two groups the motions performed are bigger, and therefore, the ability of the SfM step to impose camera constraints is more effective. This result gives also more significance to the previous results of JESS shown on the Hopkins155 database and on Hopkins12 database, which is composed exclusively by checkerboard sequences.

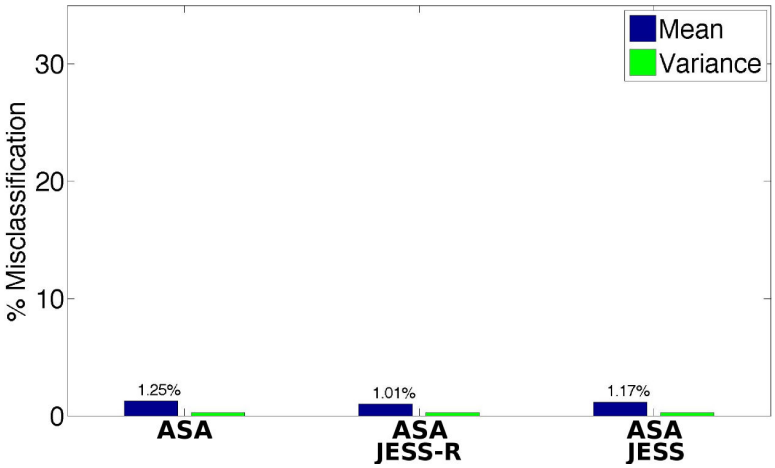
Another interesting analysis comes from the study of the histograms of the misclassifi-



(a) 2 Motions



(b) 3 Motions



(c) 2 and 3 Motions

Figure 4.14: Mean and variance misclassification rate of ASA before and after application of JESS.

2 Motions	Check.(78)		Artic.(11)		Traffic(31)		All(120)	
Method	%Avg	%Var	%Avg	%Var	%Avg	%Var	%Avg	%Var
ASA	1.00	0.32	1.75	0.10	0.57	0.01	0.96	0.22
ASA + JESS-R	**0.96**	**0.33**	**1.47**	**0.16**	**0.00**	**0.00**	**0.76**	**0.23**
ASA + JESS	**1.04**	**0.32**	**1.92**	**0.10**	**0.03**	**0.00**	**0.86**	**0.22**
3 Motions	Check.(26)		Artic.(2)		Traffic(7)		All(35)	
Method	%Avg	%Var	%Avg	%Var	%Avg	%Var	%Avg	%Var
ASA	2.41	0.65	3.72	0.28	1.11	0.03	2.23	0.49
ASA + JESS-R	**2.39**	**0.67**	**1.19**	**0.03**	**0.03**	**0.00**	**1.85**	**0.51**
ASA + JESS	**2.68**	**0.64**	**3.19**	**0.20**	**0.27**	**0.00**	**2.23**	**0.49**

Table 4.2: Misclassification rates on the Hopkins155 database of ASA with JESS-R (no reclassification) and JESS.

cation rates shown in Figure 4.15. From this histogram the effectiveness of JESS-R/JESS can be appreciated much more than from the overall misclassification rates. Note that JESS-R and JESS improved mainly the sequences with an initial rate below 5% (as expected), while for sequences with a higher initial rate the improvement was little. On the other hand, the number of sequences with no error at all increased from from 101 of ASA to 128 of JESS-R (then to 115 of JESS).

This final test showed that, under the assumption of an almost correct misclassification, JESS-R/JESS can successfully correct errors left by a MS algorithm. The reclassification strategy is able to reclassify correctly almost all of the removed points. However, in few cases the reclassification may fail. This proves that the constraints imposed by JESS are a key feature for improving the performances of classical MS algorithms. Moreover, note that when it is possible (i.e. when the remaining points after JESS-R removal are still sufficient for the desired task) the reclassification is not necessary. The reclassification strategy is also one of the steps where further investigation could lead to even better results. The final misclassification rate obtained combining ASA with JESS-R is one of the smallest in the state of the art of MS among techniques that do not require a tuning stage. More importantly, JESS produces one of the highest number of perfect segmentations (128 sequences over 156), which is essential for SfM to take place.

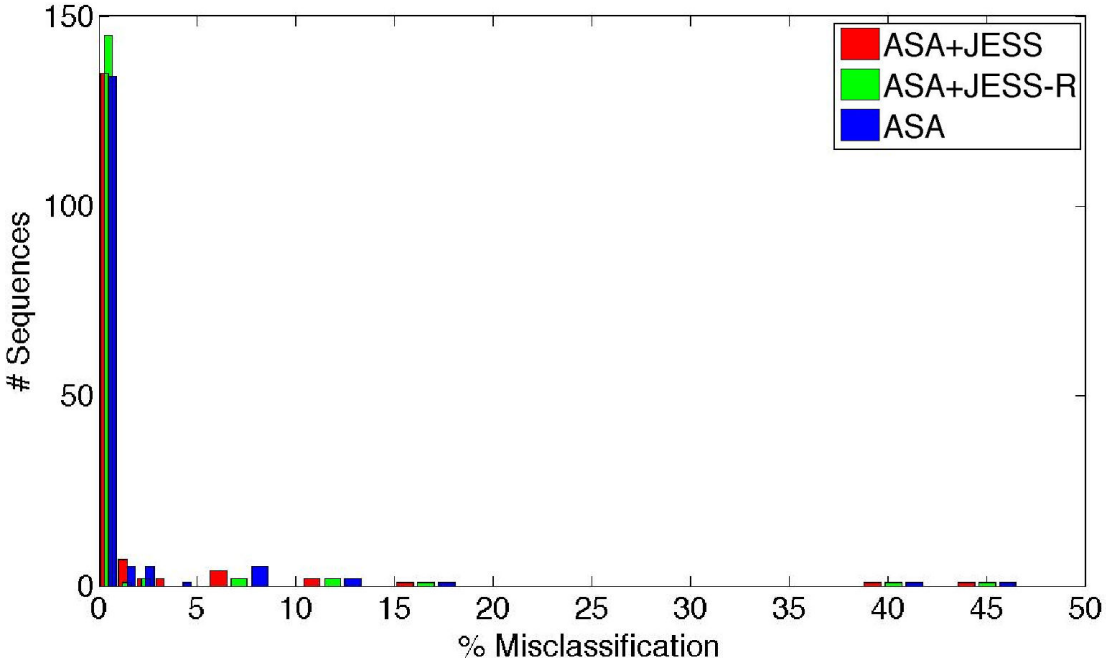


Figure 4.15: Histogram of the misclassification rate of ASA with JESS-R and JESS on the Hopkins155 database; misclassification rates from 0% to 5% are sub-sampled with bins of 1%, misclassification rates greater than 5% are sub-sampled with bins of 5%.

4.6.6 House and Hotel test

At the beginning of the Experiments section it was anticipated that the Hopkins155 and the Hopkins12 databases were not ideal test sets for JESS as they are designed for pure MS algorithms. As JESS merges MS with SfM, it is required that the tested sequences satisfy SfM constraints. The tests presented on GPCA, SSC and ASA showed this issue explicitly. In fact, the checkerboards sequences, which theoretically are among the easiest sequences in terms of MS, are those in which the improvement gained with JESS is smaller. Nevertheless, in the test presented in this section a case in which the motions are suitable for SfM reconstruction is presented. The aim of this test is to verify if the ability to correct the segmentation is more effective when SfM constraints are satisfied.

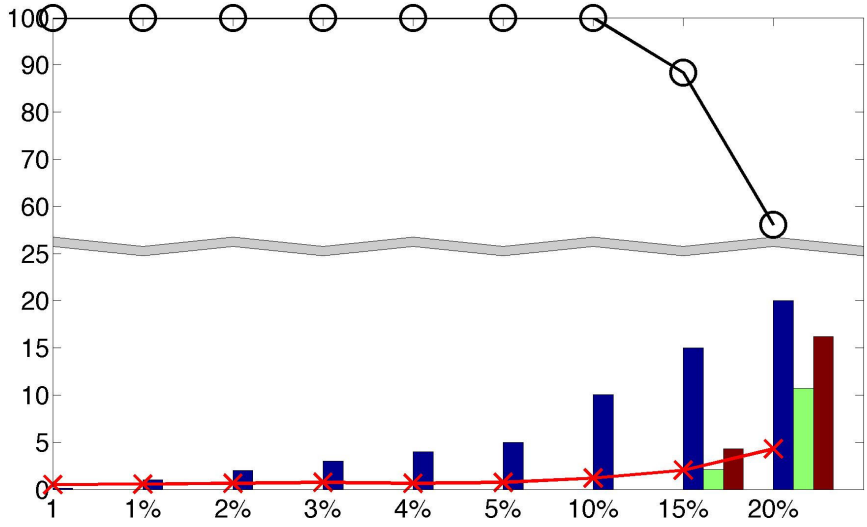
Different tests were performed on the House and Hotel dataset. First JESS-R and JESS were tested with a variable amount of initial misclassification (1 point, 1%, %2%, 3%, 4%, 5%, 10%, 15%, 20%) and no missing data. The results shown in Figure 4.16(a) are the

average results of 10 different runs; each run consisted in a different random selection of the misclassified points. Until a misclassification of 10% the error detection is perfect and the reclassification is able to reintroduce all of the removed points correctly. With 15% of initial misclassification the error detection becomes of 88.35% and therefore the misclassification rate goes from the initial 15% to 4.2% of JESS-R and finally to 8.6% of JESS. When the misclassification rate becomes of 20% around 56% of the errors are detected. These results are very important because they show that when the input sequence contains enough motion in order to impose correctly SfM constraints, then JESS becomes very effective. Moreover, it is shown that if the reclassification strategy can count on a perfect initial segmentation, then the removed points can be correctly reintroduced.

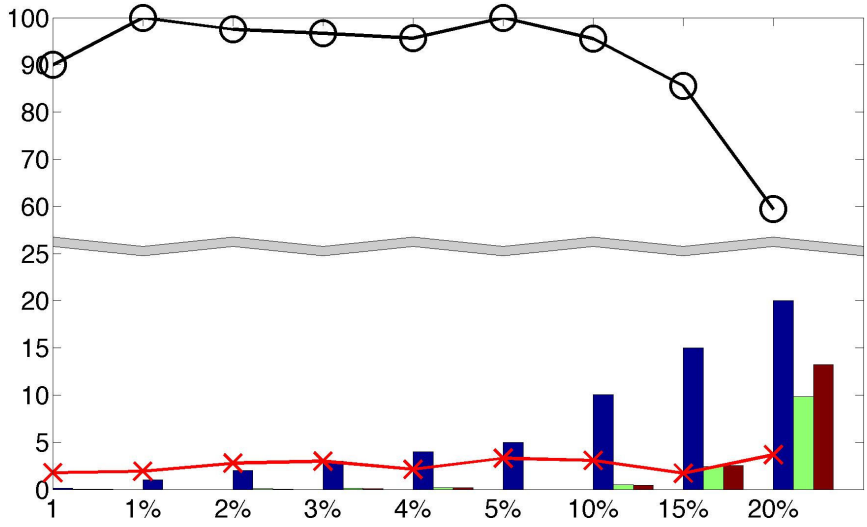
The second experiment was performed with 10% of missing data (randomly selected). Results are shown in Figure 4.16(b). As expected performances are slightly worse than the previous case, however, the trend of the error detection is still very similar.

In order to investigate further the ability to cope with missing data, another experiment was performed with an initial misclassification rate fixed to 10% and variable missing data from 0% to 40%. The average results of 10 runs are shown in Figure 4.17. In this test it is possible to appreciate the robustness of JESS with respect to missing data. In fact in all of the tests JESS is able to identify almost perfectly the 10% of initially misclassified points, remove them and reclassify them correctly.

Tests performed on the House and Hotel sequence confirmed the conclusions drawn from the analysis of the results on the Hopkins155 and Hopkins12. Specifically, JESS requires that the input sequence has enough motion so that SfM constraints can be imposed. When this happens JESS is able to remove the initially misclassified points very effectively even in presence of a high percentage of missing data. When the input sequence is not long enough, like in the Hopkins155 and the Hopkins12 database, JESS is still able to correct an initial misclassification, however, in this case an exceptional improvement cannot be expected.



(a) No missing data



(b) 10% of missing data

Figure 4.16: Average results of JESS with stop condition applied to the House and Hotel sequence. On the x-axis the initial amount of misclassified points is shown, on the y-axis the percentage of detected errors over all the misclassified points is shown (JESS-R). False positives as a percentage of the total amount of points of each sequence are shown in a red $\times$. On the bottom of the plots the bars show from left to right and for each percentage of misclassified points: initial misclassification, misclassification after removal of errors by JESS-R, misclassification after removal and reclassification by JESS.

