SPARSE SUBSPACE CLUSTERING-BASED MOTION SEGMENTATION WITH COMPLETE OCCLUSION HANDLING

by

Jana Mattheus

Submitted in partial fulfillment of the requirements for the degree
Master of Engineering (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

April 2021

SUMMARY

SPARSE SUBSPACE CLUSTERING-BASED MOTION SEGMENTATION WITH COMPLETE OCCLUSION HANDLING

by

Jana Mattheus

Supervisor(s):	Mr Hans Grobler
	Prof Adnan M. Abu-Mahfouz
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Motion Segmentation, Motion Analysis, Sparse Subspace Clustering, Sparse Representation, Spectral Clustering, Manifold Clustering, Computer Vision

Motion segmentation is part of the computer vision field and aims to find the moving parts in a video sequence. It is used in applications such as autonomous driving, surveillance, robotics, human motion analysis, and video indexing. Since there are so many applications, motion segmentation is ill-defined and the research field is vast. Despite the advances in the research over the years, the existing methods are still far behind human capabilities. Problems such as changes in illumination, camera motion, noise, mixtures of motion, missing data, and occlusion remain challenges.

Feature-based approaches have grown in popularity over the years, especially manifold clustering methods due to their strong mathematical foundation. Methods exploiting sparse and low-rank representations are often used since the dimensionality of the data is reduced while useful information regarding the motion segments is extracted. However, these methods are unable to effectively handle large and complete occlusions as well as missing data since they tend to fail when the amount of missing data becomes too large. An algorithm based on Sparse Subspace Clustering (SSC) has been proposed to address the issue of occlusions and missing data so that SSC can handle these cases with

high accuracy. A frame-to-frame analysis was adopted as a pre-processing step to identify motion segments between consecutive frames, called inter-frame motion segments. The pre-processing step is called Multiple Split-And-Merge (MSAM), which is based on the classic top-down split-and-merge algorithm. Only points present in both frame pairs are segmented. This means that a point undergoing an occlusion is only assigned to a motion class when it has been visible for two consecutive frames after re-entering the camera view. Once all the inter-frame segments have been extracted, the results are combined in a single matrix and used as the input for the classic SSC algorithm. Therefore, SSC segments inter-frame motion segments rather than point trajectories. The resulting algorithm is referred to as MSAM-SSC.

MSAM-SSC outperformed some of the most popular manifold clustering methods on the Hopkins155 and KT3DMoSeg datasets. It was also able to handle complete occlusions and 50% missing data sequences, as well as outliers. The algorithm can handle mixtures of motions and different numbers of motions. However, it was found that MSAM-SSC is more suited for traffic and articulate motion scenes which are often used in applications such as robotics, surveillance, and autonomous driving. For future work, the algorithm can be optimised to reduce the execution time so that it can be used for real-time applications. Additionally, the number of moving objects in the scene can be estimated to obtain a method that does not rely on prior knowledge.

LIST OF ABBREVIATIONS

ADMM	Alternating Direction Method of Multipliers
ALC	Agglomerative Lossy Compression
ALM	Augmented Lagrange Multiplier
ASSA	Accurate Subspace Segmentation by Successive Approximations
BP	Basis Pursuit
CNN	Convolutional Neural Network
CS	Compressed Sensing
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
EDSC	Efficient Dense Subspace Clustering
ELSA	Enhanced Local Subspace Affinity
EM	Expectation Maximisation
EMS	Enhanced Model Selection
GPCA	Generalised Principal Component Analysis
GreB	Greedy Bilateral
HQ	Half-Quadratic
IMITEC	International Multidisciplinary Information Technology and Engineering Conference
k-NN	k-Nearest Neighbours
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
KT3DMoSeg	KITTI 3D Motion Segmentation Benchmark
LADMAP	Linearised Alternating Direction Method with Adaptive Penalty
LDA	Linear Discriminant Analysis
LNRSI	Locality-constrained Non-negative Robust Shape Interaction
LRR	Low-Rank Representation
LS3C	Latent Space Sparse Subspace Clustering
LSA	Local Subspace Affinity

M-TPV	Multiple Two Perspective-View
MAP	Maximum A Posteriori Probability
MCC	Maximum Correntropy Criterion
MDL	Minimum Description Length
ML	Maximum Likelihood
MRM	Matrix Rank Minimisation
MS	Model Selection
MSAM	Motion-Split-And-Merge
MSAM-SSC	Motion-Split-And-Merge Sparse Subspace Clustering
MSMC	Multi-Scale Motion Clustering
NaN	Not a Number
NLS3C	Non-Linear Latent Space Sparse Subspace Clustering
PCA	Principal Component Analysis
PF	Particle Filter
RANSAC	Random Sample Consensus
SfM	Structure-from-Motion
SR	Sparse Representation
SSC	Sparse Subspace Clustering
SVD	Singular Value Decomposition

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

Motion segmentation forms part of the computer vision field and aims to find the moving objects within a video sequence. It is in applications such as surveillance, robotics, autonomous driving, and gait analysis. The motion segmentation problem is not well defined since it is dependent on the application [1]. This has lead to vast and active research field.

Motion segmentation methods are either dense or feature-based. Dense-based methods use all the image pixels as input while feature-based methods extract point trajectories and use these trajectories to infer the moving objects. The dense-based methods can roughly be categorised according to the key principles followed, namely image difference, optical flow, wavelet-transforms, statistical approaches, and layered approaches [2]. The main idea of image difference methods is to use the intensity difference and thresholding to identify motion between frames [3, 4, 5]. This is one of the traditional approaches which has lost popularity over the years. Optical flow methods use the apparent motion of points between frames to identify moving objects with additional methods to identify motion boundaries [6, 1, 7]. Optical flow is often combined with other techniques such as Expectation Maximisation (EM) and deep learning. Moving segments can also be identified through analysis of the frequency components of the frames by applying wavelet transforms such as the Direct Cosine Transform (DCT) [8, 9, 10, 11]. Statistical analysis is another popular approach. Commonly used statistical approaches are Maximum A Posteriori Probability (MAP), Particle Filter (PF), and EM [1, 12, 13]. Layered approaches assign moving parts to different depth levels which allow the moving parts to move behind each other and be obscured from the camera view [14, 15]. These methods can handle complete occlusions.

With feature-based methods, the aim is to group point trajectories. Manifold clustering is a feature-based approach which investigates the data subspaces and uses the information regarding the manifolds to determine the motion segments. The advantage of feature-based methods is their strong mathematical basis which can be utilised to learn and extract information from the data.

Some approaches can be used for either dense or feature-based motion segmentation. Template matching identifies motion by comparing frames to a template [16, 17]. Deep learning methods use architectures such as neural networks to learn the motion segments [18, 19, 20]. Additionally, some methods can be used to segment the motion of 3D input data such as point clouds. Deep learning, template matching, EM, and optical flow have been used to achieve 3D motion segmentation [21, 17, 22, 23].

1.1.2 Research gap

Even though the research field is vast and active, there currently is no method with performance close to human capabilities. Despite all the research efforts, motion segmentation still faces many challenges. Since motion segmentation is very problem-specific, most methods are tailored to suit specific applications. For example, in surveillance and autonomous driving applications, the aim is to learn the moving objects from the video, not necessarily the moving parts of each object, therefore, these methods tend to focus on rigid and independent motions. On the other hand, applications such as sports analysis focus on extracting the moving parts of the athlete to be able to analyse the sport technique. As a result, there is a lack of generic algorithms which can segment a variety of motion types.

Another issue is the number of motions present within the scene. The problem complexity significantly increases with the increase in the number of motions. Methods such as Generalised Principal Component Analysis (GPCA) can only segment a limited number of motions [24]. In many real-world scenarios, the video data will be corrupted by noise and outliers caused by changes in the illumination, camera motion or camera properties. Existing methods have varied performance depending on the degree of data corruption. Methods must be able to handle corrupt data so that accurate results are obtained. Missing data is also a concern and can be caused by noise, changes in illumination and camera motion. For dense based methods, missing data can cause inaccurate motion boundaries. For feature-based methods, missing data causes incomplete trajectories which can complicate the clustering process since some algorithms are unable to associate a broken trajectory with the trajectories from the

same motion.

One of the biggest challenges motion segmentation faces is the handling of large and complete occlusions. Many methods can only handle partial occlusions where most of the moving object is still visible from the camera view. A few dense-based methods, such as layered or deep learning optical flow methods, exist which focuses on solving the occlusion problem, but these methods tend to have a high complexity with long execution times. No feature-based method exists which solves the complete occlusion problem while being able to segment a wide variety of mixtures of motions.

Feature-based methods, such as manifold clustering approaches, have a strong mathematical foundation which can be exploited to simplify the extraction of motion segments and their shapes. Therefore, manifold clustering is an ideal starting point for developing a method that can handle complete occlusions and missing data. Since these methods focus on exploiting information from the subspaces, these methods can be applied to other domains such as classification and recognition. Additionally, since only a number of key points are used rather than the entire video sequence, it is possible to have shorter execution times. Further, manifold clustering approaches can be used with Structure-from-Motion (SfM) which constructs a 3D model of the object and camera motion. The majority of these methods can segment mixtures of different motion types. However, these methods are unable to effectively handle large and complete occlusions and tend to produce inaccurate results or fail under these conditions. This research aims to provide a generic feature-based algorithm that can handle large and complete occlusions and missing data such that accurate motion segments of mixtures of different motions can be produced under these conditions.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

- Is it possible to develop a manifold clustering approach that can segment motion objects that are completely occluded for consecutive frames in a video?
- Is it possible to develop a manifold clustering approach that can handle large percentages of missing data without compromising on the accuracy of the segmentation?
- Is it possible to develop a more generic manifold clustering algorithm that can extract mixtures of motions, i.e., a combination of independent, dependent and partially dependent motions, as well as rigid, non-rigid, articulated, and degenerate motions?

1.3 HYPOTHESIS AND APPROACH

The manifold clustering-based method is developed to segment the motion regions in a video. Like existing feature-based methods, independent, dependent and partially dependent motions, as well as rigid, non-rigid, articulated, and degenerate motions, are segmented. The method is designed to segment motions under partial and complete occlusions. The different motion regions are given as output. An automated evaluation framework is created to evaluate the performance of the proposed method. The evaluation framework computes performance metrics on the output. These metrics are compared to that of current state-of-the-art methods. As output of the evaluation framework, graphical representations of the metrics are given.

1.4 RESEARCH GOALS

The objective is to develop a manifold clustering method that can effectively handle large and complete occlusions such that accurate motion segments are produced under these conditions. The proposed method must also be able to produce accurate results when large portions of the data is missing. Additionally, the method must be able to handle mixtures of motion as well as different numbers of motion to provide a generic solution that can be used in different applications such as surveillance, sports analysis and robotics. The method must also have a performance that is comparable to that of current manifold clustering methods. The performance will be acceptable if the performance metrics are no more than 5% lower than that of the state-of-the-art methods.

1.5 RESEARCH CONTRIBUTION

The focus is to solve the occlusion problem as well as handling missing data that manifold clustering-based methods face. Therefore, the method must be able to handle partial and complete occlusions as well as missing data. The method must also be able to handle mixtures of motion and different numbers of motion with a performance close to that of existing methods.

1.6 RESEARCH OUTPUTS

A review paper, titled *A Review of Motion Segmentation Approaches and Major Challenges*, was published in the Proceedings of the 2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC). In this paper, the advances in the field of motion segmentation over the years were analysed and areas where more research is needed, were identified. A paper that presents and describes the performance of the SSC-based algorithm proposed in this work is being prepared and will be submitted for publication.

1.7 OVERVIEW OF STUDY

A literature review on motion segmentation methods was conducted and is presented in Chapter 2. The review focused on the approaches to motion segmentation with specific focus on manifold clustering methods. Current methods which employ occlusion handling were also investigated. Additionally, the focus was placed on the types of motion that can be segmented as well as the number of motions. It was found that these methods can handle small partial occlusions since only key points are used to segment motion. Most methods are also able to segment independent and dependent motions, but some can only handle a limited in the number of motions before the performance is significantly impacted. The literature review revealed that most methods are unable to handle large and complete occlusions, and missing data since most methods tend to either produce inaccurate results, fail, or ignore these cases. It is clear that a process for handling these cases is needed.

In Chapter 3, the development of the manifold clustering-based method is presented, including optimisations. An automated evaluation framework was also created to evaluate the performance of the proposed method. The evaluation framework runs the proposed algorithm as well as other methods and computes the misclassification error for each algorithm. The misclassification error is used to generate reports on the algorithm performance. As output, a table giving the average, median and standard deviation of the misclassification error is produced, as well as a histogram plot of the misclassification error distribution. Additionally, the average, median and standard deviation of the execution time are tabulated for each of the datasets. These results are presented in Chapter 4. In Chapter 5, observations and discussions of the results are given. Final conclusions and future work are given in Chapter 6.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

In Section 2.2, an overview of motion segmentation and the existing approaches is given, as well as the major challenges the field still faces. The existing approaches are compared and the reasons for focusing on manifold clustering approaches are given. Then, some of the most important and most popular manifold clustering approaches are discussed in Section 2.3. This is followed by an investigation into existing algorithms that focuses on occlusion handling and dealing with missing data in Section 2.4. Lastly, datasets that can be used to benchmark manifold clustering approaches are considered in Section 2.5.

2.2 APPROACHES TO MOTION SEGMENTATION

The motion segmentation research field is vast and many approaches have been developed to solve the problem. Since many methods focus on a specific application, often these methods do not generalise well. When designing a solution to the motion segmentation problem, it is important to keep several factors in mind.

The first attribute to consider is the *input data*, which can either be a sequence of frames, a set of point locations or a 3D point cloud or volumetric data. The data will determine the representation of the motion segments, i.e., dense- or feature-based representation. Dense-based methods perform pixel-wise motion segmentation which produces more precise segments, but the occlusion problem is more difficult to solve. Feature-based methods rely on several key points to represent the motion segments, which allows objects to be tracked easily under partial occlusions. In some instances, the camera is not stationary and all points in the video material have a motion parameter caused by the *camera motion*. Additionally, it is possible to receive input from *multiple cameras* mounted in different positions. In these cases, the depth of points can be inferred and used to segment the moving

objects.

It is also important to consider if any information regarding the scene is known beforehand, such as the number of motions, since any *prior knowledge* can be used to significantly simplify the problem and generate supervised or semi-supervised methods [25, 22]. Prior knowledge also includes training data if the method, such as neural networks, includes a training step. In order to improve the final segmentation results *spatial continuity* is exploited to ensure that the segments are connected and consistent [8, 9, 1, 7, 14, 22].

Depending on the application, the type of motion that must be identified and segmented is important and is used to define the segmentation problem. Motion is described using two attributes, namely the *dependency* and the *type* [2]. The dependency describes the relationship between pairs of moving objects, namely independent, dependent, or partially dependent. The type describes the kind of motion, namely rigid, non-rigid, articulated, and degenerate. An object has *rigid* motion if its shape remains unchanged over time, i.e., the relative distance between points and their relative positions remain constant over time. *Non-rigid* motion can be described by a set of rigid shapes known as the base shapes [17]. *Articulated* motion consists of parts whose motion is partially dependent on the motion of the other parts since they are connected by a joint, e.g. human limbs or robotic arms [25]. Objects which change shape and deforms over time undergo *degenerate* motion and their corresponding subspace has a lower dimension than the theoretical maximum [26]. In many instances, multiple moving objects, or *mixtures of motion*, are present in a scene and the motion of each object must be segmented simultaneously [4]. Additionally, objects are permitted to *enter* or *leave* the scene which can pose challenges for motion segmentation methods.

Another important consideration is handling the major challenges motion segmentation still faces, namely occlusions, missing and corrupt data. *Occlusions* occur when moving objects momentarily disappear from the camera view. Most methods can handle partial occlusions (when small parts of an object are obscured from the camera view) and fail when complete occlusions are encountered. In some cases, a moving object *stops temporarily* which is another challenge many methods are unable to handle [27, 28]. *Missing data* is caused by occlusions, temporary stopping of objects, noise, outliers and changes in lighting conditions.

There are many approaches used to solve the motion segmentation problem and are characterised by

the main principle followed. In some instances, methods use a combination of approaches to generate a more robust algorithm. The main approaches used are Image Difference, Optical Flow, Wavelet, Layered, Statistics, Manifold Clustering, Template Matching and Deep Learning [2].

Image difference is one of the earliest approaches to detect changes in a video sequence by applying a threshold to the difference in intensity between frames [3, 4, 5]. These methods can handle small partial occlusions but tend to be sensitive to noise, temporary stopping of objects, changes in the illumination and cameras motion [27]. The method in [3] addresses these issues by employing ego-motion compensation and statistical background modelling. The method in [5] models the background using statistical methods which allows temporary stopping and missing data to be handled effectively.

Optical flow is one of the most popular approaches used in computer vision. Optical flow by itself cannot identify the motion boundaries accurately and additional methods are required to do so [27]. Additional methods are also needed to handle occlusions and temporary stopping [6, 1, 7]. This is solved in [6], where a segmentation-embedded optical flow method is presented which determines accurate flow fields and high-quality motion boundaries. Optical flow can also be used alongside other approaches such as deep learning to achieve accurate segmentation [29]. By using a Convolutional Neural Network (CNN) architecture, the method in [29] can handle occlusions and the temporary stopping of objects.

Wavelet-based methods employ wavelet transforms to analyse frequency components of video frames [8, 9, 10, 11]. These methods can only handle simple motions, and stationary cameras [27]. The method in [8], applies double change detection on Daubechies complex wavelet coefficients across three consecutive frames to extract motion segments while reducing the shift sensitivity and improving the edge detection. To address the occlusion problem, multi-resolution analysis can be used to extract depth planes which can be used to handle large and complete occlusions.

Layered approaches incorporates depth information by layering frames according to the number of moving objects [14, 15]. The layers have a depth parameter, motion, and motion visibility parameters. These methods can handle large and complete occlusion but have long execution times due to their complexity [27]. In [14], the motion segments and layering are obtained by minimising the objective

function of the $\alpha\beta$-swap and α-expansion algorithms. Additionally, spatial continuity is exploited. The method can successfully segment rigid and articulated motion.

Statistical methods use statistical models to extract motion segments. The ability of the statistical model to describe real-world problems has a direct influence on the accuracy of these methods. Multiple motions, occlusions and temporary stopping can be handled. The traditional statistical methods require prior knowledge, but many state-of-the-art methods automatically estimate and refine these parameters [1, 13, 12]. Three of the most popular statistical approaches are MAP, PF, and EM. MAP uses Bayes rule determine the pixel-level segmentation that maximises the posterior probability [1, 30]. The MAP method in [1] uses colour and motion information to segment objects from a video. PFs use samples (or particles) to represent some posterior distribution. The changes in the samples are tracked over time to form a representation of the probability density function [13, 31]. The method in [13], addresses the dependency on parameterisation encountered by finite representations by using geometric active contours to eliminate the need for parametrisation while allowing for topology changes. Additionally, this significantly reduces the number of particles needed. The EM algorithm computes the Maximum Likelihood (ML) estimate when missing or hidden variables are present. In [12], the mixture of dynamic textures is used in an EM framework to segment a video into moving parts. This method can extract transparent motions such as smoke and fire, and therefore its characterises differ from the MAP and PF methods mentioned here.

Manifold clustering approaches are feature-based and project the data to a subspace with a lower dimensionality while preserving some properties of the original data-space [27]. Currently, there are many different approaches and methods and a few of the most popular methods are considered here. Factorisation approaches exploit properties of the subspace to extract and distinguish between different types of motion [25, 22]. Each motion has an associated subspace containing the points belonging to the motion. The subspaces have specific properties due to the motion type. Areas, where the subspaces of the different motion segments intersect, are called joints and indicate a dependency between the overlapping motion segments such as with articulate parts. GPCA is one of the earliest manifold clustering approaches which uses a set of polynomials to represent a group of subspaces [24, 32]. Algebraic geometry is used to compute the set polynomials and find the final segmentation. The performance of these methods deteriorates drastically for more than four moving objects.

Agglomerative Lossy Compression (ALC) uses lossy compression and rank minimisation to obtain

a sparse representation of the point trajectories [33]. The final segments are obtained by employing spectral clustering. Sparse Subspace Clustering (SSC) represents each point by a linear or affine set of points from the same subspace [34]. These coefficients are computed by minimising the ℓ_1-norm and spectral clustering is used to find the final segmentation. Random Sample Consensus (RANSAC) is often used as the base to segment different types of motion [35]. Automated processes are required to eliminate the need for prior knowledge. Local Subspace Affinity (LSA) segments motion by grouping point trajectories that generate similar subspaces together. It relies on parameters that have to be tuned *a priori* depending on the noise level and the number of motions present in the scene [27]. Normalised Cuts are used to estimate the number of motions, but it is unreliable. Methods such as Enhanced Local Subspace Affinity (ELSA) address the limitations of classic LSA [27]. ELSA uses Enhanced Model Selection (EMS) to automatically estimate and refines its parameters.

Manifold clustering methods can be used with SfM to construct a 3D model of the moving object and to compute the camera motion. Most of these methods can segment any type of motion, and [35, 34, 33, 24, 27] can segment multiple objects. The methods in [25, 34, 33, 35, 27] rely on prior knowledge. Partial occlusion can be handled by the methods in [34, 33, 24, 27].

Template matching methods compares the input to a template to identify moving objects. The quality of the template influences the performance, e.g., segments may not be recovered if they are occluded in the template. If the template has a high quality and all the moving parts are visible, complete occlusions can be handled since objects will be re-detected once they come into the camera view again. Other aspects that influence the performance are changes in the illumination, non-rigid transformations, and background noise. The method in [17] overcomes these issues, and can extract the articulated parts of an object from a set of 3D point clouds. The 3D point clouds depict the different positions of the object.

Deep learning approaches employ machine learning to segment motion. These methods are trained to find the motion segments on a variety of scenes and types of motion and have a greater accuracy than more traditional approaches. The method in [21] computes the deformation flow of a set of 3D point clouds to find the articulated parts of an object.

Table 2.1 shows the summary of the most important aspects of the investigated methods and follows the structure used in [2]. The methods are categorised as Image Difference, Optical Flow, Wavelet,

Table 2.1. Summary of important attributes of investigated methods. Adapted from [2], ©[2020] IEEE

Approaches		Ref.	Motion Representation (D Dense, F Feature)	Input	Prior Knowledge (N Cluster Number, T Training, S subspace Dimension)	Moving Camera	Multiple Cameras	Occlusions	Missing Data	Temporary Stopping	Spatial Continuity	Multiple Objects	Objects Enter (E) or Leave (L)	Dependency (I Independent, D Dependent)	Type (R Rigid, N Non-rigid, A Articulated, D Degenerate)
Image Difference		[3]	D	Video		Y	N	Y	Y	Y	Y	Y	EL	I	RNA
		[5]	D	Video		N	N	Y	Y	Y	Y	Y	EL	I	RA
Optical Flow		[6]	D	Stereo pair		Y	Y	Y	Y	Y	N	Y	No	I	RN
		[29]	D	Video	T	Y	N	Y	Y	Y	Y	Y	EL	I	RNA
Wavelet		[8]	D	Video		N	N	Y	Y	Y	Y	Y	E	I	RA
Statistical	MAP	[1]	D	Video		Y	N	Y	Y	N	Y	Y	EL	I	RA
	PF	[13]	D	Video		Y	N	Y	Y	N	Y	Y	No	I	RNA
	EM	[12]	D	Video		N	N	N	N	Y	Y	Y	EL	I	RNAD
Layers		[14]	D	Trajectories		Y	N	Y	Y	Y	Y	Y	EL	IDP	RNA
Manifold Clustering	Factorization	[25]	F	Trajectories	NS	Y	N	N	N	Y	Y	Y	EL	IDP	RNAD
		[22]	F	Trajectories		Y	N	N	Y	Y	Y	Y	No	IDP	RA
	GPCA	[24]	F	Trajectories		Y	Y	Y	Y	Y	Y	Y	EL	IDP	RAD
	ALC	[33]	F	Trajectories	N	Y	N	Y	Y	Y	Y	Y	EL	IDP	RAND
	SSC	[36]	F	Trajectories		Y	N	Y	Y	Y	Y	Y	EL	IDP	RNA
	RANSAC	[35]	F	Trajectories	NS	Y	N	N	N	N	Y	N	N/A	IDP	RNA
	ELSA	[27]	F	Trajectories	N	Y	N	Y	Y	Y	Y	Y	EL	IDP	RA
Template Matching		[17]	F	3D Point Clouds		N/A	N/A	Y	Y	N/A	Y	N	N/A	IDP	RNA
Deep Learning		[21]	F	3D Point Clouds	T	N/A	N/A	Y	Y	N/A	Y	N	N/A	IDP	RA

Statistical, Layers, Manifold Clustering, Template Matching and Deep Learning. "N/A" is used to indicates that an aspect is not relevant to the specific method. From Table 2.1, it is clear that none of the methods addresses all the factors since the methods were designed to address the motion segmentation problem for a specific application. Additionally, different approaches can be used for the same motion segmentation task, but some are more suited for specific applications than others.

A motion segmentation problem can be to find the independent moving objects for applications such as surveillance and traffic analysis. From Table 2.1, it can be seen that ideal methods for this application are the image difference-based methods in [3, 5], optical flow approaches in [6, 29], wavelet approach of [8], and the statistical methods in [1, 13, 12]. These methods are dense-based and can accurately determine the motion boundaries of objects undergoing complex motions such as articulate motion. However, these methods cannot extract the articulate or non-rigid parts of an object.

For applications such as robotics and human motion analysis, the articulated parts of an object are of interest and must be extracted. The articulate parts exhibit partially dependent motion. From Table

2.1, it is evident that all the manifold clustering can handle articulate motion as well as the template matching approach in [17], and the deep learning method in [21]. The manifold clustering methods in [24, 27] can only extract a limited number of articulate parts. The layered approach in [14] is the only method considered here that can accurately segment objects that undergo complete occlusions. Note that manifold cluster, layered, template matching and deep learning methods can be used to segment other types of motion as well.

Sometimes a more generic solution is required to segment mixtures of different types of motion. Therefore, the method must be able to handle different numbers of moving objects that are subjected to different types of motion. From Table 2.1, it is evident that the manifold clustering and layered methods provide a more generic solution. Generic methods are often limited and sometimes do not perform as well as methods designed for a specific application but they can be used in a wider variety of applications. Another disadvantage is these methods are unable to handle occlusions and missing data effectively since the generic motion segmentation increase the complexity of these issues.

Motion segmentation can also be used to address specific real world problems such as smoke and fire detection. Here, the method must be able to find complex and transparent motion regions. The methods in [12, 11] were designed for this specific problem and includes texture information to segment the motion of smoke and fire.

From the analysis of the different motion segmentation approaches, it is evident that manifold clustering approaches are versatile since they can handle mixtures of motion and can be applied to different motion segmentation applications. The majority of the investigated manifold clustering methods can handle partial occlusions, temporary stopping, a degree of corrupt data, and camera motion. However, large and complete occlusions and missing data remains a challenge. Taking these observations into consideration along with the strong mathematical basis, manifold clustering is an effective approach to the motion segmentation problem and is, therefore, the main focus of this work.

2.3 MANIFOLD CLUSTERING ALGORITHMS

Manifold clustering algorithms are feature-based algorithms that find the final motion segments by exploiting the information of the data subspaces. As seen previously, these methods are versatile and can be applied to many computer vision applications. Here, current and important manifold clustering methods are investigated in depth.

2.3.1 Local Subspace Affinity

LSA solves the motion segmentation problem by finding linear manifolds since trajectories from the same motion reside on the same manifold [37]. First, LSA constructs and normalises a $2F \times N$ trajectory matrix $\mathbf{X}$ from the N point trajectories extracted from the F frames. The trajectories are projected to a $\mathbb{R}^{r_{true}}$ unit sphere, where r_{true} is the rank of the trajectory matrix $\mathbf{X}$. The rank is estimated using Model Selection (MS) and is defined as

$$r_{est} = \arg\min_{r} \left(\frac{\lambda_{r+1}^2}{\sum_{k=1}^{r} \lambda_k^2} + \varkappa r \right), \tag{2.1}$$

where λ_i is singular value number i. The effective rank is zero if the sum of all the squared singular values is below a threshold. Parameter $\varkappa$ is dependent on the amount of noise present in the data and must be assigned larger values when the noise levels are high.

The projection of the trajectories onto the unit sphere is achieved by applying Singular Value Decomposition (SVD) to $\mathbf{X}$ and taking only the first r_{true} singular values. The trajectory points in the transformed $\mathbb{R}^{r_{true}}$ space reside in the same local subspace as their closest neighbours. Therefore, the subspace can be estimated from a point and its n closest neighbours. SVD is used to achieve this. The rank of the local subspace is required for the SVD operation, and MS is used once again to estimate the rank. The distance between the two subspaces is measured using the principal angle and is used to construct an affinity matrix. The principal angle between the subspaces $\mathbf{S}(\alpha)$ and $\mathbf{S}(\beta)$ of points α and β, respectively, is defined as

$$cos(\theta_m) = \max_{\mathbf{u} \in \mathbf{S}(\alpha), \mathbf{v} \in \mathbf{S}_2(\beta)} \left(\mathbf{u}^T \mathbf{v} \right) = \mathbf{u}_m^T \mathbf{v}_m, \tag{2.2}$$

where

$$||\mathbf{u}|| = ||\mathbf{v}|| = 1, \quad \mathbf{u}^T \mathbf{u}_i = 0, \quad \mathbf{v}^T \mathbf{v}_i = 0, \quad i = \{1, \ldots, m-1\}. \tag{2.3}$$

Here, $\mathbf{u}_i$ and $\mathbf{v}_i$ denote the principal vectors. The affinity is computed using the principal angle as

$$\mathbf{A}_{\alpha,\beta} = \exp\left(-\sum_i \sin^2(\theta_i) \right). \tag{2.4}$$

The affinity matrix is used to find the motion segments. The affinity matrix is a $N \times N$ matrix containing the weights between the N nodes of a graph, where each trajectory is a node. Recursive two-way spectral clustering is used to cluster the affinity matrix.

LSA can segment mixtures of motions and, therefore, can operate on affine subspaces. Geodesic and spatial continuity constraints are enforced. No prior knowledge regarding the number of manifolds or their dimensions is required beforehand. However, the number of motions is required [27].

2.3.2 Enhanced Local Subspace Affinity

ELSA builds on the LSA approach by improving the shortcomings of the classic LSA algorithm [27]. Both algorithms use the distance between the underlying subspaces to compute an affinity matrix. Spectral clustering is applied to the affinity matrix to determine the final segmentation. Unlike LSA which uses classic MS to estimate the rank of the trajectory matrix, ELSA employs EMS. The relation between the estimated rank and the affinity matrix is used to determine an appropriate value for the variable $\varkappa$, which is used in the classic MS process to estimate the rank of a matrix as seen in Equation (2.1). By exploiting this relationship, the need for prior knowledge regarding the number of motions and the noise level is eliminated. Additionally, EMS can handle different types of motion, as well as mixtures of motions.

Since classic LSA uses MS on two occasions, first, to project the feature trajectories to the unit sphere and second, to estimate the subspace of the projected points, two values, namely $\varkappa_g$ and $\varkappa_s$ are computed. Inappropriate values for $\varkappa_g$ and $\varkappa_s$, especially $\varkappa_g$, leads to an incorrect affinity matrix that cannot be used to find the true motion segments. When estimating the rank during the subspace estimation step (estimating the rank for the second time), it was observed that if the estimated rank larger than the true rank, $r_{est} > r_{true}$, the trajectory is represented its r_{true} true singular values, but also by $r_{est} - r_{true}$ unrelated components obtained from the null space of the original trajectory matrix $\mathbf{X}$. If $r_{est} < r_{true}$, the subspace is only represented by a subset of its true components. Additionally, if the estimated rank is close to the true rank, the principal angle between points from the same subspace is significantly smaller than that of points from different subspaces. Hence, the principal angle between trajectory points from the same motion is small while those between points from different motions are large. For a significantly small rank, the principal angles tend to zero, while an overestimated rank causes the principal angles to tend to $\frac{\pi}{2}$. Therefore, EMS uses entropy to evaluate the quality of the computed affinity matrix, and also to indicate the quality of the estimated value for $\varkappa_g$. The entropy is measured as

$$E\left(\mathbf{A}(r)\right) = -\sum_{i=0}^{1} h_{\mathbf{A}(r)}(i) \log_2(h_{\mathbf{A}(r)}(i)), \tag{2.5}$$

where $h_{\mathbf{A}(r)}(i)$ is the histogram count of bin i. If the rank is underestimated, the affinity values tend to one which causes the entropy to be small. An overestimation causes the affinity values to tend to zero which also causes the entropy to be small. As the estimated rank tends closer to the true rank, the entropy increases.

Once the value for $\varkappa_g$ has been determined, the value of $\varkappa_s$, which is used for the first rank estimation, is chosen such that $\varkappa_s > \varkappa_g$. This constraint is desirable since the noise level is higher when estimating the rank of the subspaces (estimating the rank for the second time) due to the small number of samples used. Similar to classic LSA, two-way spectral clustering is used to segment the affinity matrix into the final motion segments. However, the number of motions is required *a priori*, to stop the clustering process. If this knowledge is unknown, another stopping condition is required, such as when the minimum cut exceeds a threshold.

ELSA builds on the work of [38] and uses the fact that obtaining the minimum cut is equivalent to applying a threshold to the second smallest eigenvector of the Laplacian $\mathbf{L}$:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.6}$$

where $\mathbf{D}$ is a $N \times N$ diagonal matrix, which contains the sum of the weights connecting this node and the rest, and $\mathbf{A}$ is the affinity matrix. The Symmetric Normalised Laplacian $\mathbf{L}_{sym}$ can also be used and is defined as

$$\mathbf{L}_{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}. \tag{2.7}$$

A threshold value is still required as a stopping condition. Linear Discriminant Analysis (LDA) approach is used to determine the threshold. The process of using the eigenvalue spectrum to estimate the number of motions resembles a classification problem in which the data is divided into two classes. This approach attempts to find a balance between the inter-class and intra-class variance. The threshold t is computed as

$$\arg\max_t \frac{Q_1(\mu_1(t) - \mu_{all})^2 + (1 - Q_1)(\mu_2(t) - \mu_{all})^2}{Q_1\sigma_1^2(t) + (1 - Q_1)\sigma_2^2(t)}, \tag{2.8}$$

where μ_1 and σ_1 are the mean of the first class and μ_2 and σ_2 of the second, and μ_{all} is the mean of all the eigenvalues. Q_1 is a weight used to prefer the first class over the second. The numerator is a measure of the inter-class dissimilarity while the denominator measures the intra-class dissimilarity. The threshold is estimated such that the inter-class dissimilarity is maximised while the intra-class dissimilarity is minimised.

ELSA was reported to outperform ALC on the Hopkins155 dataset for sequences with no or low noise levels. However, ALC has a higher accuracy than ELSA for high levels of noise and corrupt data.

2.3.3 Generalised Principal Component Analysis

GPCA combines algebra and geometric principles to achieve subspace clustering [32]. The subspaces are represented by polynomials of the same degree as the number of subspaces. These polynomials can be estimated linearly from the data if the number of subspaces (i.e., the number of motions) is known. This reduces the segmentation problem and only one data point per subspace is classified. A distance function minimisation problem is used to select these points. Principal Component Analysis (PCA) is used to evaluate the derivatives of the polynomials at the selected points and a basis for each subspace is computed. Variations of the classic GPCA method exist that allows the GPCA method to handle data with high dimensionality and an unknown number of subspaces. GPCA can handle small amounts of noise. GPCA performs well on data with a low dimensionality (three or fewer motions) but becomes increasingly less accurate with a longer execution time as the dimensionality of the data increases. This is caused by the linear nature of GPCA in which the non-linear constraints on the coefficients are neglected since the polynomials are estimated linearly. Furthermore, GPCA is unable to handle outliers. GPCA is an algorithm that is used to cluster points from the same subspace by identifying the underlying subspaces. First, the subspaces and their dimensions are computed.

Each subspace S_i can be represented with a normal vector $\mathbf{b}_i$, and these normal vectors are linearly independent. A point $\mathbf{x}$ residing in subspace S_i must satisfy the following homogeneous polynomial:

$$p_n(\mathbf{x}) = \prod_{i=1}^{n} \mathbf{b}_i^T \mathbf{x} = 0. \tag{2.9}$$

Using this polynomial, the subspace S_i can be obtained by solving for all the normal vectors $\{\mathbf{b}_i\}_{i=1}^{n}$. Since this problem is non-linear, it can be made linear by projecting the problem to a higher dimensional space. Consider $R_n(K) = R_n[\mathbf{x}_1; \ldots; \mathbf{x}_K]$, the set of homogeneous polynomials. $R_n(K)$ can be transformed to the vector space, and consists of $\mathbf{x}^{\mathbf{n}} = \{x_1^{n_1} x_2^{n_2} \ldots x_K^{n_K}\}$, a set of M_n monomials. The projection of the problem from $\mathbb{R}^K$ to higher dimension $\mathbb{R}^{M_n}$ is achieved using

$$p_n(\mathbf{x}) = \mathbf{c}_n^T v_n(\mathbf{x}) = 0, \tag{2.10}$$

where $\mathbf{c}_n = \{c_{n_1}, \ldots, c_{n_K}\}$ are the coefficients of monomial $\mathbf{x}^n$, and $v_n(\mathbf{x}) = \mathbf{x}^{\mathbf{n}}$ is the Veronese map. This can be extended for the case of $N > M_n$ points:

$$\mathbf{c}_n^T \mathbf{V}_n \doteq \mathbf{c}_n^T \begin{bmatrix} v_n(\mathbf{x}^1)^T \\ \vdots \\ v_n(\mathbf{x}^N)^T \end{bmatrix} = 0. \tag{2.11}$$

From this linear system, it can be seen that the set of coefficients $\mathbf{c}$ can only be computed if the number of subspaces n is known. Therefore, n is estimated using

$$n = \min(i : rank(\mathbf{V}_i) = M_i). \tag{2.12}$$

Once the set of coefficients $\mathbf{c}$ has been computed, the normal vectors $\mathbf{b}$ can be determined using Equation (2.9). PCA is applied to the normal vectors to compute a basis B_i for each subspace S_i. These bases are used to cluster the points by labelling point $\mathbf{x}_j$ to subspace S_i using

$$i = \min_{q=1,\ldots n} ||B_q^T \mathbf{x}_j||. \tag{2.13}$$

Each subspace represents a motion segment.

2.3.4 Agglomerative Lossy Compression

ALC is a subspace separation method, inspired by lossy compression, the sparse representation and rank minimisation [33]. ALC achieves segmentation by computing the segmentation which most simply describes the data. The simplicity of the data can be described by the minimum coding length, which has been applied to the segmentation problem before. The agglomerative clustering algorithm from [39] is used to compute the coding length of subsets of the data. This algorithm has only one parameter, namely the distortion level. Additionally, data from multiple subspaces with mixed dimensionalities can be segmented and is suited for segmenting mixtures of motions. The computed coding length is then used to define the distance between subsets and is used to determine the matrix rank. The data can also be represented with a sparse structure, which is robust and allows missing and corrupt data to be handled. The sparse structure is computed using convex optimisation the nuclear norm. There is a strong relation between the coding length and sparsity, and ALC exploits these similarities. Subspace separation is achieved with Matrix Rank Minimisation (MRM), which aims to partition data matrix $\mathbf{X}$ into sub-matrices $\{\mathbf{X}_n\}_{n=1}^N$ such that each $\mathbf{X}_n$ has the lowest dimension possible and is maximally rank deficient.

MRM is difficult to solve since the function is not smooth or convex. However, for small ranks, minimising the rank over a convex domain is the same as minimising the nuclear norm. This minimisation problem is solved by employing semi-definite programming. The minimum rank of $\mathbf{M}^* \in \mathbf{\Omega}$, where $\mathbf{\Omega}$ is a convex set of symmetric positive semidefinite matrices, is computed using

$$\mathbf{M}^* = \arg\min_{\mathbf{M}\in\mathbf{\Omega}} J_\delta(\mathbf{M}) \doteq \log_2 det(\mathbf{I} + \frac{1}{\delta}\mathbf{M}), \tag{2.14}$$

where δ is a positive regularising parameter. Function J_δ is approximately the sum of the logarithm of the singular values. It is non-convex and smooth and has the same minimum as the true rank

function. This function can be changed to resemble the principle of Minimum Description Length (MDL). Therefore, the following function can be used to estimate the number of bits needed to encode $\mathbf{X}_n$:

$$\begin{aligned} L(Y_n, \varepsilon) &\doteq \frac{D+P_n}{2} J_{\varepsilon^2/D}\left(\frac{1}{P_n}\mathbf{X}_n\mathbf{X}_n^T\right) \\ &\doteq \frac{D+P_n}{2}\log_2 det\left(\mathbf{I}+\frac{1}{P_n\varepsilon^2}\mathbf{X}_n\mathbf{X}_n^T\right), \end{aligned} \tag{2.15}$$

where ε^2 is the distortion. Here, $M = \frac{1}{P_n}\mathbf{X}_n\mathbf{X}_n^T$ and $\delta = \varepsilon^2/D$. To partition $\mathbf{X} \in \mathbb{R}^{D\times P}$ into disjoint subsets $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_N]$ of size $P_1 + \cdots + P_N = P$, the number of bits need for the encoding is determined as

$$L^s([\mathbf{X}_1, \ldots \mathbf{X}_N], \varepsilon) \doteq \sum_{n=1}^{N} L(\mathbf{X}_n, \varepsilon) - P_n \log_2 \frac{P_n}{P}. \tag{2.16}$$

The distortion level is determined from the statistics of the data heuristically and is inversely proportional to the number of motions N. Therefore, it is assumed that the number of motions is known *a priori* and used is to calculate ε. The value for ε is determined by executing ALC on the data for different values of ε. Any values which do not result in the correct number of segments are discarded. Then, all the distinct computed segmentations are determined. The value, from the remaining set of values for ε, which minimises the coding length for the most segmentation, is selected as the final value. This voting scheme is simple and provides accurate practical results, but it is not optimal. As already stated, the coding length is used to compute the rank of the sub-matrices.

ALC can produce sub-optimal segments if the point trajectories of high dimensional data do not span a sufficient area of the subspace. This phenomenon is addressed in two ways. The first solution, for affine motions, projects the data onto a 5D subspace. However, the data must be projected to a larger subspace if mixtures of motions are encountered. The second solution deals with mixtures of motions and projects the data onto a subspace with a dimensionality larger than that of the minimum motion subspaces. The sparsity-preserving dimension d_{sp} is defined as

$$d_{sp} = \min d, \quad \text{s.t. } d \geq 2k\log\left(\frac{D}{d}\right), \tag{2.17}$$

where D is the dimensionality of the ambient space and k is the true dimensionality of the data. ALC can handle missing and data as well as occlusions. Therefore, two variations of ALC exist, namely ALC_5 and ALC_{sp}. ALC_5 assumes that $d = 5$ while ALC_{sp} uses Equation (2.17) to estimate d.

2.3.5 Low-Rank Representation

Low-Rank Representation (LRR) is a compressed sensing technique that builds on the classic Sparse Representation (SR), but instead of obtaining a sparse representation for each vector, the representation

with the lowest rank over a group of vectors is computed [40]. Additionally, LRR extends SR to operate on multiple subspaces. The LRR presentation can express the global structure of the data which allows for more robust segmentation of noisy data. A set of feature trajectories are received in vector format, and each vector is represented by a linear combination of the other vectors, called the bases. This can be written as

$$\mathbf{X} = \mathbf{AZ} + \mathbf{E}, \tag{2.18}$$

where coefficient matrix $\mathbf{Z} = [z_1, \ldots, z_n]$ contains the representation of each corresponding input vector, and $\mathbf{E}$ contains the noise. LRR finds the lowest rank representation and uses this representation to construct an affinity matrix. Spectral clustering is applied to the affinity matrix to obtain the final segmentation. Therefore, the low-rank optimisation problem can be written as

$$\min_{\mathbf{Z},\mathbf{E}} ||\mathbf{Z}||_* + \lambda ||\mathbf{E}||_l, \quad \text{s.t. } \mathbf{X} = \mathbf{XZ} + \mathbf{E}, \tag{2.19}$$

where $||\cdot||_*$ is the nuclear norm and $||\cdot||_l$ is an arbitrary regularisation strategy. λ is a parameter that is used to set the importance of the two norms, and its value is dependent on the properties of the two norms. For independent motions, this representation leads to an affinity matrix that is dense for points residing in the same cluster while the affinity measure for points in different clusters is approximately 0. LRR is robust to noise and corrupted data.

2.3.6 Sparse Subspace Clustering

SSC uses SR to cluster data with a high dimensionality [34]. SSC finds the sparsest combination to obtain the rest of the points residing in the same subspace. This can be achieved by solving the ℓ_0 optimisation problem. Solving the non-convex ℓ_0-norm is NP-hard, but can be simplified to solving for the ℓ_1 minimisation problem, subject to certain constraints. This sparse representation is used to construct an affinity matrix, which is then used by spectral clustering to obtain the final segmentation. SSC is robust to corrupt data, requires no initialisation, and can handle missing data. The exponential complexity, faced by spectral clustering approaches such as GPCA, is addressed with the use of sparse representations.

SSC extends Compressed Sensing (CS) to operate on multiple subspaces of which a sparsifying basis is unknown. CS represents vectors as a proper basis, which has the advantage of reducing the signal information rate to less than that of the maximum signal frequency. A vector $\mathbf{x}$ in $\mathbb{R}^D$, can be expressed by D basis vectors $\{\psi_i \in \mathbb{R}^D\}_{i=1}^D$ and is used to construct the basis matrix $\Psi = [\psi_1, \ldots, \psi_D]$. Then, $\mathbf{x}$ can be written as

$$\mathbf{x} = \sum_{i=1}^{D} s_i \psi_i = \Psi \mathbf{s}, \tag{2.20}$$

where $\mathbf{s} = [s_1, \ldots, s_D]^T$ and represents $\mathbf{x}$ in the ψ domain. Vector $\mathbf{x}$ cannot be measured directly but is represented by m linear combinations of entries, of the form $\mathbf{x}_i = \phi_i^T \mathbf{x}$ for $i \in \{1, \ldots, m\}$. Therefore,

$$\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^T = \Phi\mathbf{x} = \Phi\Psi\mathbf{s} = A\mathbf{s}, \tag{2.21}$$

where $\Phi = [\phi_1, \ldots, \phi_m]^T \in \mathbb{R}^{m \times D}$ is the measurement matrix. Vector $\mathbf{x}$ is K-sparse if it has a maximum of K significantly large coefficients. Using $\mathbf{x}$, the K-sparse vectors can be determined if $K \lesssim m/\log(D/m)$ as follows:

$$\min ||\mathbf{s}||_0 \quad \text{s.t. } A\mathbf{s}, \tag{2.22}$$

where $||\mathbf{s}||_0$ is the ℓ_0-norm of $\mathbf{s}$ (the number of non-zero elements). This problem is non-convex and NP-hard, therefore, the Basis Pursuit (BP) algorithm is used to solve the ℓ_1 optimisation, which is non-convex, instead:

$$\min ||\mathbf{s}||_1 \quad \text{s.t. } A\mathbf{s}. \tag{2.23}$$

The BP algorithm computes a K-sparse vector by subjecting the isometry constant of A to certain conditions. This can be extended to represent disjoint subspaces. In this instance, a basis for each subspace is known. The block-sparse vector $\mathbf{s}$ is solved using the ℓ_1/ℓ_2 optimisation problem under a modified isometry constant. Let $\{A_i \in \mathbb{R}^{D \times d_i}\}_{i=1}^n$ be a set of bases for n disjoint linear subspaces with dimensions $\{d_i\}_{i=1}^n$. For $\mathbf{x}$ in subspace i, the sparse solution is

$$\mathbf{x} = A\mathbf{s} = [A_1, \ldots, A_n][s_1^T, \ldots, s_n^T]^T, \tag{2.24}$$

where $s_i \in \mathbb{R}^{d_i}$ is a non-zero vector and all other vectors are zero. Then, $\mathbf{s}$ is obtained by optimising the non-convex problem:

$$\min \sum_{i=1}^{n} 1(||s_i||_2 > 0), \quad \text{s.t. } \mathbf{x} = A\mathbf{s}, \tag{2.25}$$

where the indicator function $1(||s_i||_2 > 0)$ is 1 when $||s_i||_2 > 0$ and zero otherwise.

SSC extends this CS representation to express each point located in a union of subspaces as a linear or affine combination of all the other points. First, the case of linear subspaces will be considered. Let $X_i \in \mathbb{R}^{D \times N_i}$ be the set of N_i points from subspace i. Then the data matrix is expressed as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] = [X_1, \ldots, X_n]\Gamma \in \mathbb{R}^{D \times N}$ where $N = \sum_{i=1}^n N_i$ and $\Gamma \in \mathbb{R}^{N \times N}$ is an unknown matrix which containing the final motion segmentation. The subspace bases are chosen from the columns of X. Then, the points are self-expressive and point $\mathbf{x}$ in S_i can be expressed as a linear combination of d_i points from the same subspace. If $s = \Gamma^{-1}[\mathbf{s}_1^T, \ldots, \mathbf{s}_n^T]^T \in \mathbb{R}^N$, where $\mathbf{s}_i \in \mathbb{R}^{N_i}$, then $\mathbf{x}$ can be represented in d_i dimensions that is obtained as a sparse solution of $\mathbf{x} = Xs$ with $s_i \neq 0$ and $s_j = 0$ for all $j \neq i$. This is the solution to the ℓ_0 optimisation problem. As previously discussed, this problem is NP-hard, therefore the ℓ_1 optimisation problem is used instead when the subspaces are independent. The non-zero block

of the sparse solution obtained corresponds to the points in the same subspace as $\mathbf{x}$. This sparse representation is used to cluster the data. Let $X_{\hat{i}} \in \mathbb{R}^{D\times N-1}$ be matrix $\mathbf{X}$ without column i. If $\mathbf{x}_j$ belongs to subspace j, then its sparse representation is obtained with respect to the basis matrix $X_{\hat{i}}$ by solving the ℓ_1 program:

$$\min ||c_i||_1 \quad \text{s.t. } \mathbf{x}_i = X_{\hat{i}} c_i. \tag{2.26}$$

For affine subspaces, point $\mathbf{x}$ can be written as an affine combination of points from the same subspace:

$$\mathbf{x} = c_1\mathbf{x}_1 + \cdots + c_N\mathbf{x}_N, \quad \sum_{i=1}^{N} c_i = 1. \tag{2.27}$$

The ℓ_1 minimisation problem becomes

$$\min ||c_i||_1, \quad \text{s.t. } \mathbf{x}_i = \hat{X}_i c_i \quad \text{and } c_i^T \mathbf{1} = 1. \tag{2.28}$$

The non-zero entries of the optimal solution $c_i \in \mathbb{R}^{N-1}$ corresponds to the points/columns in $X_{\hat{i}}$ which reside in the same subspace as $\mathbf{x}_i$. This solution is transformed to vector $c_{\hat{i}} \in \mathbb{R}^N$ by inserting a 0 in row i. c_i for all $i = 1,\ldots N$ are computed to form the coefficient matrix $C = [c_{\hat{1}}, \ldots, c_{\hat{n}}]$. The coefficient matrix is used to construct a directed graph $G = (V,E)$ where the vertices correspond to the N points and the edges $(v_i, v_j) \in E$ correspond to the entry in the coefficient matrix $C_{ji} \neq 0$. C is the adjacency matrix of G. Since G is an unbalanced digraph in most cases, a balanced graph $\tilde{G}$ is constructed with adjacency matrix $\tilde{C}$ where $\tilde{C}_{ij} = |C_{ij}| + |C_{ji}|$. The new adjacency matrix is a valid similarity since $\mathbf{x}_i$ can be represented by a linear or affine combination of points including $\mathbf{x}_j$, then $\mathbf{x}_j$ can do the same. All connected vertices in graph $\tilde{G}$ represent points belonging to the same subspace while vertices representing points from different subspaces are not connected. The Laplacian of $\tilde{G}$ is: $L = D - \tilde{C}$ where the entries of diagonal matrix D is defined as $D_{ii} = \sum_j \tilde{C}_{ij}$. The final segmentation is obtained by applying K-means to a subset of eigenvectors of the Laplacian.

The use of the ℓ_1-norm is not the best approach since larger coefficients can be corrupted with large errors. There are several variations of the classic SSC algorithm that aim to address these issues. In [36], the ℓ_1-norm is replaced with a pseudo norm as the minimisation problem. This norm is called the $\ell_{q,\varepsilon}$-norm and is defined as

$$||c_i||_{q,\varepsilon} = N \sum_{j=1} (|c_{ij}| + \varepsilon)^q := N \sum_{j=1} g[h(c_{ij})], \tag{2.29}$$

where $0 < q < 1$ and $\varepsilon > 0$. Function $h(t) = |t|$, $\mathbb{R} \longrightarrow \mathbb{R}$ is convex and $g(s) = (s+\varepsilon)^q$, $Im(h) \longrightarrow \mathbb{R}$ is convex. This norm re-weights the coefficients by assigning larger weights to small coefficients and smaller weights to large coefficients. Unlike the ℓ_1-norm, the $\ell_{q,\varepsilon}$-norm is non-convex which better

approximates the similarity matrix coefficients, but it is significantly more complex. To address the increased complexity, Alternating Direction Method of Multipliers (ADMM) is used.

2.3.7 Latent Space Sparse Subspace Clustering

Latent Space Sparse Subspace Clustering (LS3C) builds on the SSC method which aims to reduce the data dimensionality while clustering the data [41]. LS3C projects the data to a lower-dimensional subspace such that manifold structure is preserved. The sparse coefficients are computed from the projected data and a similarity matrix is constructed. Spectral clustering is applied to the similarity matrix to find the final segmentation. This process is optimised using an iterative procedure that presents the projection to the lower-dimensional subspace as a linear combination of the data samples.

LS3C projects the data to a lower-dimensional subspace while computing the sparse codes simultaneously. Let $\mathbf{P} \in \mathbb{R}^{t \times D}$ be the transformation matrix which projects the input data, $\mathbf{X}$, from the $\mathbb{R}^D$ space to the $\mathbb{R}^t$ space. This transformation matrix can be recovered while computing the sparse codes by minimising the cost function

$$[\mathbf{P}^*, \mathbf{C}^*] = \min_{P,C} \mathcal{J}(\mathbf{P}, \mathbf{C}, \mathbf{X}), \quad \text{s.t. } \mathbf{P}\mathbf{P}^T = \mathbf{I}, \quad diag(\mathbf{C}) = \mathbf{0}. \tag{2.30}$$

For affine subspaces, the minimisation problem becomes:

$$[\mathbf{P}^*, \mathbf{C}^*] = \min_{P,C} \mathcal{J}(\mathbf{P}, \mathbf{C}, \mathbf{X}), \quad \text{s.t. } \mathbf{P}\mathbf{P}^T = \mathbf{I}, \quad \mathbf{C}^T\mathbf{1} = \mathbf{1}, \quad diag(\mathbf{C}) = \mathbf{0}. \tag{2.31}$$

In both versions of the cost minimisation function,

$$\mathcal{J}(\mathbf{P}, \mathbf{C}, \mathbf{X}) = ||\mathbf{C}||_1 + \lambda_1 ||\mathbf{P}\mathbf{X} - \mathbf{P}\mathbf{X}\mathbf{C}||_F^2 + \lambda_2 ||\mathbf{X} - \mathbf{P}^T\mathbf{P}\mathbf{X}||_F^2. \tag{2.32}$$

The terms $||\mathbf{C}||_1$ and $\lambda_1 ||\mathbf{P}\mathbf{X} - \mathbf{P}\mathbf{X}\mathbf{C}||_F^2$ are used to encourage sparsity while the term $\lambda_2 ||\mathbf{X} - \mathbf{P}^T\mathbf{P}\mathbf{X}||_F^2$ is a regularisation term which encourages information from the original data to be retained. Parameters λ_1 and λ_2 are non-negative and are used to indicate the importance of the sparsity and regularisation terms, respectively. The rows of $\mathbf{P}$ are orthogonal, and normalised, to prevent degenerate solutions.

There is an optimal solution for the transformation in which has the form of a linear combination of the original data:

$$\mathbf{P}^* = \mathbf{\Psi}^T\mathbf{X}^T. \tag{2.33}$$

Then, function $\mathcal{J}$ becomes

$$\begin{aligned} &\mathcal{J}(\mathbf{\Psi}, \mathbf{C}, \mathbf{X}) = ||\mathbf{C}||_1 + \lambda_1 ||\mathbf{\Psi}^T\mathbf{K}(\mathbf{I} - \mathbf{C})||_F^2 + \lambda_2 ||\mathbf{X} - (\mathbf{I} - \mathbf{\Psi}\mathbf{\Psi}^T\mathbf{K})||_F^2, \\ &\text{s.t. } \mathbf{P}\mathbf{P}^T = \mathbf{\Psi}^T\mathbf{K}\mathbf{\Psi} = \mathbf{I}, \end{aligned} \tag{2.34}$$

where $\mathbf{K} = \mathbf{X}^T\mathbf{X}$, and the optimisation problem becomes

$$[\mathbf{\Psi}^*, \mathbf{C}^*] = \min_{\mathbf{\Psi},\mathbf{C}} \mathcal{J}(\mathbf{\Psi}, \mathbf{C}, \mathbf{X}), \quad \text{s.t. } \mathbf{\Psi}^T\mathbf{K}\mathbf{\Psi} = \mathbf{I}, \quad diag(\mathbf{C}) = \mathbf{0}. \tag{2.35}$$

Therefore, $\mathbf{P}$ is updated by updating $\mathbf{\Psi}$. This formulation of the optimisation function is solved by iteratively optimising over $\mathbf{\Psi}$ and $\mathbf{C}$. First, $\mathbf{C}$ is fixed and $\mathbf{\Psi}$ is optimised using

$$\mathbf{M}^* = \min_{\mathbf{M}} trace(\mathbf{M}^T \triangle \mathbf{M}), \quad \text{s.t. } \mathbf{M}^T\mathbf{M} = \mathbf{I}, \tag{2.36}$$

where $\mathbf{M} = \mathbf{S}^{1/2}\mathbf{V}^T\mathbf{\Psi}$ and $\triangle = \mathbf{S}^{1/2}\mathbf{V}^T(\lambda_1(\mathbf{I}-\mathbf{C})(\mathbf{I}-\mathbf{C})^T - \lambda_2\mathbf{I})\mathbf{V}\mathbf{S}^{1/2}$. Here, $\mathbf{V}$ and $\mathbf{S}$ are obtained from the eigenvalue decomposition of $\mathbf{K}$: $\mathbf{K} = \mathbf{V}\mathbf{S}\mathbf{V}^T$. After the optimal $\mathbf{M}^*$ has been computed, $\mathbf{\Psi}$ is determined as $\mathbf{\Psi} = \mathbf{V}\mathbf{S}^{-1/2}\mathbf{M}^*$.

After $\mathbf{\Psi}$ has been updated, it is fixed and $\mathbf{C}$ is updated using the following minimisation function:

$$\min_{\mathbf{C}} ||\mathbf{C}||_1 + \lambda_1||\mathbf{\Psi}^T\mathbf{K} - \mathbf{\Psi}^T\mathbf{K}\mathbf{C}||_F^2, \quad \text{s.t. } diag(\mathbf{C}) = \mathbf{0}. \tag{2.37}$$

This optimisation problem is equivalent to the SSC optimisation problem, therefore ADMM is used to update $\mathbf{C}$.

A variation of the LS3C method, called Non-Linear Latent Space Sparse Subspace Clustering (NLS3C), exists that handles data exhibiting non-linear manifolds using kernel-based methods. The non-linear transformation matrix $\mathcal{P}$ can be determined by employing a kernel function. The kernel function is used to define a kernel Gram matrix as

$$[\mathcal{K}(\mathbf{Y}, \mathbf{Y})]_{i,j} = \kappa(\mathbf{y}_i, \mathbf{y}_j), \tag{2.38}$$

where κ is the kernel function which applies the following mapping: $\mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R}$. For non-linear manifolds, the cost function used in the optimisation problem becomes

$$\begin{aligned} &\mathcal{J}(\mathbf{\Psi}, \mathbf{C}, \mathcal{K}) = ||\mathbf{C}||_1 + \lambda_1||\mathbf{\Psi}^T\mathcal{K}(\mathbf{I}-\mathbf{C})||_F^2 + \lambda_2\, trace\left[(\mathbf{I} - \mathbf{\Psi}\mathbf{\Psi}^T\mathcal{K})^T\mathcal{K}(\mathbf{I} - \mathbf{\Psi}\mathbf{\Psi}^T\mathcal{K})\right], \\ &\text{s.t. } \mathcal{P}\mathcal{P}^T = \mathbf{I}, \quad \mathbf{\Psi}^T\mathcal{K}\mathbf{\Psi} = \mathbf{I}. \end{aligned} \tag{2.39}$$

Not only does LS3C reportedly outperform both SSC and LRR, but is also more stable. This is since the performance of LS3C is not influenced by the projection used to reduce the dimensionality of the data, but rather LS3C computes the features automatically. It is also robust to noise; however, the method was only tested on the Hopkins155 dataset which contains only simple motions by two or three moving objects.

2.4 OCCLUSION HANDLING METHODS

Occlusions are one of the major challenges the field of motion segmentation faces. There have been many efforts to design methods to effectively handle large and complete occlusions, but despite all the efforts, there are no solutions close to human capabilities. Here, current efforts to address the occlusion problem as well as missing data are investigated to give an indication of where the research is heading and identify possible solutions which can be adapted to suit the manifold clustering approach.

2.4.1 Accurate Subspace Segmentation by Successive Approximations

Accurate Subspace Segmentation by Successive Approximations (ASSA) performs subspace clustering of data with a high dimensionality while estimating the missing data [42]. The method builds on classic SSC by adapting the completion of self-representation matrices. The motion segments are computed iteratively by approximating and refining the subspace structure. The method can segment both high- and low-dimensional data, which means that ASSA can handle multiple motions, as well as a mixture of different types of motion.

ASSA builds on SSC and describes $\mathbf{X} = \mathbf{XC}$, where $\mathbf{X}$ is the data matrix and $\mathbf{C}$ is the coefficient matrix with $diag(\mathbf{C}) = 0$. Since some of the data entries in $\mathbf{X}$ are missing, $\mathbf{X}$ can be written in terms of the observed and missing entries as

$$\mathbf{X} = \mathbf{X}_{\Omega} + \mathbf{X}_{\Omega^C}, \tag{2.40}$$

where $\mathbf{X}_{\Omega}$ contains the observed data entries, and $\mathbf{X}_{\Omega^C}$ the missing data entries in the complimentary positions to $\mathbf{X}_{\Omega}$. An estimation for the missing data $\mathbf{X}_{\Omega^C}$ is obtained by solving the ℓ_1 optimisation problem

$$\begin{aligned} &\mathbf{X}_{\Omega^C}, \min_{\mathbf{C},\mathbf{E},\mathbf{Z}} ||\mathbf{C}||_1 + \lambda_e||\mathbf{E}||_1 + \frac{\lambda_z}{2}||\mathbf{Z}||_F^2 \\ &\text{s.t. } \mathbf{X}_{\Omega} + \mathbf{X}_{\Omega^C} = (\mathbf{X}_{\Omega} + \mathbf{X}_{\Omega^C})\mathbf{C} + \mathbf{E} + \mathbf{Z}, \quad diag(\mathbf{C}) = 0, \quad (\mathbf{X}_{\Omega^C})_{\Omega} = 0, \end{aligned} \tag{2.41}$$

where $\mathbf{E}$ accounts the outliers and $\mathbf{Z}$ for the noise. By solving for the ℓ_1-norm, the sparse representation of the complete data can be obtained if the subspaces are adequately separated, and the points span each subspace sufficiently. The optimisation problem is non-convex but can be solved using a relaxed convex problem under tight constraints along with an effective initialisation strategy. The relaxed convex problem is defined as

$$\begin{aligned} &\min_{\triangle\mathbf{C},\triangle\mathbf{X},\mathbf{E},\mathbf{Z}} ||\mathbf{C}^{(i)} + \triangle\mathbf{C}||_1 + \lambda_e||\mathbf{E}||_1 + \frac{\lambda_z}{2}||\mathbf{Z}||_F^2 \\ &\text{s.t. } \mathbf{X}^{(i)} + \triangle\mathbf{X} = (\mathbf{X}^{(i)} + \triangle\mathbf{X})\mathbf{C}^{(i)} + \mathbf{X}^{(i)}\triangle\mathbf{C} + \mathbf{E} + \mathbf{Z}, \\ &||\triangle\mathbf{X}||\infty \leq \delta_{\mathbf{X}}, \quad ||\triangle\mathbf{C}||_{\infty} \leq \delta_{\mathbf{C}}, \quad diag(\triangle\mathbf{C}) = 0, \quad \triangle\mathbf{X}_{\Omega} = 0. \end{aligned} \tag{2.42}$$

To solve the relaxed optimisation problem, the Accurate Algorithm is used which iteratively updates the current estimates for $\mathbf{x}$ and $\mathbf{C}$. For the given points $\mathbf{X}^{(i)}$ and $\mathbf{C}^{(i)}$, the precise model update model is defined as

$$\mathbf{X}^{(i)} + \triangle\mathbf{X} = (\mathbf{X}^{(i)} + \triangle\mathbf{X})(\mathbf{C}^{(i)} + \triangle\mathbf{C}) + \mathbf{E} + \mathbf{Z}, \tag{2.43}$$

where $\triangle\mathbf{X}$ consists only of the change in the missing entries, namely $\triangle\mathbf{X}_{\Omega^c}$ while $\triangle\mathbf{X}_{\Omega} = 0$. To determine $\triangle\mathbf{X}$ and $\triangle\mathbf{C}$, the precise update model is linearised by removing the $\triangle\mathbf{X}\triangle\mathbf{C}$. This leads to the convex problem with new constraints:

$$\begin{aligned} &\min_{\triangle\mathbf{C},\triangle\mathbf{X},\mathbf{E},\mathbf{Z}} ||\mathbf{C}^{(i)} + \triangle\mathbf{C}||_1 + \lambda_e||\mathbf{E}||_1 + \frac{\lambda_z}{2}||\mathbf{Z}||_F^2 \\ &\text{s.t. } \mathbf{X}^{(i)} + \triangle\mathbf{X} = (\mathbf{X}^{(i)} + \triangle\mathbf{X})\mathbf{C}^{(i)} + \mathbf{X}^{(i)}\triangle\mathbf{C} + \mathbf{E} + \mathbf{Z}, \\ &||\triangle\mathbf{X}||_\infty \leq \delta_{\mathbf{X}}, \quad ||\triangle\mathbf{C}||_\infty \leq \delta_{\mathbf{C}}, \quad diag(\triangle\mathbf{C}) = 0, \quad \triangle\mathbf{X}_{\Omega} = 0. \end{aligned} \tag{2.44}$$

Therefore, $\mathbf{X}$ and $\mathbf{C}$ are updated iteratively using

$$\begin{aligned} \mathbf{X}^{(i+1)} &= \mathbf{X}^{(i)} + \triangle\mathbf{X} \\ \mathbf{C}^{(i+1)} &= \mathbf{C}^{(i)} + \triangle\mathbf{C}. \end{aligned} \tag{2.45}$$

Note that an initial estimate of $\mathbf{X}$ and $\mathbf{C}$ is required and is computed using Alternate Convex Search. Alternate Convex Search iteratively fixes one variable (either $\mathbf{X}_{\Omega^c}$ or $\mathbf{C}$) to estimate the other, and can be used since the minimisation problem in Equation (2.41) is convex in $\mathbf{X}$ when $\mathbf{C}$ is fixed and vice versa. Therefore, for the current $\mathbf{X}_{\Omega^c}^{(i)}$, $\mathbf{C}$ is updated using

$$\begin{aligned} &\min_{\mathbf{C},\mathbf{E},\mathbf{Z}} ||\mathbf{C}||_1 + \lambda_e||\mathbf{E}_{\Omega}||_1 + \frac{\lambda_z}{2}||\mathbf{Z}_{\Omega}||_F^2 \\ &\text{s.t. } \mathbf{X}_{\Omega} + \mathbf{X}_{\Omega^c}^{(i)}, \quad diag(\mathbf{C}) = 0, \end{aligned} \tag{2.46}$$

where $\mathbf{E}_{\Omega}$ and $\mathbf{Z}_{\Omega}$ are the noise and error matrices indexed by Ω. Once $\mathbf{C}^{(i+1)} = \mathbf{C}$ is computed, it is fixed and $\mathbf{X}_{\Omega^C}^{(i)}$ is computed using

$$\mathbf{X}_{\Omega^C}^{(i+1)} = (\mathbf{X}^{(i)}\mathbf{C}^{(i+1)})_{\Omega^C}, \tag{2.47}$$

where $\mathbf{X}^{(i)} = \mathbf{X}_{\Omega} + \mathbf{X}_{\Omega^C}^{(i)}$.

Once the missing data has been estimated, the coefficient matrix $\mathbf{C}$ is used to define the affinity matrix, and spectral clustering is applied to the affinity matrix to obtain the final segmentation.

2.4.2 Locality-constrained Non-negative Robust Shape Interaction

Classic SSC is a point-wise approach that consists of n sub-problems, and the global properties of the subspace are not taken into account. Even though SSC produces a sparse affinity graph, this can reduce the accuracy of the segmentation. Locality-constrained Non-negative Robust Shape Interaction

(LNRSI) addresses this issue by extending SSC and adding a low-rank property inspired approach. LNRSI exploits the manifold structure of the data in combination with robust shape interaction (RSI) to segment motion in a video [43]. The constructed affinity matrix is sparse and exhibits low-rank properties. Unlike existing low-rank methods, LNRSI achieves low-rank regularisation while keeping the locality of the manifold of the data. It is also robust to noise and occlusions.

LNRSI exploits the low-rank property which states that points residing in the same linear subspace can be represented with similar vectors. For points residing in different linear subspaces, the coefficients of the representation are zero, leading to a sparse local structure. Therefore, the affinity matrix has a block diagonal shape and is discriminative and sparse. LNRSI solves the following optimisation problem:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{D},\mathbf{E}} rank(\mathbf{Z}) + \lambda_1 trace(\mathbf{W}^T\mathbf{Z}) + \lambda_2 ||\mathbf{E}||_{2,1} \\ &\text{s.t. } \mathbf{X} = \mathbf{D} + \mathbf{E}, \quad \mathbf{D} = \mathbf{D}\mathbf{Z}, \quad \mathbf{Z} \geq 0, \end{aligned} \tag{2.48}$$

where $||\mathbf{E}||_{2,1} = \sum_i ||\mathbf{E}_i||_2$, $\mathbf{E}_i$ is column i of $\mathbf{E}$. Parameters λ_1 and λ_2 determines the importance of the locality constraint and the noise level, respectively. The term $rank(\mathbf{Z})$ is responsible for the global structure and to cluster points from the same subspace to be together. The locality constraint $trace(\mathbf{WZ})$ relies on weight $\mathbf{W}_{ij}$ which is used to describe the contribution of other points $\mathbf{X}_j$ to the reconstruction of point $\mathbf{X}_i$, and encourages similar points to contribute to the representation. Here, $\mathbf{W}_{ij}$ is computed using k-Nearest Neighbours (k-NN):

$$\mathbf{W}_{ij} = \begin{cases} 0 & \text{if } \mathbf{X}_j \in N(\mathbf{X}_i) \\ \infty & \text{otherwise,} \end{cases} \tag{2.49}$$

where $N(X_i)$ is the $k = 5$ nearest neighbours of data $\mathbf{X}_i$. The weight is infinitely large for points located far from the current point X_i, which forces the term $trace(\mathbf{WZ})$ to be zero and the optimisation problem is minimised. For points located close to X_i, the term $trace(\mathbf{WZ})$ is discarded, and rank regularisation is used to determine the coefficients. Note that solving the optimisation problem is NP-hard, therefore the problem is relaxed to obtain the convex optimisation problem:

$$\begin{aligned} &\min_{\mathbf{Z},\mathbf{D},\mathbf{E}} ||\mathbf{Z}||_* + \lambda_1 trace(\mathbf{W}^T\mathbf{Z}) - \lambda_2 ||\mathbf{E}||_{2,1} \\ &\text{s.t. } \mathbf{X} = \mathbf{D} + \mathbf{E}, \quad \mathbf{D} = \mathbf{D}\mathbf{Z}, \quad \mathbf{Z} \geq 0. \end{aligned} \tag{2.50}$$

This problem is still difficult to solve. Therefore, the relaxed optimisation problem is divided into two sub-problems and solved using Augmented Lagrange Multiplier (ALM) approach, and a Linearised Alternating Direction Method with Adaptive Penalty (LADMAP), respectively.