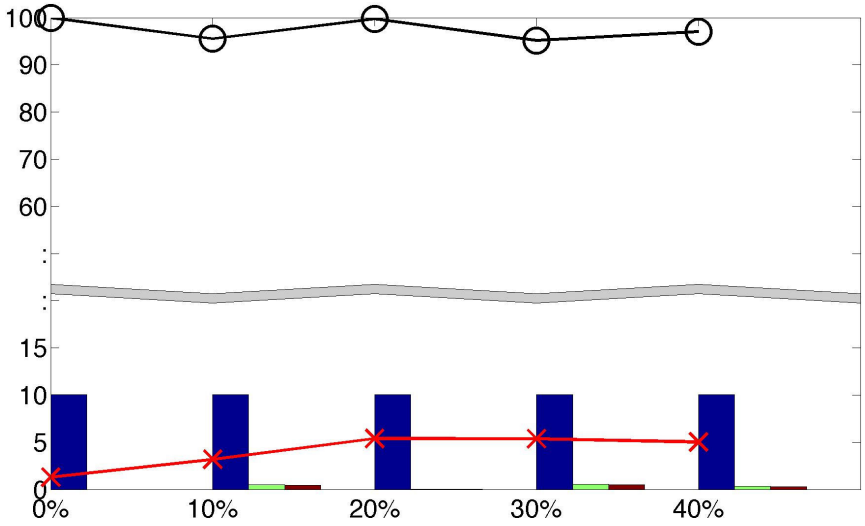


Figure 4.17: Average results of JESS with stop condition applied to the House and Hotel sequence with an initial amount of misclassification equal to 10%. On the x-axis the amount of missing data, points is shown, on the y-axis the percentage of detected errors over all the misclassified points is shown (JESS-R). False positives as a percentage of the total amount of points of each sequence are shown in a red $\times$. On the bottom of the plots the bars show from left to right and for each percentage of misclassified points: initial misclassification, misclassification after removal of errors by JESS-R, misclassification after removal and reclassification by JESS.

4.6.7 3D Reconstruction

An additional aspect left to evaluate is the quality of the 3D reconstruction with synthetic experiments (whose 3D ground truth is known). Therefore, the 3D Euclidean error between the 3D estimation of JESS and the ground truth shape was computed. Errors are always shown as a percentage of the ground truth depth size. Results on the synthetic database with different number of motions, a noise level of $\sigma = 1.0$ pixel, and 10% of missing data are shown. Each cube had an edge size of 1 unit in the metric space. Figure 4.18 shows on the first row the error detection rate, and on the second row the average 3D error ($+$) and the median error ($\square$) per point after performing the Procrustes analysis (which aligns the reconstructed 3D shape to the ground truth). With two motions, the mean error values remain very stable even when the amount of misclassification increases. With more motions, the mean error is slightly higher, however, the trends with 3, 4 and 5 motions remain similar. When the amount of misclassification rises the chances of leaving some

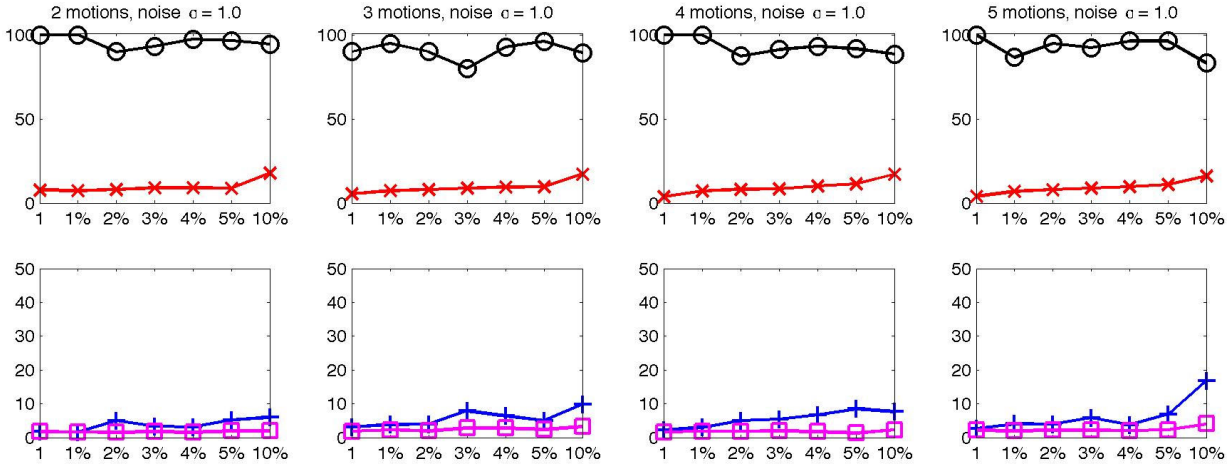
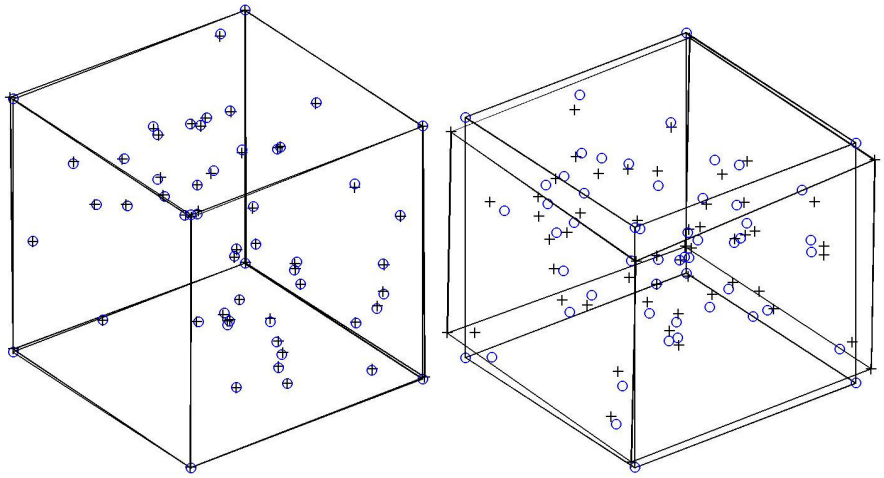


Figure 4.18: Average results of JESS with stop condition applied to the synthetic database with noise of $\sigma = 1.0$ and 10% of missing data per sequence. On the x-axis the initial amount of misclassified points is shown, on the y-axis a percentage value is shown. On the first row: ◯ is the percentage of the removed misclassified points over all the misclassified points.False positives as a percentage of the total amount of points of each sequence are shown in a red ×. On the second row: + is the average 3D error per point, while □ is the median 3D error per point. Errors shown as a percentage of the ground truth depth size.

errors in the segmentation becomes higher, consequently, the 3D reconstruction quality may be affected.

In Figure 4.19 two examples of 3D reconstructions compared to the ground truth are shown. Figure 4.19(a) presents the reconstruction of a cube with 8.96% of missing data and with a final segmentation completely corrected by JESS. It is possible to appreciate the accurate reconstruction with an average 3D error per point of only 0.80%. In Figure 4.19(b) a case with 9.32% of missing data, but with 1 error left in the segmentation, is presented. In this case it is possible to appreciate that the presence of even only 1 misclassified point greatly affects the reconstruction. The final average 3D error becomes of 7.34%.

Finally, the last test was performed on the House and Hotel sequence. In this case there is no ground truth of the 3D reconstruction, therefore, the reconstruction of the two buildings when there is no error in the segmentation is used as main reference. The 3D reconstruction of each of the two buildings is shown when the cloud of points of each object contained 10%, 5% and 0% of misclassified points. Only the points that belong to the correct object are then taken into account for the Procrustes analysis in order to

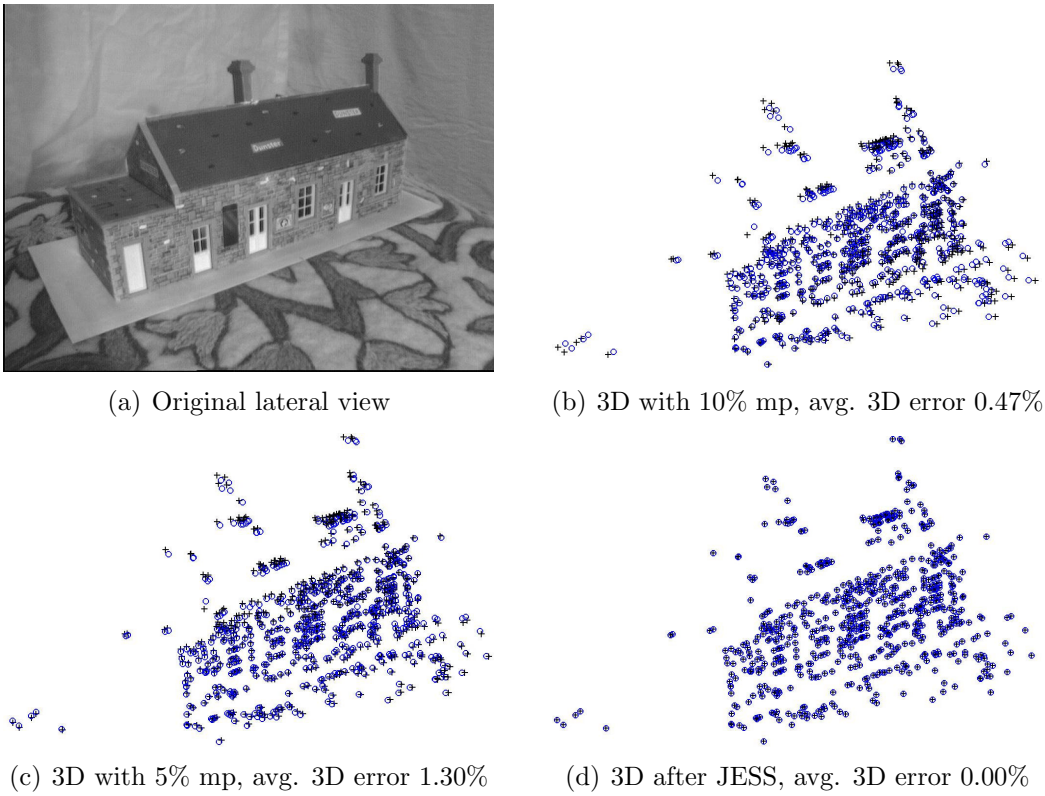


(a) 0 error left (b) 1 error left

Figure 4.19: 3D reconstruction of JESS. ◯ are the ground truth point positions, + are the JESS estimation point positions. (a) 8.96% MD, avg. 3D error: 0.80%. (b) 9.32% MD, avg. 3D error: 7.34%. Errors shown as a percentage of the ground truth depth size.

align the reference reconstruction with the one obtained by JESS. The results are shown in Figure 4.20 to 4.23. These results confirm the test on the synthetic cubes: the presence of even only few misclassified points greatly affect the 3D reconstruction. Moreover, in these examples the misclassified points were removed from the analysis because the ground truth of the segmentation was known. However, in an unknown scenario it is not possible to rely on prior knowledge and therefore the ability of JESS to remove misclassified points is vital. For both buildings JESS was able to completely correct the segmentation and finally provide the reconstructions shown in Figure 4.20(d), 4.21(d), and Figure 4.22(d), 4.23(d).

Note that the 3D error is always higher for the House sequence. This is due to the small amount of motion contained in this sequence (refer to Figure 4.6 and Figure 4.7 to compare respectively the House and the Hotel motions). This shows once again that if the amount of motion is not sufficiently long, metric constraints cannot be satisfied. In the case of the House and Hotel sequence the presence of a long motion of one of the two objects was sufficient for the JESS algorithm to detect misclassified points. However, in cases when none of the motions are long enough, like for the Hopkins155 and Hopkins12 databases,



(a) Original lateral view (b) 3D with 10% mp, avg. 3D error 0.47%

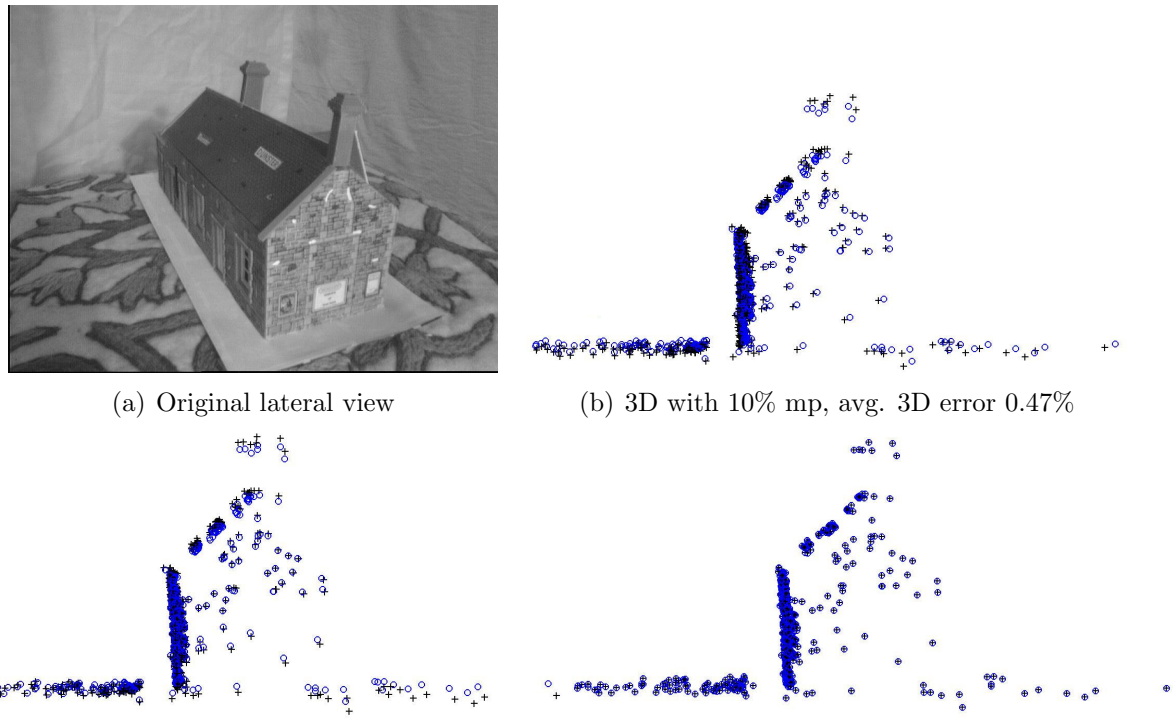
(c) 3D with 5% mp, avg. 3D error 1.30% (d) 3D after JESS, avg. 3D error 0.00%

Figure 4.20: 3D reconstruction of JESS on the House and Hotel sequence with different amount of misclassified points (mp). Metric size of House ground truth is $3.88 \times 6.25 \times 7.67$. Lateral view of House. ◯ are the ground truth point positions, + are the 3D reconstructions. Errors shown as a percentage of the ground truth depth size.

metric constraints are greatly effected and this is reflected by JESS performance.

4.7 Conclusion

In this chapter a generic optimisation framework, JESS, has been presented. JESS is able to estimate the multi-body motion segmentation and the 3D reconstruction from image trajectories even in the presence of missing data. This approach takes advantage of an initial segmentation to jointly include the metric constraints, given by an orthographic camera model, and the constraint that arises from the fact that the shape matrix that describes the multi-body 3D shape is generally sparse. The metric constraints are used to compute the 3D metric shapes and to fill the missing entries, while the sparse optimisation of the shape matrix detects wrong assignments of the trajectories and reclassifies them



(a) Original lateral view

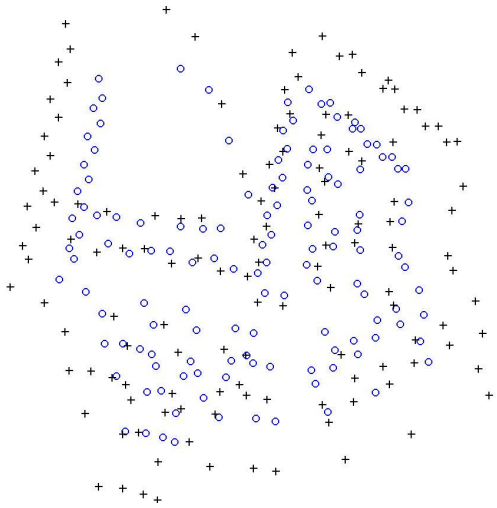
(b) 3D with 10% mp, avg. 3D error 0.47%

(c) 3D with 5% mp, avg. 3D error 1.30%

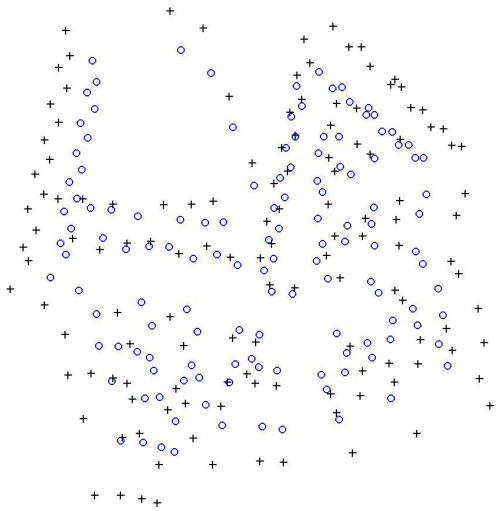
(d) 3D after JESS, avg. 3D error 0.00%

Figure 4.21: 3D reconstruction of JESS on the House and Hotel sequence with different amount of misclassified points (mp). Metric size of House ground truth is $3.88 \times 6.25 \times 7.67$. Back view of House ◯ are the ground truth point positions, + are the 3D reconstructions. Errors shown as a percentage of the ground truth depth size.

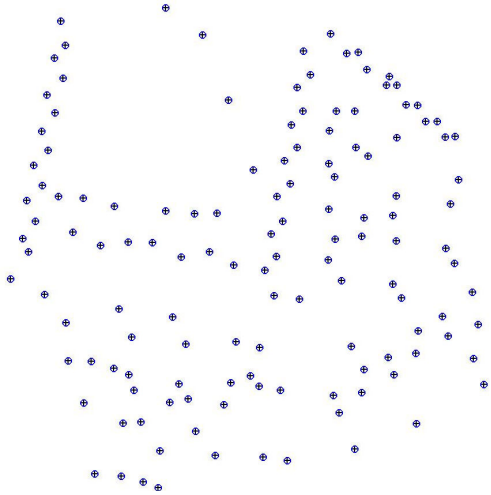
(a) Original frontal view



(b) 3D with 10% mp, avg. 3D error 8.84%

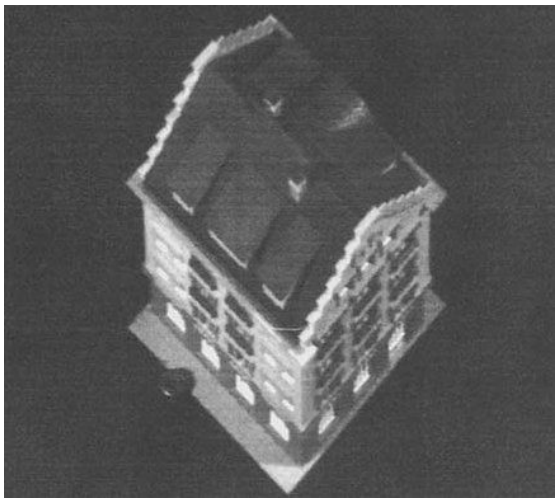


(c) 3D with 5% mp, avg. 3D error 6.25%

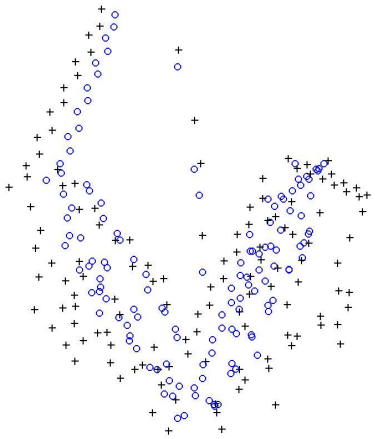


(d) 3D after JESS, avg. 3D error 0.00%

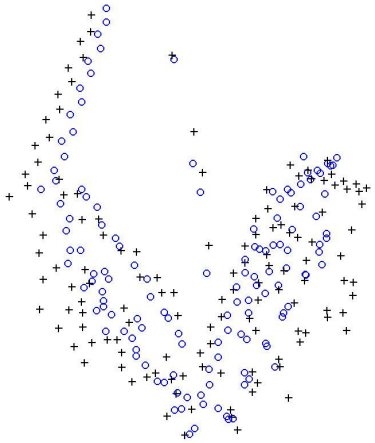
Figure 4.22: 3D reconstruction of JESS on the House and Hotel sequence with different amount of misclassified points (mp). Metric size of Hotel ground truth is $3.32 \times 3.80 \times 4.64$. Lateral view of Hotel. ◯ are the ground truth point positions, + are the 3D reconstructions. Errors shown as a percentage of the ground truth depth size.



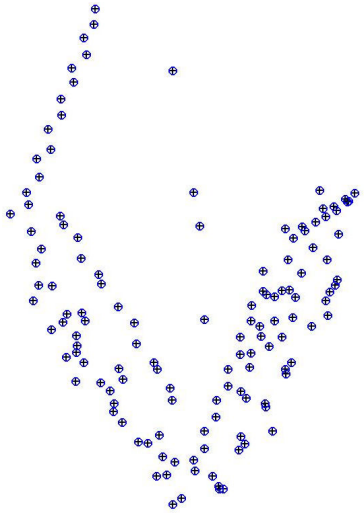
(a) Original top view



(b) 3D with 10% mp, avg. 3D error 8.84%



(c) 3D with 5% mp, avg. 3D error 6.25%



(d) 3D after JESS, avg. 3D error 0.00%

Figure 4.23: 3D reconstruction of JESS on the House and Hotel sequence with different amount of misclassified points (mp). Metric size of Hotel ground truth is $3.32 \times 3.80 \times 4.64$. Top view of Hotel. ◯ are the ground truth point positions, + are the 3D reconstructions. Errors shown as a percentage of the ground truth depth size.

to the correct motion. Note that while camera matrix and sparsity constraint have been previously used in SfM and multi-body SfM, they have never been used to solve the motion segmentation problem. Moreover, to these authors knowledge, JESS is unique in this field as there is no previous framework able to correct the results of a motion segmentation algorithm and, simultaneously, to compute the 3D structure of the moving objects.

The experiments proved the validity of JESS in spite of the fact that two of the three databases used, the Hopkins155 and the Hopkins12, are mainly composed by sequences that contain very small motions, and therefore, they may not guarantee the effectiveness of the camera constraints. Moreover, JESS could be improved by taking into account the specific nature of each motion. In fact, in the SfM step all the motions were treated as if they were rigid, even if in the Hopkins155 database there are some articulated and non-rigid motions. Therefore, by treating non-rigid and articulated motions properly the performance of JESS could be further improved. Moreover, as explained in Section 4.6.4 also the performance in terms of computational time could be boosted by adopting implementations of sparse optimisation on Graphic Processing Units [158].

5

Conclusion

In this final chapter a summary of the thesis and its main contributions are presented. Future directions connected to this work are also pointed out. The chapter is ended with a summary of publications and remarks related to this thesis.

5.1 Summary and contributions

In this thesis the problem of *motion segmentation* has been discussed.

- The *analysis of the state of the art*, Chapter 2, showed how big the effort that the research community has invested in this field is. A possible *classification of the state of the art* has been proposed and the most important attributes of motion segmentation algorithms have been described. Among all of the different techniques presented one category, the motion segmentation by *manifold clustering*, has emerged as particularly promising. One of the advantages of manifold clustering techniques is that they are usually based on features, therefore they are faster than dense-based approaches. Moreover, they can deal with different kinds of motion, provided that the dimension of the global space is known or can be estimated. Finally, manifold clustering can be easily extended to structure from motion. Within this group, the Local Subspace Affinity (LSA) [72] algorithm has been studied in more detail because

of the elegance of its framework and its good performance.

- In Chapter 3 the weaknesses of LSA have been pointed out and two new algorithms that fix some of those weaknesses have been presented. The first algorithm, the *Enhanced Model Selection+* (EMS+), solves the difficult task of estimating the rank of the trajectory matrix by analysing the relationship between the estimated rank and the affinity matrix. EMS+ uses the affinity matrix with the highest entropy value. This choice has the property of discarding homogeneous, and thus, not useful matrices while selecting the matrix with the highest content of information. The second algorithm, the *Adaptive Subspace Affinity* (ASA) takes a step further and studies the trend of the principal angles (PAs) in order to be able to select the rank that provides the best separation between angles of different motions. Moreover, ASA uses a new affinity measure that takes into account, so far neglected, issues related to the use of PAs as a measure of distance between subspaces. Both EMS+ and ASA outperformed any other LSA-based technique. EMS+ scored on the whole Hopkins155 database a misclassification rate of 4.23%, while ASA is one of the best performing algorithms of the state of the art as its misclassification rate is only 1.47%.

- The last contribution presented in Chapter 3, is an algorithm for the *estimation of the number of motions*. Such estimation is performed by exploiting a property that comes from the spectral graph theory. Specifically, the Symmetric Normalised Laplacian matrix is built using the value of the estimated affinity matrix. Hence, the eigenvalue spectrum of the Symmetric Normalised Laplacian matrix is thresholded using a formula inspired by the Linear Discriminative Analysis (LDA). The number of eigenvalues below the threshold indicates the number of connected components of the graph described by the matrix, and therefore, the number of motions. EMS+ with the automatic estimation of the number of motions is named *Enhance Local Subspace Affinity* (ELSA), while ASA with the automatic estimation is called *Automatic-ASA* (A-ASA). Thanks to this estimation ELSA and A-ASA, differently from most of

the techniques in the state of the art of motion segmentation, are able to perform their task without requiring any additional parameter than the trajectory matrix. In fact, the few free parameters of ELSA and A-ASA are either fixed or automatically tuned by the algorithms themselves. ELSA and A-ASA have been compared, on the Hopkins155 database and on a synthetic database, with the only known robust algorithm able to perform the segmentation without any parameter: the Agglomerative Lossy Compression [65] (ALC). Both ELSA and A-ASA performed better than ALC in terms of final misclassification rate but also in terms of correct estimation of the number of motions. In fact, on the Hopkins155 database, ALC had a misclassification rate of 12.86%, while ELSA had a rate of 10.75% and A-ASA of 9.50%. In addition, the average error in absolute value of the estimation of the number of motions was 1.16 with ALC, but only 0.37 and 0.42 for ELSA and A-ASA respectively.