The first sub-problem is defined as

$$\min_{\mathbf{D},\mathbf{E}} rank(\mathbf{D}) + \lambda_2 ||\mathbf{E}||_{2,1}, \quad \text{s.t. } \mathbf{X} = \mathbf{D} + \mathbf{E}, \tag{2.51}$$

and is solved using an inexact ALM.

The second sub-problem is defined as

$$\min_{\mathbf{Z}} ||\mathbf{Z}||_* + \lambda_1 trace(\mathbf{W}^T\mathbf{Z}), \quad \text{s.t. } \mathbf{D} = \mathbf{D}\mathbf{Z}, \quad \mathbf{Z} \geq 0. \tag{2.52}$$

In order to make the second sub-problem separable, variables $\mathbf{Y}$ and $\mathbf{P}$ are used:

$$\min_{\mathbf{Z},\mathbf{P}} ||\mathbf{P}||_* + \lambda_1 trace(\mathbf{W}^T\mathbf{Z}), \quad \text{s.t. } \mathbf{D} = \mathbf{D}\mathbf{Z}, \quad \mathbf{Z} = \mathbf{P}, \quad \mathbf{Z} = \mathbf{Y}, \quad \mathbf{Y} \geq 0. \tag{2.53}$$

Then, the augmented Lagrangian function is written as

$$\begin{aligned} L(\mathbf{Z},\mathbf{P},\mathbf{Y},L_1,L_2,L_3) =& ||P||_* + \lambda_1 trace(\mathbf{W}^T\mathbf{Z}) + < L_1, \mathbf{D} - \mathbf{D}\mathbf{Z} > + < L_2, \mathbf{Z} - \mathbf{Y} > + < L_3, \mathbf{Z} - \mathbf{P} > \\ &+ \frac{\beta_1}{2}||\mathbf{D} - \mathbf{D}\mathbf{Z}||_2^2 + \frac{\beta_2}{2}||\mathbf{Z} - \mathbf{Y}||_2^2 + \frac{\beta_3}{2}||\mathbf{Z} - \mathbf{P}||_2^2. \end{aligned} \tag{2.54}$$

and LADMAP is used to update $\mathbf{Z}$, $\mathbf{P}$ and $\mathbf{Y}$ alternately, by minimising L while the other variables remain fixed. The update process is defined as

$$\begin{aligned} \mathbf{Z} =& \min_{\mathbf{Z}} \lambda_1 trace(\mathbf{W}^T\mathbf{Z}) + < L_1, \mathbf{D} - \mathbf{D}\mathbf{Z} > + < L_2, \mathbf{Z} - \mathbf{Y} > + < L_3, \mathbf{Z} - \mathbf{P} > \\ &+ \frac{\beta_1}{2}||\mathbf{D} - \mathbf{D}\mathbf{Z}||_2^2 + \frac{\beta_2}{2}||\mathbf{Z} - \mathbf{Y}||_2^2 + \frac{\beta_3}{2}||\mathbf{Z} - \mathbf{P}||_2^2 \\ \mathbf{P} =& \min_{\mathbf{P}} ||\mathbf{P}||_* + \frac{\beta_3}{2}||\mathbf{Z} - \mathbf{P} + \frac{L_3}{\beta_3}||_2^2 \\ \mathbf{Y} =& \max(0, \mathbf{Z} + \frac{L_2}{\beta_2}). \end{aligned} \tag{2.55}$$

Once the two sub-problems have been solved, an affinity matrix is constructed using

$$\mathbf{A} = \frac{(\mathbf{Z}^* + (\mathbf{Z}^*)^T)}{2}. \tag{2.56}$$

The affinity matrix has a block-diagonal shape which allows LNRSI to handle noise and occlusions effectively. Spectral clustering is applied to the affinity matrix to obtain the final segmentation.

2.4.3 GoDec+

GoDec+ is an algorithm that can be used for tasks such as classification and subspace clustering, and is suited for motion segmentation [44]. This method can handle different types of noise, as well as missing data caused by large occlusions. GoDec+ builds on GoDec, which is an algorithm used to decompose matrix $\mathbf{X}$ into the form $\mathbf{X} = \mathbf{L} + \mathbf{S} + \mathbf{G}, \quad rank(\mathbf{L}) \leq r, \quad card(\mathbf{S}) \leq k$, where $\mathbf{L}$ is the low-rank part, $\mathbf{S}$ the sparse part and $\mathbf{G}$ the Gaussian noise. This problem is solved iteratively by

alternately solving for

$$\begin{aligned}\mathbf{L}^{(t)} &= \min_{rank(\mathbf{L})\leq r} ||\mathbf{X}-\mathbf{L}-\mathbf{S}^{(t-1)}||_F^2 \\ \mathbf{S}^{(t)} &= \min_{card(\mathbf{S})\leq k} ||\mathbf{X}-\mathbf{L}^{(t)}-\mathbf{S}||_F^2.\end{aligned} \tag{2.57}$$

Noise does not always have a Gaussian distribution, and a sparse model combined with a Gaussian model, cannot model motion trajectories corrupted by large occlusions. Therefore GoDec+ models noise using Maximum Correntropy Criterion (MCC). Correntropy is defined as $V_\sigma(x,y) = \mathbf{E}[\kappa_\sigma(x-y)]$, where $\mathbf{E}[\cdot]$ is the expectation of the kernel function $\kappa_\sigma(\cdot)$. Maximising the correntropy is equivalent to minimising the Welsch M-estimator, which is defined as $1-g_\sigma(x)$, where $g_\sigma(x)$ is the Gaussian kernel. The MCC is used as the objective, to obtain

$$\min_{\mathbf{L}} \sum_{i=1}^{m}\sum_{j=1}^{n}[1-g_\sigma(\mathbf{N}_{i,j})], \quad \text{s.t. } \mathbf{X}=\mathbf{L}+\mathbf{N}, \quad rank(\mathbf{L})\leq r. \tag{2.58}$$

Drawing inspiration from Half-Quadratic (HQ) minimisation, an auxiliary variable $\mathbf{E}$ is added. Let $\varphi_v(\mathbf{E}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\varphi(\mathbf{E}_{i,j})$, where $\varphi(\cdot)$ is the dual function of $g_\sigma(\cdot)$. Then, the MCC objective problem becomes

$$\min_{\mathbf{L},\mathbf{E}} ||\mathbf{X}-\mathbf{L}-\mathbf{E}||_F^2 + \varphi_v(\mathbf{E}), \quad \text{s.t. } rank(\mathbf{L})\leq r. \tag{2.59}$$

This problem is solved iteratively by alternately solving for

$$\begin{aligned}\mathbf{L}^{(t)} &= \min_{rank(\mathbf{L})\leq r} ||\mathbf{X}-\mathbf{L}-\mathbf{E}^{(t-1)}||_F^2 + \varphi_v(\mathbf{E}_{(t-1)}) \\ \mathbf{E}^{(t)} &= \min_{\mathbf{E}} ||\mathbf{X}-\mathbf{L}^{(t)}-\mathbf{E}||_F^2 + \varphi_v(\mathbf{E}).\end{aligned} \tag{2.60}$$

When $\mathbf{E}$ is fixed, the update for $\mathbf{L}$ becomes equivalent to the classic GoDec update for $\mathbf{L}$, as denoted in Equation (2.57). $\mathbf{L}$ is solved using a Greedy Bilateral (GreB) approach which models the low-rank estimate as a bilateral factorisation. When $\mathbf{L}$ is fixed, solving for $\mathbf{E}$ becomes equivalent to $\mathbf{E} = \mathbf{N} - \mathbf{N}\circ g_\sigma(\mathbf{N})$, where $\circ$ is the entry-wise product. The rate of convergence for $\mathbf{L}$ and $\mathbf{E}$ is linear.

GoDec+ can be adapted for subspace clustering and is based on Efficient Dense Subspace Clustering (EDSC). The adapted GoDec+ problem is equal to the additive form of HQ and is defined as

$$\begin{aligned}&\min_{\mathbf{H}} \sum_{i=1}^{m}\sum_{j=1}^{n}[1-g_\sigma^2((\mathbf{L}-\mathbf{L}\mathbf{H})_{i,j})] + \lambda_2||\mathbf{H}||_F^2 \\ =&\min_{\mathbf{H},\mathbf{E}} ||\mathbf{L}-\mathbf{L}\mathbf{H}-\mathbf{E}||_F^2 + \varphi_v(\mathbf{E}) + \lambda_2||\mathbf{H}||_F^2.\end{aligned} \tag{2.61}$$

GoDec+ for subspace clustering is solved iteratively by alternately solving for $\mathbf{H}$ and $\mathbf{E}$. When $\mathbf{H}$ is fixed, $\mathbf{E} = \mathbf{N} - \mathbf{N}\circ g_\sigma(\mathbf{N})$, where $\mathbf{N} = \mathbf{L}-\mathbf{L}\mathbf{H}$. When $\mathbf{E}$ is fixed, $\mathbf{H} = (\mathbf{L}^T\mathbf{L}+\lambda_2\mathbf{I})^{-1}\mathbf{L}^T(\mathbf{L}-\mathbf{E})$. The

affinity matrix is computed as

$$\mathbf{A}_{i,j} = [\mathbf{Z}\mathbf{Z}^T]^{\alpha}_{i,j}. \tag{2.62}$$

Here, $\mathbf{Z}$ is computed from the SVD of $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$, and is defined as $\mathbf{Z} = \mathbf{U}_{1:r}\Sigma_{1:r}^{1/2}$ where matrix $\Sigma_{1:r}$ contains the r largest singular values along its main diagonal, and $\mathbf{U}_{1:r}$ contains the r corresponding columns of $\mathbf{U}$. Spectral clustering is applied to the affinity matrix to obtain the final segmentation.

2.4.4 Multi-Scale Motion Clustering

The Multi-Scale Motion Clustering (MSMC) algorithm is an affinity-based motion segmentation method which consists of two main algorithms, namely the Motion-Split-And-Merge (MSAM) and MSMC algorithms [45]. The MSAM algorithm builds on the traditional split-and-merge algorithm presented in [46]. MSAM uses the split-and-merge operation to segment the motion between two frames. This is referred to as inter-frame motion segmentation. Then, the MSMC algorithm merges the inter-frame motion segments computed by the MSAM algorithm to find the final motion segments over the entire input sequence. The number of motions is also computed and therefore not required beforehand. The MSMC is also able to combine inter-frame motions with various scales, that is motions between non-adjacent frames (e.g. every second or third frame pair) as well. The MSMC method can handle occlusions and missing data. Trajectories with different lengths can also be segmented, which makes the method robust to objects which enter and/or leave the scene.

The MSAM algorithm extracts inter-frame motion segments. The algorithm starts by assigning all points to the same segment S_1. Then, split and merge operations are performed iteratively. The first split operation estimates the parameters $\mathbf{p}_k$ using RANSAC. The outliers are identified using

$$d^2(\mathbf{x}_i|\mathbf{p}_k) < \varepsilon^2, \tag{2.63}$$

where $d^2(\mathbf{x}_i|\mathbf{p}_k)$ is the distance metric for the point $\mathbf{x}_i$ between the two frames which relies on model parameters $\mathbf{p}_k$ of segment S_k. The outliers are re-assigned to the outlier class S_0. Note that this can only be done if the inlier ratio is smaller than threshold θ_s, since RANSAC was unable to compute a valid motion due to distortion caused by the outliers of multiple segments still assigned to segment S_k. In this case, the split operation splits an inconsistent motion segment into two segments using an approach based on J-linkage. The process employs RANSAC to determine the segment inliers as well as the model parameters $\mathbf{p}_k$. The estimation of the parameters is improved by selecting the best result from a set of r RANSAC estimates. Once the RANSAC sampling process and parameter estimation are complete, the Jaccard distance between the point positions in the two frames is computed for each of

the inlier points. Spatial consistency is added by combining the Jaccard distance with the Mahalanobis distance between the inlier point positions in the second frame. The Jaccard and Mahalanobis distances are combined to form the affinity, as follows:

$$\mathbf{A}_{i,j} = [1 - J(\mathbf{P}_i, \mathbf{P}_j)] \cdot \min [M_k(\mathbf{x}_i, \mathbf{x}_j), 1.0], \tag{2.64}$$

where

$$J(\mathbf{P}_i, \mathbf{P}_j) = \begin{cases} \frac{|\mathbf{P}_i \cup \mathbf{P}_j| - |\mathbf{P}_i \cap \mathbf{P}_j|}{|\mathbf{P}_i \cup \mathbf{P}_j|}, & \mathbf{P}_i \cup \mathbf{P}_j \neq 0 \\ 1, & \text{otherwise} \end{cases} \tag{2.65}$$

denotes the Jaccard distance and

$$M_k(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\mathbf{C}_{\mathbf{x}}^{-1}(\mathbf{x}_i - \mathbf{x}_j)} \tag{2.66}$$

the Mahalanobis distance. The affinity matrix is then used as the input for k-means to split the class inliers into two segments.

After the split operation, a merge operation is performed. Two segments S_k and S_l are merged if the inlier ratio of the smallest segment is larger than threshold θ_m. Next, a split operation is performed on the outlier segment S_0. The inliers of the class are determined. If the inlier count is larger than the minimum class size, these inlier points are assigned to a new segment using the same steps as with the first split operation where segment S_k were split into two. Lastly, a merge operation is performed on the outlier segment S_0. Points from the outlier segment are re-assigned to segment S_k if these points are detected as inliers. The two split and two merge operations are performed iteratively until no action is taken or a maximum number of iterations has been reached.

To discourage continual split and merge operations of the same segments and points, the following constraint is used on the split and merge thresholds:

$$1 + \theta_m \leq 2\theta_s. \tag{2.67}$$

The MSMC algorithm is used to obtain the final segmentation from the inter-frame segments computed by MSAM. First, the MSMC algorithm identifies the inlier correspondences between the sets of inter-frame segments and assigns them to the same trajectory T_i. Each trajectory T_i can be defined by parameter set H_i which consists of the parameters $\mathbf{p}_{k,t}$ of segments $S_{k,t}$, where k denotes the segment number and t the frame number. The Jacobian distance between the trajectories is used to populate the affinity matrix:

$$\mathbf{A}_{i,j} = 1 - J(H_i, H_j). \tag{2.68}$$

Lastly, k-means is applied to the affinity matrix to compute the final motion segments.

The MSAM algorithm can not only be applied to two consecutive frames, but also non-consecutive frames, say every fifth frame. Therefore, MSAM can be applied to different time scales. The MSMC algorithm receives the inter-frame segments produced by MSAM as input and can handle a set of inter-frame segments of different time scales. The MSMC algorithm identifies inlier correspondences as before, but also across the different time scale inter-frame segments, and assigns these inlier correspondences to the same trajectory T_i. The affinity matrix is built, as before, using Equation (2.68).

2.4.5 Multiple Two Perspective-view

Multiple Two Perspective-View (M-TPV) is an extension of the classic SSC algorithm which uses approaches based on two-frame analysis [47]. M-TPV utilises the epipolar constraint between an image pair to perform motion segmentation. To take advantage of the linkage between the image pairs, the point correspondence between all image pairs from the video sequence is combined by optimising a mixed norm optimisation problem, called the $\ell_{1,1,2}$-norm. The coefficients computed by solving the optimisation problem is used to construct an affinity matrix. This method does not require the number of motions beforehand and can effectively handle missing data and perspective effects.

Let an input video have L image pairs with point correspondence matrix $\mathbf{W}^{(l)}$. Using SSC, for each image pair the coefficient matrix $\mathbf{C}^{(l)}$ can be determined. Since there is a relation between the sets of image pairs, this method aims to compute all L sets of coefficients $\mathbf{C}^{(l)}$ simultaneously, subjected to the same sparse profile. Therefore, $\mathbf{C}^{(l)}$ must be sparse and the columns across different $\mathbf{C}^{(l)}$ corresponding to trajectory T_i across different must have similar representations. To achieve this, a joint optimisation problem called the $\ell_{1,1,2}$-norm is defined as

$$\begin{aligned} &\min \sum_{i=1}^{P} \sum_{j=1}^{P} \sqrt{\sum_{l=1}^{L} \left(c_{ij}^{(l)}\right)^2}, \\ &\text{s.t. } \mathbf{W}^{(l)} = \mathbf{W}^{(l)}\mathbf{C}^{(l)}, \quad diag(\mathbf{C}^{(l)}) = 0, \end{aligned} \tag{2.69}$$

where $c_{ij}^{(l)}$ is element in position (i, j) of $\mathbf{C}^{(l)}$ for image pair l. To ensure that all data matrices $\mathbf{W}^{(l)}$ have the same dimensions, empty column vectors are used to represent any correspondences that are missing in one or both images. This does not affect the sparse representation computed by solving the $\ell_{1,1,2}$-norm, since zero entries are not selected to represent a point.

To handle noise and corrupted trajectories, the $\ell_{1,1,2}$ optimisation problem is extended to become

$$\begin{aligned} &\min \sum_{i=1}^{P}\sum_{j=1}^{P}\sqrt{\sum_{l=1}^{L}\left(c_{ij}^{(l)}\right)^2} + \lambda \sum_{l=1}^{L}||\mathbf{E}^{(l)}||_{\ell} \\ &\text{s.t. } \mathbf{E}^{(l)} = \mathbf{W}^{(l)} - \mathbf{W}^{(l)}\mathbf{C}^{(l)}, \quad diag(\mathbf{C}^{(l)}) = 0, \end{aligned} \tag{2.70}$$

where λ is the weight used to balance the $\ell_{1,1,2}$-norm and the noise $\mathbf{E}^{(l)}$ parts. The noise $\mathbf{E}^{(l)}$ is modelled using the $\ell_{1,2}$-norm in order to handle corruptions and outliers. After the above optimisation problem has been solved, corrupt trajectories are removed from image pair l if the corresponding column in the error matrix $\mathbf{E}^{(l)}$ contains large values.

After the coefficients have been determined, the affinity matrix is constructed using

$$\mathbf{A}_{i,j} = \sqrt{\sum_{l=1}^{L}\left(c_{ij}^{(l)}\right)^2} + \sqrt{\sum_{l=1}^{L}\left(c_{ji}^{(l)}\right)^2}. \tag{2.71}$$

The final segments are retrieved using spectral clustering. To eliminate the need for knowing the number of motions beforehand for the spectral clustering step, model selection is achieved using an over-segmentation and merge step. The merge step utilises the ℓ_1-norm of the global sparse representation over the two over-segmented groups. To obtain an initial estimate, the Laplacian of the affinity matrix is used to over-segment the data into two classes. During the merge step, the data in one class is used to form a representation of each of the data points in the other class. This representation is defined as

$$\min ||\mathbf{C}||_1 \quad \text{s.t. } \mathbf{P} = \mathbf{Q}\mathbf{C}, \tag{2.72}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are the two classes, and $\mathbf{C}$ contains the coefficient vectors corresponding to the points in $\mathbf{P}$. A point is identified as an inlier with respect to the other class if the ℓ_1-norm of the corresponding sparse vector is below a threshold, and inlier points are merged to one class.

The merge step is extended to L classes and the representation in Equation (2.72) becomes

$$\begin{aligned} &\min \sum_{i=1}^{N}\sum_{j=1}^{M}\sqrt{\sum_{l=1}^{L}\left(c_{ij}^{(l)}\right)^2}, \\ &\text{s.t. } \mathbf{P}^{(l)} = \mathbf{Q}^{(l)}\mathbf{C}^{(l)}, \quad diag(\mathbf{C}^{(l)}) = 0. \end{aligned} \tag{2.73}$$

2.5 DATASETS

Datasets with ground truth annotations are required to test and benchmark algorithms. Two feature-based datasets were used to evaluate the algorithm performance, namely the Hopkins155 and KITTI 3D Motion Segmentation Benchmark (KT3DMoSeg) datasets.

2.5.1 Hopkins155

The Hopkins155 dataset is used to benchmark feature-based motion segmentation algorithms and consists of 155 motion sequences [48]. The dataset contains only affine motions and no perspective motions. The sequences contain independent and partially dependent motions, as well as rigid, non-rigid, articulate, and degenerate motions. There are 120 sequences containing two motions while the rest contain three motions. For some of the sequences, the camera is moving which means that the background pixels have motion as well. Some of the sequences, including their feature trajectories and ground truth segmentation, were obtained from the following papers: [49, 50, 37]. For the rest of the sequences, the feature trajectories were determined using the OpenCV library for feature extraction and tracking [51]. The ground truths were obtained by labelling the trajectories by hand.

The Hopkins155 dataset sequences can be grouped into three categories, namely checkerboard, traffic and articulated. The checkerboard sequences consist of 104 indoor videos of moving checkerboards while there are 38 outdoor traffic sequences and 13 sequences containing articulated and/or non-rigid motions. The 104 checkerboard sequences contain moving objects, covered in checkerboard patterns. These objects undergo rotations and/or translations. The 38 traffic sequences are of vehicles on a street or in a parking lot. Most of these sequences contain linear and planar motions which are degenerate. The articulate group of 13 sequences contains articulate and non-rigid motions such as people, facial expressions, and cranes. The Hopkins155 dataset has an additional 12 missing data sequences. The dataset is freely available for use under free use non-commercial copyright license.

2.5.2 KT3DMoSeg

The KT3DMoSeg dataset is derived from the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) dataset, presented in [52], in order to create a feature-based motion segmentation dataset which contains more challenging and realistic sequences [53]. The dataset contains sequences of perspective motions as well as forward translations not captured by previous motion segmentation datasets, such as the Hopkins155 dataset. The Hopkins155 dataset has limitations such as a small number of moving objects and limited scene depths. The sequences from the Hopkins155 dataset have a maximum of 3 moving objects and the depths within the scenes are limited. These sequences are complete and contain no outliers. This unrealistic since most real-world sequences are corrupted by noise, missing data and outliers. Even though some of the Hopkins155 sequences were taken with a moving camera, the camera motions are controlled and limited i.e., none of the sequences were taken by a camera undergoing large transitions such as a camera mounted on a vehicle. Also, most camera

motions are rotations. The KT3DMoSeg data set seeks to address these limitations.

The KT3DMoSeg dataset consists of 22 sequences, each containing between 10 to 20 frames. Since the KITTI dataset is dense-based, feature trajectories had to be extracted from the video frames as well as their corresponding ground truth segmentation. Dense trajectories were extracted using the feature extraction method presented in [54] and trajectories with a length of fewer than five frames were discarded. The 22 sequences are short video sequences selected from the KITTI dataset according to the specific criteria to increase the complexity of the dataset and make the dataset more realistic. Significant camera translations are preferred, therefore sequences taken with cameras on moving vehicles were selected. Scenes with complex backgrounds and strong perspective motion increase the complexity of the motion segmentation task significantly and closely model real-world types of sequences expected for tasks such as self-driving cars. Additionally, large numbers of motions are difficult to segment, therefore sequences with more than three moving objects were selected. The dataset is freely available for use with no copyright restrictions.

2.6 CONCLUSION

The main approaches used to address the motion segmentation problem can be classified as follows: Image Difference, Wavelets, Optical Flow, Statistics, Layers, Manifold Clustering, Template Matching and Deep Learning. A few key methods from each of these categories were considered and compared. It is clear that the existing methods are far behind human capabilities and that occlusions and missing data remain a challenge. Additionally, manifold clustering approaches were found to be effective approaches since they have a strong mathematical foundation and can be used for a variety of applications. Therefore, manifold clustering approaches were investigated further, focusing on the most popular algorithms to date. These include GPCA, LSA, ALC, SSC, and LRR. Since the focus of this work is to address the occlusion problem as well as dealing with missing data, existing methods that aim to solve these problems were investigated. However, it was found that most of these methods focused on handling missing data and were not explicitly tested on large and complete occlusion scenarios. Therefore, there is a need for a manifold clustering algorithm that can handle large and complete occlusions. Lastly, the Hopkins155 and KT3DMoSeg datasets were discussed since these datasets are used to benchmark manifold clustering algorithms.

CHAPTER 3 METHODS

3.1 CHAPTER OVERVIEW

In Section 3.2, the metrics used to evaluate the performance of manifold clustering algorithms are discussed, namely the average, mean and median of the misclassification error. In Section 3.3, a manifold clustering approach was selected to form the base on which to build a method that can handle large and complete occlusions as well as missing data. The most popular manifold clustering approaches are compared theoretically, according to the performance reported by the original authors. Subsequently, the algorithms were executed on the Hopkins155 and KT3DMoSeg datasets and their performance were compared to select a base algorithm. SSC was selected accordingly. Then, methods to add occlusion handling capabilities were discussed in Section 3.4. A multiple split-and-merge approach was selected. In Section 3.5, the new algorithm, referred to as MSAM-SSC, was introduced, explained and verified to be a viable solution by comparing the performance to that of SSC and MSMC. Next, the algorithm was optimised in Section 3.6 and the final algorithm is given in Section 3.7.

3.2 PERFORMANCE METRICS

Performance metrics are essential to effectively evaluate and benchmark algorithms. These metrics give an insight into the strengths and weaknesses of an algorithm. For motion segmentation problems, the misclassification error is used to evaluate the algorithm performance. The misclassification error computes the percentage of feature points that were classified to the wrong motion segment across the entire input sequence:

$$Misclassification = \frac{\#\ of\ misclassified\ points}{total\ \#\ of\ points}. \tag{3.1}$$

To evaluate the algorithm performance over an entire dataset, the average, median and standard deviation of the misclassification error are considered. This gives insight into the distribution of the misclassification errors such that the performance can be evaluated. These metrics were used throughout the development and optimisation stages to evaluate the developed manifold clustering

algorithm.

3.3 MANIFOLD CLUSTERING ALGORITHM EVALUATION

As a first step, some of the most popular manifold clustering algorithms were investigated and their performance were compared. The aim was to identify the strengths and weaknesses and identify an algorithm on which to improve on. The implementations of the ALC, ELSA, LRR, LS3C and SSC algorithms are freely available on the UdGMS-19 dataset website [55]. The implementation of SSC is also available on the Hopkins155 dataset website [56].

ALC is a classic approach that adopts principles from rank minimisation and lossy compression to form a sparse representation for the motion trajectories [33]. ELSA builds on the traditional LSA approach by automatically setting and tuning its parameters and was reported to outperform ALC on the Hopkins155 dataset [27]. SSC is another classic approach that employs compressed sensing to express points located at a union of subspaces as a linear or affine combination of all the other points using a sparse framework. The coefficients are determined by optimising the ℓ_1-norm and used to construct an affinity matrix to which spectral clustering is applied to find the final motion segments. SSC was only tested on the Hopkins155 dataset and was reported to outperform ALC [34]. LRR aims to obtain the joint lowest-rank representation of the set of point trajectories. It builds on classic sparse representation by finding the global sparse structure of all the point trajectories rather than for each trajectory. This structure allows LRR to handle multiple motions as well as noise and corrupt data. LRR outperformed SSC on the Hopkins155 dataset and the Extended Yale B dataset, which is a facial recognition dataset [40]. LS3C aimed to improve the classic SSC algorithm by projecting the data to a latent space of lower dimensionality while the sparse coefficients are computed using an adapted optimisation function. LS3C was tested on the Hopkins155 dataset and reportedly outperformed LRR and SSC [41].

All the algorithms were initially tested on the Hopkins155 dataset, and algorithms such as LRR and LS3C were tested in domains other than motion segmentation as well (LRR was tested in the facial recognition domain and LS3C on handwritten characters) [40, 41]. ELSA was tested on synthetic data which contained four and five motions as well as different noise levels, and the results were compared with that of ALC [27]. It was reported that ELSA outperformed ALC for both sets of synthetic data. SSC was also tested on additional data containing missing data and outliers, and the performance was compared to that of ALC [34]. For the missing data test, the 12 additional missing data sequences

from the Hopkins155 dataset was used along with generated missing data sequences using the original Hopkins155 sequences. Outlier sequences were also generated from the Hopkins155 sequences. It was reported that SSC outperformed ALC in both cases. None of these algorithms were tested on large or complete occlusion scenarios by the original authors, therefore, they are all candidates that can be improved upon.

3.3.1 Comparison of Manifold-clustering Algorithms

As an initial step, the algorithm implementations were compared with each other. This is to compare the performance of the algorithms since the original papers did not compare all the algorithms with each other. The algorithms were executed on the Hopkins155 and KT3DMoSeg datasets to form a more comprehensive understanding of the performance. This is necessary since most algorithms were only tested on the Hopkins155 dataset and may not generalise well. The misclassification metrics of ALC, ELSA, LRR, LS3C and SSC on the Hopkins155 and KT3DMoSeg datasets are shown in Tables 3.1 and 3.2, respectively. It is clear that none of these algorithms is an optimal approach and the average misclassification errors are relatively high.

Considering the performance metrics on all the Hopkins155 sequences in Table 3.1 (the last column), it is evident that ELSA outperformed the other algorithms since it had the lowest average misclassification error. It also has the lowest median misclassification error; however, the standard deviation is relatively large. The next best-performing algorithms are ALC and LS3C. SSC and LRR had the worst overall performance. Considering the performance on the subsets of sequences, it is evident that ALC performed significantly better than the rest of the algorithms on all the subsets containing three motions, as well as the articulate set containing two motions. ELSA performed the worst on the subsets containing three motions but the best on the checkerboard and traffic subsets containing two motions. Additionally, ELSA and LS3C performed the best on the missing data subset while SSC and ALC performed the worst. LRR performed significantly better on the traffic subsets than on the checkerboard and articulate subsets, while LS3C performed significantly better on the articulate subsets. SSC performed relatively consistent on all the subsets but had the best performance on the articulate subset containing two motions.

Note that ELSA has two cases where the standard deviation is higher than the mean. This is for the checkerboard and articulate subsets containing two motions. Since the standard deviation is a measure of the spread of the misclassification errors around the mean, this indicates that the distribution of

Table 3.1. Misclassification metrics of different manifold clustering algorithms on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
Average								
ALC	34.44	30.13	8.87	6.78	6.03	7.25	37.78	25.75
ELSA	10.69	22.37	18.70	54.20	46.11	51.00	23.95	22.54
LRR	32.91	25.21	27.38	42.43	29.17	47.55	34.82	31.99
LS3C	25.51	27.52	18.73	39.62	35.83	15.90	25.61	27.31
SSC	30.92	27.29	19.27	43.47	36.75	31.81	39.10	31.52
Median								
ALC	31.97	19.58	0.95	0.92	1.35	7.25	39.43	15.89
ELSA	0.53	16.67	12.82	54.04	52.52	51.00	13.42	5.56
LRR	33.47	23.81	27.27	42.67	21.23	47.55	33.36	33.60
LS3C	25.97	32.27	10.96	40.77	35.55	15.90	24.39	27.93
SSC	32.72	36.34	20.51	44.89	45.07	31.81	44.35	34.85
Standard Deviation								
ALC	26.11	29.17	14.90	10.78	11.17	9.31	30.88	26.70
ELSA	19.63	21.83	19.67	9.26	11.54	1.41	26.35	24.63
LRR	10.04	14.07	14.57	11.55	13.50	10.07	10.28	13.46
LS3C	15.37	17.00	16.83	12.58	20.39	12.12	19.23	17.00
SSC	13.21	18.62	16.77	11.20	14.78	16.70	16.69	16.31

the misclassification errors does not follow a normal distribution or that outliers with large values are present.

Different observations are made on the KT3DMoSeg dataset in Table 3.2. Here, LRR had the lowest average misclassification error followed closely by SSC. ALC and ELSA performed considerably worse with a significantly higher average and median misclassification.

The significant difference in the performance of ALC, ELSA and LS3C between the datasets is due

Table 3.2. Misclassification metrics of different manifold clustering algorithms on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
ALC	58.52	53.10	21.24
ELSA	48.86	48.23	11.25
LRR	32.02	31.33	15.33
LS3C	44.96	43.62	16.65
SSC	34.02	34.79	12.32

to the difference in the complexity of the datasets. The KT3DMoSeg sequences contain a wider variety of motions with complex camera motions, while the Hopkins155 sequences contain two or three motions and the camera motions are limited to simple rotations and translations. Additionally, these three algorithms were only tested on the Hopkins155 dataset by the original authors, therefore, these algorithms do not generalise well. In contrast with this observation, LRR and SSC had similar performance on both datasets which show that these algorithms generalise well. Therefore, either LRR or SSC is considered as the preferred approach to be extended to include occlusion handling. Comparing the performance of LRR and SSC on all the sequences from both datasets, LRR had slightly better accuracy and misclassification, but SSC performed better than LRR on most of the subsets of the Hopkins155 dataset. Therefore, SSC was selected as the preferred approach to be improved upon.

3.4 OCCLUSION HANDLING

The objective is to improve the SSC algorithm to be able to handle large and complete occlusions and missing data efficiently. Existing methods that employ occlusion handling are investigated to formulate approaches that can be used to improve the SSC framework to handle occlusions effectively.

3.4.1 Model Missing Data

Occlusions is one way in which data can become missing, therefore, occlusions can be handled if an algorithm has an effective method to handle large amounts of missing data. One way to handle these cases is to explicitly model missing data. Therefore, the data matrix can be written as

$$\mathbf{X} = \mathbf{X}_p + \mathbf{X}_m, \tag{3.2}$$

where $\mathbf{X}_p$ is the present data $\mathbf{X}_m$ is the missing data. As discussed in section 2.4.1, ASSA is an extension of SSC which models the missing data explicitly. ASSA uses the ℓ_1 optimisation to solve for both present and missing data. The ℓ_1 optimisation for the missing data is

$$\begin{aligned} &\min_{\mathbf{C},\mathbf{E},\mathbf{Z}} ||\mathbf{C}||_1 + \lambda_e||\mathbf{E}||_1 + \frac{\lambda_z}{2}||\mathbf{Z}||_2^F \\ &\text{s.t. } \mathbf{X}_p + \mathbf{X}_m = (\mathbf{X}_p + \mathbf{X}_m)\mathbf{C} + \mathbf{E} + \mathbf{Z}, \quad diag(\mathbf{C}) = 0, \quad (\mathbf{X}_m)_p = 0, \end{aligned} \tag{3.3}$$

where $\mathbf{E}$ accounts the outliers and $\mathbf{Z}$ for the noise. Other methods can be used to model missing data. The constraints were derived from the constraints used in classic SSC when solving for the ℓ_1 optimisation problem [42]. However, ASSA was not tested on complete occlusions, but only on missing data sequences. The ℓ_1 optimisation may not be the optimal method to recover the missing data.