- Given the already good performance of motion segmentation algorithms, in Chapter 4, a slightly different approach has been followed. This approach aimed to bring the problems of motion segmentation and Structure from Motion (SfM) closer. These two problems are related by the fact that motion segmentation can be seen as a preprocessing step that has to be applied before SfM. In this thesis the two problems have been connected by exploiting some constraints usually employed in SfM, but never used in motion segmentation: the *camera matrix* constraints and the *sparse* structure of the aggregate shape matrix. Therefore, the proposed *Joint Estimation of Motion Segmentation and Structure from Motion* (JESS) framework, begins with an initial almost-correct segmentation, and by imposing these mentioned constraints, is able to correct the misclassified points and provide the 3D shape of all of the moving objects in the scene. The combination of ASA and JESS provided the one of the lowest misclassification rates in the state of the art on the Hopkins155 database (among techniques that do not require tuning of parameters): 0.97%. More importantly, of the 155 sequences, 128 were segmented without any error. Moreover, for these cases, a 3D reconstruction also became available. To the knowledge of these authors, JESS

is the first framework simultaneously able to correct the segmentation of any motion segmentation algorithm and to provide the 3D reconstruction of the moving objects.

Before continuing, Table 2.2 with the summary of manifold clustering algorithms presented in Chapter 2 is extended with the algorithms presented in this thesis. The new Table 5.1 should allow the reader to quickly understand which are the benefits and the drawbacks of the proposed techniques in comparison with the rest of the state of the art.

5.2 Future directions

Future work connected to this thesis is now presented. Future work can be divided into immediate future work, which can be accomplished by extending the work presented in this thesis, and further future work, which can be seen as long term objectives.

5.2.1 Immediate future work

- Nowadays the misclassification rates, when the number of motions is known, are already good. Despite the fact that the misclassification rate could be further improved, future work should focus on the *ability to estimate the number of motions* in a more efficient way. ALC, ELSA and A-ASA already provide satisfactory results without having this information, however, there is room for improvement especially when the motions become dependant.

- The ability of JESS to recover missing data by exploiting 3D information has been described. However, the overall performance, including the estimation of missing data, could be improved by a *better initialisation of the missing entries*. Currently, the initialisation is performed by computing the central point of the group of trajectories to which each missing entry is associated. However, a more precise initialisation could be used [121, 128] and it could lead to better reconstruction and also a more effective ability of the framework to detect errors in the initial segmentation.

Manifold Clustering	Iter	*Ho et al. 2003* [56]	F			✓	✓✓X	✓	✓	CD
		da Silva et al. 2008 [57]	F			✓	✓✓X	✓	✓	CD
		da Silva et al. 2009 [58]	F			✓	✓✓X	✓	✓	CD
	Stat	*Fishler et al. 1981* [59]	F			✓	✓✓–	I	RA	C
		Kanatani et al. 2002 [60]	F			✓	✓✓✓	I	R	
		Sugaya et al. 2004 [61]	F			✓	✓✓X	I	R	C
		Gruber et al. 2004 [62]	F	✓		✓	✓✓X	I	R	X
		Gruber et al. 2006 [63]	F	✓	✓	✓	✓✓X	I	R	X
		Sugaya et al. 2010 [64]	F			✓	✓✓✓	I	R	C
	Sparse	*Rao et al. 2008/2010* [65, 66]	F	✓		✓	X✓–	I	R	
		Elhamifar et al. 2009 [67]	F	✓		✓	✓✓–	✓	✓	DX
	Fact	*Costeira et al. 1998* [68]	F			✓	XX–	I	R	
		Ichimura et al. 2000 [69]	F			✓	XX–	I	R	
		Zelnik-Manor et al. 2003 [70]	F			✓	✓X–	✓	RA	CD
	Subspaces	*Vidal et al. 2004* [71]	F	✓		✓	XX–	✓	R	C
		Yan et al. 2006/08 [31, 72]	F			✓	X✓–	✓	✓	CDX
		Goh et al. 2007 [73]	F			✓	X✓–	✓	R	CD
		Julià et al. 2008 [74]	F	✓		✓	X✓–	I	R	D
		Vidal et al. 2008 [75]	F	✓		✓	X✓–	✓	R	C
		Goh et al. 2008 [76]	F			✓	X✓–	✓	R	CD
		Chen et al. 2009 [77, 78]	F			✓	✓✓–	✓	✓	CD
		Kim et al. 2009 [79]	F			✓	✓✓–	✓	R	C
		Yang et al. 2009 [80]	D			✓	✓✓–	I	R	CD
		Lauren et al. 2009 [81]	F			✓	✓✓–	✓	✓	CD
		This Thesis: ELSA	F			✓	✓✓–	✓	✓	
		This Thesis: A-ASA	F			✓	✓✓–	✓	✓	
		This Thesis: ASA+JESS	F	✓		✓	✓✓X	✓	✓	C
Features (F) / Dense (D)										
Occlusion or Missing Data										
Spatial Continuity										
Temporary Stopping										
Robustness (Noise, Outliers, Initialisation: ✓yes, Xno, - not related)										
Dependency (Independent, Dependent, ✓all)										
Kind (Rigid, Non-rigid, Articulated, ✓all)										
Prior knowledge (C Number of clusters, D Subspace dimensions, X Other)										

Table 5.1: Updated summary of the manifold clustering-based techniques including the algorithms proposed in this thesis.

- Another improvement related to JESS is to take into account the kind of motion of each sequence. This information could be derived by the global space size estimated by EMS+, or by ASA, and the number of motions of the scene. Once this information has been recovered the specific kind of motion could be used in the structure from motion step, which at this stage, assumes rigid motions.

Further future work

- From the analysis of the state of the art it has emerged that very few approaches in the manifold clustering group *exploit spatial information*. However, when the motion of two objects become dependent the only possibility of distinguishing the two objects is via spatial information. Spatial information could be included in the affinity matrix, so that this additional information could help the clustering step and solve cases of dependant motions.

- As seen in the state of the art discussion, in general, feature-based techniques, such as manifold clustering, have the advantage over dense-based approaches of reducing dramatically the computational cost. However, feature-based techniques have to rely on the ability of the tracker to find salient points and track them successfully throughout the video sequence. Nowadays, such an assumption is not unrealistic, however, it would be important to develop algorithms able to deal with *only a few points per motion* instead of requiring larger amounts of them.

- A motion segmentation system, useful in real time applications, should work *incrementally*. An ideal incremental algorithm should be able to refine the segmentation at every new frame (or every small group of frames) without recomputing the whole solution from the beginning. In this respect, a bigger framework could be designed, where not only motion segmentation and SfM are merged, but also the tracking step is included. In such a framework the initial estimation of the segmentation could be used to predict where the points are moving so that the tracker could limit its area

of research, and outliers could be automatically discarded. Furthermore, in this ideal framework interrupted trajectories (i.e. missing data) could be managed more easily as their contribution could be exploited while the points are visible and temporarily frozen when they disappear. The frozen points could then be discarded if they do not appear within a predefined number of frames.

- Finally, a very ambitious aim is *to complete the merging between motion segmentation and SfM problems* that was started in this thesis. Currently, an initial motion segmentation is required so that, by imposing SfM constraints, the segmentation can be corrected. However, as explained in Chapter 4, if one additional constraint is found, the segmentation could be computed from scratch without requiring any initialisation. This would be an extraordinary advance as the two problems would be solved simultaneously by the same algorithm.

5.3 Publications and Code

The work developed in this thesis led to the following publications.

Journals articles

[**CVIU2011**] L. Zappella, A. Del Bue, X. Lladó, J. Salvi: *Joint Estimation of Segmentation and Structure from Motion with Missing Data.* In: Computer Vision and Image Understanding (2011). *Under Review.*

[**PR2011**] L. Zappella, X. Lladó, E. Provenzi, J. Salvi: *Enhanced Local Subspace Affinity for Feature-Based Motion Segmentation.* In: Pattern Recognition Ed. Elsevier (2011) Volume 44, Pages 454-470.

[**EL2009**] L. Zappella, X. Lladó, J. Salvi: *Rank estimation of trajectory matrix in motion segmentation.* In: Electronic Letters (2009) Volume 45, Number 11, Pages 540-541.

Books

[**LAP2011**] L. Zappella, X. Lladó, J. Salvi: *Motion Segmentation From Tracked Features*. LAP LAMBERT Academic Publish. GmbH & Co. KG. (2011), ISBN: 978-3-8443-0060-4.

Book Chapters

[**PR2009**] L. Zappella, X. Lladó, J. Salvi: *New Trends in Motion Segmentation*. In: Pattern Recognition book In-TECH, Ed. Intechweb.org. In-TECH, Ed. Intechweb.org (2009), ISBN: 978-953-307-014-8, Pages 31-46.

Conferences

[**WMVC2011**] L. Zappella, A. Del Bue, X. Lladó, J. Salvi: *Simultaneous Motion Segmentation and Structure from Motion*. In: Proceedings of the IEEE workshop on Motion and Video Computing, Kona, Hawaii, US (2011), Pages 679–684.

[**ACCV2010**] L. Zappella, E. Provenzi, X. Lladó, J. Salvi: *Adaptive Motion Segmentation Algorithm Based on the Principal Angles Configuration*. Lecture Notes Computer Science (ACCV, Queenstown, New Zealand 2010), Volume 6494/2011, Pages 15–26.

[**ICIP2009**] L. Zappella, X. Lladó, J. Salvi: *Enhanced model selection for motion segmentation*. In: Proceedings of the International Conference on Image Processing, Cairo, Egypt (2009), Pages 4053–4056.

[**CCIA2008**] L. Zappella, X. Lladó, J. Salvi: *Motion Segmentation: A Review*. In: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence, Sant Martí d'Empúries, Spain (2008), Pages 398–407.

5.4 Remarks

Part of the work of this thesis has been realised during a stage (1^{st} October 2009 to 1^{st} April 2010) in the Institute for Systems and Robotics laboratories coordinated by Professor

João Paulo Costeira at the University of Lisbon, Portugal. During this stage, thanks to the collaboration with Doctor Alessio Del Bue, the initial study of the JESS algorithm has been performed.

5.5 Code

The Matlab source code of the developed algorithms are publicly available at: `http://eia.udg.edu/∼zappella`. Specifically, the code available is described in this section.

- *Enhanced Model Selection* (EMS) is a novel rank estimation technique for trajectory matrices that can be used within the Local Subspace Affinity (LSA) framework. EMS is based on the relationship between the rank estimated by a model selection technique and the affinity matrix built with LSA. The result is a more robust and precise model selection by which it is possible to automate LSA without requiring any a priori knowledge (about the kind of motion) and to improve the final segmentation.
  Link: `http://eia.udg.edu/∼zappella/code/EMS_v0.1.tgz`

- *Enhanced Local Subspace Affinity* (ELSA) is a new feature-based motion segmentation technique. Unlike LSA, ELSA is robust in a variety of conditions even without manual tuning of its parameters. This result is achieved thanks to two improvements. The first is the EMS+ technique for the estimation of the trajectory matrix rank. The second is an estimation of the number of motions based on the analysis of the eigenvalue spectrum of the Symmetric Normalized Laplacian matrix.
  Link: `http://eia.udg.edu/∼zappella/code/ELSA_v0.1.tgz`

- *Adaptive Subspace Affinity* (ASA). This motion segmentation algorithm is a manifold clustering-based technique. The two most important steps are: the new rank estimation of the trajectory matrix (PAC) and the new similarity measure adopted (SCbA). Both these two techniques take into account the trend of the principal angles when the rank estimation of the trajectory matrix changes. The rank estimation is performed

by analysing which rank leads to a configuration where small and large angles are best separated. The affinity measure is a new function automatically parametrised so that it is able to adapt to the actual configuration of the principal angles.
Link: `http://eia.udg.edu/~zappella/code/ASA_v1.1.tgz`