GoDec+ is an existing manifold clustering method that can handle large and complete occlusions, as well as missing data and noise [44]. As discussed in Section 2.4.3, GoDec+ models corruptions using MCC and optimises the correntropy by minimising the Welsch M-estimator. A similar approach can be followed by updating the SSC minimisation function to include the MCC noise model. Here, the missing data matrix can be retrieved by solving the MCC objective, namely

$$\min_{\mathbf{X}_p} \sum_{i}^{F}\sum_{j}^{N}[1 - g_\sigma(\mathbf{X}_{m(i,j)})], \quad \text{s.t. } \mathbf{X} = \mathbf{X}_p + \mathbf{X}_m, \quad \mathbf{X}_p = A\mathbf{s}, \tag{3.4}$$

where $g_\sigma(\mathbf{X}_{m(i,j)})$ is the Gaussian kernel. Then, $\mathbf{X}_p$ is estimated by optimising the ℓ_1 optimisation, as with classic SSC.

3.4.2 Depth Information

Depth information provides information regarding the distance of the object from the camera. These distances can be used to identify objects which undergo occlusions. Often this information is obtained along with the video material using an RGB-D camera. However, as with the Hopkins155 and KT3DMoSeg datasets, this information is not always available and must be inferred. If multiple cameras are used to generate input, visual odometry techniques can be used to compute the point correspondences in each frame and compute the corresponding depth. If only a single camera is used, optical flow or visual odometry techniques can be used to compute the depth of points such as presented in [57, 58, 59].

Classic SSC can operate on 3D point trajectories but additional processes to handle the incomplete trajectories caused by occlusions and missing data are still required. One method is to assign trajectories to depth layers, as is done in layered approaches [14, 15]. Traditionally, layered approaches use a dense

representation rather than a feature-based representation. This principle can be adapted to fit the SSC framework by assigning each motion trajectory to a depth layer. This approach has some drawbacks. First, a pre-processing step is required to infer the depth information of each point trajectory. Then, the depths must be assigned to a layer. The layering information can be exploited to simplify the segmentation task since trajectories assigned to the same depth layer are more likely to belong to the same segment. However, by adding depth information (to achieve the layering) the data dimensionality and problem complexity are increased.

3.4.3 Frame-to-frame Analysis

Frame-to-frame analysis aims to extract information from two frames that can be used to segment point trajectories to the corresponding motion segments. The extracted information can range from point correspondences to inter-frame motion segments. In this case, the only time when a point is not assigned to a motion is when it is not visible in one of the two consecutive frames (i.e., when the point disappears or reappears from the camera view).

An existing frame-to-frame improvement of the traditional SSC method is M-TPV [47], which was discussed in Section 2.4.5. M-TPV incorporated the frame-to-frame analysis by changing the ℓ_1-norm to the $\ell_{1,1,2}$-norm. M-TPV was tested on missing data sequences and performed better than classic SSC, but it was not tested on complete occlusions.

Another manifold clustering method that utilises frame-to-frame analysis is MSMC. The MSMC is unrelated to SSC and performs frame-to-frame analysis using an algorithm called MSAM [45], as discussed in Section 2.4.4. It is possible to adapt the MSAM algorithm to be a pre-processing step for the traditional SSC algorithm. The challenge is to combine these inter-frame motion segments in such a manner that it can be used by SSC to obtain the final motion segments. One advantage of this approach is dimensionality reduction which can improve the SSC execution time. However, this is a pre-processing step that must be applied to each consecutive frame pair and may increase the overall execution, even though the execution time of SSC may decrease.

3.4.4 Selected Occlusion Handling Approach

Three possible approaches to handle occlusions are considered. The first approach models missing data explicitly. This approach can be effective if the model accurately reflects the occlusion scenario. Note that modelling the missing data increases the problem complexity and data dimensionality, and therefore this approach is not ideal. The second approach relies on depth information which must

be inferred as a pre-processing step. Similar to modelling the missing data, the addition of depth information increases the problem complexity and execution time due to the increase in the data dimensionality. Therefore, this approach is also not ideal. The frame-to-frame analysis approach can handle large amounts of missing data. M-TPV is an existing frame-to-frame adaption of the SSC algorithm which has shown to be able to handle large occlusions. The MSAM algorithm, which forms part of the MSMC algorithm, is also effective since the MSMC algorithm can reportedly handle up to 98% missing data [45]. One advantage of the frame-to-frame analysis is that the data dimensionality can be reduced which reduces the problem complexity and execution time. Therefore, the MSAM algorithm was selected and to be used as a pre-processing step for classic SSC to add occlusion handling.

3.5 THE MSAM-SSC ALGORITHM

An algorithm based on the SSC which aims to solve the occlusion problem and handle missing data, referred to as the Motion-Split-And-Merge Sparse Subspace Clustering (MSAM-SSC) algorithm, is proposed. The MSAM algorithm is used as a pre-processing step before the classic SSC algorithm is executed. The MSAM algorithm extracts the motion regions between pairs of image frames. These regions are placed in a 2D data matrix which is then given to the SSC algorithm as input. Therefore, the SSC algorithm groups inter-frame motion regions rather than point trajectories. No additional modifications to the SSC algorithm are required. The MSAM-SSC algorithm is discussed in detail, giving focus to the data manipulation and computations.

3.5.1 Multiple Split-and-merge

The MSAM algorithm computes the motion regions between each consecutive frame pair from the input video. This is achieved by iteratively applying split-and-merge operations. The MSAM algorithm iteratively executes the following split and merge operations:

- Split class: a class is split into two classes if the homogeneity is less than the split threshold.
- Merge classes: merge two classes if the homogeneity computed over both classes is above the merge threshold.
- Split outliers: Remove points identified as outliers.
- Merge outliers: Merge a class with the outlier class.

Additionally, the MSAM algorithm has the following thresholds used by the split and merge operations:

- Split threshold, θ_s: Used to determine if a class must be split into two.
- Merge threshold, θ_m: Used to determine if two classes must be merged to form a single class.
- Inlier threshold, θ_i: Used to determine which points are class inliers.
- Outlier threshold, θ_o: Used to determine if outliers must be removed from the class.

MSAM has the following parameters:

- Minimum class size, ϕ_1: the minimum number of points required to form a class.
- RANSAC minimum sample size, ϕ_2: The minimum number of class inliers required for the homogeneity computation.
- Second RANSAC sample size, ϕ_3: If the number of class inliers is larger than this threshold, the inliers and homogeneity measure must be recalculated using the first set of computed class inliers. It is set to double the RANSAC minimum sample size.

The algorithm receives two $3 \times N$ matrices containing the N point locations for each of the two frames. First, the values of the entries corresponding to missing points are set to *Not a Number (NaN)*. These two matrices are used to identify the motion regions between the two frames. It is important to note that if a point is visible in the first frame, but not the second, the corresponding location for the point in the second frame will contain *NaN*. This phenomenon occurs for image pairs that contain key points that appear or disappear from the view of the camera. Points that are only visible in one frame will not be classified to a motion class and will only be assigned to a class for frame pairs in which the point is visible in both frames. This allows for points that disappear and reappear at a later stage, such as when points are occluded, to be detected and segmented. Since the SSC algorithm is only used to group inter-frame segments, points and segments which undergo occlusions can still be segmented since these segments have been identified by MSAM.

Initially, all the points visible in both frames are assigned to the same class. A vector is used to store the class assignment of each point. Class label 0, is used to identify any outliers. MSAM iteratively performs the above-mentioned split and merge operations. At the start of each iteration, before any split or merge operations are performed, the class size is verified to be larger than the minimum allowed class size. If the class does not have enough points, the class is dissolved and all the points are classified as outliers. If the class is not dissolved, the homogeneity factor is computed, and class inliers are

determined. Points not identified as inliers, i.e., outliers, are removed if the homogeneity factor is larger than the outlier threshold θ_o.

3.5.1.1 Homogeneity Factor and Class Inliers

The homogeneity factor is a measure of the similarity between points and lies in the range $[0,1]$ where 0 is the worst and 1 the best possible homogeneity. The homogeneity is computed using a RANSAC process. RANSAC is used to find the set of points from the input data which fits the same model. Here the model is defined as the error between the true and projected point locations in both images. The objective is to find the set of inliers that minimises the residual value. The inliers are then used to compute the homogeneity factor.

The RANSAC process starts by randomly sampling ϕ_2 points. The homography $\mathbf{H}$ from the sample set of points in the two images is computed. Then a residual vector is constructed. The residual combines the error of projecting the points in the first image to the second and vice versa. For true class inlier points, the residual value is low. The residual value is computed as follows: the points in the first image are projected onto the second image using the homography $\mathbf{H}$. The Euclidean distance between the true point locations and the projected point locations in the second image is computed for each point in class c. The same process is followed for the points in the second image, which are projected onto the first using the inverse homography $\mathbf{H}_I = \mathbf{H}'$. The final residual value is computed as

$$d_{12} = \frac{1}{2}(\sum_{n=1}^{N_c} d_{1,n} + \sum_{n=1}^{N_c} d_{2,n}), \tag{3.5}$$

where N_c is the number of points in class c, $d_{1,n}$ is the distance between the location of projecting point n onto the second image and its true location in the second image, and $d_{2,n}$ the distance between the true and projected point locations in image 1. The class inliers are the points for which the square root of the residual value is below the class inlier threshold θ_i. If the number of inliers is larger than the second sample size ϕ_3, the homography and residual vector are recomputed using only the set of inliers. Since $\phi_3 = 2\phi_2$, it means that the class inliers and homogeneity is recomputed using the class inliers if the number of inliers is larger than double the first RANSAC minimum sample size. The purpose of this is to enhance the inlier and homogeneity computation. To further enhance the computation of the model parameters (the homography), the best result, meaning the highest number of inliers, from $r = 11$ iterations are selected. Once the highest number of inliers have been determined, the homogeneity factor is computed as

$$homogeneity = \frac{\text{number of inliers}}{N_c}. \tag{3.6}$$

The homogeneity function returns both the homogeneity factor and the residual vector since the residual vector is used in other steps, such as to split a class into two.

3.5.1.2 Split Class

Once all the outliers are removed, the classes are split if their corresponding homogeneity is smaller than the split threshold θ_s. If a class must be split, a residuum matrix is generated first by re-sampling the class. The re-sample process uses the homogeneity computation to obtain the residual vector for 10 samples. These residual vectors form the columns of the residuum matrix, and the Jacobian distance d_m between the residual vectors are computed. d_m is a measure of the distance between the locations of each feature in both frames. Since the residual value measures the error of projecting a point from the first to the second frame and vice versa, the Jacobian distance is a measure of similarity and is used to identify points that underwent similar motion between the two frames. Additionally, the standardised Euclidean distance between all points in the second frame, d_s, is computed since points from the same class tend to be in close proximity. These two distance measures are combined to form the affinity measure as follows

$$\mathbf{A} = (1 - d_m) * \min\{d_s, 1\}, \tag{3.7}$$

where d_m is the Jacobian distance between the locations of each feature in both frames and d_s and the standardised Euclidean distance between all points in the second frame. The affinity measure is used to construct the affinity matrix, which is used by k-means to split the class into two.

3.5.1.3 Merge Classes

After the split operation was performed on each class, the merge operation is performed on all pairs of classes. For each class pair, the combined homogeneity is computed, and the classes are merged if the homogeneity is above the merge threshold θ_m.

3.5.1.4 Split Outlier Class

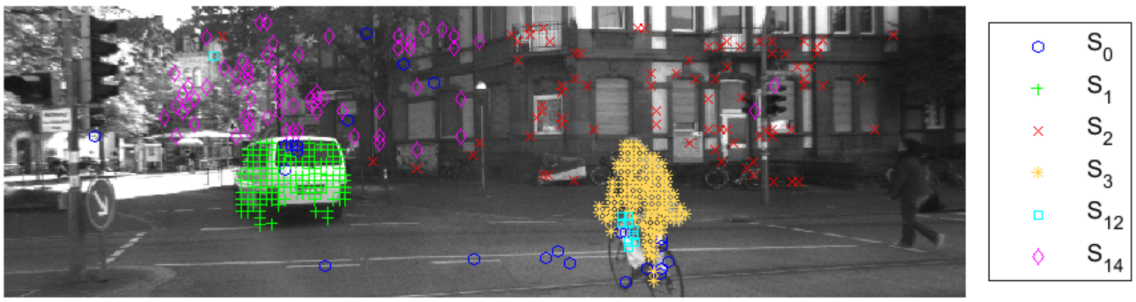
Next, points from the outlier class are removed to form a new class as follows: the RANSAC step for the homogeneity computation is used to identify inliers or points in close proximity. If the number of inliers is larger than the minimum class size, these points are assigned to a new class.

3.5.1.5 Merge Outliers

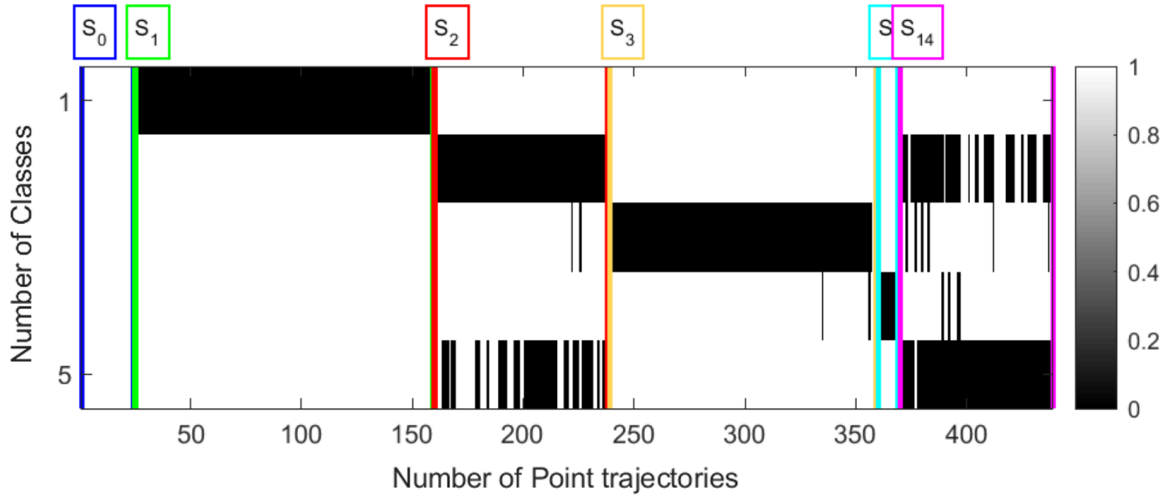
Lastly, any redundant classes are identified and merged with the outlier class. Once again, the RANSAC process is used to identify the class inliers but instead computed over the outlier class and another class. The points from the other class which are identified as inliers are closer to the points in the outlier class than the other class. Therefore, these points are assigned to the outlier class.

3.5.1.6 Residual Matrix

After the class and outlier split and merge steps have been completed, a total of $S_{i,j}$ classes are obtained for a frame pair (i, j). The final step of the MSAM algorithm is to compute the residual matrix. The residual matrix is a $S_{i,j} \times N$ matrix which indicates which points belong to each of the $S_{i,j}$ classes. The matrix has the familiar block structure associated with the affinity matrix. The residual matrices for each frame pair are used to construct a global residual matrix that forms the input for the classic SSC algorithm. Figures 3.1, 3.2, 3.3 and 3.4 show the inter-frame motion segments determined by applying MSAM to different frame pairs from the *Seq005_Clip01* sequence from the KT3DMoSeg dataset. The corresponding residual matrices are also shown.



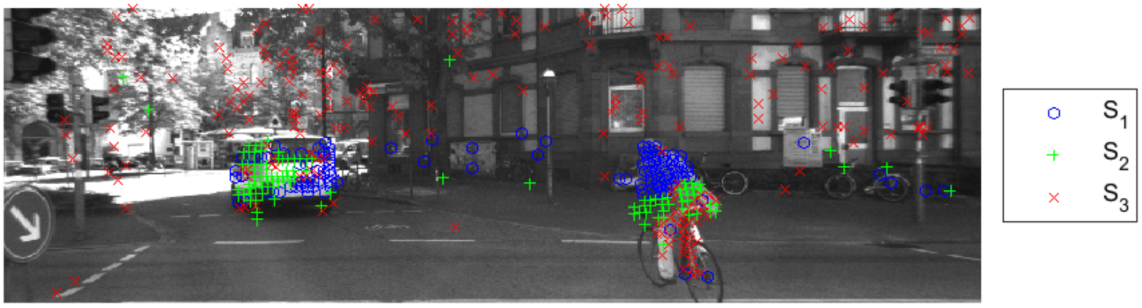
(a) Inter-frame motion segments.



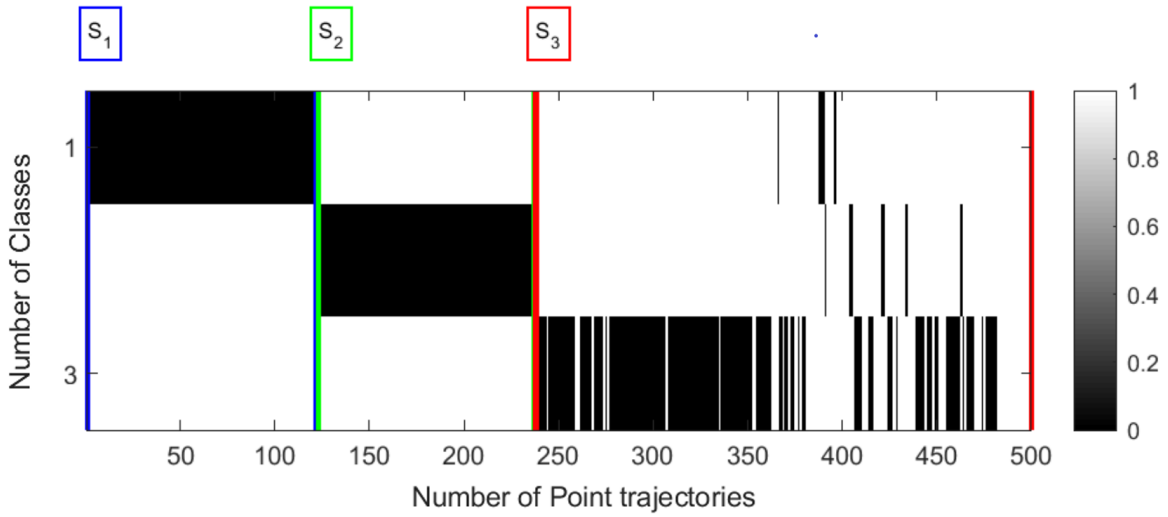
(b) Residual matrix.

Figure 3.1. Inter-frame motion segment of frames 1 and 2 from *Seq005_Clip01* sequence. (a) Inter-frame motion segments. (b) Residual matrix.

Adapted from [53], ©2018 IEEE



(a) Inter-frame motion segments.

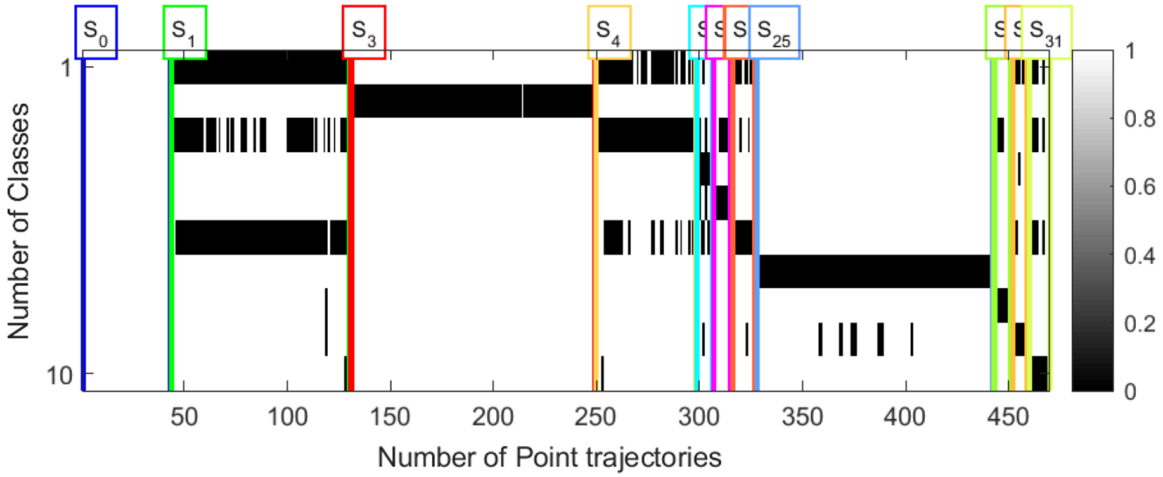


(b) Residual matrix.

Figure 3.2. Inter-frame motion segment of frames 8 and 9 from *Seq005_Clip01* sequence. (a) Inter-frame motion segments. (b) Residual matrix.

Adapted from [53], ©2018 IEEE

(a) Inter-frame motion segments.

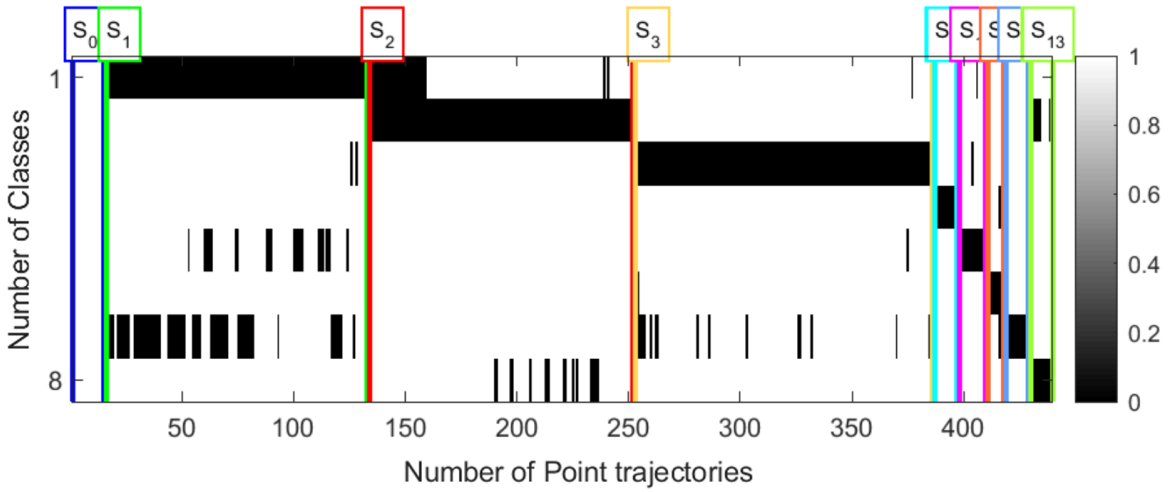


(b) Residual matrix.

Figure 3.3. Inter-frame motion segment of frames 14 and 15 from *Seq005_Clip01* sequence. (a) Inter-frame motion segments. (b) Residual matrix.

Adapted from [53], ©2018 IEEE

(a) Inter-frame motion segments.



(b) Residual matrix.

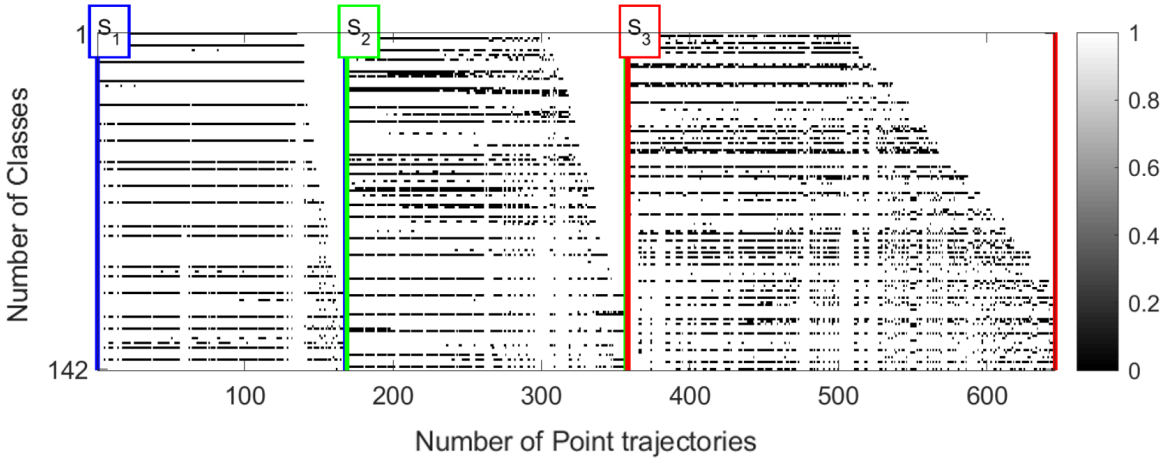
Figure 3.4. Inter-frame motion segment of frames 18 and 19 from *Seq005_Clip01* sequence. (a) Inter-frame motion segments. (b) Residual matrix.

Adapted from [53], ©2018 IEEE

3.5.1.7 Inter-frame Motion Segments

Once all the inter-frame motion segments and the residual matrices for each frame pair has been computed, the data can be prepared for the classic SSC algorithm. The residual matrices are combined to form a single global residual matrix. Therefore, the global residual matrix contains the inter-frame segmentation for each feature point. To obtain the final segmentation, the SSC algorithm must group these inter-frame segments such that a matrix with diagonal block structure, namely an affinity, is obtained and spectral clustering is applied to find the final motion segments. Figure 3.5 shows the global residual matrices.

(a) Inter-frame motion segments.



(b) Global residual matrix.

Figure 3.5. Global residual matrix of *Seq005_Clip01* sequence. (a) Inter-frame motion segments. (b) Global residual matrix.

Adapted from [53], ©2018 IEEE

3.5.2 SSC

Once the inter-frame motion segments and the global residual matrix have been computed, the data can now be given to the classic SSC algorithm. The classic SSC algorithm consists of the following steps:

- Recovery of the Sparse Coefficients
- Adjacency matrix construction
- Spectral clustering

3.5.2.1 Recovery of the Sparse Coefficients

The sparse coefficients are recovered by solving for the ℓ_1-norm for the data matrix $\mathbf{X}_p$. Here $\mathbf{X}_p$ is the global residual matrix computed by MSAM which contains the inter-frame motion segments. Since the noise level of the data is unknown, the Lasso optimisation algorithm [60] is used to solve the ℓ_1-norm and obtain the $N \times N$ matrix of coefficients, where column i corresponds to the sparse coefficients of the point located in column i of $\mathbf{X}_p$. The Lasso optimisation function has a single regularisation parameter which is set to $\lambda_L = 0.001$. The CVX MATLAB package is used to model convex optimisation and is utilised by the Lasso optimisation algorithm to improve the computation [61, 62].

Figure 3.6 shows a visual representation of the coefficient matrix computed by MSAM-SSC on *Seq005_Clip01* from the KT3DMoSeg dataset. Considering the visual representation, there are three blocks on the main diagonal, which indicate the three classes of motion to be extracted by the spectral clustering process. Since the coefficient matrix is used to construct the affinity matrix, which in turn is used by spectral clustering to compute the final segmentation, it is expected that the affinity matrix has a similar structure. The coefficient matrix cannot be used as is by spectral clustering since the coefficient values are relatively close, as can be seen by considering the heat map. Therefore, the construction of the affinity matrix must favour the values in the same block such that the difference in values between the values inside and outside the block diagonals are larger to increase the accuracy of spectral clustering.

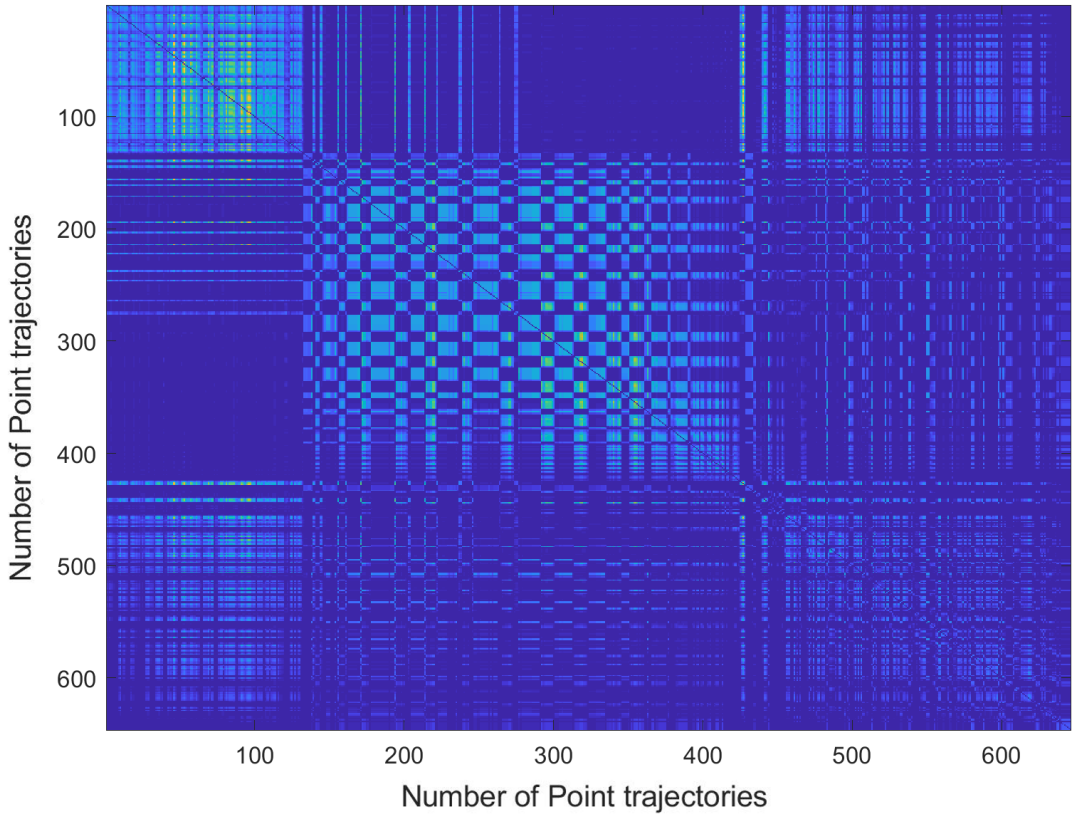


Figure 3.6. Coefficient matrix of *Seq005_Clip01* sequence

3.5.2.2 Adjacency matrix construction

The computed $N \times N$ sparse coefficient matrix is used to compute a $N \times N$ adjacency matrix and spectral clustering is applied to the adjacency matrix to obtain the final motion segmentation. A similarity graph is constructed using the coefficient matrix and the adjacency matrix is the $N \times N$ matrix containing all the graph connection weights.

Figure 3.7 shows a visual representation of the affinity matrix computed by MSAM-SSC on *Seq005_Clip01* from the KT3DMoSeg dataset. The affinity matrix has the same block diagonal structure of the coefficient matrix shown in Figure 3.7 but the difference between the values inside the block diagonals and outside the block diagonals are larger here. This leads to increased accuracy of the spectral clustering step.

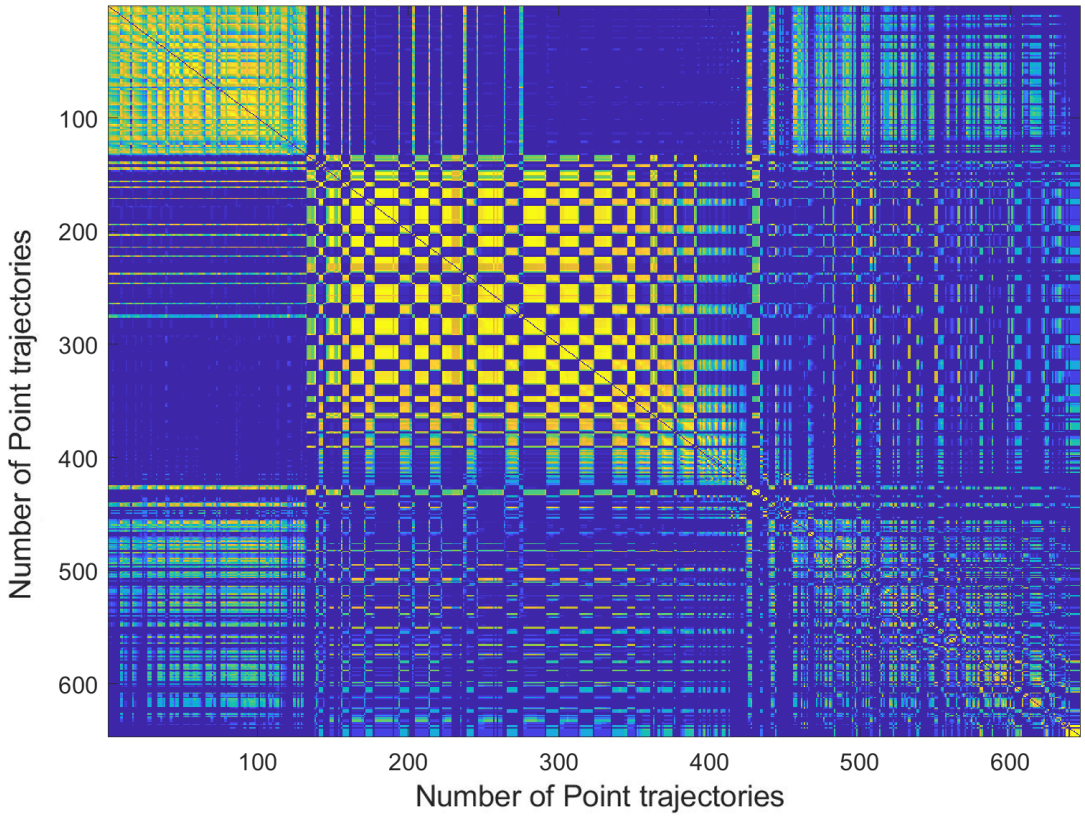


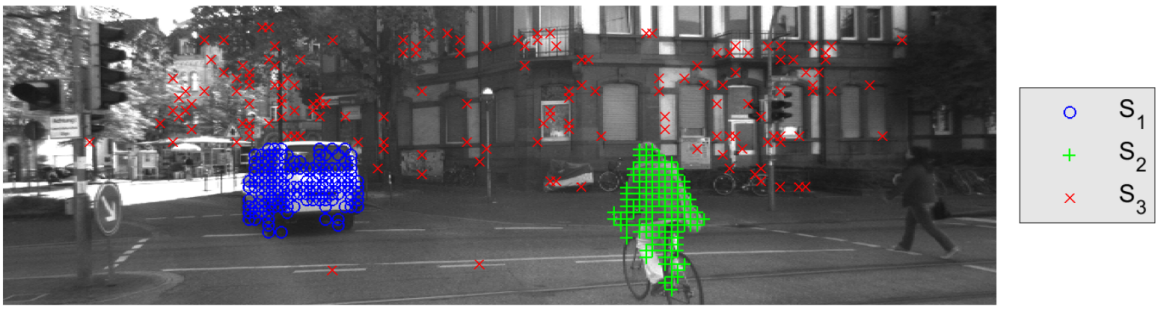
Figure 3.7. Affinity matrix of *Seq005_Clip01* sequence

3.5.2.3 Spectral clustering

Spectral clustering is applied to the adjacency matrix to obtain the final motion segments. K-means spectral clustering is used. The maximum iteration for k-means algorithm is set to 1000 and the number of times the clusters must be replicated is set to 100. Additionally, the number of motion segments is required beforehand.

Figure 3.8 shows the final results of the MSAM-SSC algorithm on *Seq005_Clip01* from the KT3DMoSeg dataset. The ground truth annotation is shown on the left and the MSAM-SSC classification on the right. It is evident that MSAM-SSC has a high accuracy on this sequence.

(a) Ground truth.



(b) MSAM-SSC results.

Figure 3.8. Final results of *Seq005_Clip01* Sequence. (a) Ground truth. (b) MSAM-SSC results.