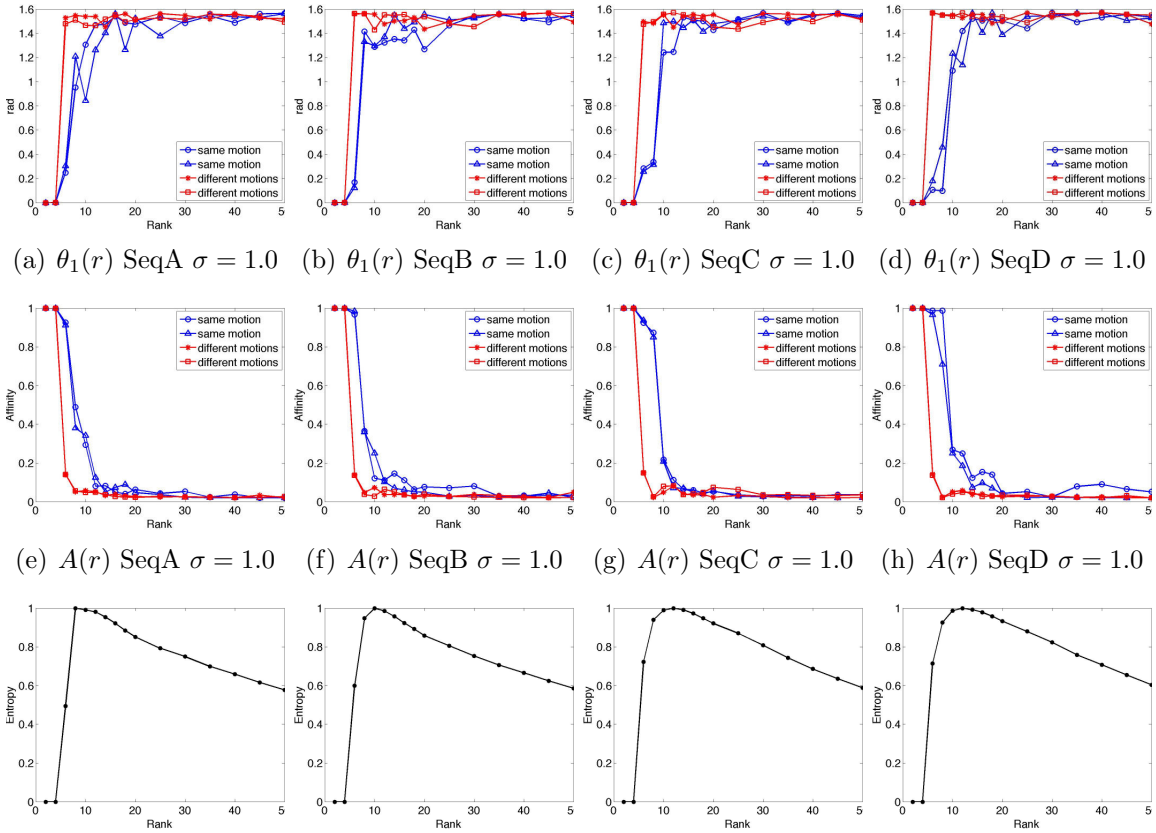
- *Jointly Estimation of Segmentation and Structure from Motion* (JESS) is a framework that partially merges the problems of motion segmentation and Structure from Motion. JESS exploits an initial, possibly wrong, motion segmentation result and corrects it by imposing constraints that arise from the Structure from Motion theory: the camera matrix constraints and the sparse structure of the aggregate shape matrix. The final JESS result is a corrected segmentation, the 3D structure of each of the moving objects and a motion description of each object.
  Link: `http://eia.udg.edu/~zappella/code/JESS_v0.1.tgz`

A

Principal Angles, Affinity and Entropy

Appendix

In Section 3.2.1 the trends of PAs, affinity, and entropy were discussed. In this appendix more examples are provided for a synthetic sequence with 2 motions and a noise level from $\sigma = 1.0$ up to $\sigma = 3.0$ pixels. The same trends are also provided for some sequences of the Hopkins155 database.

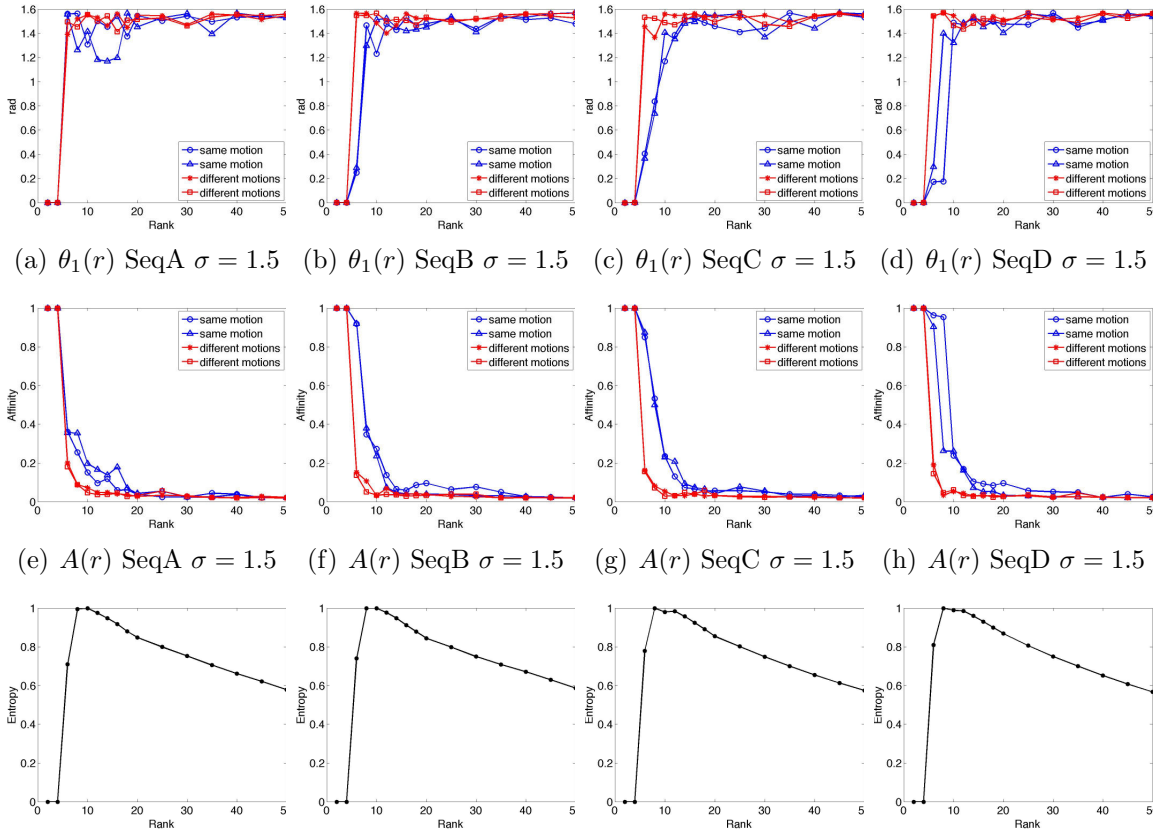


(a) $\theta_1(r)$ SeqA $\sigma = 1.0$ (b) $\theta_1(r)$ SeqB $\sigma = 1.0$ (c) $\theta_1(r)$ SeqC $\sigma = 1.0$ (d) $\theta_1(r)$ SeqD $\sigma = 1.0$

(e) $A(r)$ SeqA $\sigma = 1.0$ (f) $A(r)$ SeqB $\sigma = 1.0$ (g) $A(r)$ SeqC $\sigma = 1.0$ (h) $A(r)$ SeqD $\sigma = 1.0$

(i) $E(\mathtt{A}(r))$ SeqA $\sigma = 1.0$ (j) $E(\mathtt{A}(r))$ SeqB $\sigma = 1.0$ (k) $E(\mathtt{A}(r))$ SeqC $\sigma = 1.0$ (l) $E(\mathtt{A}(r))$ SeqD $\sigma = 1.0$

Figure A.1: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from synthetic sequences with Gaussian noise $\sigma = 1.0$) with two rigid motions, hence the maximum rank is 8 (NNs are estimated). Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.

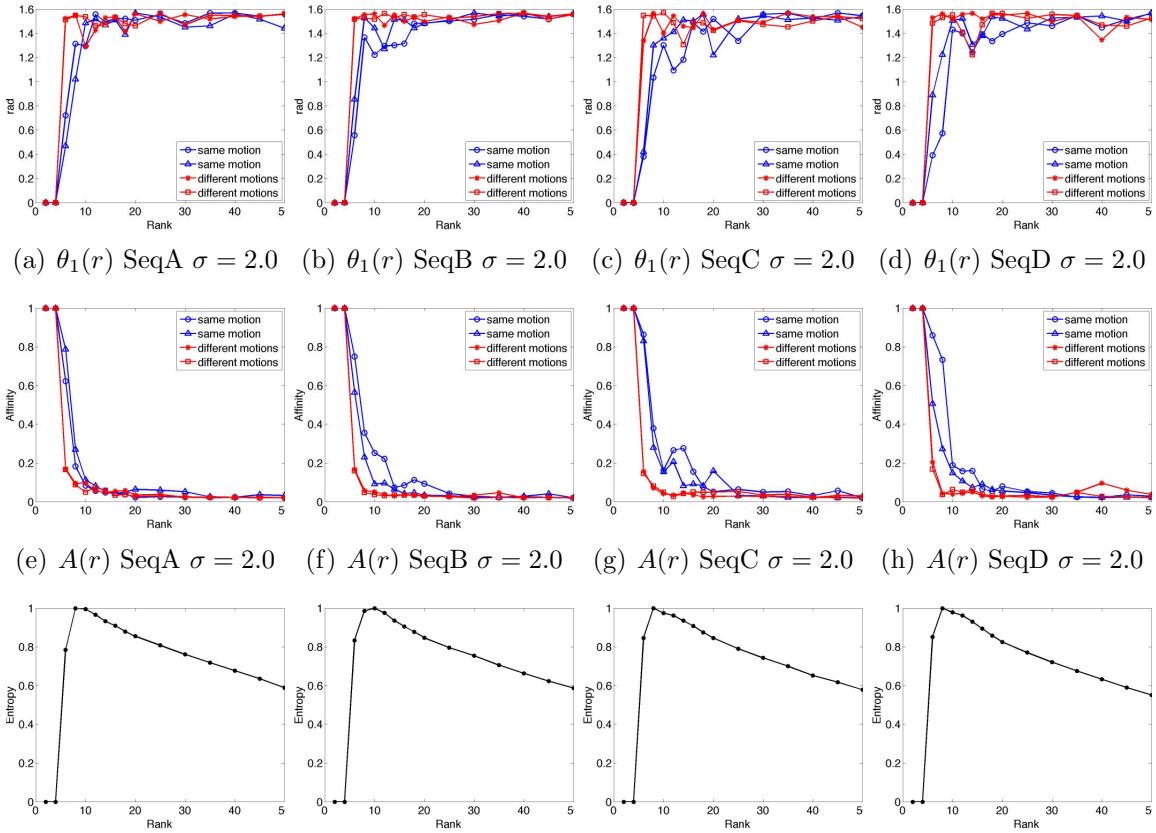


(a) $\theta_1(r)$ SeqA $\sigma = 1.5$ (b) $\theta_1(r)$ SeqB $\sigma = 1.5$ (c) $\theta_1(r)$ SeqC $\sigma = 1.5$ (d) $\theta_1(r)$ SeqD $\sigma = 1.5$

(e) $A(r)$ SeqA $\sigma = 1.5$ (f) $A(r)$ SeqB $\sigma = 1.5$ (g) $A(r)$ SeqC $\sigma = 1.5$ (h) $A(r)$ SeqD $\sigma = 1.5$

(i) $E(\mathtt{A}(r))$ SeqA $\sigma = 1.5$ (j) $E(\mathtt{A}(r))$ SeqB $\sigma = 1.5$ (k) $E(\mathtt{A}(r))$ SeqC $\sigma = 1.5$ (l) $E(\mathtt{A}(r))$ SeqD $\sigma = 1.5$

Figure A.2: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from synthetic sequences with Gaussian noise $\sigma = 1.5$) with two rigid motions, hence the maximum rank is 8 (NNs are estimated). Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.

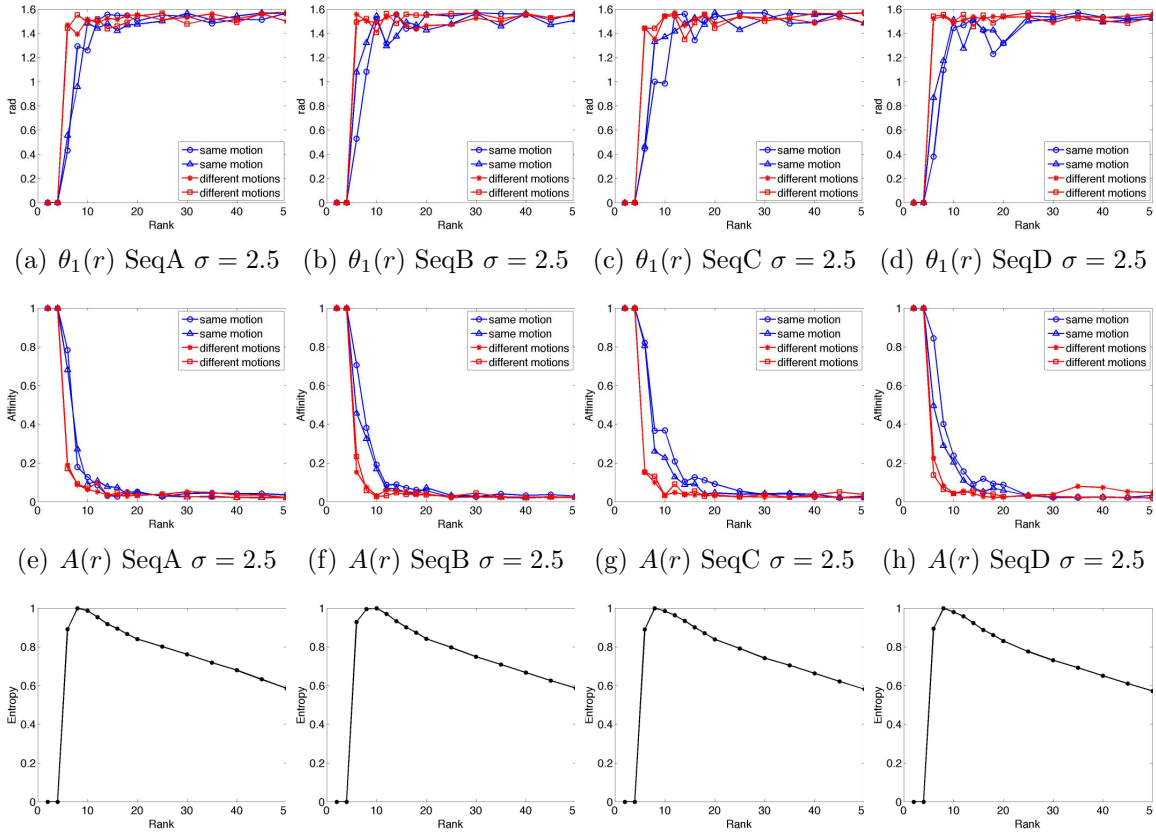


(a) $\theta_1(r)$ SeqA $\sigma = 2.0$ (b) $\theta_1(r)$ SeqB $\sigma = 2.0$ (c) $\theta_1(r)$ SeqC $\sigma = 2.0$ (d) $\theta_1(r)$ SeqD $\sigma = 2.0$

(e) $A(r)$ SeqA $\sigma = 2.0$ (f) $A(r)$ SeqB $\sigma = 2.0$ (g) $A(r)$ SeqC $\sigma = 2.0$ (h) $A(r)$ SeqD $\sigma = 2.0$

(i) $E(\mathtt{A}(r))$ SeqA $\sigma = 2.0$ (j) $E(\mathtt{A}(r))$ SeqB $\sigma = 2.0$ (k) $E(\mathtt{A}(r))$ SeqC $\sigma = 2.0$ (l) $E(\mathtt{A}(r))$ SeqD $\sigma = 2.0$

Figure A.3: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from synthetic sequences with Gaussian noise $\sigma = 2.0$) with two rigid motions, hence the maximum rank is 8 (NNs are estimated). Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.

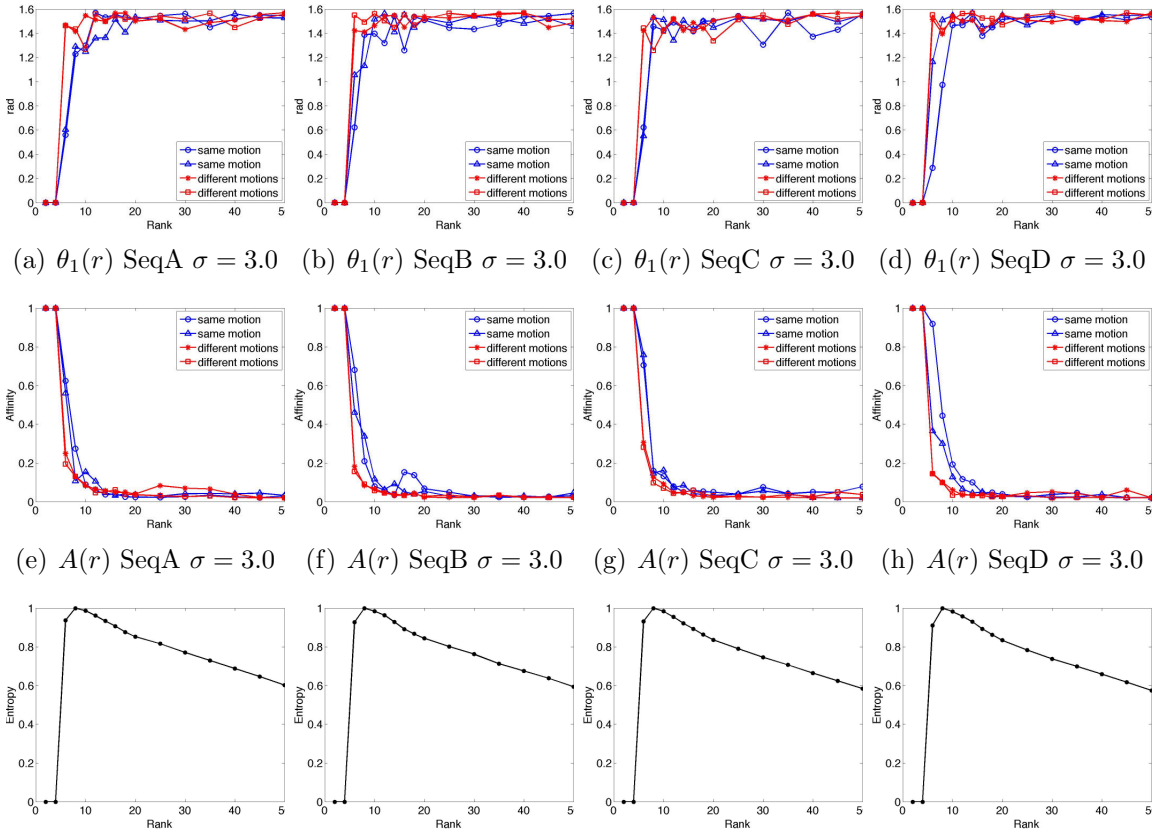


(a) $\theta_1(r)$ SeqA $\sigma = 2.5$ (b) $\theta_1(r)$ SeqB $\sigma = 2.5$ (c) $\theta_1(r)$ SeqC $\sigma = 2.5$ (d) $\theta_1(r)$ SeqD $\sigma = 2.5$

(e) $A(r)$ SeqA $\sigma = 2.5$ (f) $A(r)$ SeqB $\sigma = 2.5$ (g) $A(r)$ SeqC $\sigma = 2.5$ (h) $A(r)$ SeqD $\sigma = 2.5$

(i) $E(\mathtt{A}(r))$ SeqA $\sigma = 2.5$ (j) $E(\mathtt{A}(r))$ SeqB $\sigma = 2.5$ (k) $E(\mathtt{A}(r))$ SeqC $\sigma = 2.5$ (l) $E(\mathtt{A}(r))$ SeqD $\sigma = 2.5$

Figure A.4: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from synthetic sequences with Gaussian noise $\sigma = 2.5$) with two rigid motions, hence the maximum rank is 8 (NNs are estimated). Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.

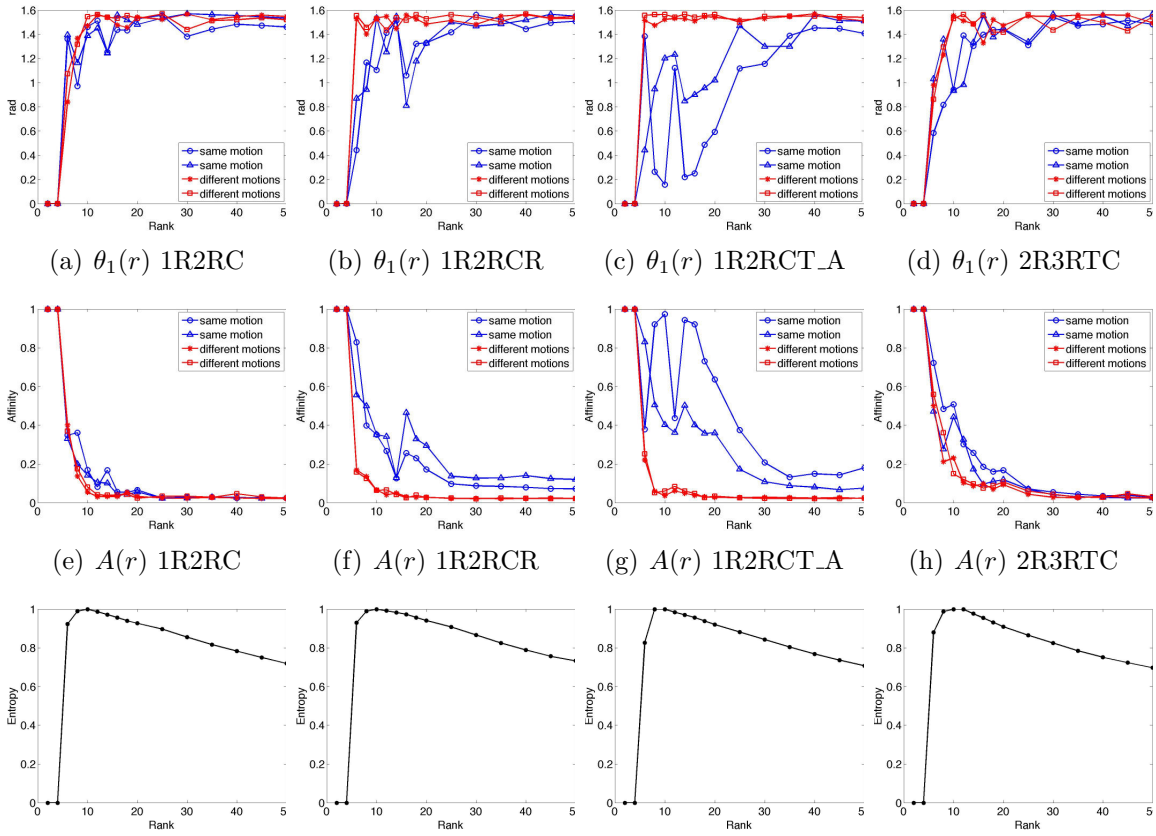


(a) $\theta_1(r)$ SeqA $\sigma = 3.0$ (b) $\theta_1(r)$ SeqB $\sigma = 3.0$ (c) $\theta_1(r)$ SeqC $\sigma = 3.0$ (d) $\theta_1(r)$ SeqD $\sigma = 3.0$

(e) $A(r)$ SeqA $\sigma = 3.0$ (f) $A(r)$ SeqB $\sigma = 3.0$ (g) $A(r)$ SeqC $\sigma = 3.0$ (h) $A(r)$ SeqD $\sigma = 3.0$

(i) $E(\mathtt{A}(r))$ SeqA $\sigma = 3.0$ (j) $E(\mathtt{A}(r))$ SeqB $\sigma = 3.0$ (k) $E(\mathtt{A}(r))$ SeqC $\sigma = 3.0$ (l) $E(\mathtt{A}(r))$ SeqD $\sigma = 3.0$

Figure A.5: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from synthetic sequences with Gaussian noise $\sigma = 3.0$) with two rigid motions, hence the maximum rank is 8 (NNs are estimated). Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.

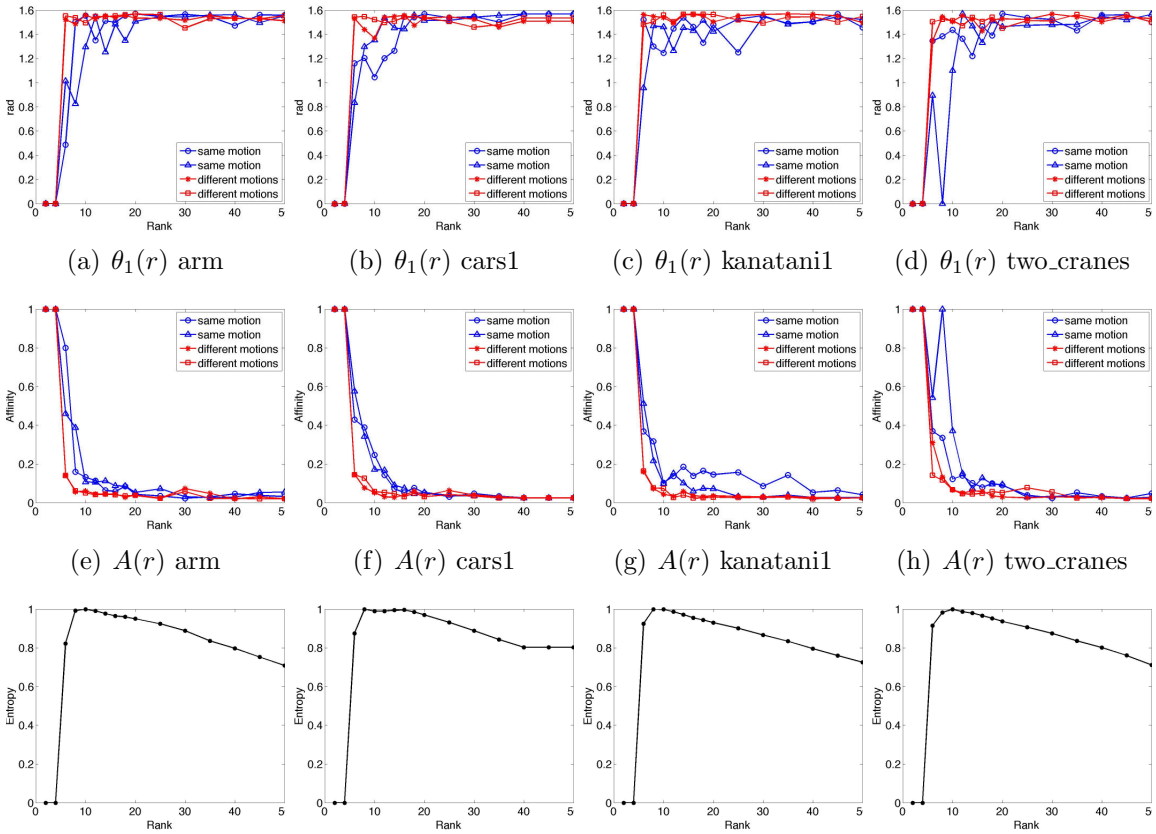


(a) $\theta_1(r)$ 1R2RC (b) $\theta_1(r)$ 1R2RCR (c) $\theta_1(r)$ 1R2RCT_A (d) $\theta_1(r)$ 2R3RTC

(e) $A(r)$ 1R2RC (f) $A(r)$ 1R2RCR (g) $A(r)$ 1R2RCT_A (h) $A(r)$ 2R3RTC

(i) $E(\mathtt{A}(r))$ 1R2RC (j) $E(\mathtt{A}(r))$ 1R2RCR (k) $E(\mathtt{A}(r))$ 1R2RCT_A (l) $E(\mathtt{A}(r))$ 2R3RTC

Figure A.6: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from some sequences of the Hopkins155 database. Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.