Adapted from [53], ©2018 IEEE

3.5.3 Performance

To verify that MSAM-SSC is a viable solution to the motion segmentation problem, it must have a performance similar or better than classic SSC and MSMC which are the two base algorithms. The MSAM-SSC algorithm was executed on the Hopkins155 and KT3DMoSeg datasets and the results are shown in Tables 3.3 and 3.4, respectively.

Considering the performance on all the sequences in the Hopkins155 dataset in Table 3.3, MSAM-SSC had the best performance with the best average, median and standard deviation for the misclassification error. Similar observations are made for both checkerboard and traffic subsets. SSC performed the best on the two articulate subsets while MSMC performed the best on the missing data subset, followed closely by MSAM-SSC. The main purpose of MSAM-SSC is to be able to effectively handle large and complete occlusions and missing data. It is evident that MSAM-SSC already improves on the SSC algorithm when encountering missing data. Looking at the performance on the KT3DMoSeg dataset in

Table 3.4, MSAM-SSC had the best performance once again. It is clear that the MSAM-SSC algorithm has a better performance than the two algorithms it is derived from. The next step is to further improve the algorithm to increase the overall performance.

Table 3.3. Misclassification metrics of SSC, MSMC and MSAM-SSC on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
Average								
SSC	30.92	27.29	19.27	43.47	36.75	31.81	39.10	31.52
MSMC	37.20	19.82	33.94	44.63	24.43	52.38	33.67	33.50
MSAM-SSC	29.33	14.18	20.74	39.56	13.68	49.81	34.67	26.87
Median								
SSC	32.72	36.34	20.51	44.89	45.07	31.81	44.35	34.85
MSMC	46.58	11.95	41.03	60.07	21.90	52.38	40.82	43.84
MSAM-SSC	32.06	6.77	23.29	40.03	13.69	49.81	33.06	30.29
Standard Deviation								
SSC	13.21	18.62	16.77	11.20	14.78	16.70	16.69	16.31
MSMC	17.52	21.32	16.20	22.51	17.45	14.68	17.80	20.86
MSAM-SSC	13.10	16.72	16.79	10.81	7.13	8.76	10.09	16.23

Table 3.4. Misclassification metrics of SSC, MSMC and MSAM-SSC on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
SSC	34.02	34.79	12.32
MSMC	29.30	34.58	16.79
MSAM-SSC	21.59	20.18	12.04

3.6 OPTIMISATION OF THE MSAM-SSC ALGORITHM

The MSAM-SSC algorithm was verified to be an improvement on the classic SSC algorithm as well as the MSMC algorithm. MSAM-SSC also performed better on the missing data subset from the Hopkins155 dataset. The next step was to further improve and optimise MSAM-SSC. To achieve this, the MSAM part of the algorithm was investigated, and areas for possible improvements were identified. Possible improvements were implemented and tested on the Hopkins155 and KT3DMoSeg datasets. To increase the diversity of the input sequences (e.g., number of motions, motion types, scene types and noise), all the sequences from both datasets were used. A greater diversity of the input sequences prevented MSAM-SSC to be optimised for sequences with specific properties (e.g., complete and noise free sequences if only the original 155 sequences of the Hopkins155 dataset were used). Parameter values, thresholds, distance measurements and sub-algorithms were considered for improvements.

3.6.1 Minimum Class Size and RANSAC Minimum Sample Sizes

The MSAM algorithm has a minimum class ϕ_1 size parameter which determines the minimum number of points per class. Additionally, the algorithm has two parameters that are used by the RANSAC process during the homogeneity computation. The first RANSAC minimum sample size ϕ_2 defines the number of samples required to compute the homogeneity and its value must be less than or equal to the minimum class size. The second RANSAC minimum sample size ϕ_3 is used as a threshold to determine if the inliers and homogeneity must be recomputed from the first set of computed inliers. This is done only if the number of inliers is larger than ϕ_3. The value of ϕ_3 must be less than or equal to the minimum class size and larger than or equal to the first RANSAC sample size. These values were originally set to the values given for the MSMC algorithm, namely $\{\phi_1, \phi_2, \phi_3\} = \{8, 4, 8\}$ [45]. Two different combinations of these three parameters were executed to determine the effect of different combinations of these parameter values on the performance, namely $\{\phi_1, \phi_2, \phi_3\} = \{8, 4, 4\}, \{4, 4, 4\}$. For the first combination, the second RANSAC minimum sampling size is set to the same value as the first RANSAC minimum sample size which means that the inliers and homogeneity measure will be recalculated for a smaller number of inliers. This may improve the accuracy of the inliers and homogeneity measure since only the inliers computed during the first computation are used during the second. The disadvantage is that this can lead to longer execution times since the inliers and the homogeneity measure are recomputed more often. For the second combination, the minimum class size is decreased and set to be equal to the two RANSAC minimum sample sizes. This allows fewer points to form a class and can improve the performance, especially for sequences that contain objects

with few point trajectories (e.g. moving objects further away from the camera appear smaller and contain fewer point trajectories). When the minimum class size and first RANSAC sample size have the same values, all the points sampled from the class are assumed to be inliers and used to compute the homogeneity if there are only 4 points in the class. In this case, the inliers and homogeneity measure will not be recomputed which may decrease the overall performance. These parameters values were used and the algorithm was executed on the Hopkins155 and KT3DMoSeg datasets. The results are shown in Tables 3.5 and 3.6, respectively.

Considering the results on all the Hopkins155 sequences in Table 3.5, all combinations of the minimum class size parameters had similar performance, since all the average, median and standard deviation of the misclassification error are similar. Considering the subsets of sequences, the same observation is made for the missing data and both checkerboard subsets. For the traffic subset containing two motions, the parameter values $\{\phi_1, \phi_2, \phi_3\} = \{8,4,4\}$ has the lowest average misclassification. Parameter values $\{\phi_1, \phi_2, \phi_3\} = \{8,4,8\}$ performed better on the articulate subset containing two motions and on the traffic subset containing three motions but performed the worst on the articulate subset containing three motions. Overall, all three parameter sets had very similar performance. This can be due to the nature of the input sequences which contain similar numbers of motion and no noise, except for the missing data subset.

For the KT3DMoSeg dataset in Table 3.6, a different observation is made. Here the parameter values $\{\phi_1, \phi_2, \phi_3\} = \{8,4,4\}$ had a significant better performance with the lowest average and median misclassification errors. Since the second RANSAC sample size is smaller than originally, the inliers and homogeneity measure are recomputed more often which results in better performance. The minimum class size is double the size of the first RANSAC sample size, which means that the re-computation can occur even if a class has only the minimum number of samples. Therefore, the parameters are set to $\{\phi_1, \phi_2, \phi_3\} = \{8,4,4\}$ for the final MSAM-SSC algorithm.

3.6.2 Thresholds

The MSAM algorithm has four thresholds: two to determine if a split or merge operation must be executed, respectively; one to determine the class inliers during the homogeneity computation; and one to identify and remove outlier points from a class. The threshold values were originally set to the values used for MSMC [45]:

Table 3.5. Misclassification metrics of different size parameters on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
8,4,8	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
8,4,4	29.1	12.8	24.8	39.3	18.5	34.5	34.6	26.7
4,4,4	28.7	14.3	24.5	38.6	17.6	34.5	34.6	26.7
Median								
8,4,8	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
8,4,4	31.5	6.5	25.0	40.6	17.5	34.5	33.2	30.3
4,4,4	31.0	6.7	25.0	39.3	17.9	34.5	32.7	30.1
Standard Deviation								
8,4,8	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
8,4,4	13.2	16.1	16.5	10.5	10.3	12.9	10.1	16.0
4,4,4	13.6	17.0	16.2	9.9	9.4	12.9	10.1	15.9

Table 3.6. Misclassification metrics of different size parameters on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
8,4,8	21.6	20.2	12.0
8,4,4	18.8	16.0	11.7
4,4,4	23.9	21.1	10.8

- Split threshold: $\theta_s = 0.95$
- Merge threshold: $\theta_m = 0.9$
- Inlier threshold: $\theta_i = 5$
- Outlier rejection threshold: $\theta_o = 1$

The values of these thresholds were varied and executed on the Hopkins 155 and KT3DMoSeg datasets.

3.6.2.1 Split Threshold

The split threshold value was set to $\theta_s = \{0.95, 0.98, 0.9, 0.8\}$. One value higher than the original value was selected. The higher the threshold value, $\theta_s = 0.98$, the higher the homogeneity factor of the class must be to keep the class and not split it into two new classes. A higher split threshold can improve performance. Since the data can be corrupted by noise, a higher threshold can lead to over-segmentation which deteriorates the performance, therefore a lower threshold value was selected as well, namely $\theta_s = 0.9$. To evaluate the performance on a significantly lower value, $\theta = 0.8$ was selected as well. The performance metrics of the varied split threshold value on the Hopkins155 and the KT3DMoSeg datasets are shown in Tables 3.7, and 3.8, respectively.

When the performance on all the sequences from the Hopkins155 dataset is considered, it can be seen that $\theta_s = 0.98$ resulted in the best performance with the lowest average and median misclassification errors while $\theta_s = 0.95$ had the second-best performance. On the other hand, $\theta_s = 0.8$ resulted in the worst performance. It is clear a higher split threshold value leads to increased performance. Similar observations are made for the subset of sequences. However, for the articulate sequences containing three motions and the traffic sequences containing three motions, $\theta_s = 0.95$ had the best performance.

For the KT3DMoSeg dataset, it can be seen from Table 3.8 that all the split threshold values, except for $\theta_s = 0.95$, resulted in similar average misclassification errors. The lower threshold value of $\theta_s = 0.8$ and $\theta_s = 0.9$ had the lowest median values which indicate that a lower threshold value has a slightly increased performance. Note that the slight improvement has no impact on the average value. Therefore, the choice of the split threshold value does not have a significant impact on the performance. Since the higher threshold value of $\theta_s = 0.98$ resulted in slightly better performance across the Hopkins155 dataset, this value was selected for the final MSAM-SSC algorithm.

3.6.2.2 Merge Threshold

The merge threshold value was set to $\theta_m = \{0.9, 0.98, 0.95, 0.8\}$. Similar reasoning was used with the value selection as with the split threshold. A higher merge value requires that the points from two classes must have a higher homogeneity factor to be merged, therefore classes with higher homogeneity factors are encouraged. If the threshold value is too high, over-segmentation can be encouraged instead,

Table 3.7. Misclassification metrics of different split thresholds on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
Average								
0.95	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
0.98	27.7	10.9	21.3	38.6	15.2	23.5	31.6	25.0
0.9	31.3	15.8	25.0	42.7	19.9	49.8	34.7	29.1
0.8	31.7	19.1	27.1	43.6	16.5	50.8	34.7	30.0
Median								
0.95	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
0.98	30.5	0.6	23.3	39.7	16.1	23.5	30.4	28.4
0.9	33.3	11.8	25.0	42.2	17.0	49.8	33.4	32.2
0.8	33.2	14.0	27.6	44.6	15.6	50.8	33.1	32.9
Standard Deviation								
0.95	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
0.98	13.8	16.0	17.2	9.3	7.2	28.5	11.8	16.3
0.9	12.0	16.0	17.2	10.4	11.9	8.8	10.0	15.8
0.8	11.1	16.6	14.1	10.0	11.3	10.2	10.2	15.2

Table 3.8. Misclassification metrics of different split thresholds on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
0.95	21.6	20.2	12.0
0.98	20.2	19.6	11.0
0.9	20.8	17.6	13.2
0.8	20.0	17.2	12.2

and the performance deteriorates. The performance metrics of the different merge threshold values on the Hopkins155 and KT3DMoSeg datasets are shown in Tables 3.9 and 3.10.

The performance metrics of the Hopkins155 dataset in Table 3.9 indicate that the choice of merge factor does not have a significant effect on the performance on the sequences, since the algorithm performed similarly for all four values. Similar observations were made for the checkerboard and traffic sequences containing two motions. The articulate sequences containing two motions and the traffic sequences containing three motions had the best performance when $\theta_m = 0.9$. For the group of articulate sequences containing three motions, the merge values of $\theta_m = \{0.95, 0.98\}$ had the best performance. The performance of $\theta_m = 0.95$ was the best for the missing data sequences. The articulate subset containing three motions is the only instance where a significant difference in the average and median misclassification errors is observed. Here, the two lowest threshold values had the worst performance. Overall, the choice of merge threshold did not have a significant impact on the performance of the MSAM-SSC algorithm.

In contrast with the observations made on the Hopkins155 dataset, the choice of the merge threshold influenced the performance on the KT3DMoSeg dataset, as can be seen in Table 3.10. Here, the highest threshold value $\theta_m = 0.98$ had the best performance with the best average, median and standard deviation misclassification error while the lowest value $\theta_m = 0.8$ had the second-best performance. It is important to note that the resulting difference in performance is not significant.

It is evident from the observations that the merge threshold value does not have a significant effect on the algorithm performance, but a higher value can result in a slightly better performance in some cases such as for sequences containing articulate motions or for corrupted data. If the threshold is too high, the performance can decrease as seen on the KT3DMoSeg dataset. Therefore, a value of $\theta_m = 0.95$ was selected for the final algorithm.

3.6.2.3 Inlier Threshold

The inlier threshold value is used during the RANSAC process of the homogeneity calculation to determine which points within the class are inliers since only the inliers must be used to compute the homogeneity factor. The inlier threshold was set to the following values: $\theta_i = \{5, 2, 1, 0.5\}$. It was decided that the original threshold value of $\theta_i = 5$, used for MSMC [45], should be the maximum value since larger values can lead to decreased performance. A smaller threshold value encourages

Table 3.9. Misclassification metrics of different merge thresholds on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
Average								
0.9	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
0.98	29.0	14.5	24.8	41.2	14.4	34.5	34.8	27.2
0.95	29.3	14.3	24.3	39.4	14.4	34.1	32.2	26.8
0.8	28.8	13.6	24.7	39.8	14.9	50.8	34.6	26.9
Median								
0.9	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
0.98	31.7	6.8	25.0	42.3	15.6	34.5	33.8	30.3
0.95	32.5	9.5	25.0	39.9	15.0	34.1	30.4	29.1
0.8	30.9	6.2	25.0	39.7	15.6	50.8	33.0	29.8
Standard Deviation								
0.9	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
0.98	13.3	16.2	16.4	10.3	7.6	12.9	10.1	16.0
0.95	13.3	16.3	16.0	10.2	7.1	13.4	10.7	15.8
0.8	13.3	16.1	16.3	10.4	7.1	10.2	10.1	16.1

Table 3.10. Misclassification metrics of different merge thresholds on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
0.95	21.6	20.2	12.0
0.98	18.5	16.1	11.0
0.9	22.3	20.1	11.4
0.8	19.1	17.2	10.6

points in close proximity, or with a small distance, to be labelled as class inliers which leads to an improved homogeneity computation. If the threshold becomes too small, true inliers can be detected as outliers which can lead to an incorrect homogeneity computation and over-segmentation can be encouraged. $\theta_i = \{2, 1, 0.5\}$ were chosen to determine how small the threshold value can be before the performance decreases. The performance results on the Hopkins155 and KT3DMoSeg datasets are shown in Tables 3.11 and 3.12.

Considering the results on all the Hopkins155 sequences in Table 3.11, the lower the inlier distance threshold, the better the algorithm performance. The same observation is made for all the subsets of sequences. A different observation was made for the KT3DMoSeg as seen in Table 3.12. The best performance was achieved with $\theta_i = \{5, 2\}$ with the best average, median and standard deviation of the misclassification error. This observation is due to corrupt data as well as the complexity of the data. As the inlier threshold becomes too small at $\theta_i = 1$, true inliers are detected to be outliers and removed, and the homogeneity computation degenerates. Taking the observations of both datasets into consideration, a smaller inlier threshold is preferred but not too small such that the performance degenerates, therefore, the inlier threshold is set to $\theta_i = 2$ for the final MSAM-SSC algorithm.

3.6.2.4 Outlier Threshold

Once all the class inliers have been determined and the homogeneity computed, the homogeneity is used to determine if the outliers must be removed. The outliers are removed if the homogeneity is below the outlier threshold. Using two thresholds, one to determine the inliers and another to remove the outliers, increases the robustness to noisy and corrupted data. The outlier threshold was originally set to $\theta_o = 1$ according to the threshold values used for the MSMC algorithm [45]. Since the homogeneity measure falls in the range $[0, 1]$, it is set to the maximum allowed value and indicates the outliers are removed if the best possible homogeneity is achieved. However, some classes may not achieve the best possible homogeneity, but rather a high homogeneity. In this case, if the outlier threshold is too high, true outliers will not be removed from the class, and the performance deteriorates. Therefore, two low values, $\theta_o = \{0.8, 0.5\}$ were selected. On the other hand, if the threshold is too low, true inliers corrupted by noise can be removed from the class; therefore, values lower than $\theta_o = 0.5$ were not considered. The results on the Hopkins155 and KT3DMoSeg datasets are shown in Tables 3.13 and 3.14, respectively.

From Table 3.13, it can be seen that $\theta_o = 1$ had the best performance for all the Hopkins155 sequences

Table 3.11. Misclassification metrics of different inlier thresholds on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
5	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
2	18.1	3.3	12.9	31.2	7.2	21.8	23.1	16.6
1	11.3	0.3	9.0	23.3	5.2	8.5	12.5	10.6
0.5	3.5	0.2	3.3	18.2	2.3	5.3	9.6	5.4
Median								
5	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
2	18.0	0.0	5.1	31.6	9.4	21.8	20.8	13.3
1	3.1	0.0	0.0	25.8	1.4	8.5	7.3	0.6
0.5	0.0	0.0	0.0	20.5	0.0	5.3	2.7	0.0
Standard Deviation								
5	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
2	15.8	9.7	15.8	10.7	6.1	30.8	16.2	16.3
1	14.5	1.7	14.4	13.0	6.2	12.0	13.9	14.0
0.5	7.1	0.5	5.2	13.9	5.4	7.5	11.8	9.9

Table 3.12. Misclassification metrics of different inlier thresholds on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
5	21.6	20.2	12.0
2	21.3	20.5	11.5
1	26.5	25.4	14.3
0.5	28.3	26.0	13.2

and for all the subsets except for the articulate subset containing three motions. Here, $\theta_o = 0.5$ had the best performance. The subsets containing two motions are simple sequences containing no corruptions;

therefore, the homogeneity measure for each class is higher, and a higher outlier threshold produces more accurate results. As the complexity of the sequences increases with the increase in the number of motions, as with the case of the articulate subset containing three motions, lower homogeneity measures are computed for the classes, and a higher threshold decreases the performance since true inliers are not removed. However, an increase in the complexity does not always result in a decrease in the homogeneity measure, and a higher threshold still produces more accurate results, as seen with the missing data subset. Similar observations are made on the KT3DMoSeg dataset in Table 3.14, where $\theta_o = 1$ had the best performance once again. Therefore, the outlier threshold value, $\theta_o = 1$, is kept for the final algorithm.

Table 3.13. Misclassification metrics of different outlier thresholds on Hopkins155

	Two Motions			**Three Motions**			**Missing**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	**Data** (12)	(171)
1	29.33	14.18	20.74	39.56	13.68	49.81	34.67	26.87
0.8	32.80	18.45	27.33	45.43	21.18	50.48	34.85	30.87
0.5	33.54	24.88	28.30	46.69	29.62	48.48	34.71	32.94
Median								
1	32.06	6.77	23.29	40.03	13.69	49.81	33.06	30.29
0.8	33.64	13.24	29.87	45.67	17.45	50.48	33.68	33.33
0.5	34.58	24.10	29.87	46.15	26.34	48.48	33.25	35.00
Standard Deviation								
1	13.10	16.72	16.79	10.81	7.13	8.76	10.09	16.23
0.8	9.71	15.92	13.87	8.01	9.57	9.70	10.02	14.61
0.5	9.09	15.32	14.57	8.37	13.86	6.87	10.06	13.66

3.6.3 Affinity Measure

The affinity measure is used to construct the affinity matrix that is used by the split operator to split a class into two. As previously discussed, the affinity measure consists of two distance measures, namely the motion distance d_m and the spatial distance between the points in the second frame d_s. The motion distance d_m describes the motion the points underwent between the two frames and is measured

Table 3.14. Misclassification metrics of different outlier thresholds on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
1	21.6	20.2	12.0
0.8	26.1	27.1	12.6
0.5	28.3	30.1	14.2

using the Jacobian distance since it can be used as a similarity measure. Points from the same motion segment will have a similar Jacobian distance. The spatial distance d_s between the points is measured using the standardised Euclidean distance. Points from the same motion segment tend to be in close proximity to each other. The affinity matrix is constructed by taking the product of the Jacobian and standardised Euclidean distances:

$$\mathbf{A} = (1 - d_m) \times \min\{d_s, 1\}. \tag{3.8}$$

The effect on the algorithm performance when using only the motion distance or the spatial distance as the affinity measure was investigated on the Hopkins 155 and KT3DMoSeg datasets. The results are shown in Tables 3.15 and 3.16.

First, consider the performance on the Hopkins155 dataset in Table 3.15. Over the entire dataset, the combined affinity measure d and the d_s affinity measure had similar performance and outperformed the d_m affinity measure. Similar observations were made for both checkerboard and both traffic subsets. For the missing data sequences and the articulate subset with three motions, d_s outperformed d and the d_m. For the articulate subset with two motions, d had the best performance. For the KT3DMoSeg dataset in Table 3.16, the d_s affinity measure had the best performance, followed by d. Since the combined affinity measure d and the d_s affinity measure had similar performance over both datasets, either can be used, therefore, the combined affinity measure d was selected.

3.6.4 Split Operation

If the homogeneity measure for a class is larger than the split threshold θ_s, the class must be divided into two. K-means is the algorithm used for dividing a class into two but this may not be the best method. Three other methods were considered to improve the split operation and the overall performance of

Table 3.15. Misclassification metrics of different distance affinity measures on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board	Traffic	Articulate	Checker-board	Traffic	Articulate		
	(78)	(31)	(11)	(26)	(7)	(2)	(12)	(171)
d	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
d_m	33.4	23.1	27.3	45.1	25.7	51.1	34.6	32.1
d_s	29.4	13.2	25.1	40.0	15.1	34.8	33.0	26.8
Median								
d	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
d_m	33.9	19.9	29.9	45.3	19.4	51.1	33.1	34.5
d_s	31.7	5.8	25.0	39.9	14.5	34.8	30.1	29.9
Standard Deviation								
d	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
d_m	9.9	16.8	14.1	9.2	15.2	10.6	10.2	14.4
d_s	12.9	16.5	16.6	10.3	5.9	12.5	9.6	15.9

Table 3.16. Misclassification metrics of different distance affinity measures on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
d	21.6	20.2	12.0
d_m	26.7	25.7	13.8
d_s	20.4	16.6	14.5

the MSAM-SSC algorithm. First, mean shift clustering was considered. The mean shift algorithm is centroid-based and finds clusters by updating the centroid candidates to the average of a group of points within a specific distance from each other [63]. The MATLAB implementation of the mean shift algorithm was used [64]. Next, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was considered as a possible candidate for the splitting operation. DBSCAN clusters data by finding groups of data which are in close proximity; i.e. find dense areas of points within the feature

space [65]. This is achieved by identifying core points and finding all other points with a specified distance from the core point. Lastly, J-linkage was considered. J-linkage is a type of agglomerative clustering which fits multiple models to the data. Agglomerative clustering starts by assigning each point to its own class, then, during each iteration, the two classes with the smallest Jaccard distance are merged [66].

The changes were tested on the Hopkins155 and KT3DMoSeg datasets and the results shown in Tables 3.17 and 3.18, respectively. The changes were also executed on the KT3DMoSeg dataset; however, the mean shift algorithm had an extremely long execution time. Even though the aim is not to produce a fast, near real-time manifold clustering algorithm, the mean shift algorithm was excluded from consideration since an algorithm that takes more than a day to run on a single short sequence of only 20 frames is not a feasible solution. Therefore, only k-means, DBSCAN and J-linkage were considered.

Consider the results on the Hopkins155 dataset, shown in Table 3.17. K-means had the lowest average and median misclassification error for all the Hopkins155 sequences as well as for all the subsets containing two motions, and the checkerboard and traffic subsets containing three motions. For the articulate subset containing three motions, DBSCAN had a significantly better performance, while all three clustering algorithms had similar performance on the missing data subset. Overall, mean shift had the worst performance. Looking at the results on the KT3DMoSeg dataset in table 3.18, k-means also had the best performance, and was therefore selected as the clustering algorithm to perform the split operation.

3.6.5 Alternate SSC Implementation

Each part of the MSAM algorithm was investigated and optimised. The classic SSC implementation was considered next. The original author of the SSC method created two implementations of the SSC algorithm. The first uses the CVX MATLAB package to solve complex mathematical problems, while the second implementation employs ADMM [61, 62]. The CVX package is used to model convex optimisation problems in MATLAB. CVX uses convex analysis to form a set of rules which is used to build functions from a base library containing sets and functions. These constructed functions are then solved automatically. ADMM is used to solve convex optimisation problems by decomposing the problem into sub-problems that are simpler to solve. It resembles other algorithms such as the method of multipliers, Douglas–Rachford splitting and dual composition [67].

Table 3.17. Misclassification metrics of different split operations on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
k-means	29.33	14.18	20.74	39.56	13.68	49.81	34.67	26.87
DBSCAN	30.00	21.55	27.81	40.92	21.69	35.14	34.78	29.34
J-linkage	33.51	24.20	28.17	45.96	24.77	50.48	34.56	32.50
Meanshift	33.34	25.62	27.83	47.46	28.12	52.81	34.76	33.07
Median								
k-means	32.06	6.77	23.29	40.03	13.69	49.81	33.06	30.29
DBSCAN	33.00	18.43	29.87	40.89	20.07	35.14	33.58	32.23
J-linkage	34.81	21.28	29.87	46.15	17.01	50.48	32.60	35.00
Meanshift	34.04	24.34	29.87	46.57	25.83	52.81	33.68	34.78
Standard Deviation								
k-means	13.10	16.72	16.79	10.81	7.13	8.76	10.09	16.23
DBSCAN	12.67	16.34	14.21	10.72	7.37	11.99	10.02	14.66
J-linkage	9.06	15.82	14.93	8.47	16.13	9.70	10.20	13.96
Meanshift	9.14	15.27	14.21	9.09	12.63	13.00	10.03	13.83

Table 3.18. Misclassification metrics of different split operations on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
k-means	21.59	20.18	12.04
DBSCAN	28.62	30.05	13.81
J-linkage	27.07	28.09	13.66

Both implementations were executed on the Hopkins 155 and the KT3DMoSeg datasets and the results are shown in Tables 3.19 and 3.20, respectively. It was found that the ADMM implementation converged significantly faster than the CVX implementation for both datasets, but performed worse on

both datasets. Considering the performance on the subsets of the Hopkins155 dataset as seen in Table 3.19, the ADMM optimisation had a slightly better performance for the articulate subset with three motions, but the CVX implementation performed significantly better on the rest of the subsets. Since a more accurate performance is preferred over a faster computation time, the CVX implementation is chosen.

Table 3.19. Misclassification metrics of different SSC implementation strategies on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board	Traffic	Articulate	Checker-board	Traffic	Articulate		
	(78)	(31)	(11)	(26)	(7)	(2)	(12)	(171)
CVX	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
ADMM	36.3	27.5	29.7	54.1	41.1	48.3	37.5	36.6
Median								
CVX	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
ADMM	40.1	23.4	33.3	58.7	44.7	48.3	39.2	39.3
Standard Deviation								
CVX	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
ADMM	12.3	13.7	13.1	10.6	13.2	14.5	17.1	15.9

Table 3.20. Misclassification metrics of different SSC implementation strategies on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
CVX	21.6	20.2	12.0
ADMM	37.8	37.4	17.6

3.7 THE FINAL ALGORITHM

Up until this point, only a single part of the MSAM-SSC was changed at a time while the rest were kept constant, therefore, the effect of changing multiple parts, whether it be parameters, thresholds, or sub-algorithms, has not been investigated yet. It is possible that selecting the best values and metrics as discovered when changing only a single aspect of the algorithm does not lead to the best possible

performance. Therefore, three sets of settings were tested on both the Hopkins155 and KT3DMoSeg datasets and compared to the original set. The original set contains the values and sub-algorithms used for the original MSAM-SSC algorithm and consists of:

- Minimum class size and RANSAC minimum sample sizes: $\{\phi_1, \phi_2, \phi_3\} = \{8, 4, 8\}$
- Thresholds: $\theta_s = 0.95$, $\theta_m = 0.9$, $\theta_i = 5$, $\theta_o = 1$
- Affinity measure: $\mathbf{A}_{i,j} = d_{i,j}$ for each point pair i, j
- Split operation: k-means
- SSC implementation: CVX

Set 1 contains the individual best choices for parameters, thresholds, measurements and sub-algorithms as follows:

- Minimum class size: $\{\phi_1, \phi_2, \phi_3\} = \{8, 4, 4\}$
- Thresholds: $\theta_s = 0.98$, $\theta_m = 0.9$, $\theta_i = 5$, $\theta_o = 1$
- Affinity measure: $\mathbf{A}_{i,j} = d_{i,j}$ for each point pair i, j
- Split operation: k-means
- SSC implementation: CVX

Set 2 contained the following:

- Minimum class size: $\{\phi_1, \phi_2, \phi_3\} = \{8, 4, 4\}$
- Thresholds: $\theta_s = 0.95$, $\theta_m = 0.9$, $\theta_i = 2$, $\theta_o = 1$
- Affinity measure: $\mathbf{A}_{i,j} = d_{i,j}$ for each point pair i, j
- Split operation: k-means
- SSC implementation: CVX

Lastly, set 3 consisted of

- Minimum class size: $\{\phi_1, \phi_2, \phi_3\} = \{8, 4, 4\}$
- Thresholds: $\theta_s = 0.95$, $\theta_m = 0.95$, $\theta_i = 2$, $\theta_o = 1$

- Affinity measure: $\mathbf{A}_{i,j} = d_{i,j}$ for each point pair i, j
- Split operation: k-means
- SSC implementation: CVX

Only the split, merge and inlier thresholds changed for each set. This is since it was observed that MSAM-SSC is the least sensitive to these values. Therefore, it is possible that the algorithm can have increased performance for different combinations of these three thresholds. For the rest of the parameters, thresholds and sub-algorithms, there were clear best choices, therefore, there was no need to vary these. The performance of these three sets of changes are compared to that of the original MSAM-SSC algorithm and the metrics are summarised in Tables 3.21 and 3.22 for the Hopkins155 and KT3DMoSeg datasets, respectively.

From the results on the Hopkins155 dataset in Table 3.21, it can be seen that all three sets significantly improved on the original MSAM-SSC algorithm. Set 1 had the best performance with the lowest average, median and standard deviation for the misclassification error. Similar observations are made for all the subsets of sequences except for the traffic subset containing two motions where Set 2 resulted in the best performance. For the KT3DMoSeg dataset, Set 1 also had the best performance. Therefore, the final MSAM-SSC algorithm is set to include all the parameters, thresholds, distance measures and sub-algorithms as stipulated in Set 1. Considering the performance on the KT3DMoSeg dataset given in Table to 3.22, Set 1 had the best performance. All three sets also performed better than the original MSAM-SSC algorithm. Therefore, Set 1 is selected as the final optimisation.

3.8 CONCLUSION

As a first step, the performance of current manifold clustering methods on the Hopkins155 and KT3DMoSeg datasets are compared and a base algorithm was selected. The base algorithm had to have similar performance on both datasets since this indicated that the algorithm generalised better. Therefore, SSC was selected. Next, a method to handle large and complete occlusions, as well as missing data, was derived. Several possible methods were investigated and a frame-to-frame approach, named MSAM, was selected. MSAM was used as a pre-processing step to identify inter-frame motion regions between frame pairs. The inter-frame motion segments extracted by MSAM were combined to form the input for SSC. The resulting algorithm was called the MSAM-SSC algorithm. It was found that MSAM-SSC is a viable solution to the motion segmentation problem since it performed better on both datasets than SSC and MSMC from which it was derived. Once the performance of MSAM-SSC

Table 3.21. Comparison of MSAM-SSC with different parameter sets on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
Average								
Original	29.3	14.2	20.7	39.6	13.7	49.8	34.7	26.9
Set 1	16.31	2.25	12.30	30.72	4.83	20.21	15.86	14.86
Set 2	22.14	5.25	16.31	32.95	7.45	32.21	26.02	19.62
Set 3	18.85	1.92	13.27	32.05	6.92	21.81	25.70	17.02
Median								
Original	32.1	6.8	23.3	40.0	13.7	49.8	33.1	30.3
Set 1	13.38	0.00	0.00	31.98	0.34	20.21	10.42	9.05
Set 2	26.37	0.00	15.15	33.15	9.43	32.21	26.19	20.55
Set 3	19.74	0.00	5.12	31.65	9.43	21.81	26.64	14.55
Standard Deviation								
Original	13.1	16.7	16.8	10.8	7.1	8.8	10.1	16.2
Set 1	15.38	7.70	16.06	11.08	6.17	28.59	16.76	15.93
Set 2	15.75	12.51	17.87	11.78	5.48	11.61	16.82	16.96
Set 3	15.66	6.59	16.05	9.55	5.84	30.84	16.20	16.32

Table 3.22. Comparison of MSAM-SSC with different parameter sets on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
Original	21.6	20.2	12.0
Set 1	17.92	14.43	11.50
Set 2	18.96	15.60	10.92
Set 3	19.31	15.25	11.19

was verified, the algorithm was then investigated further to optimise and improve the performance. The parameters, thresholds and sub-algorithms of the MSAM algorithm were changed until a set was found which further reduced the misclassification error to improve the performance.