(a) $\theta_1(r)$ arm (b) $\theta_1(r)$ cars1 (c) $\theta_1(r)$ kanatani1 (d) $\theta_1(r)$ two_cranes

(e) $A(r)$ arm (f) $A(r)$ cars1 (g) $A(r)$ kanatani1 (h) $A(r)$ two_cranes

(i) $E(\mathtt{A}(r))$ arm (j) $E(\mathtt{A}(r))$ cars1 (k) $E(\mathtt{A}(r))$ kanatani1 (l) $E(\mathtt{A}(r))$ two_cranes

Figure A.7: First row: trend of the largest principal angle between two pairs of trajectories of the same motion (blue) and two pairs of trajectories of different motions (red). The pairs of trajectories are randomly taken from some sequences of the Hopkins155 database. Second row: trend of the affinity value for the same pairs of trajectories showed in the first row. Third row: trend of the entropy value of the whole affinity matrix.

Bibliography

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Addison-Wesley Pub, 2007.

[2] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," in *Proc. Intern. Workshop on Image Proc.: Real-time Edge and Motion Detection/Estimation*, 1979.

[3] F. G. Smith, K. R. Jepsen, and P. F. Lichtenwalner, "Comparison of neural network and markov random field image segmentation techniques," in *Proc. of the 18th Annual Review of progress in quantitative nondestructive evaluation*, vol. 11, 1992, pp. 717–724.

[4] A. Blake, "Active contours," *Robotica*, vol. 17, no. 4, pp. 459–462, 1999.

[5] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *IEEE Trans. Pattern Analysis Machine Intell.*, 2000, pp. 888–905.

[6] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry. Fluid Mechanics, Computer Vision and Materials Sciences.* Cambridge University Press, 1999.

[7] D. Cremers, M. Rousson, and R. Deriche, "A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape," *Int. J. Comput. Vision*, vol. 72, no. 2, pp. 195–215, 2007.

[8] Y. Sugaya and K. Kanatani, "Multi-stage optimization for multi-body motion segmentation," *IEICE Trans. on Information and Syst.*, vol. E87-D(7), pp. 1935–1942, 2004.

[9] J. Carme, "Missing data matrix factorization addressing the structure from motion problem," PhD in Computer Science, Universitat Autònoma de Barcelona, 2007.

[10] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Int. J. Comput. Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[11] B. Raducanu, J. Vitria, and D. Gatica-Perez, "You are fired! nonverbal role analysis in competitive meetings," in *IEEE Conf. on Acoustics, Speech and Signal Proc.*, 2009, pp. 1949–1952.

[12] S. Escalera, R. Martnez, J. Vitriá, and T. Radeva P., Anguera, "Dominance detection in face-to-face conversations," in *CVPR Workshop on Human communicative Behavior analysis*, 2009, pp. 97–102.

[13] A. A. Salah, T. Gevers, N. Sebe, and A. Vinciarelli, "Challenges of human behavior understanding," in *Lect. Notes Comput. Sc.*, vol. 6219, 2010, pp. 1–12.

[14] D. Zhang and G. Lu, "Segmentation of moving objects in image sequence: a review," *Circ. Syst. Signal Proc.*, vol. 20, no. 2, pp. 143–183, 2001.

[15] MPEG, "Coding of audio-visual objects: visual, final draft international standard," *ISO/IEC JTC1/SC29/WG11*, vol. N2502, pp. 1–318, 1998.

[16] A. Del Bue, D. Comaniciu, V. Ramesh, and C. Regazzoni, "Smart cameras with real-time video object generation," in *Proc. ICIP IEEE*, vol. 3, 2002, pp. 429–432.

[17] E. Trucco, K. Plakas, N. Brandenburg, P. Kauff, M. Karl, and O. Schreer, "Real-time disparity analysis for immersive 3-d teleconferencing," in *IEEE ICCV Workshop Video Regist.*, 2001.

[18] F. Isgrò, E. Trucco, P. Kauff, and O. Schreer, "3-d image processing in the future of immersive media," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, pp. 288–303, 2004.

[19] M. Li and S. Drew, *Fundamentals of Multimedia*. Prentice Hall, 2003.

[20] L. Zappella, X. Lladó, and J. Salvi, "Motion segmentation: A review," in *Proc. of the Intern. Conf. of the Cat. Assoc. for Art. Intell.*, 2008, pp. 398–407.

[21] L. Zappella, X. Lladó, and J. Salvi, "New trends in motion segmentation," in *Pattern Recognition*, P.-Y. Yin, Ed. INTECH, 2009, pp. 31–46.

[22] R. Tron and R. Vidal, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *Proc. CVPR IEEE*, 2007, pp. 1–8.

[23] J. Shi and C. Tomasi, "Good features to track," in *Proc. CVPR IEEE*, 1994, pp. 593–600.

[24] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. CVPR IEEE*, 1999, pp. 1150–1157.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," *Comput. Vis. Image Und.*, vol. 110, no. 3, pp. 346–359, 2008.

[26] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[27] M. P. Kumar, P. H. Torr, and A. Zisserman, "Learning layered motion segmentations of video," *Int. J. Comput. Vision*, vol. 76, no. 3, pp. 301–319, 2008.

[28] A. Bronstein, M. Bronstein, M. Bronstein, and R. Kimmel, *Numerical geometry of non-rigid shapes*. Springer-Verlag New York Inc, 2008.

[29] A. Yezzi and S. Soatto, "Deformotion: Deforming motion, shape average and the joint registration and approximation of structures in images," *Int. J. Comput. Vision*, vol. 53, no. 2, pp. 153–167, 2003.

[30] X. Lladó, A. Del Bue, and L. Agapito, "Non-rigid metric reconstruction from perspective cameras," *Image Vision Comput.*, vol. 28, no. 9, pp. 1339–1353, 2010.

[31] J. Yan and M. Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," in *Lect. Notes Comput. Sc. (ECCV)*, vol. 3954, 2006, pp. 94–106.

[32] A. Cavallaro, O. Steiger, and T. Ebrahimi, "Tracking Video Objects in Cluttered Background," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 4, pp. 575–584, 2005.

[33] F.-H. Cheng and Y.-L. Chen, "Real time multiple objects tracking and identification based on discrete wavelet transform," *Pattern Recogn.*, vol. 39, no. 6, pp. 1126–1139, 2006.

[34] R. Li, S. Yu, and X. Yang, "Efficient spatio-temporal segmentation for extracting moving objects in video sequences," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 3, pp. 1161–1167, 2007.

[35] A. Colombari, A. Fusiello, and V. Murino, "Segmentation and tracking of multiple video objects," *Pattern Recogn.*, vol. 40, no. 4, pp. 1307–1317, 2007.

[36] C. Rasmussen and G. D. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 23, no. 6, pp. 560–576, 2001.

[37] D. Cremers and S. Soatto, "Motion competition: A variational approach to piecewise parametric motion segmentation," *Int. J. Comput. Vision*, vol. 62, no. 3, pp. 249–265, 2005.

[38] H. Shen, L. Zhang, B. Huang, and P. Li, "A map approach for joint motion estimation, segmentation, and super resolution," *IEEE Trans. Image Processing*, vol. 16, no. 2, pp. 479–490, 2007.

[39] N. Vaswani, A. Tannenbaum, and A. Yezzi, "Tracking deforming objects using particle filtering for geometric active contours," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 29, no. 8, pp. 1470–1475, 2007.

[40] R. Stolkin, A. Greig, M. Hodgetts, and J. Gilby, "An em/e-mrf algorithm for adaptive model based tracking in extremely poor visibility." *Image Vision Comput.*, vol. 26, no. 4, pp. 480–495, 2008.

[41] N. Thakoor, J. Gao, and V. Devarajan, "Multibody structure-and-motion segmentation by branch-and-bound model selection," *IEEE Trans. on Image Proc.*, vol. 19, no. 6, pp. 1393–1402, 2010.

[42] L. Wiskott, "Segmentation from motion: Combining Gabor- and Mallat-wavelets to overcome aperture and correspondence problem," in *Lect. Notes Comput. Sc. (CAIP)*, vol. 1296, 1997, pp. 329–336.

[43] M. Kong, J.-P. Leduc, B. Ghosh, and V. Wickerhauser, "Spatio-temporal continuous wavelet transforms for motion-based segmentation in real image sequences," in *Proc. ICIP IEEE*, vol. 2, 1998, pp. 662–666.

[44] E. Trucco, T. Tommasini, and V. Roberto, "Near-recursive optical flow from weighted image differences," *IEEE Trans. Syst., Man, Cybern.*, vol. 35, no. 1, pp. 124–129, 2005.

[45] J. Zhang, F. Shi, J. Wang, and Y. Liu, "3d motion segmentation from straight-line optical flow," in *Multimedia Content Analysis and Mining*, 2007, pp. 85–94.

[46] L. Xu, J. Chen, and J. Jia, "A segmentation based variational model for accurate optical flow estimation," in *Lect. Notes Comput. Sc. (ECCV)*, 2008, pp. 671–684.

[47] J. Klappstein, T. Vaudrey, C. Rabe, A. Wedel, and R. Klette, "Moving object segmentation using optical flow and depth information," in *Pacific-Rim Symposium on Image and Video Technology*, 2009, p. 611623.

[48] A. Bugeau and P. Pérez, "Detection and segmentation of moving objects in complex scenes," *Comput. Vis. Image Und.*, vol. 113, pp. 459–476, 2009.

[49] B. Ommer, T. Mader, and J. M. Buhmann, "Seeing the objects behind the dots: Recognition in videos from a moving camera," *Int. J. Comput. Vision*, vol. 83, pp. 57–71, 2009.

[50] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *Lect. Notes Comput. Sc. (ECCV)*. Springer, 2010, pp. 282–295.

[51] C. Min and G. Medioni, "Inferring segmented dense motion layers using 5d tensor voting," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 30, no. 9, pp. 1589–1602, 2008.

[52] K. Nordberg and V. Zografos, "Multibody motion segmentation using the geometry of 6 points in 2d images," in *Proc. Int. C. Patt. Recog.*, 2010, pp. 1783–1787.

[53] V. Zografos, K. Nordberg, and L. Ellis, "Sparse motion segmentation using multiple six-point consistencies," in *VECTaR workshop (ACCV)*, 2010.

[54] F. Xu, K.-M. Lam, and Q. Dai, "Video-object segmentation and 3d-trajectory estimation for monocular video sequences," *Image and Vision Computing*, vol. 29, no. 2-3, pp. 190–205, 2011.

[55] Y. Wang, J. Gong, D. Zhang, C. Gao, J. Tian, and H. Zeng, "Large disparity motion layer extraction via topological clustering," *IEEE Trans. on Image Proc.*, vol. 20, no. 1, pp. 43–52, 2011.

[56] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Proc. CVPR IEEE*, vol. 1, 2003, pp. 11–18.

[57] N. Pinho da Silva and J. Costeira, "Subspace segmentation with outliers: a grassmannian approach to the maximum consensus subspace," in *Proc. CVPR IEEE*, 2008, pp. 1–6.

[58] N. Pinho da Silva and J. Costeira, "The normalized subspace inclusion: Robust clustering of motion subspaces," in *Proc. ICCV IEEE*, 2009, pp. 1444–1450.

[59] M. A. Fischler and R. C. Bolles, "Ransac random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, 1981.

[60] K. Kanatani and C. Matsunaga, "Estimating the number of independent motions for multibody motion segmentation," in *Proceedings of the Fifth Asian Conference on Computer Vision*, vol. 1, 2002, pp. 7–12.

[61] Y. Sugaya and K. Kanatani, "Geometric structure of degeneracy for multi-body motion segmentation," in *Lect. Notes Comput. Sc.*, 2004, pp. 13–25.

[62] A. Gruber and Y. Weiss, "Multibody factorization with uncertainty and missing data using the em algorithm," in *Proc. CVPR IEEE*, vol. 1, 2004, pp. 707–714.

[63] A. Gruber and Y. Weiss, "Incorporating non-motion cues into 3d motion segmentation," in *Lect. Notes Comput. Sc. (ECCV)*, 2006, pp. 84–97.

[64] Y. Sugaya and K. Kanatani, "Improved multistage learning for multibody motion segmentation," in *Proc. of Int. Conf. Computer Vision Theory and App.*, vol. 1, 2010, pp. 199–206.

[65] S. R. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories," in *Proc. CVPR IEEE*, 2008, pp. 1–8.

[66] S. Rao, R. Tron, R. Vidal, and Y. Ma, "Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 32, pp. 1832–1845, 2010.

[67] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *Proc. CVPR IEEE*, 2009, pp. 2790–2797.

[68] J. P. Costeira and T. Kanade, "A multibody factorization method for independently moving objects," *Int. J. Comput. Vision*, vol. 29, no. 3, pp. 159–179, 1998.

[69] N. Ichimura and F. Tomita, "Motion segmentation based on feature selection from shape matrix," *Syst. Comput. Jpn.*, vol. 31, no. 4, pp. 32–42, 2000.