CHAPTER 4 RESULTS

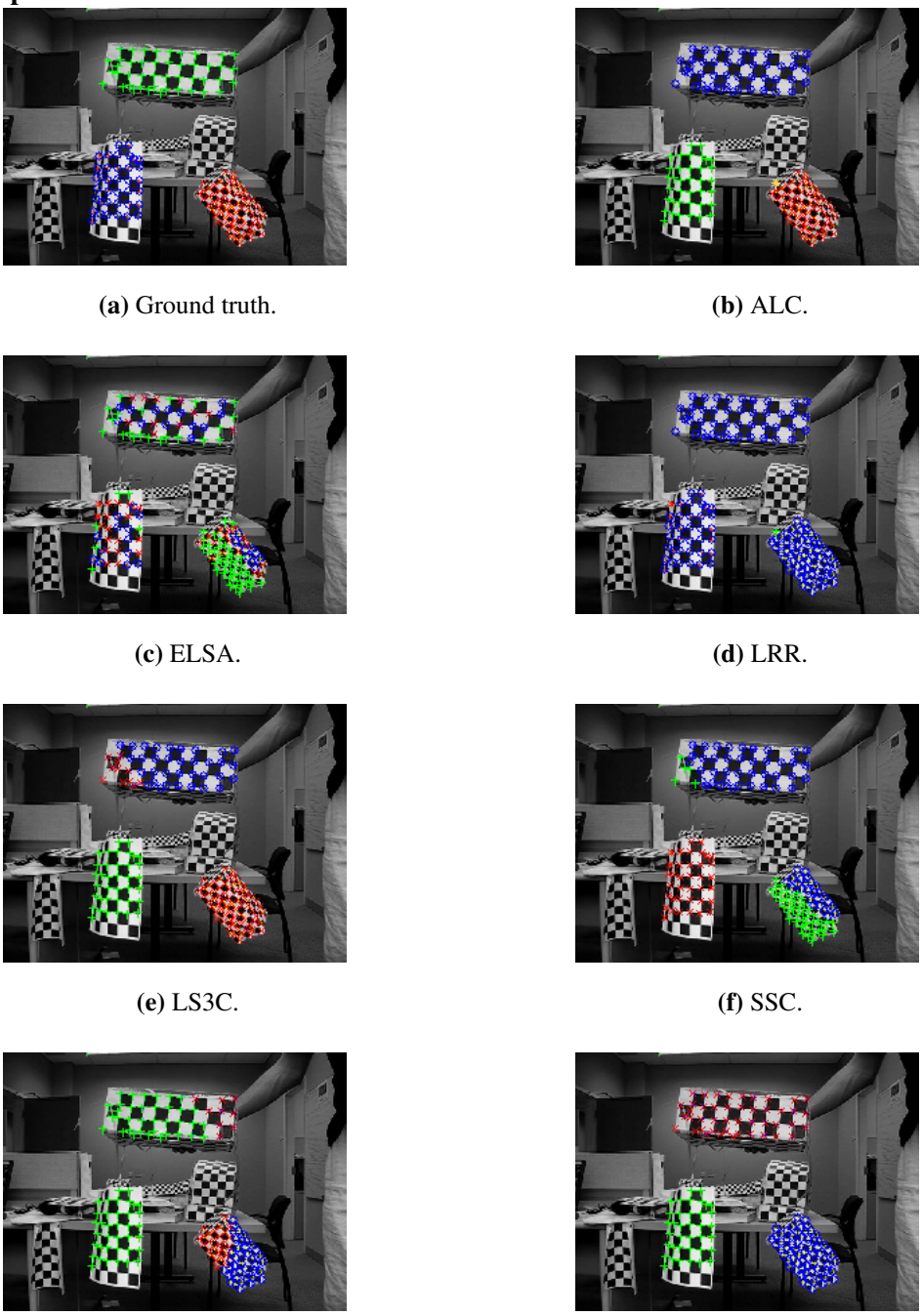
4.1 CHAPTER OVERVIEW

In Section 4.2, visual results of the ALC, ELSA, LRR, LS3C, SSC, MSMC and MSAM-SSC algorithms are given next to the ground truth segmentation. In the remaining sections, the performance metrics of these algorithms are presented to evaluate the performance for different challenges motion segmentation faces. These metrics consist of tables showing the average, median and standard deviation of the misclassification error for a set of sequences, as well as graphs illustrating the histogram distribution of the misclassification error. In Section 4.3, the performance metrics on the Hopkins155 and KT3DMoSeg datasets are given. Additionally, the average, median and standard deviation of the execution times are given to form a better understanding of the trade-off between accuracy and speed. In Section 4.4, the results on synthetic complete occlusion sequences are given while Section 4.5 contains the results on synthetic missing data sequences of which 50% of the original data is missing. Section 4.6 gives the results on different motion types, namely rigid, non-rigid, articulate and degenerate motions. In Section 4.7 the results on sequences with different numbers of motions, namely 2, 3, 4 or 5 motions are given. In Section 4.8, the results on different types of camera motion, namely rotation, small translation, rotation with small translation, handheld and large translation, are presented. Lastly, in Section 4.9, the performance of the algorithms are evaluated on sequences containing synthetic outliers.

4.2 QUALITATIVE EVALUATION

Before the performance of MSAM-SSC is evaluated, visual representations of the final output are presented. The ground truth and segmentation results of ALC, ELSA, LRR, LS3C, SSC, MSMC and MSAM-SSC are shown for a few sequences from the Hopkins155 and KT3DMoSeg datasets.

4.2.1 Hopkins155



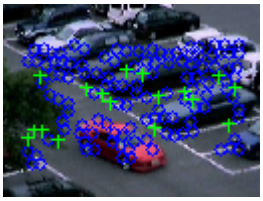
(a) Ground truth. **(b)** ALC. **(c)** ELSA. **(d)** LRR. **(e)** LS3C. **(f)** SSC. **(g)** MSMC. **(h)** MSAM-SSC.

Figure 4.1. Final motion segments of *articulated* sequence. (a) Ground truth. (b) ALC. (c) ELSA. (d) LRR. (e) LS3C. (f) SSC. (g) MSMC. (h) MSAM-SSC.

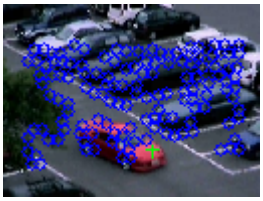
Adapted from [48, 49, 50], ©[2004-2012] Vision Lab

(a) Ground truth.

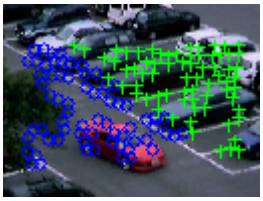
(b) ALC.



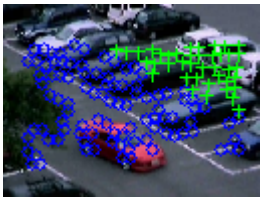
(c) ELSA.



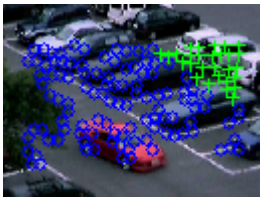
(d) LRR.



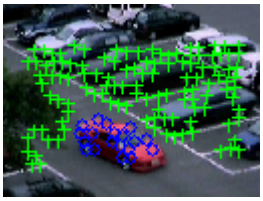
(e) LS3C.



(f) SSC.



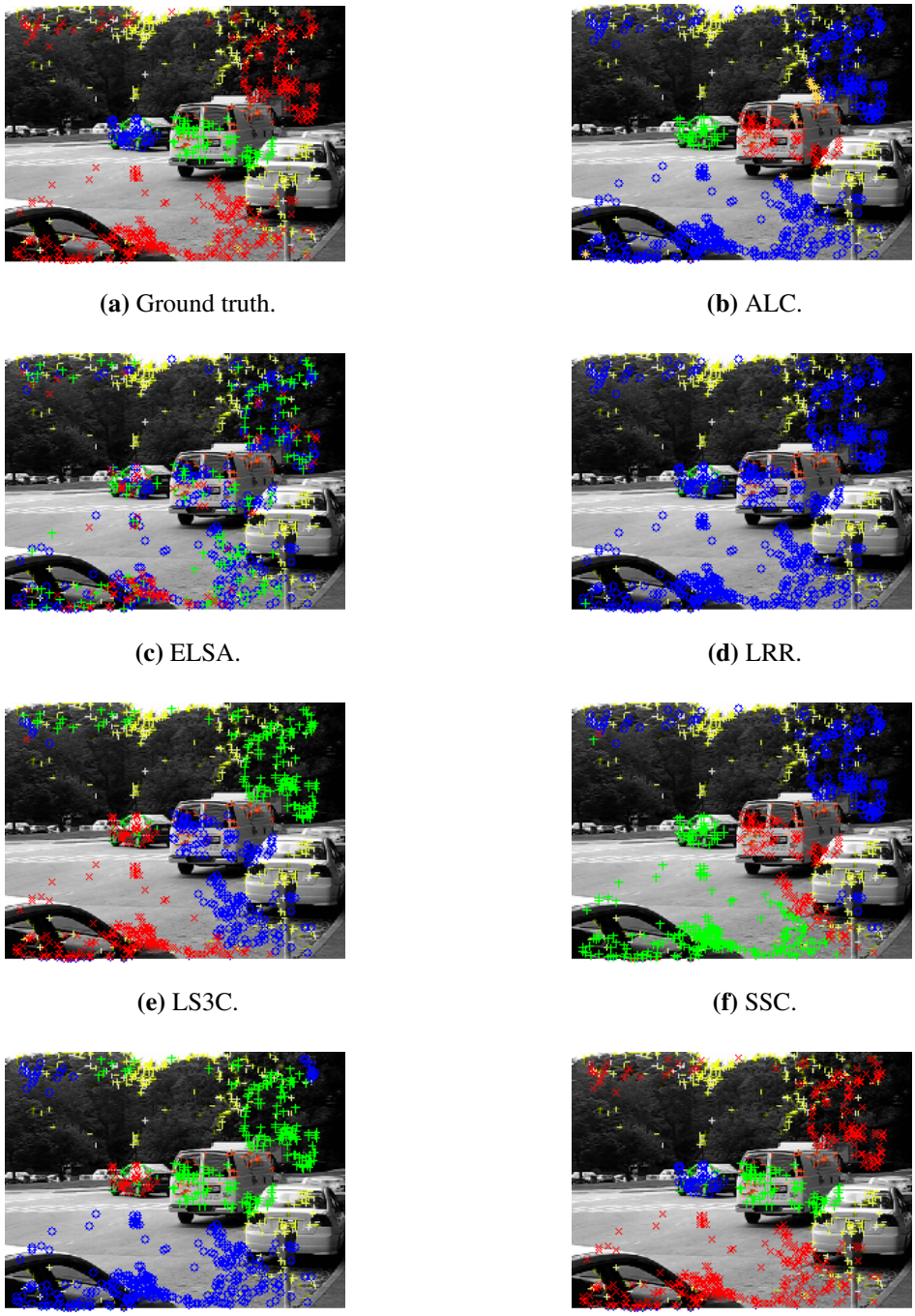
(g) MSMC.



(h) MSAM-SSC.

Figure 4.2. Final motion segments of *kanatani1* sequence. (a) Ground truth. (b) ALC. (c) ELSA. (d) LRR. (e) LS3C. (f) SSC. (g) MSMC. (h) MSAM-SSC.

Adapted from [48, 49, 50], ©[2004-2012] Vision Lab

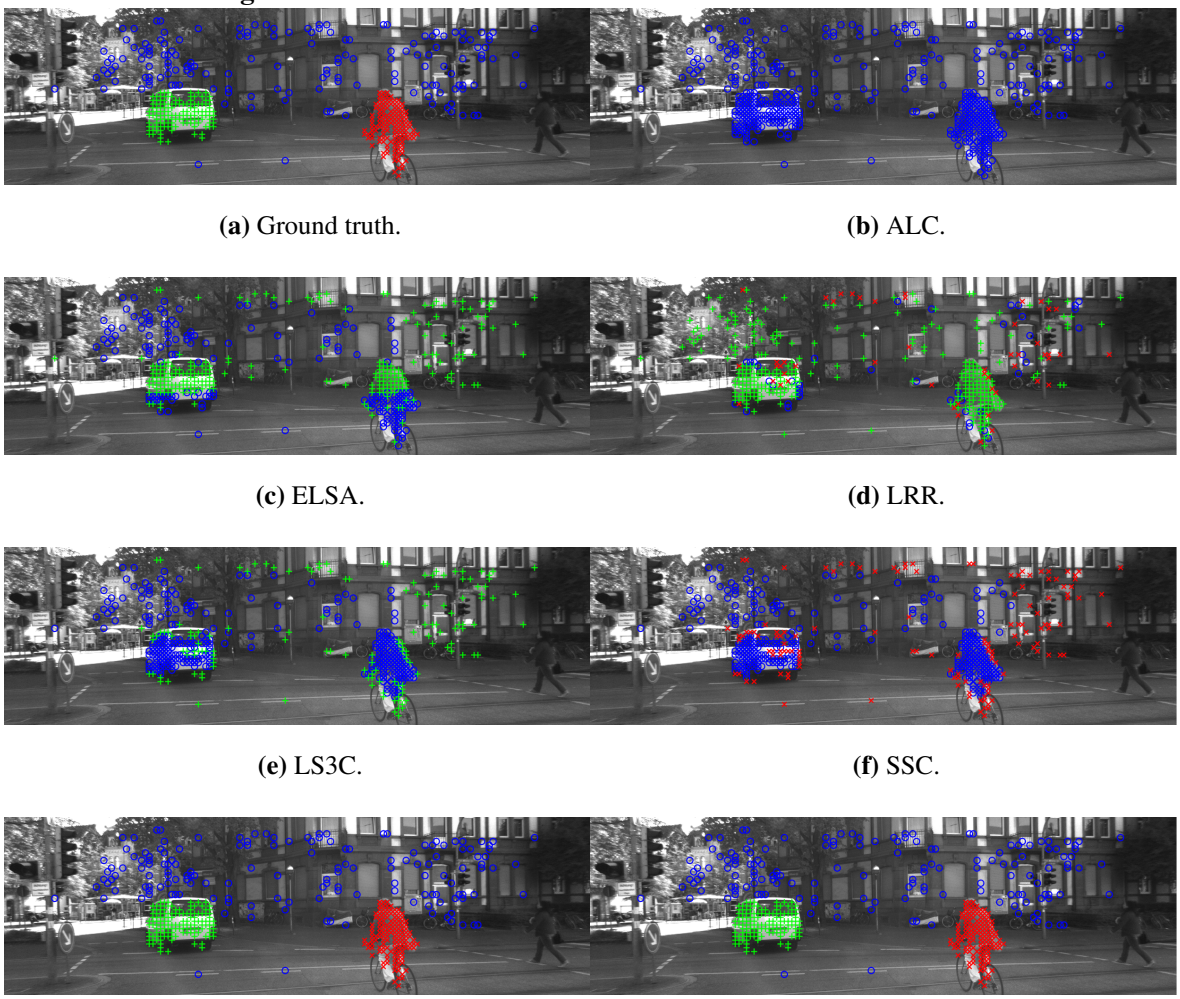


(a) Ground truth. **(b)** ALC. **(c)** ELSA. **(d)** LRR. **(e)** LS3C. **(f)** SSC. **(g)** MSMC. **(h)** MSAM-SSC.

Figure 4.3. Final motion segments of *cars3* sequence. (a) Ground truth. (b) ALC. (c) ELSA. (d) LRR. (e) LS3C. (f) SSC. (g) MSMC. (h) MSAM-SSC.

Adapted from [48, 49, 50], ©[2004-2012] Vision Lab

4.2.2 KT3DMoSeg



(a) Ground truth. **(b)** ALC.

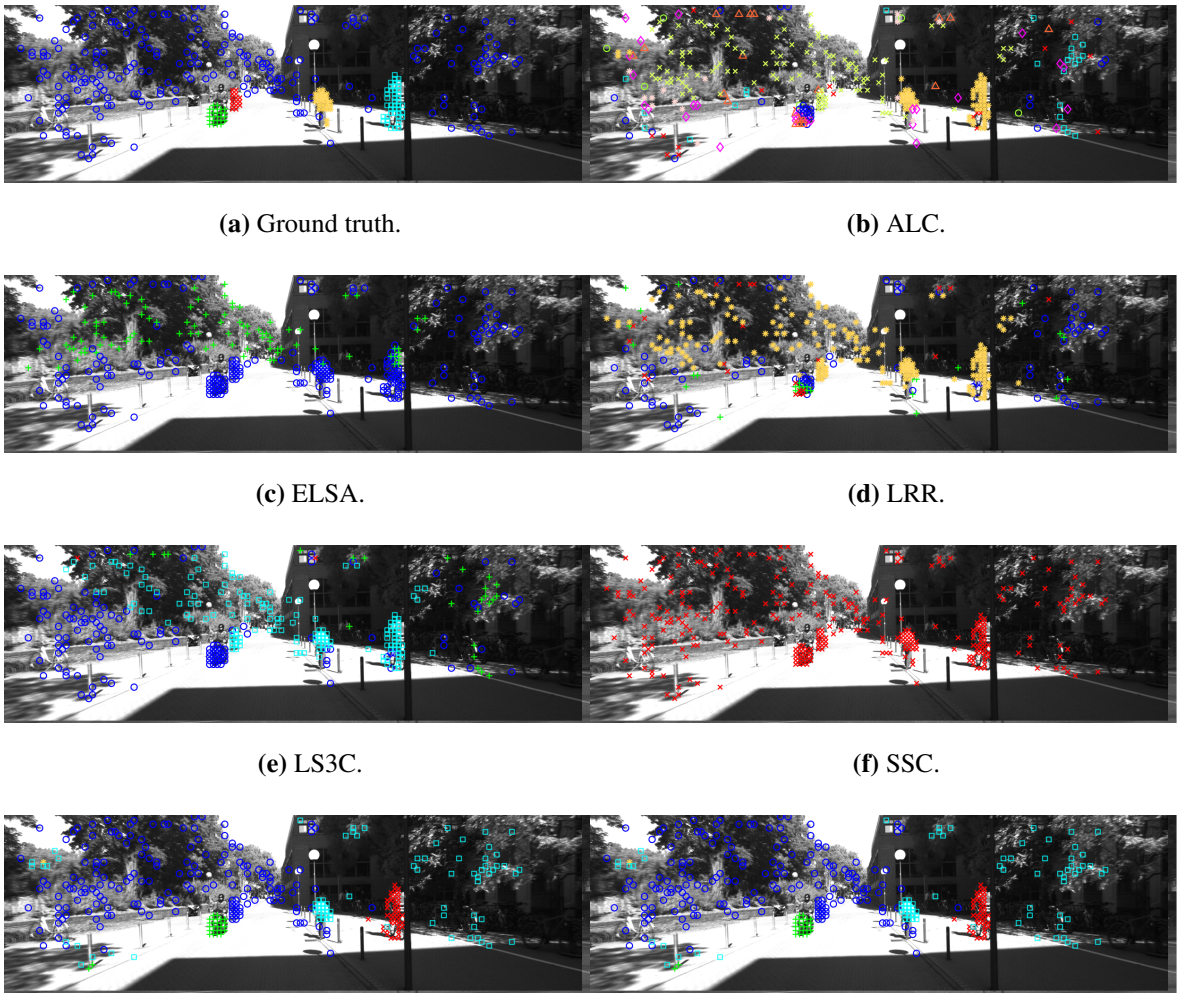
(c) ELSA. **(d)** LRR.

(e) LS3C. **(f)** SSC.

(g) MSMC. **(h)** MSAM-SSC.

Figure 4.4. Final motion segments of *Seq005_Clip01* sequence. (a) Ground truth. (b) ALC. (c) ELSA. (d) LRR. (e) LS3C. (f) SSC. (g) MSMC. (h) MSAM-SSC.

Adapted from [53], ©2018 IEEE



(a) Ground truth. **(b)** ALC.

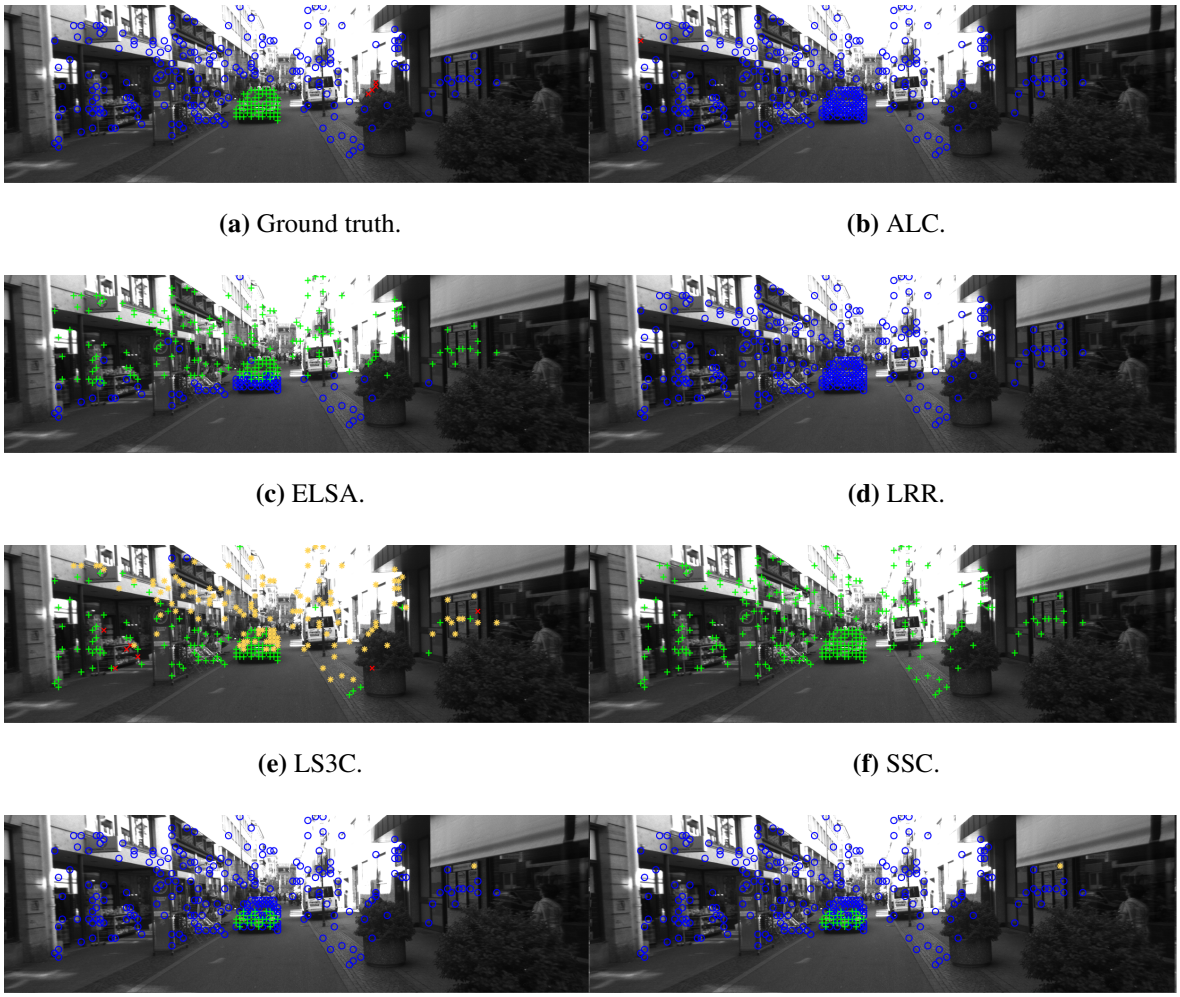
(c) ELSA. **(d)** LRR.

(e) LS3C. **(f)** SSC.

(g) MSMC. **(h)** MSAM-SSC.

Figure 4.5. Final motion segments of *Seq038_Clip01* sequence. (a) Ground truth. (b) ALC. (c) ELSA. (d) LRR. (e) LS3C. (f) SSC. (g) MSMC. (h) MSAM-SSC.

Adapted from [53], ©2018 IEEE



(a) Ground truth. **(b)** ALC.

(c) ELSA. **(d)** LRR.

(e) LS3C. **(f)** SSC.

(g) MSMC. **(h)** MSAM-SSC.

Figure 4.6. Final motion segments of *Seq071_Clip01* sequence. (a) Ground truth. (b) ALC. (c) ELSA. (d) LRR. (e) LS3C. (f) SSC. (g) MSMC. (h) MSAM-SSC.

Adapted from [53], ©2018 IEEE

4.3 COMPARISON OF MANIFOLD CLUSTERING ALGORITHMS

The MSAM-SSC algorithm was verified to be a viable solution to the motion segmentation problem by comparing the performance to that of existing manifold clustering methods. The performance of ALC, ELSA, LRR, LS3C, SSC and MSMC were used to benchmark the performance of MSAM-SSC. The algorithms were executed on the Hopkins155 and KT3DMoSeg datasets. Here, not only the misclassification error was considered, but also the execution time to draw a better conclusion on the

performance of the MSAM-SSC algorithm. Even though the focus was not to produce an algorithm with a short execution time while maintaining high accuracy, the execution time is an important factor to consider when evaluating algorithm performance. An algorithm that has a low misclassification and execution time is desired, but it is important to determine if the algorithm is compromising execution time to achieve higher accuracy since in many cases there is a trade-off between these two factors.

4.3.1 Hopkins155 Dataset

The average, median and standard deviation of the misclassification error on the Hopkins155 dataset for all the manifold clustering algorithms under consideration is shown in Table 4.1. Figure 4.7 shows the histogram distribution of the misclassification per sequence. Table 4.2 shows the average, median and standard deviation of the execution times (in seconds) for each of the algorithms.

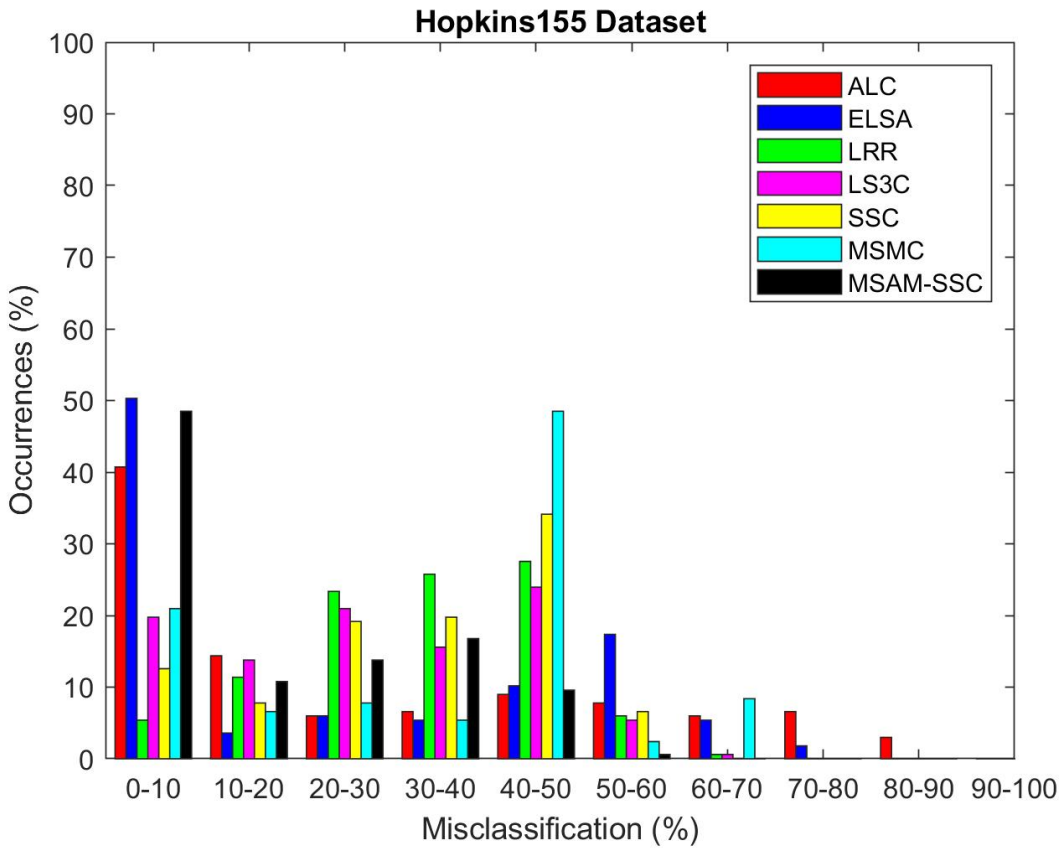


Figure 4.7. Percentage of occurrences of the misclassification error on Hopkins155

Table 4.1. Misclassification metrics of all algorithms on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
Average								
ALC	34.44	30.13	8.87	6.78	6.03	7.25	37.78	25.75
ELSA	10.69	22.37	18.70	54.20	46.11	51.00	23.95	22.54
LRR	32.91	25.21	27.38	42.43	29.17	47.55	34.82	31.99
LS3C	25.51	27.52	18.73	39.62	35.83	15.90	25.61	27.31
SSC	30.92	27.29	19.27	43.47	36.75	31.81	39.10	31.52
MSMC	37.20	19.82	33.94	44.63	24.43	52.38	33.67	33.50
MSAM-SSC	16.74	3.54	11.85	32.05	6.81	21.81	16.28	15.59
Median								
ALC	31.97	19.58	0.95	0.92	1.35	7.25	39.43	15.89
ELSA	0.53	16.67	12.82	54.04	52.52	51.00	13.42	5.56
LRR	33.47	23.81	27.27	42.67	21.23	47.55	33.36	33.60
LS3C	25.97	32.27	10.96	40.77	35.55	15.90	24.39	27.93
SSC	32.72	36.34	20.51	44.89	45.07	31.81	44.35	34.85
MSMC	46.58	11.95	41.03	60.07	21.90	52.38	40.82	43.84
MSAM-SSC	14.67	0.00	0.00	31.41	9.43	21.81	10.22	10.10
Standard Deviation								
ALC	26.11	29.17	14.90	10.78	11.17	9.31	30.88	26.70
ELSA	19.63	21.83	19.67	9.26	11.54	1.41	26.35	24.63
LRR	10.04	14.07	14.57	11.55	13.50	10.07	10.28	13.46
LS3C	15.37	17.00	16.83	12.58	20.39	12.12	19.23	17.00
SSC	13.21	18.62	16.77	11.20	14.78	16.70	16.69	16.31
MSMC	17.52	21.32	16.20	22.51	17.45	14.68	17.80	20.86
MSAM-SSC	16.16	9.80	16.04	9.59	5.78	30.84	16.24	16.33

Table 4.2. Execution times (s) of all manifold clustering algorithms on Hopkins155

	Two Motions			**Three Motions**			**Missing Data**	**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(12)	(171)
ALC	103.89	104.52	74.58	224.66	151.67	7.57	228.32	92.37
ELSA	4.30	3.08	2.12	8.46	5.16	0.69	7.78	3.25
LRR	0.57	0.48	0.37	0.66	0.59	0.44	0.70	0.36
LS3C	4.62	3.57	2.02	10.37	5.91	0.83	10.19	3.86
SSC	85.12	68.50	49.64	138.79	103.15	26.80	155.74	62.13
MSMC	1.59	4.03	2.58	3.85	6.24	0.76	3.26	2.16
MSAM-SSC	69.36	60.13	40.72	131.04	95.90	28.44	160.57	57.08
Median								
ALC	88.48	19.62	2.93	239.27	50.54	7.57	212.46	9.38
ELSA	4.01	1.39	0.32	8.17	3.36	0.69	7.62	0.78
LRR	0.60	0.48	0.30	0.64	0.58	0.44	0.64	0.40
LS3C	2.68	1.44	0.44	11.36	2.77	0.83	9.28	0.87
SSC	82.37	44.97	20.05	139.27	89.72	26.80	138.44	29.81
MSMC	1.47	3.02	0.69	3.01	6.60	0.76	3.15	0.99
MSAM-SSC	74.13	47.91	19.62	130.48	96.85	28.44	144.20	32.94
Standard Deviation								
ALC	87.37	144.41	158.76	116.63	168.05	4.64	174.58	135.54
ELSA	2.99	3.59	4.05	3.60	4.73	0.42	4.69	4.27
LRR	0.19	0.22	0.17	0.13	0.28	0.04	0.14	0.32
LS3C	4.52	4.71	3.58	5.08	5.79	0.38	6.76	5.50
SSC	44.49	53.10	72.12	44.89	59.12	10.33	79.01	70.27
MSMC	0.93	3.92	3.95	2.34	3.45	0.14	1.53	2.95
MSAM-SSC	30.80	41.29	49.45	38.63	51.10	13.99	88.05	65.23

4.3.2 KT3DMoSeg Dataset

The average, median and standard deviation of the misclassification error on the KT3DMoSeg dataset for all the manifold clustering algorithms under consideration is shown in Table 4.3. Figure 4.8 shows the histogram distribution of the misclassification per sequence. Table 4.4 shows the average, median and standard deviation of the execution times (in seconds) for each of the algorithms.

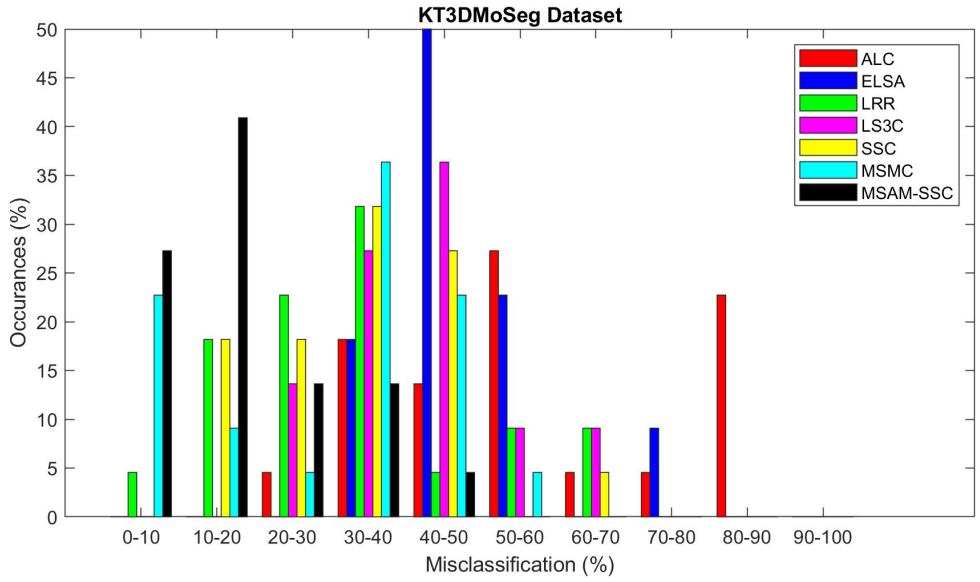


Figure 4.8. Percentage of occurrences of the misclassification error on KT3DMoSeg

Table 4.3. Misclassification metrics of all algorithms on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
ALC	58.52	53.10	21.24
ELSA	48.86	48.23	11.25
LRR	32.02	31.33	15.33
LS3C	44.96	43.62	16.65
SSC	34.02	34.79	12.32
MSMC	29.30	34.58	16.79
MSAM-SSC	17.92	14.43	11.50

Table 4.4. Execution times (s) of all manifold clustering algorithms on KT3DMoSeg

	All (22)		
	Average	**Median**	**Standard Deviation**
ALC	245.59	177.68	288.36
ELSA	18.60	17.01	15.12
LRR	0.65	0.69	0.24
LS3C	14.58	13.08	14.14
SSC	147.20	134.04	97.39
MSMC	12.22	10.23	7.38
MSAM-SSC	9.65	9.65	0.00

4.4 OCCLUSIONS

Even though some of the sequences in the Hopkins155 and KT3DMoSeg datasets contain small partial occlusions, none of the sequences contain large or complete occlusions. To date, and to the best of the author's knowledge, there are no comprehensive datasets available that can be used specifically to test feature-based algorithms on large and complete occlusions. Therefore, sequences containing more than one complete occlusion of a moving object were created using the Hopkins155 and KT3DMoSeg datasets. The occlusions were created by removing the point locations of each object for 5 consecutive frames. Therefore, the object completely disappears for 5 frames before it reappears. In practice, objects generally will not disappear instantaneously, but points will gradually disappear then reappear. Only objects which are located at a great distance can disappear instantaneously. However, this is an extreme example of occlusions, therefore the algorithms were tested on these cases. Table 4.5 shows the results of the algorithms on the extreme occlusion scenario. For the Hopkins155 dataset, the names of the subsets are represented as follows: C for Checkerboard, T for Traffic, and A for Articulate. Figure 4.9 shows the histogram distribution of the misclassification per sequence. Note that only the results for MSMC and MSAM-SSC are shown since ALC, ELSA, LRR, LS3C and SSC all failed to segment the extreme occlusion scenario sequences.

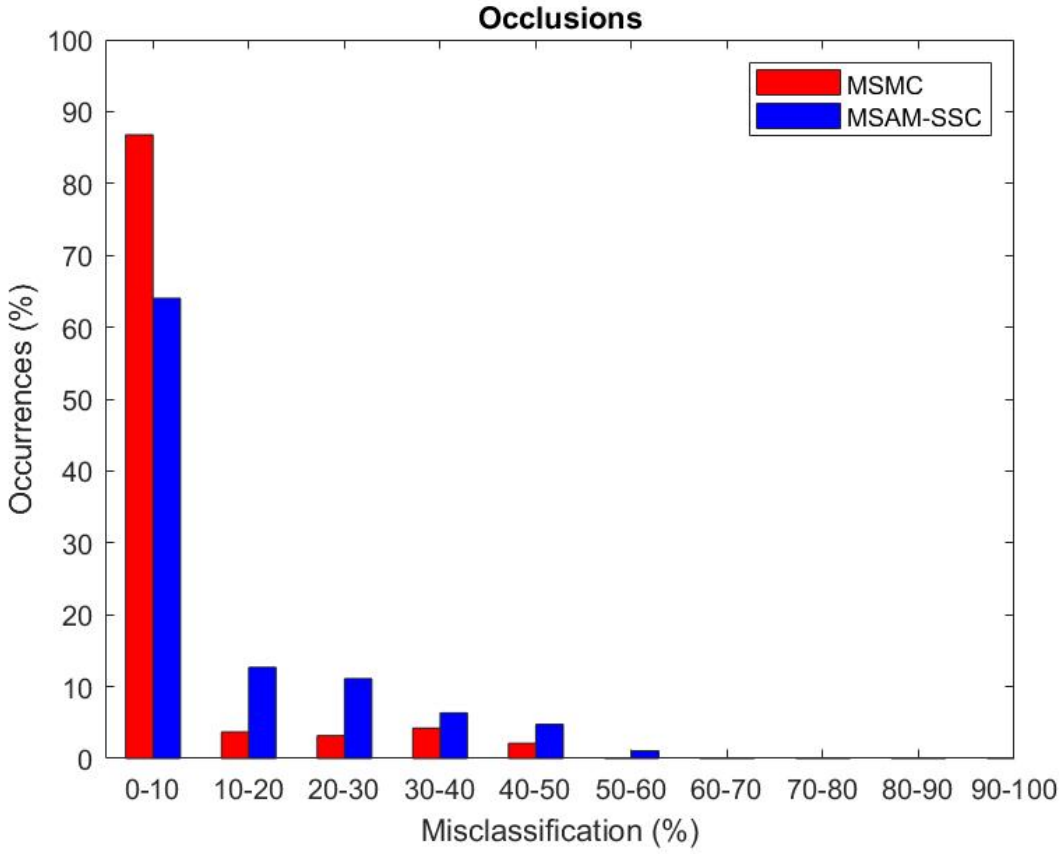


Figure 4.9. Percentages of occurrences of the misclassification error on data with generated occlusions

Table 4.5. Misclassification metrics of manifold clustering algorithms on data with generated occlusions

	Hopkins155							**KT3DMoSeg**	**All**
	Two Motions			**Three Motions**			**Missing Data**		
	C (78)	T (31)	A (11)	C (26)	T (7)	A (2)	(12)	(22)	(189)
Average									
MSMC	0.01	2.36	0.00	0.51	9.70	0.00	0.10	26.98	3.88
MSAM-SSC	3.81	1.43	1.19	25.82	6.75	12.23	10.71	29.42	9.71
Median									
MSMC	0.00	0.00	0.00	0.00	0.00	0.00	0.00	26.23	0.00
MSAM-SSC	0.00	0.00	0.00	27.80	9.43	12.23	2.76	30.80	0.00
Standard Deviation									
MSMC	0.07	9.22	0.00	2.24	15.41	0.00	0.19	12.30	10.51
MSAM-SSC	8.19	3.60	3.96	12.77	6.04	17.30	14.16	13.53	14.02

4.5 MISSING DATA

In real-world applications, the data sequences can have missing entries caused by noise, camera motion or changes in the illumination, and it is, therefore, important to evaluate if the MSAM-SSC algorithm can handle such cases. The Hopkins155 dataset contains an additional 16 sequences with missing data, and the KT3DMoSeg dataset is also corrupted with missing entries and noise. The performance on these sequences has already been provided. However, to further test the ability to tolerate missing data, new missing sequences were generated. These sequences were generated by randomly removing half of the point locations throughout the video sequence. This was done for both the Hopkins155 and KT3DMoSeg datasets. As with the extreme occlusion case, ALC, ELSA, LRR, LS3C and SSC all failed to segment any of the adapted missing data sequences. The results of MSMC and MSAM-SSC are shown in Table 4.6. For the Hopkins155 dataset, the names of the subsets are represented as follows: C for Checkerboard, T for Traffic, and A for Articulate. Figure 4.10 shows the histogram distribution of the misclassification per sequence.

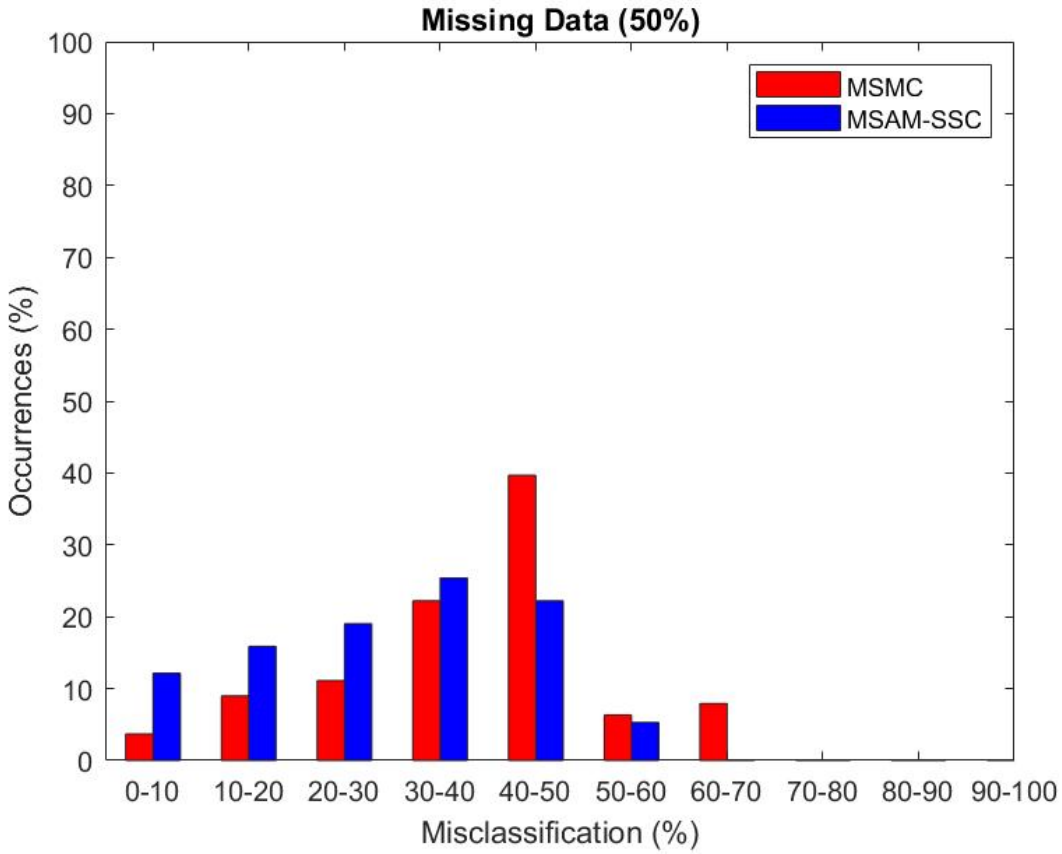


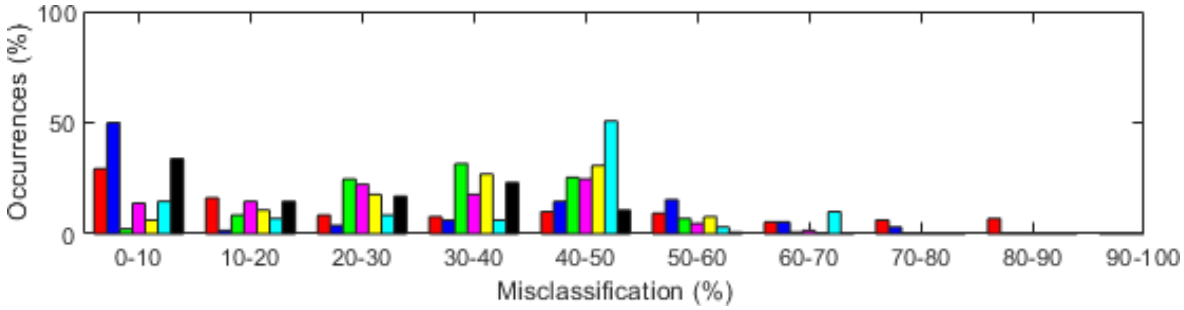
Figure 4.10. Percentages of occurrences of the misclassification error on generated missing data sequences

Table 4.6. Misclassification metrics of manifold clustering algorithms on generated missing data sequences

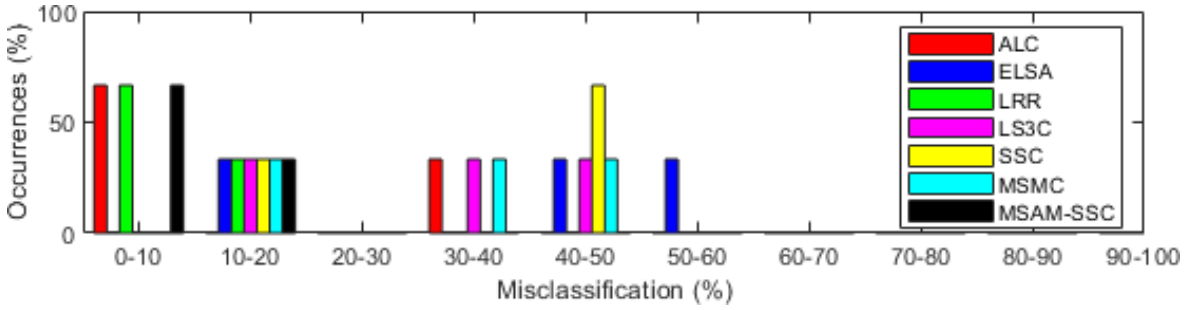
	Hopkins155							**KT3DMoSeg**	**All**
	Two Motions			**Three Motions**			**Missing Data**		
	C (78)	T (31)	A (11)	C (26)	T (7)	A (2)	(12)	(22)	(189)
Average									
MSMC	38.02	29.24	32.76	52.71	39.61	57.26	43.15	39.08	38.20
MSAM-SSC	29.40	17.05	21.86	43.68	22.59	34.48	34.79	31.56	28.69
Median									
MSMC	40.66	34.23	38.94	54.76	39.06	57.26	47.46	34.67	41.67
MSAM-SSC	31.42	11.92	23.29	43.89	20.62	34.48	33.45	32.15	30.70
Standard Deviation									
MSMC	9.76	16.13	15.57	10.23	17.78	1.78	18.00	11.78	15.09
MSAM-SSC	12.97	15.40	16.12	8.89	11.24	12.93	10.34	12.63	15.30

4.6 MOTION TYPES

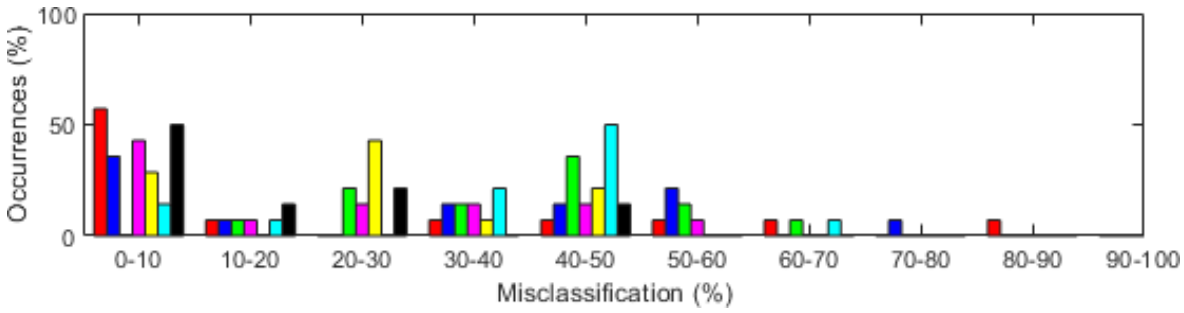
Scenes often contain different types of motion, and often a mixture of motion types. Algorithms must be able to handle these different motions as well as mixtures of motions. The Hopkins155 and KT3DMoSeg datasets contain a variety of different motions. The misclassification error metrics on the different types of motions, namely rigid, non-rigid, articulate and degenerate, are given in Table 4.7, The histogram distribution of the misclassification error for each motion type is illustrated in Figure 4.11.



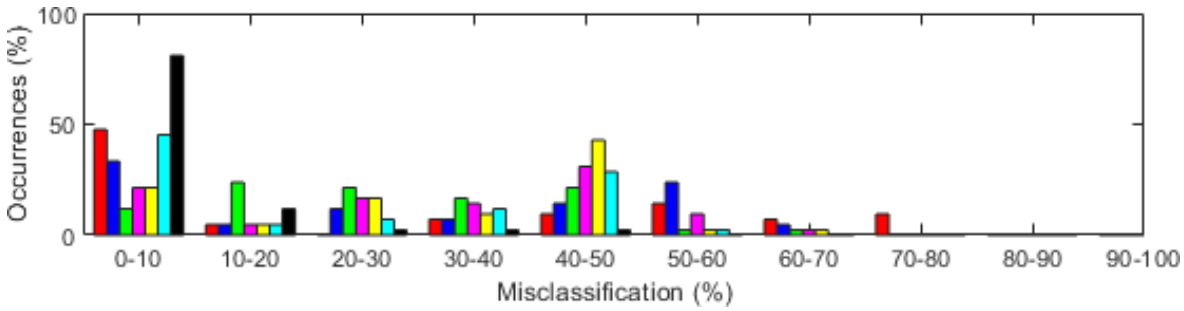
(a) Rigid.



(b) Non-rigid.



(c) Articulate.



(d) Degenerate.

Figure 4.11. Percentages of occurrences of the misclassification error on different types of motions. (a) Rigid. (b) Non-rigid. (c) Articulate. (d) Degenerate.

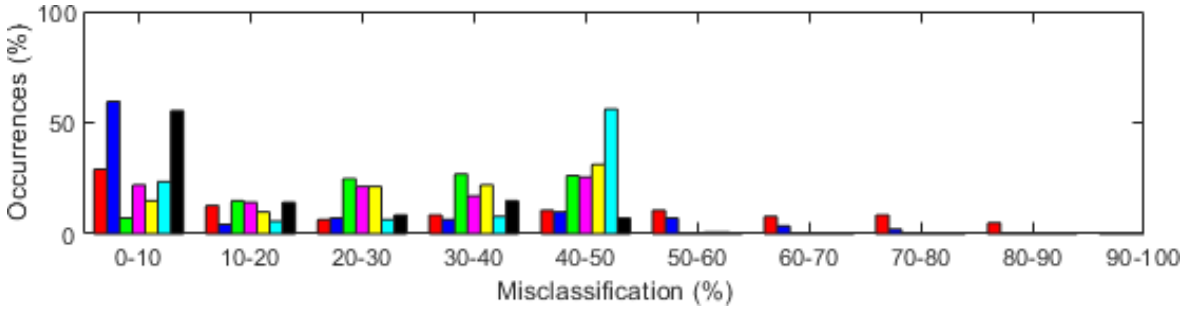
Table 4.7. Misclassification metrics of manifold clustering algorithms on different types of motions

	Motion Type				All
	Rigid (130)	Non-rigid (3)	Articulate (14)	Degenerate (42)	(189)
Average					
ALC	32.02	16.53	21.23	28.15	30.11
ELSA	24.43	38.58	29.73	29.07	26.08
LRR	34.33	8.81	37.87	27.51	32.67
LS3C	30.29	34.88	21.61	31.28	29.94
SSC	34.19	35.35	22.00	30.45	32.48
MSMC	37.77	32.52	35.37	20.74	33.72
MSAM-SSC	20.08	3.37	15.22	5.39	16.19
Median					
ALC	24.49	9.52	1.77	19.15	21.72
ELSA	10.72	43.45	34.76	29.50	26.06
LRR	34.21	6.15	40.57	26.10	33.46
LS3C	29.82	37.37	17.38	35.42	30.00
SSC	36.46	44.44	21.13	38.59	36.00
MSMC	46.14	34.75	40.87	16.43	43.45
MSAM-SSC	21.95	0.00	10.88	0.00	12.75
Standard Deviation					
ALC	28.15	13.40	29.22	27.98	28.06
ELSA	25.69	17.47	26.15	22.42	24.91
LRR	11.37	5.52	13.74	14.65	12.96
LS3C	17.14	15.34	18.33	18.24	17.49
SSC	13.85	18.12	15.73	18.27	15.37
MSMC	18.77	14.36	16.91	20.33	20.08
MSAM-SSC	15.78	5.83	16.71	9.83	15.84

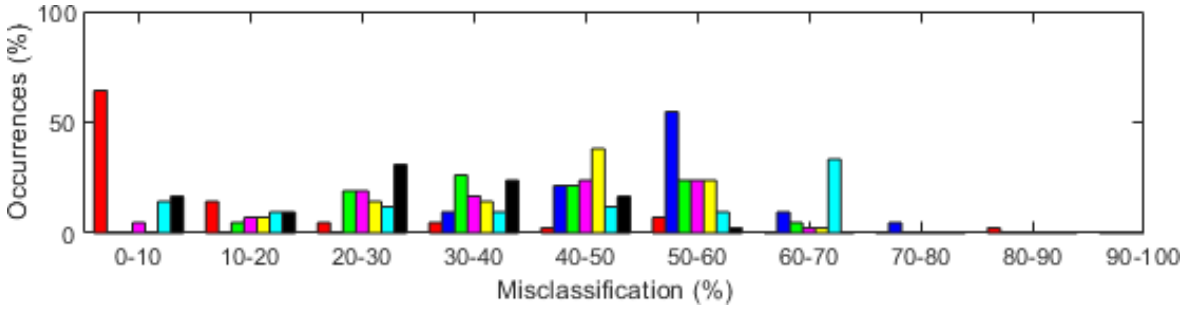
4.7 MULTIPLE MOTIONS

In most cases, the scene contains multiple objects in motion, and the algorithm must be able to segment these objects. In some cases, the background points have a motion component as well, and

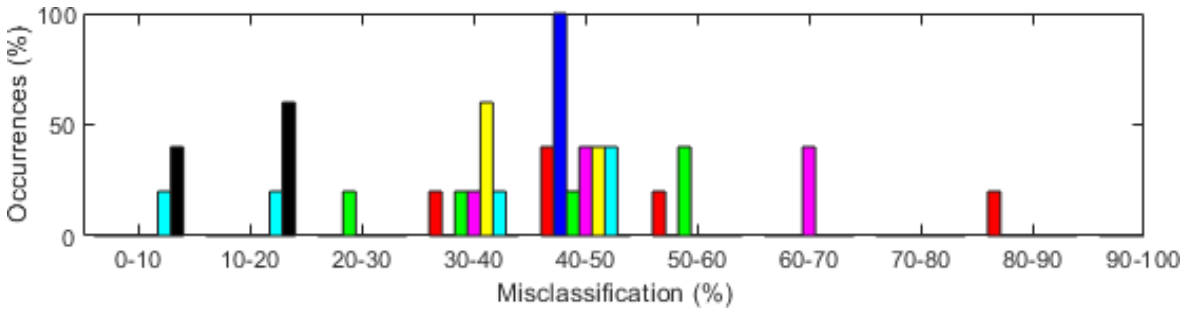
the algorithm must be able to group these points. The sequences from the Hopkins155 dataset contain either two or three motions, while the number of moving objects in the KT3DMoSeg dataset range from 1 to 4, with a moving background. Table 4.8 shows the results of the manifold clustering algorithms on the sequences with different numbers of motion. Figure 4.12 shows the histogram distribution of the misclassification errors for the different numbers of motion.



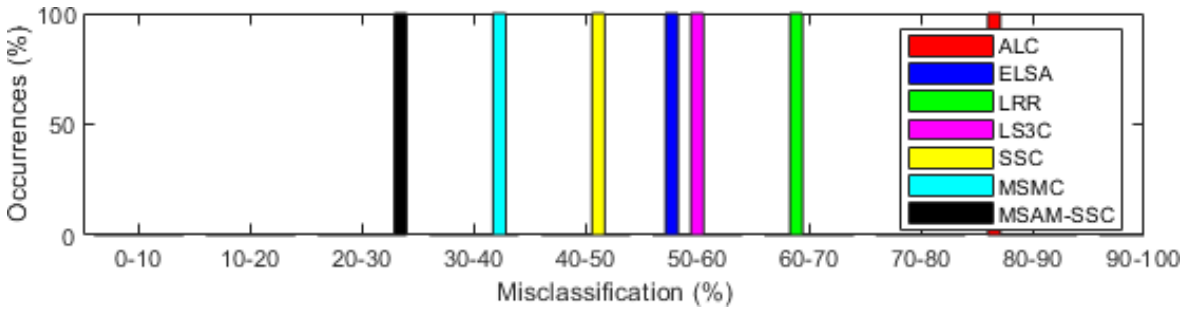
(a) Two motions.



(b) Three motions.



(c) Four motions.



(d) Five motions.

Figure 4.12. Percentages of occurrences of the misclassification error on different numbers of motions. (a) Two motions. (b) Three motions. (c) Four motions. (d) Five motions.

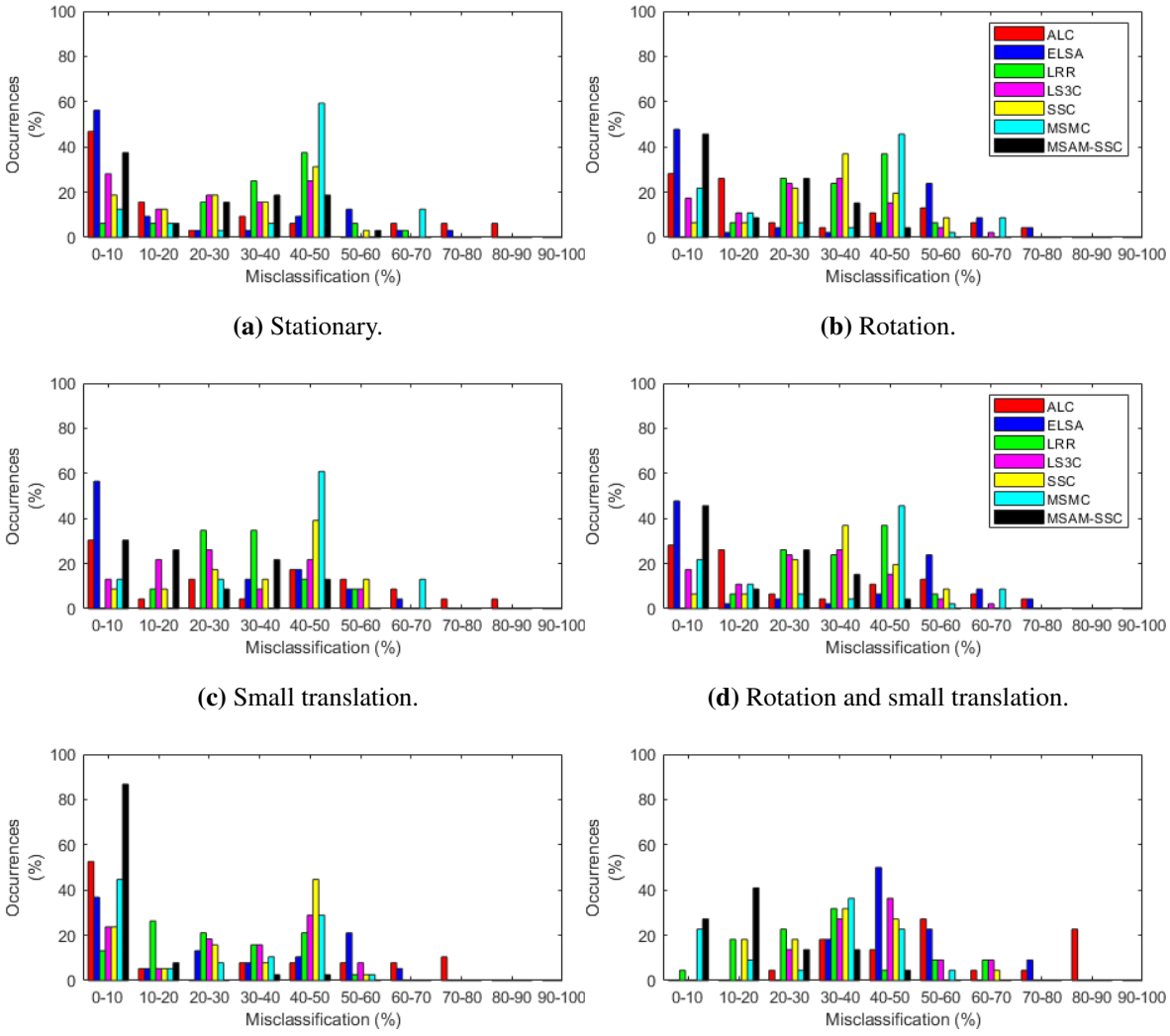
Table 4.8. Misclassification metrics of manifold clustering algorithms on different numbers of motions

	Number of Motions				All
	2	**3**	**4**	**5**	
	(141)	(42)	(5)	(1)	(189)
Average					
ALC	34.25	12.29	51.53	87.92	30.11
ELSA	17.36	52.14	47.37	54.59	26.08
LRR	29.93	40.09	42.09	60.40	32.67
LS3C	25.95	40.12	51.09	59.96	29.94
SSC	29.29	41.93	40.42	45.19	32.48
MSMC	31.96	40.20	27.86	39.82	33.72
MSAM-SSC	13.20	26.67	10.64	26.17	16.19
Median					
ALC	33.67	1.09	43.76	-	21.72
ELSA	2.44	52.49	48.32	-	26.06
LRR	30.63	39.99	42.49	-	33.46
LS3C	27.45	41.74	47.80	-	30.00
SSC	32.22	44.89	39.09	-	36.00
MSMC	43.68	45.26	31.57	-	43.45
MSAM-SSC	6.42	27.82	11.67	-	12.75
Standard Deviation					
ALC	27.88	19.95	19.40	-	28.06
ELSA	22.49	9.27	2.12	-	24.91
LRR	11.85	12.68	13.40	-	12.96
LS3C	15.76	17.52	10.57	-	17.49
SSC	15.06	12.89	6.43	-	15.37
MSMC	19.35	21.83	18.89	-	20.08
MSAM-SSC	15.36	13.93	3.72	-	15.84

4.8 CAMERA MOTION

For some applications, such as autonomous driving, the camera may not be stationary. In these cases, all the scene points, both stationary and moving points, have a motion parameter caused by the camera

motion. The motion segmentation algorithm needs to be able to separate the stationary background from the moving objects as well as to distinguish between the different moving objects despite the motion caused by the camera. The sequences from the Hopkins155 dataset were taken by a stationary or a moving camera. The camera motion can be categorised as rotation, small translation or rotation with small translation. Additionally, some sequences were taken by a handheld camera which is subjected to affine transforms. The KT3DMoSeg dataset was taken by a camera mounted on a moving vehicle, therefore, the camera underwent large translations. The misclassification error on the different categories of camera motion is shown in Table 4.9. The histogram of the misclassification distribution for each of the different categories of camera motion is shown in Figure 4.13.



(a) Stationary.

(b) Rotation.

(c) Small translation.

(d) Rotation and small translation.

(e) Handheld.

(f) Large translation.

Figure 4.13. Percentages of occurrences of the misclassification error on different camera motions. (a) Stationary. (b) Rotation. (c) Small translation. (d) Rotation and small translation. (e) Handheld. (f) Large translation.

4.9 OUTLIERS

The Hopkins155 dataset does not contain any outliers, therefore, outliers were generated using a random walk approach. First, a point is randomly selected from the current frame as the start of the random walk. Then, a trajectory between any two consecutive frames is randomly selected and used to compute the increment that must be taken by the random walk to generate the outlier trajectory. This was done for each of the Hopkins155 sequences, excluding the additional 12 missing data sequences. Synthetic outlier sequences were not created for the KT3DMoSeg dataset since many of these already

Table 4.9. Misclassification metrics of manifold clustering algorithms on different camera motions

	Camera Motion						All
	Stationary (32)	Rotation (28)	Small Translation (23)	Rotation with Small Translation (46)	Handheld (38)	Large Translations (22)	(189)
Average							
ALC	24.31	23.14	33.21	26.92	25.69	58.52	30.11
ELSA	19.42	17.88	21.42	26.61	26.74	48.86	26.08
LRR	35.93	33.09	32.15	36.27	25.94	32.02	32.67
LS3C	25.07	28.99	28.83	28.01	29.05	44.96	29.94
SSC	28.05	37.03	35.35	33.45	29.04	34.02	32.48
MSMC	39.59	43.28	39.93	33.62	20.67	29.30	33.72
MSAM-SSC	21.50	23.00	20.61	15.27	4.14	17.92	16.19
Median							
ALC	14.61	14.51	39.56	17.13	8.51	53.10	21.72
ELSA	3.55	1.72	3.04	18.62	26.56	48.23	26.06
LRR	37.48	33.10	30.82	36.16	22.63	31.33	33.46
LS3C	24.37	27.13	26.14	29.44	32.97	43.62	30.00
SSC	28.53	39.96	40.80	33.47	37.88	34.79	36.00
MSMC	45.75	46.75	46.59	44.28	16.43	34.58	43.45
MSAM-SSC	26.02	26.34	19.18	16.31	0.00	14.43	12.75
Standard Deviation							
ALC	28.35	27.65	26.55	24.18	28.28	21.24	28.06
ELSA	24.22	23.50	24.67	27.46	22.25	11.25	24.91
LRR	13.15	13.43	9.74	10.01	13.87	15.33	12.96
LS3C	16.35	17.50	15.66	16.49	17.67	16.65	17.49
SSC	16.85	12.66	15.84	13.67	18.18	12.32	15.37
MSMC	18.08	14.20	18.54	20.91	20.52	16.79	20.08
MSAM-SSC	18.80	16.38	15.10	14.73	9.22	11.50	15.84

contain outliers. The misclassification error metrics are shown in Table 4.10. Figure 4.14 shows the histogram distribution of the misclassification error.

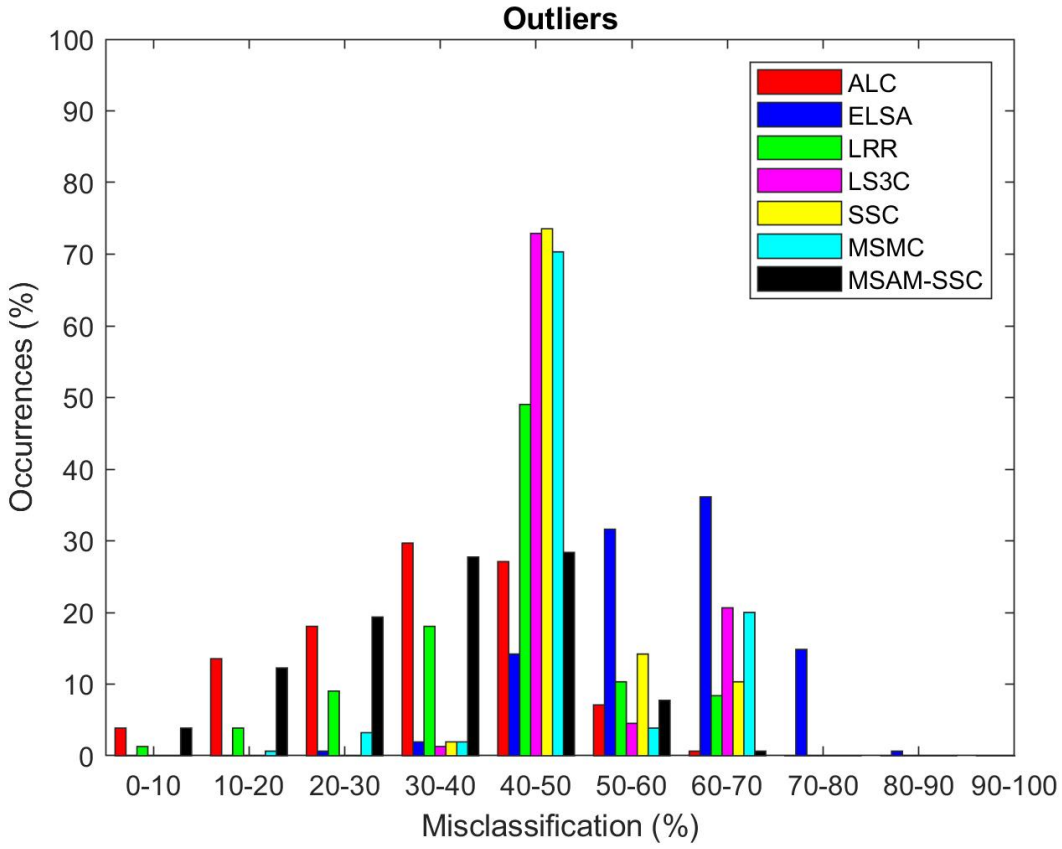


Figure 4.14. Percentages of occurrences of the misclassification error on Hopkins155 sequences with outliers

Table 4.10. Misclassification metrics of manifold clustering algorithms on Hopkins155 sequences with outliers

	Two Motions			**Three Motions**			**All**
	Checker-board (78)	Traffic (31)	Articulate (11)	Checker-board (26)	Traffic (7)	Articulate (2)	(155)
ALC	33.27	25.10	27.75	46.36	29.28	48.08	31.05
ELSA	61.01	55.31	58.18	64.27	58.66	65.50	55.84
LRR	41.20	35.49	32.81	58.54	50.75	49.08	39.82
LS3C	47.57	48.38	46.62	63.29	60.69	59.52	47.38
SSC	47.01	45.94	45.86	60.41	55.79	55.33	45.91
MSMC	45.84	45.67	42.62	62.37	63.30	57.86	45.75
MSAM-SSC	34.11	24.99	28.74	46.94	29.29	47.55	31.57
Median							
ALC	33.44	23.28	28.57	45.92	21.48	48.08	33.79
ELSA	59.89	54.93	60.32	63.98	57.58	65.50	59.82
LRR	43.26	41.67	36.19	59.10	55.47	49.08	42.94
LS3C	47.81	47.40	47.47	63.52	60.16	59.52	48.10
SSC	47.89	46.34	46.15	60.46	56.93	55.33	47.92
MSMC	47.59	47.23	45.21	63.71	63.55	57.86	47.92
MSAM-SSC	34.39	22.75	27.27	46.47	22.25	47.55	34.09
Standard Deviation							
ALC	9.01	14.07	14.64	8.31	13.50	9.32	14.95
ELSA	8.84	11.97	13.94	5.11	9.22	9.67	18.23
LRR	7.54	12.91	12.40	5.75	12.12	10.73	16.12
LS3C	1.99	9.90	2.43	1.11	1.97	3.09	15.22
SSC	2.75	2.78	3.07	2.81	6.05	0.01	14.11
MSMC	6.03	5.59	10.02	5.04	1.07	2.09	15.57
MSAM-SSC	8.96	13.64	13.92	8.58	13.23	10.07	14.99

4.10 CONCLUSION

The results of the proposed algorithm, MSAM-SSC, was presented here. To evaluate the performance, and MSAM-SSC was benchmarked against existing manifold clustering methods, namely ALC, ELSA, LRR, LS3C, SSC and MSMC. The misclassification error was used as the performance metric. Specifically, the average, median and standard deviation were considered since this indicates the spread and overall performance of the algorithms on a group of input sequences. Additionally, the histogram distribution of the misclassification error is also used to evaluate the performance as it shows the spread of the misclassification errors for the input set. First, the performance on the original Hopkins155 and KT3DMoSeg datasets are given to benchmark the performance of MSAM-SSC. The average, median and standard deviation of the execution times are also presented since the execution time is another indication of the performance. Then, the performance on data containing complete occlusions is given followed by that of missing data sequences. For both cases, synthetic data is used where the Hopkins155 and KT3DMoSeg dataset sequences were adapted. Next, the performance on different motion types, numbers of motion and camera motion is evaluated to form a complete picture of the algorithm performance. Lastly, the algorithm performance on Hopkins155 sequences containing synthetic outliers were given.