[70] L. Zelnik-Manor and M. Irani, "Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations," in *Proc. CVPR IEEE*, vol. 2, 2003, pp. 287–93.

[71] R. Vidal and R. Hartley, "Motion segmentation with missing data using powerfactorization and gpca," in *Proc. CVPR IEEE*, vol. 2, 2004, pp. 310–316.

[72] J. Yan and M. Pollefeys, "A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 30, no. 5, pp. 865–877, 2008.

[73] A. Goh and R. Vidal, "Segmenting motions of different types by unsupervised manifold clustering," in *Proc. CVPR IEEE*, 2007, pp. 1–6.

[74] C. Julia, A. Sappa, F. Lumbreras, J. Serrat, and A. Lopez, "Rank estimation in 3d multibody motion segmentation," *Electron. Lett.*, vol. 44, no. 4, pp. 279–280, 14 2008.

[75] R. Vidal, R. Tron, and R. Hartley, "Multiframe motion segmentation with missing data using powerfactorization and gpca," *Int. J. Comput. Vision*, vol. 79, pp. 85–105, 2008.

[76] A. Goh and R. Vidal, "Clustering and dimensionality reduction on riemannian manifolds," in *Proc. CVPR IEEE*, 2008, pp. 1–7.

[77] G. Chen and G. Lerman, "Spectral curvature clustering (scc)," *Int. J. Comput. Vision*, vol. 81, pp. 317–330, 2009.

[78] G. Chen and G. Lerman, "Motion segmentation by scc on the hopkins 155 database," in *Proc. ICCV IEEE*, 2009, pp. 759–764.

[79] J.-H. Kim and L. Agapito, "Motion segmentation using the hadamard product and spectral clustering," in *Proc. of the IEEE Intern. Conf. on Motion and Video Computing*, 2009, pp. 126–133.

[80] H. Yang, G. Welch, J.-M. Frahm, and M. Pollefeys, "3d motion segmentation using intensity trajectory," in *Lect. Notes Comput. Sc. (ACCV)*, 2009, pp. 157–168.

[81] F. Lauer and C. Schnrr, "Spectral clustering of linear subspaces for motion segmentation," in *Proc. ICCV IEEE*, 2009, pp. 678–685.

[82] A. Bobick and J. Davis, "An appearance-based representation of action," in *Proc. Int. C. Patt. Recog.*, 1996, pp. 307–312.

[83] Kalman, Rudolph, and Emil, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[84] M. A. Tanner, *Tools for statistical inference: Methods for exploration of posterior distributions and likelihood functions (3rd Edition).* Springer-Verlag, 1996.

[85] A. Doucet, "On sequential Monte Carlo methods for Bayesian filtering," Dept. Eng., Univ. Cambridge, Tech. Rep., 1998.

[86] I. Rekleitis, "Cooperative localization and multi-robot exploration," PhD in Computer Science, School of Computer Science, McGill University, Montreal, Quebec, Canada, 2003.

[87] Y. Shi and W. C. Karl, "Real-time tracking using level sets," in *Proc. CVPR IEEE*, 2005, pp. 34–41.

[88] S. Borman, "The expectation maximization algorithm – a short tutorial," 2004.

[89] K. Kanatani, "Statistical optimization and geometric visual inference," in *Lect. Notes Comput. Sc.*, 1997, pp. 306–322.

[90] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[91] Russle and Norvig, *AI, A Modern Approach*. Prentice Hall, 1995.

[92] O. Faugeras, R. Deriche, and N. Navab, "Information contained in the motion field of lines and the cooperation between motion and stereo," *International Journal of Imaging Systems and technology*, vol. 2, pp. 356–370, 1990.

[93] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 17, no. 8, pp. 790–799, 1995.

[94] J. Wang and E. Adelson, "Layered representation for motion analysis," in *Proc. CVPR IEEE*, 1993, pp. 361–366.

[95] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *Proc. CVPR IEEE*, 2006, pp. 53–60.

[96] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," in *Proc. ICCV IEEE*, 1999, pp. 377–384.

[97] L. Quan, "Invariants of six points and projective reconstruction from three uncalibrated images," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 17, pp. 34–46, 1995.

[98] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proc. of the IEEE*, 1989, pp. 257–286.

[99] X. Llado, A. D. Bue, and L. Agapito, "Euclidean reconstruction of deformable structure using a perspective camera with varying intrinsic parameters," in *Proc. Int. C. Patt. Recog.*, vol. 1, 2006, pp. 139–142.

[100] S. Koterba, S. Baker, I. Matthews, C. Hu, J. Xiao, J. F. Cohn, and T. Kanade, "Multi-view aam fitting and camera calibration," in *Proc. ICCV IEEE*, 2005, pp. 511–518.

[101] P. Tresadern and I. Reid, "Articulated structure from motion by factorization," in *Proc. CVPR IEEE*, vol. 2, 2005, pp. 1110–1115.

[102] J. Xiao and T. Kanade, "Non-rigid shape and motion recovery: Degenerate deformations," in *Proc. CVPR IEEE*, vol. 1, 2004, pp. 668–675.

[103] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (gpca)," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 27, pp. 1945–1959, 2005.

[104] G. Taubin, "Estimation of planar curves, surfaces, and non-planar space curves defined by implicit equations with applications to edge and range image segmentation," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 13, pp. 1115–1138, 2010.

[105] A. Gruber and Y. Weiss, "Factorization with uncertainty and missing data: Exploiting temporal coherence," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004, vol. 16.

[106] D. Donoho, "Compressed sensing," *IEEE Trans. on Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[107] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 29, no. 9, pp. 1546–1562, 2007.

[108] L. Zappella, X. Lladó, E. Provenzi, and J. Salvi, "Enhanced local subspace affinity for feature-based motion segmentation," *Pattern Recogn.*, vol. 44, pp. 454–470, 2011.

[109] C. J. Poelman and T. Kanade, "A paraperspective factorization method for shape and motion recovery," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 19, no. 3, pp. 206–218, 1997.

[110] T. Morita and T. Kanade, "A sequential factorization method for recovering shape and motion from image streams," in *Proceedings of the 1994 ARPA Image Understanding Workshop*, vol. 2, 1994, pp. 1177–1188.

[111] P. Chen and D. Suter, "Recovering the missing components in a large noisy low-rank matrix: application to sfm," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 26, no. 8, pp. 1051–1063, 2004.

[112] P. Anandan and M. Irani, "Factorization with uncertainty," *Int. J. Comput. Vision*, vol. 49, no. 2-3, pp. 101–116, 2002.

[113] T. Okatani and K. Deguchi, "On the wiberg algorithm for matrix factorization in the presence of missing components," *Int. J. Comput. Vision*, vol. 72, no. 3, pp. 329–337, 2007.

[114] T. Wiberg, "Computation of principal components when data are missing," in *Proceedings of the Second Symposium of Computational Statistics*, Berlin, 1976, pp. 229–236.

[115] R. Hartley and F. Schaffalizky, "Powerfactorization: 3d reconstruction with missing or uncertain data," in *Australia-Japan Advanced Workshop on Computer Vision*, 2003.

[116] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore, USA: Hopkins University Press, 1989.

[117] L. Zelnik-Manor and M. Irani, "Temporal factorization vs. spatial factorization," in *Lect. Notes Comput. Sc. (ECCV)*, vol. 2, 2004, pp. 434–445.

[118] A. V. Knyazev and M. E. Argentati, "Principal angles between subspaces in an a-based scalar product: Algorithms and perturbation estimates," *SIAM J. Sci. Comput.*, vol. 23, no. 6, pp. 2008–2040, 2002.

[119] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[120] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, 2003.

[121] R. Vidal and R. Hartley, "Motion segmentation with missing data using powerfactorization and gpca," in *Proc. CVPR IEEE*, vol. 2, 2004, pp. 310–316.

[122] L. Zappella, X. Llado, and J. Salvi, "Rank estimation of trajectory matrix in motion segmentation," *Electron. Lett.*, vol. 45, no. 11, pp. 540–541, 2009.

[123] L. Zappella, X. Lladó, and J. Salvi, "Enhanced model selection for motion segmentation," in *Proc. ICIP IEEE*, 2009, pp. 4053–4056.

[124] L. Zappella, E. Provenzi, X. Lladó, and J. Salvi, "Adaptive motion segmentation algorithm based on the principal angles configuration," in *Lect. Notes Comput. Sc. (ACCV)*, vol. 6494/2011, 2010, pp. 15–26.

[125] R. Vidal, Y. Ma, S. Soatto, and S. Sastry, "Two-view multibody structure from motion," *Int. J. Comput. Vision*, vol. 68, no. 1, pp. 7–25, 2006.

[126] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *Proc. ICCV IEEE*, vol. 2, 2001, pp. 586–591.

[127] L. Zappella, X. Lladó, and J. Salvi, *Motion Segmentation From Tracked Features*, ser. Lectures in Mathematics. LAP LAMBERT Academic Publish. GmbH & Co. KGBirkhauser, 2011.

[128] C. Julià, A. Sappa, F. Lumbreras, J. Serrat, and A. Lpez, "An iterative multiresolution scheme for sfm with missing data: Single and multiple object scenes," *Image Vision Comput.*, vol. 28, pp. 164–176, 2010.

[129] J. Cheeger, "A lower bound for the smallest eigenvalue of the laplacian," *Problems in Analysis*, pp. 195–199, 1970.

[130] A. Björck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Mathematics of Computation*, vol. 27, no. 123, pp. 579–594, 1973.

[131] P.-A. Absil, A. Edelman, and P. Koev, "On the largest principal angle between random subspaces," *Linear Algebra and its Applications*, vol. 414, no. 1, pp. 288–294, 2006.

[132] L. Ambrosio, N. Gigli, and G. Savaré, *Gradient flows in metric spaces and in the space of probability measures*, ser. Lectures in Mathematics. Birkhauser, 2005.

[133] S. J. Orfanidis, *Introduction to Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.

[134] F. Chung, *Spectral Graph Theory*, ser. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[135] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[136] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[137] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[138] C. Zach, I. A., and H. Bischof, "What can missing correspondences tell us about 3d structure and motion?" in *Proc. CVPR IEEE*, 2008.

[139] L. Zappella, A. Del Bue, X. Lladó, and J. Salvi, "Simultaneous motion segmentation and structure from motion," in *Proc. of the IEEE Intern. Conf. on Motion and Video Computing*, 2011, pp. 679–684.

[140] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[141] J. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *International Conference on Machine Learning*. ACM, 2005, pp. 713–719.

[142] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *Information Theory, IEEE Transactions on*, vol. 56, no. 6, pp. 2980–2998, 2010.

[143] A. Cheriyadat and R. Radke, "Non-negative matrix factorization of partial track data for motion segmentation," in *Proc. ICCV IEEE*, 2009, pp. 865–872.

[144] C. Bregler, A. Hertzmann, and H. Biermann, "Recovering non-rigid 3D shape from image streams," in *Proc. CVPR IEEE*, 2000, pp. 690–696.

[145] P. Tresadern and I. Reid, "Articulated structure from motion by factorization," in *Proc. CVPR IEEE*, vol. 2, 2005, pp. 1110–1115.

[146] A. M. Buchanan and A. Fitzgibbon, "Damped newton algorithms for matrix factorization with missing data," in *Proc. CVPR IEEE*, vol. 2, 2005, pp. 316–322.

[147] M. Marques and J. Costeira, "Estimating 3d shape from degenerate sequences with missing data," *Comput. Vis. Image Und.*, vol. 113, no. 2, pp. 261–272, 2009.

[148] J. Costeira and T. Kanade, "A multi-body factorization method for motion analysis," in *Proc. ICCV IEEE*, 1995, pp. 1071–1076.

[149] A. W. Fitzgibbon and A. Zisserman, "Multibody structure and motion: 3-d reconstruction of independently moving objects," in *Lect. Notes Comput. Sc. (ECCV)*, 2000, pp. 891–906.

[150] R. Vidal and R. Hartley, "Three-view multibody structure from motion," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 30, no. 2, pp. 214–227, 2008.

[151] K. Ozden, K. Schindler, and L. Van Gool, "Multibody structure-from-motion in practice," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 32, no. 6, pp. 1134–1141, 2010.

[152] K. Schindler, D. Suter, and W. Wang, Hanzi, "A model-selection framework for multibody structure-and-motion of image sequences," *Int. J. Comput. Vision*, vol. 79, no. 2, pp. 159–177, 2008.

[153] A. Galántai, *Projectors and Projection Methods*, ser. Advances in Mathematics. Assinippi Park, MA, USA: Kluwer Academic Publishers, 2009.

[154] A. Del Bue, J. Xavier, L. Agapito, and M. Paladini, "Bilinear factorization via augmented lagrange multipliers," in *Lect. Notes Comput. Sc. (ECCV)*, vol. 4, 2010, pp. 283–296.

[155] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Jour. Scientic Comput.*, vol. 20, pp. 33–61, 1998.

[156] S. Wright, R. Nowak, and M. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[157] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 28, no. 11, pp. 1768–1783, 2006.

[158] S. Lee and S. Wright, "Implementing algorithms for signal and image reconstruction on graphical processing units," Computer Sciences Department, University of Wisconsin-Madison, Tech. Rep, Tech. Rep., 2008.