CHAPTER 5 DISCUSSION

5.1 CHAPTER OVERVIEW

The results presented in Chapter 4 are discussed and evaluated here. Section 5.2 evaluates the results on the original data from the Hopkins155 and KT3DMoSeg datasets. This includes a discussion on the execution times of the algorithms. Section 5.3 contains the discussion on the performance on the synthetic complete occlusion sequences, and Section 5.4 discusses the results on the synthetic missing data sequences where 50% of the original data is missing. Section 5.5 contains the discussion of the results on different types of motions (namely rigid, non-rigid, articulate or degenerate), and Section 5.6 that of different numbers of motion (namely 2, 3, 4, or 5 motions). In Section 5.7, the discussion on different categories of camera motion is given. The categories are rotation, small translation, rotation with small translation, handheld, and large translation.

5.2 COMPARISON OF MANIFOLD CLUSTERING ALGORITHMS

The performance of the MSAM-SSC algorithm was compared to that of ALC, ELSA, LRR, LS3C, SSC and MSMC in order to benchmark the performance. The Hopkins155 dataset is the most popular datasets used to benchmark feature-based motion segmentation methods such as the manifold clustering methods. Additionally, a more complex dataset, namely the KT3DMoSeg dataset was used. The performance of the manifold clustering algorithms on these two datasets is discussed next.

5.2.1 Hopkins155 Dataset

Consider the results of all the manifold clustering algorithms on the Hopkins155 dataset in Table 4.1. The MSAM-SSC algorithm outperformed all the other algorithms on all the Hopkins155 sequences. MSAM-SSC also fared best on the checkerboard and traffic subsets containing two motions with a median misclassification error of 0%. MSAM-SSC also had the best performance on the missing data subset and similar performance of ALC on the traffic subset containing three motions. ALC had the best performance on the subsets containing three motions, and the articulate subset containing two

motions. It is also the only method that was able to outperform MSAM-SSC on any of the subsets. As discussed in section 3.3, ALC performs well on the Hopkins155 dataset, but not on the KT3DMoSeg dataset since it is overfitted to the Hopkins155 dataset. This means that its parameters were chosen such that the misclassification error is minimised for the Hopkins155 dataset and is the reason ALC performed better than MSAM-SSC on some of the subsets. Looking at the standard deviation of the misclassification error, it is evident that most of the values of the MSAM-SSC subsets are below 17% while most median values are below 15%. This indicates that most of the sequences from these subsets had a misclassification error close to the mean.

Considering the different types of scenes, MSAM-SSC performed worse on the checkerboard and articulate subsets containing three motions than on the corresponding subsets containing two motions. However, the difference in the misclassification metrics for the traffic subsets containing two and three motions is significantly smaller than the difference observed for the other subsets. This indicates that MSAM-SSC is sensitive to the increase in the number of motions, but the sensitivity varies with the type of scene. Another observation is that the performance of MSAM-SSC resembles the performance of its two base algorithms. Just like MSMC, MSAM-SSC performed the best on the traffic subsets. MSAM-SSC also performed well on the articulate subsets, just as SSC performed better on these two subsets. Unlike SSC and MSMC, MSAM-SSC also performed well on the missing data subset which indicates that it is more robust to noise than its base algorithms. MSAM-SSC performed the worst on the two checkerboard subsets. These scenes contain checkerboard objects undergoing small motions such as rotations and translations and were taken by either a stationary camera or a camera subject to small controlled motions. Possible reasons for the decreased performance of MSAM-SSC on the checkerboard subsets are explored in subsequent sections.

The histogram distribution of the misclassification error per Hopkins155 sequence, shown in Figure 4.7, is an indication of the spread of the misclassification errors for each algorithm. For approximately half of the sequences, MSAM-SSC had a misclassification error lower than 10%. MSAM-SSC had the best performance since it has the lowest maximum misclassification error (between 50% - 60%). For ALC and ELSA, similar observations are made where their respective largest portion of misclassification errors are below 10%. These algorithms also have the highest recorded misclassification errors which are higher than 70%. This is supported by their respective standard deviation values recorded in Table 4.1. The largest portions of misclassification errors for LRR, LS3C, SSC and MSMC lie between 20% and 50% which indicate that these algorithms had a moderate performance.

Considering the execution times of the manifold clustering algorithms on the Hopkins155 dataset, shown in Table 4.2. For all the sequences, MSAM-SSC had the third-highest execution time of 57.08s. The execution time for the subsets containing two motions is also lower than those of the subsets containing three motions since the increase in the number of motions means an increase in the problem complexity since three classes, excluding the background class must be computed by the algorithms. The execution times for the missing data subsets is also higher than for those subsets containing two motions and is due to the high complexity caused by corrupt data. For MSAM-SSC, as with SSC, these differences in execution time are significant, which means that the execution time increases significantly with the increase of the data complexity. However, for MSMC, which also relies on the MSAM algorithm, the difference in execution time is relatively small. This difference in execution times can be attributed to the complexity of the SSC algorithm since the MSMC algorithm uses a simpler method to infer the final segmentation from the inter-frame motion segments that MSAM-SSC. It is clear that there is a trade-off in execution time to achieve higher accuracy for MSAM-SSC. MSAM-SSC has an average misclassification of less than half of that achieved by MSMC (and SSC) which justifies the loss of execution speed.

5.2.2 KT3DMoSeg Dataset

Considering the misclassification metrics of all the manifold clustering algorithms on the KT3DMoSeg dataset in Table 4.3, MSAM-SSC outperformed the rest of the algorithms by a large margin. When comparing the performance of MSAM-SSC on the Hopkins155 and KT3DMoSeg datasets, it is evident that MSAM-SSC performed slightly worse on the KT3DMoSeg dataset with an average misclassification of 17.92% than on the Hopkins155 dataset which had an average misclassification of 15.59%. Similar observations are made for LRR and SSC. On the other hand, for algorithms such as ALC, ELSA and LS3C the difference in performance is significant and indicates that these algorithms do not generalise well. The difference is since the KT3DMoSeg dataset contains more complex scenes with larger camera translations, and more noise and missing data. Since the difference in performance for MSAM-SSC is small, MSAM-SSC handles corrupt data well. In contrast with this observation, MSMC performed better on the KT3DMoSeg dataset than on the Hopkins155 dataset. MSMC also utilises the MSAM algorithm to perform frame-to-frame analysis which allows corrupt data to be handled more effectively. The type of scene also plays an important role here since it has already been established that MSMC performs well on traffic scenes. These scenes are characterised by rigid, degenerate and articulate motions, large motion parameters and are often taken by a camera undergoing large translations (such as when the camera is mounted on a vehicle). Since the KT3DMoSeg dataset

consists only of traffic scenes, the difference in performance between the KT3DMoSeg and the Hopkins155 datasets makes sense. Comparing the MSMC performance of only the two traffic subsets, MSMC performed significantly worse on the KT3DMoSeg dataset due to the increased complexity of the scenes. Similar observations are made when comparing the performance of MSAM-SSC with that of the two traffic subsets from the Hopkins155 dataset. The difference in the performance can be attributed to the increased complexity of the KT3DMoSeg dataset and it is evident that an increase in the data complexity influences the performance of MSAM-SSC and MSMC.

From Figure 4.8, which shows the histogram distribution of the misclassification error per KT3DMoSeg sequence, it can be seen that the highest percentage of misclassification errors for MSAM-SSC lie below 20% while the highest recorded misclassification errors fall in the range 40% - 50%, which is once again the lowest maximum error of all the algorithms. Therefore, MSAM-SSC outperformed the rest. On the other hand, the largest percentage for ALC lies between 50% - 60% and 80% - 90%, which is significantly worse than the performance on the Hopkins155 dataset. ELSA and LS3C also exhibited a significant decrease in performance with the majority of misclassification errors lying in the range 30% - 60%. Comparing the performance of LRR and SSC on the Hopkins155 and KT3DMoSeg datasets, respectively, the overall performance is similar. The largest portion of the misclassification errors lies between 20% - 50% for both algorithms on both datasets.

Consider the execution times of all the algorithms on the KT3DMoSeg dataset given in Table 4.4. It is interesting to note MSAM-SSC had the same execution time for all the sequences from the KT3DMoSeg dataset. This is due to the similar complexity, scene type and similar length of the input sequences. MSAM-SSC had the best average, median and standard deviation for the execution time, followed by MSMC. Their execution time was significantly lower than the rest of the algorithms. These two algorithms are also the only two algorithms where the execution times for the KT3DMoSeg dataset is lower than that of the Hopkins155 dataset. The sequences from the KT3DMoSeg dataset are shorter than the majority of the Hopkins155 sequences; therefore, fewer inter-frame motion segments must be computed by the MSAM-SSC and MSMC algorithms which leads to shorter execution times. Their execution times are dependent on the number of frames from the input video. For the rest of the algorithms, this significant increase in execution time can be attributed to the increased complexity of the KT3DMoSeg datasets. For these algorithms, the length of the input sequence does not influence the execution times, but rather the complexity of the data. As with the traffic subsets from the Hopkins155 dataset, MSAM-SSC and MSMC performed significantly well on the

KT3DMoSeg dataset with significantly short execution times. Since the KT3DMoSeg dataset only contains sequences of traffic scenes, MSAM-SSC is well suited for applications such as autonomous vehicles and traffic analysis.

5.3 OCCLUSIONS

The main objective is to improve the SSC algorithm to be able to handle large and complete occlusions. For the extreme occlusion case where objects completely disappeared from the camera view for five consecutive frames, ALC, ELSA, LRR, LS3C, and SSC failed to segment any of the adapted Hopkins155 and KT3DMoSeg sequences. MSMC and MSAM-SSC were both able to segment the sequences. From the misclassification metrics in Table 4.5, it can be seen that MSMC had the lowest average misclassification error over all the adapted sequences. MSMC outperformed MSAM-SSC due to the difference in the way the inter-frame motion segments are combined and clustered. MSMC uses Jaccard distance between the inter-frame motion segments to construct an affinity matrix which is used to infer the final segmentation. This is simpler than the SSC method which MSAM-SSC relies on and is more robust to occlusions. The same observation is made for the KT3DMoSeg sequences as well as the missing data, and both of the checkerboard and articulate Hopkins155 subsets. MSAM-SSC only had a better average misclassification error for the two traffic subsets from the Hopkins155 dataset and was outperformed by MSMC. Considering the median misclassification error, MSMC had a median of 0% over all the sequences and all the subsets, except for the KT3DMoSeg dataset. MSAM-SSC had a median of 0% over all the sequences, but the median for the KT3DMoSeg and Hopkins155 subsets containing three motions were relatively high. The median for the Hopkins155 missing subset was relatively low. When considering the standard deviation, it is evident that MSMC has a lower standard deviation for all subsets other than the two Hopkins155 traffic subsets. Taking the average, median and standard deviation of the misclassification error into account, it is evident that MSAM-SSC performed better than MSMC on the traffic subsets from the Hopkins155 dataset. It is also clear that the average misclassification error for MSAM-SSC significantly increased with the increase of sequence complexity. In other words, the misclassification errors for the subsets with three motions is higher than that of the corresponding subset with two motions, while it is even higher for the missing data subset and significantly higher for the KT3DMoSeg dataset which has the highest complexity.

Now comparing these results to those on the original data results from the Hopkins155 dataset given in Table 4.1, the average, median, and standard deviation of both algorithms on the Hopkins155 subsets is lower for the occlusion case than for the original sequences. This is unexpected behaviour since

occlusions cause missing data which significantly increases the complexity of the motion segmentation task. Note that the data from the Hopkins155 dataset is complete with no noise or missing entries (except for the missing data subset) and has a low complexity since there are only two or three motions. When one of the objects completely disappears for five consecutive frames, the complexity of performing inter-frame motion segmentation significantly decreases. For the sequences which originally contain only two motions, for these five frames, there is only one moving object that must be distinguished from the background. Since the data is otherwise complete, the motion segmentation problem becomes simple, and both algorithms can segment the sequences with higher accuracy. Similar observations are made for the Hopkins155 sequences containing three motions as well as the missing data subset. MSAM-SSC is more sensitive to the increase in the number of motions and scene complexity than MSMC since the difference in the average and median misclassification errors are higher for the subsets containing three motions as well as for the missing data subset. This observation can be attributed to the sensitivity of the SSC algorithm to increases in the number of motions. In contrast with the observation of the improved performance on the Hopkins155 sequences, the MSAM-SSC metrics for the KT3DMoSeg dataset are significantly higher for the occlusion case than for the original data, as shown in Table 4.3. This is expected since the complexity of the occlusion case is significantly higher than that of the original data. However, the metrics for MSMC are lower than on the original KT3DMoSeg data as observed for the Hopkins155 subsets.

The histogram distribution of the misclassification error per sequence in Figure 4.9 supports the above observations. From the plot, MSMC performed better than MSAM-SSC. More than 85% of the sequences had a misclassification error of less than 10% for MSMC while more than 65% of the sequences for MSAM-SSC fall in the same range. MSAM-SSC also has a higher maximum misclassification (between 50% - 60%) than MSMC (between 40% - 50%).

It is clear that MSMC handles complete occlusions better than MSAM-SSC, but MSAM-SSC shows similar behaviour to that observed by MSMC. The method used by MSMC to infer the final segmentation from the inter-frame motion segments is more robust to occlusions and increases in the number of motions than the classic SSC algorithm which causes the MSAM-SSC algorithm to be sensitive to changes in the problem complexity. However, since classic SSC completely fails on these sequences, the MSAM-SSC algorithm is a significant improvement on the traditional approach and has high accuracy. Note that the sequences from the Hopkins155 and KT3DMoSeg datasets are short where the minimum sequence length is 10 frames. Due to the short sequence length, the MSAM-SSC and MSMC

algorithms were not tested on occlusions that lasted longer than 5 frames since for short sequences, this means that the objects are occluded for half of the sequence length. To allow a minimum of two objects to be occluded for all sequences, the length of the occlusions were set to 5 frames. Ideally, the effect of longer occlusion times must be investigated on longer sequences to investigate if longer occlusions deteriorate the performance.

5.4 MISSING DATA

Often the point trajectories can have missing entries due to noise and data corruption caused by factors such as changes in the illumination, and camera motion. Table 4.6 shows the results of MSMC and MSAM-SSC on missing data sequences which were generated by randomly removing half of the entries for each sequence from the Hopkins155 and KT3DMoSeg datasets. ALC, ELSA, LRR, LS3C and SSC were unable to segment any of the adapted sequences. From Table 4.6, it can be seen that MSAM-SSC had the lowest average and median misclassification error over all the sequences as well as for each of the subsets. Therefore, MSAM-SSC outperformed MSMC. However, MSMC had a lower standard deviation over all the sequences, as well as for the KT3DMoSeg dataset sequences and some of the Hopkins155 subsets such as the articulate subsets. Comparing the results on the generated missing sequences with the corresponding Hopkins155 subsets in Table 4.1, both algorithms performed worse on the missing data sequences than on the original Hopkins155 data. Similar observations are made when comparing the KT3DMoSeg missing data sequence results with that of the original data shown in Table 4.3. This behaviour is expected since only half of the original data is available. The difference in the average misclassification for MSAM-SSC between the adapted missing data sequences and the original data from both datasets is less than 15%. Considering that there is a loss of 50% of the original data, this difference in the average misclassification error is not significantly high, and it is evident that MSAM-SSC handles missing data well.

Looking at the histogram plot of the misclassification error per sequence in Figure 4.10, MSAM-SSC outperformed MSMC since more sequences fall into lower misclassification ranges with the majority of the sequences having a misclassification error of less than 50%. Both methods were still able to segment the sequences with relative success unlike ALC, ELSA, LRR, LS3C and SSC which failed.

Occlusions are one of the causes of missing data since the entries of points for the corresponding frames are missing. However, algorithms can react differently to missing data caused by occlusions than

other causes of missing data, as is seen when comparing Tables 4.5 and 4.6. MSMC and MSAM-SSC performed better under the occlusion conditions than on the original data, while both their performances deteriorated under the random missing data conditions. This shows that it is important to explicitly test the algorithms under occlusion conditions.

5.5 MOTION TYPE

In most real-world applications, scenes can contain different types of motion, and it is important to evaluate the performance of algorithms on each type. Looking at the misclassification metrics in Table 4.7, MSAM-SSC outperformed the rest of the algorithms for each motion type. The algorithm had the best performance on the non-rigid subset that contains human motion such as that of the head. MSAM-SSC also performed well on the degenerate and articulate motion sets with an average misclassification error of less than 6% and a median of 0%. The degenerate motion sequences are predominantly traffic scenes where the object shapes change from the camera perspective due to the movement of people and vehicles. The articulate subset contains sequences of human and robotic motions. This is consistent with the observations made on the Hopkins155 dataset where MSAM-SSC performed the best on the subsets on which MSMC and SSC also had the best performance, respectively. MSAM-SSC performed the worst on the rigid motion subset which contains sequences of objects moving independently without changing their shapes. This behaviour is expected since both of its base algorithms, namely MSMC and SSC, did not perform as well on this subset as on the degenerate and articulate subsets. Also, the rigid subset consists mainly of checkerboard sequences from the Hopkins155 dataset and the scene type may have an influence on the performance of MSAM-SSC. The size of the displacement may also contribute to the decreased performance, e.g. if an object moves slowly, the displacement between frames is small and the MSAM part of the MSAM-SSC algorithm can mistake these points to be part of the background. Factors such as the camera motion could also contribute to the decreased performance on the checkerboard sequences. However, MSAM-SSC performs well on traffic scenes that also contain rigid motion, therefore, the decreased performance on the checkerboard scenes cannot be attributed to the type of motion.

Now consider the histogram distribution of the misclassification error for each motion type as illustrated in Figure 4.11. When looking at the distribution for rigid motions, MSAM-SSC had the best performance since it is the only algorithm for which all the misclassification errors are below 50%. Even though ALC and ELSA had some of the highest portions of misclassification errors below 10%, they are also the only algorithms with misclassification errors higher than 70%. MSMC did not perform

well since most of its misclassification errors fall in the range 40% -70%. LRR, LS3C and SSC had a moderate performance with most of the misclassification errors between 20% - 50%. Looking at the distribution for the non-rigid motion sequences, it is evident that LRR and MSAM-SSC had similar performance and outperformed the rest since all their misclassification errors are below 20%. ALC had the second-best performance, followed by LS3C, MSMC and SSC. ELSA had the worst performance since approximately 30% of its misclassification errors fall in the range 50% - 60%.

Turning the attention to the articulate motion misclassification distribution, it can be seen that MSAM-SSC also had the best performance with the majority of its misclassification errors having a value of lower than 30%. Even though similar trends for ALC and ELSA are observed here as for the rigid motion case, both algorithms performed better since larger portions of the misclassification errors are below 10%. LS3C and SSC both performed better here than on the rigid motion subset since all the misclassification errors for SSC and the majority of the misclassification errors for LS3C are below 50%. LRR and MSMC are the only algorithms that had similar performance to the rigid case. From the distribution for the degenerate motion case, it is clear that MSAM-SSC performed significantly well once again with 80% of the misclassification errors below 10%. Even though ALC has the largest maximum misclassification error, which falls in the range 70% - 80%, the overall performance is better relative to the previous motion cases since the majority of misclassification errors are below 50%. The same is observed for ELSA. MSMC also performed better than on the previous motion cases with a larger portion of misclassification errors having a value lower than 10%. The largest portions of misclassification errors for LRR, LS3C and SSC are between 20% - 50%.

Considering all the results, it is evident that MSAM-SSC performed the best on non-rigid and degenerate motions, while its performance on the rigid and articulate cases had more variation. MSAM-SSC is most suited for tasks such as autonomous driving, traffic surveillance and segmentation and tracking of human motion. There are some drawbacks to the evaluation done here. The subsets vary in length and, ideally, all subsets must have the same number of sequences to draw more accurate conclusions about the performance. The non-rigid subset only contains three sequences which are not enough to draw accurate conclusions regarding the performance of the algorithms. Also, the majority of the sequences in each category contain similar scene types and conditions. Ideally, the sequences should be from a variety of different scenes under different conditions, such as noise and illumination.

5.6 NUMBER OF MOTIONS

The majority of scenes contain more than one moving object; therefore, it is important to evaluate the performance of an algorithm on multiple motions and establish if an increase in the number of motions negatively impacts the performance. For the Hopkins155 dataset, it has already been observed that the performance of MSAM-SSC decreases significantly with the increase in the number of motions, depending on the scene type. Consider the misclassification metrics given in Table 4.8 which takes the performance of the algorithms on both the Hopkins155 and KT3DMoSeg datasets into account. MSAM-SSC had the best performance across all the different numbers of motions, except for the case of three motions where ALC had the best performance. Overall, the performance of ALC decreased with the increase in the number of motions, except for the subset containing three motions but this may be due to other factors such as the type of scene. This subset also consists largely of Hopkins155 sequences and since ALC is overfitted to the Hopkins155 dataset, it has increased performance. A similar phenomenon is observed for MSMC and MSAM-SSC that performed better on the subset containing four motions, but this is since this subset consists predominantly of traffic scenes. The difference in the performance between the subsets containing two and three motions for ELSA, LRR, LS3C and SSC is significant but remains relatively stable with any further increases in the number of motions. Since there is only one sequence that contains five moving objects, no real conclusions can be made on the algorithm performance on sequences containing five motions.

Consider Figure 4.12 the histogram plot of the misclassification error per sequence for each subset containing different numbers of motion. For the subset containing two motions, it is evident the largest portion of the misclassification errors for ELSA and MSAM-SSC has a value of less than 10%. ELSA has sequences with misclassification errors which fall in every bin up until 70% - 80% whereas the highest misclassification bin range for MSAM-SSC is 40% - 50% and, therefore, MSAM-SSC had the best performance. ALC had the worst performance and is the only algorithm with misclassification errors between 80% - 90%. LRR, LS3C, SSC and MSMC have their largest percentage of sequences with misclassification errors between 20% - 50%. From the distribution of the misclassification errors on the subset containing three motions, it can be seen that ALC had the most sequences with a misclassification error below 10% and had the best performance but is also the only algorithm with sequences that have a misclassification error between 80% - 90%. ELSA and MSMC performed significantly worse than they did on the subsets with two motions having the highest number of misclassification errors between 50% - 60% and 60% - 70%, respectively. MSAM-SSC also performed worse with fewer misclassification errors below 10%. LRR, LS3C and SSC also showed a decrease in

performance since most misclassification errors fall in the range 30% - 60%.

Considering the subset containing four motions, MSAM-SSC performed the best with all the misclassification errors having a value of less than 20%. Even though about 40% of the misclassification errors for MSMC falls in the same range, the other 60% falls in the range 30% - 50%, therefore MSMC had the second-best performance. Both algorithms performed better here than on the subset containing three motions, but this is because most of the scenes in these subsets are traffic scenes. Most of the misclassification errors for ALC are located between 30% - 60%, but about 20% is located between 80% - 90% which is the highest recorded error for any of the algorithms. ELSA performed significantly worse on this subset than on the previous subsets since all the misclassification errors are in the range 40% - 50%. SSC also showed decreased performance in comparison to the previous subsets which contain fewer motions with all its misclassifications between 30% - 50%. LRR performed slightly worse than SSC since its misclassification errors are spread over a larger range (20% - 60%). LS3C performed worse than SSC and LRR since most of its misclassification errors are between 40% - 70%.

Overall, MSAM-SSC had the best performance across all the sequences, and an increase in the number of motions does not significantly impact its performance. MSMC showed a similar trend. This observation is made for mostly traffic scenes and may differ for other types of scenes. On the other hand, ALC, ELSA, LS3C, LRR and SSC are sensitive to the increase in the number of moving objects present in the scene. Even though trends were identified in the algorithm performance when the number of motions were increased, these trends would be much clearer for subsets of the same size containing sequences from a wide variety of scenes types.

5.7 CAMERA MOTION

For some applications, the camera may move which causes all points to a motion parameter. Algorithms must be able to handle these cases and the effect of camera motion on the algorithm performance is evaluated. From the misclassification metrics shown in Table 4.9, it can be seen that MSAM-SSC outperformed the other algorithms for all camera motion categories, except for the stationary and rotation categories. MSAM-SSC performed particularly well on the handheld category where the camera undergoes small affine transformations. MSAM-SSC also performed well when the camera undergoes rotation with small translations, and for large transitions, such as occurs for a camera mounted on a moving vehicle. MSAM-SSC and MSMC had the worst performance on the rotation

subset which consists of checkerboard sequences from the Hopkins155 dataset. The camera rotation for these sequences are small and the rotations and/or translations underwent by the checkerboard objects are small. The MSAM part of the MSAM-SSC algorithm performs poorly under these conditions since sub-optimal inter-frame motion segments are obtained due to the small difference between the displacement of background and moving objects. However, the SSC part of the MSAM-SSC algorithm refines the inter-frame motion segments to obtain more accurate final motion segments than the method applied by MSMC. Similar observations are made for the small translation and rotation with small translation subsets. On the other hand, the objects from the handheld and large translation categories undergo larger motions which allow MSAM to find more accurate inter-frame motion segments to increase the overall performance of MSAM-SSC and MSMC. In contrast, ELSA handles smaller motions of objects with higher accuracy while its performance deteriorates when larger motions are encountered. MSMC performed the best on the handheld and large transition categories which predominantly consists of traffic sequences. SSC had the best performance on the stationary and handheld categories. ALC, ELSA and LS3C performed the worst on the large transition category and are not suited for tasks such as autonomous driving. LRR performed the best on the handheld category while showing similar performance on the rest.

Consider the histogram distribution of the misclassification errors shown in Figure 4.13. For the case where the video was taken with a stationary camera, the largest portion of the misclassification errors for MSAM-SSC is below 40%. Approximately half of the misclassification errors for ALC and ELSA is below 10% and, therefore, they have high accuracy. Their remaining misclassification errors are widely spread across most of the bins with the maximum misclassification errors falling in the range of 80% - 90% for ALC and 70% - 80% for ELSA. The largest portion of misclassification errors for LRR, LS3C, SSC and MSMC are between 20% - 50%.

For the rotation, small translation, and the rotation with small translation categories, ALC and ELSA showed similar trends as observed for the stationary category. However, the performance on the small translation and the rotation with small translation categories is slightly worse with fewer misclassification errors below 10%. In contrast with this observation, MSMC had better performance on these two categories than the rest with a larger portion of misclassification errors smaller than 10%. MSAM-SSC showed similar trends for the rotation, and small translation categories as that of the stationary category. A larger portion of the MSAM-SSC misclassification errors for the rotation with small translation category lies below 10%.

For the handheld category, MSAM-SSC performed significantly well since approximately 90% of the misclassification errors are below 10%. ALC, LS3C, SSC and MSMC also performed better on this category since both algorithms have a larger portion of misclassification errors with a value less than 10%. For the large translation category, once again, MSAM-SSC had the best performance with most of its misclassification errors having a value of less than 30%. ALC, ELSA and LS3C performed significantly worse since all their misclassification errors are higher than 20%. ALC also has a larger portion of misclassification errors in the range 80% - 90%. MSMC performed better here than on the first three categories with a larger portion of misclassification errors below 20% than previously. SSC had a similar performance to that of all the previous categories except the stationary and handheld categories. LRR had a similar performance across all the categories.

5.8 OUTLIERS

In many real-world instances, the input sequences are corrupted with outlier,s and algorithms must be able to handle these cases. From Table 4.10, it can be seen that ALC and MSAM-SSC had similar performance, but ALC performed slightly better than MSAM-SSC on most subsets. ALC is overfitted to the Hopkins155 data, therefore, it was expected to have one of the best performance even if the original data is corrupted by outliers. ELSA performed the worst with the highest average misclassification error, followed by LS3C. SSC and MSMC had similar performance but were outperformed by LRR. Comparing the results on the synthetic outlier sequences with that of the original sequences in Table 4.1, as expected, all the algorithms performed significantly worse than originally. The performance of MSAM-SSC deteriorated significantly more than ALC and LRR which indicates that it is less robust to noise. Also, note that the performance for the subsets containing three motions decreased significantly more than for the subsets containing two motions since MSAM-SSC is sensitive to the increase in the number of motions. Similar observations are made for ELSA, LS3C, SSC and MSMC. This is a drawback since most real-world sequences are corrupted by outliers. Now consider the performance of MSAM-SSC and MSMC on the synthetic outliers and the synthetic missing data sequences 4.6. Both algorithms performed better on the missing data sequences than on the outlier sequences. This indicates that both algorithms handle missing data better than outliers.

Consider the histogram distribution of the misclassification errors. The majority of the misclassification errors are lower than 50% for ALC and MSAM-SSC. ELSA had the worst performance since it has the most misclassification errors above 50% and is the only algorithm with misclassification errors higher than 70%. LS3C, SSC and MSMC had similar performance with the largest portion of their

misclassification errors lying between 30% - 60%. LRR is the only algorithm other than ALC and MSAM-SSC that has misclassification errors smaller than 10%.

5.9 CONCLUSION

The performance of MSAM-SSC was evaluated and it was found that MSAM-SSC had a higher accuracy on the Hopkins155 and KT3DMoSeg dataset than ALC, ELSA, LRR, LS3C, SSC and MSMC. MSAM-SSC had some of the worst execution times. Only MSAM-SSC and MSMC were able to successfully segment the complete occlusion and missing data sequences. It was found that both algorithms performed better on the complete occlusion sequences than on the original data due to the nature of the MSAM algorithm. MSMC performed better on the complete occlusion sequences than MSAM-SSC. On the missing data sequences, MSAM-SSC had the best performance. The evaluation of the performance on different motion types revealed that MSAM-SSC handles non-rigid and degenerate motions the best, but still outperformed the rest of the algorithms on rigid and articulate motions. MSAM-SSC, like MSMC, is also less sensitive to the increase in the number of motions than the rest of the algorithms. It was observed that MSAM-SSC handles affine and large camera transitions the best but still had the best performance on the remaining camera motion categories. Lastly, it was found that MSAM-SSC can handle outliers, but it handles occlusions and missing data better.

CHAPTER 6 CONCLUSION

Motion segmentation is an active field that still faces many challenges since the current methods are far from human capabilities. Two of the greatest challenges are occlusions and missing data. An extension of the SSC algorithm, called the MSAM-SSC algorithm, was proposed to address these two issues. The algorithm extends the classic SSC algorithm by adding a pre-processing step, called MSAM, which is based on the classic top-down split-and-merge algorithm. The MSAM algorithm performs motion segmentation on pairs of frames to extract inter-frame motion segments that are used as the input for the classic SSC algorithm. The occlusion problem is solved by grouping points belonging to the same motion between two frames rather than over the entire sequence. Occlusions cause incomplete trajectories where data entries are missing from the input matrix for the frames in which points disappear from the camera view. MSAM only assigns points visible in both frames to a moving object class while points that are occluded in one of the frames are ignored. Therefore, the problem of grouping incomplete trajectories is avoided using the frame-to-frame analysis approach. All the inter-frame motion segments are combined and used as the input for the SSC algorithm to find the final motion segments.

6.1 CONCLUSIONS

From the results, it is clear that the proposed MSAM-SSC algorithm outperforms ALC, ELSA, LRR, LS3C, SSC and MSMC on the Hopkins155 and KT3DMoSeg datasets. The algorithm generalises well since the difference in performance on the two datasets is small, unlike other algorithms such as ALC and ELSA. Another observation made on both datasets is that MSAM-SSC, like MSMC which also employs the MSAM algorithm, has a high accuracy for traffic scenes. These scenes contain mixtures of motions such as rigid, articulate and degenerate. MSAM-SSC also fared well on scenes containing articulate motion, just like the classic SSC algorithm. However, unlike MSMC, MSAM-SSC is more sensitive to increases in the number of motions and the complexity of the sequences but less sensitive than SSC.

For both datasets, it was observed that MSAM-SSC has long execution times. MSAM-SSC compromises execution time for accuracy which makes it unsuited for real-time applications. The algorithm can possibly be sped up by using fewer frame pairs to obtain inter-frame motion segments. Depending on the camera and speed of the moving objects, scenes do not undergo major changes between two consecutive frames, and it is unnecessary to use each of the consecutive frame pairs. Instead of using every consecutive frame to form frame pairs, every second or third frame can be used, which will lead to an improved execution time. For future work, the effect on the algorithm performance and execution time can be investigated. Another way to speed up the execution time is to employ parallel computing. Since MSAM operates on each frame pair independently, it is possible to execute the MSAM algorithm on multiple frame pairs simultaneously. By using fewer frame pairs and employing parallel computing, the execution time can be reduced significantly.

MSAM-SSC and MSMC are the only two algorithms that were able to successfully segment any of the generated complete occlusion and missing data sequences. The complete occlusion sequences were generated by removing the point locations for each object for five consecutive frames. Therefore, each of the objects disappeared completely for frames. The missing data sequences were generated by randomly removing half of the entries in the trajectory matrix. For the complete occlusion case, it was observed that both algorithms performed significantly better than on the original data. This is since the problem of finding the inter-frame motion segments was simplified for the frames where one object was completely occluded since fewer objects are present. For the Hopkins155 sequences containing only two motions originally, for these frames, only one object is visible, therefore the problem is significantly simpler, and the MSAM part of both algorithms can identify the objects with greater accuracy. However, MSMC outperformed MSAM-SSC due to the difference in which the inter-frame motion segments are used to infer the final segmentation by the sub-algorithm of MSMC, which is more robust to the occlusion case than classic SSC used by MSAM-SSC. For the generated missing data sequences, both algorithms had lower accuracy than on the original sequences, which is expected behaviour due to the increase in the problem complexity. The difference in the performance was not severe considering that half of the original data was missing. Here, MSAM-SSC performed better than MSMC and is more robust to missing data caused by factors such as noise and changes in the illumination.

The performance of MSAM-SSC on different types of motion and different numbers of motion were investigated since scenes can contain mixtures of different motions. It was found that MSAM-SSC

handles non-rigid and degenerate motions the best. These types of motions are encountered in scenes containing human and robotic motions as well as traffic scenes. This is in line with the observations made on the original sequences from the Hopkins155 and KT3DMoSeg dataset. From the performance on the different number of motions, it was observed that the performance decreased with the increase of the number of motions since the problem complexity increased. For MSAM-SSC, like MSMC, the deterioration in performance was not as significant. It was observed that MSAM-SSC is less sensitive to the number of motions than to the scene type and types of motions.

Since the input can be obtained from a moving camera, the performance of the algorithms under different camera motions was investigated. It was found that MSAM-SSC had the best performance for scenes taken by a handheld camera (where the camera undergoes affine transforms) as well as large camera translations such as when the camera is mounted on a moving vehicle. It did not perform well on the rotation, small translation and rotation with small translation sequences. These three categories consisted largely of checkerboard sequences from the Hopkins155 dataset, which exhibits small object motions. Therefore, the object displacement between frames is too small for the MSAM-SSC algorithm to distinguish these points from the background points.

Often, the input sequences are corrupted by noise such as outliers, and algorithms must be able to deal with these cases efficiently. MSAM-SSC can handle outliers, but the performance deteriorates significantly. However, just like ALC, MSAM-SSC still outperformed the rest by a significant margin. It was also observed that MSAM-SSC handles occlusions and missing data better than outliers.

6.2 SUMMARY OF CONTRIBUTIONS

The MSAM-SSC algorithm presented can handle large and complete occlusions with higher accuracy than sequences in which no or partial occlusions are present. The algorithm can also handle large percentages of missing data, as well as mixtures of motions. Additionally, MAMS-SSC outperformed other popular manifold clustering algorithms. A paper was written presenting the literature review on motion segmentation and was published in the Proceedings of the 2020 2nd IMITEC. A second paper presenting the findings of this work is being prepared and will be submitted for publication.

6.3 FUTURE RESEARCH WORK

MSAM-SSC not only added capabilities to the classic SSC algorithm to be able to handle large and complete occlusions and missing data, but it also has higher accuracy. The capabilities of MSAM-SSC must be investigated further on longer sequences (e.g. 2 min videos) from a larger variety of scenes

containing more complex problems. Specifically, the effect of longer occlusions must be investigated as well as larger numbers of motion and more complex mixtures of motions.

MSAM-SSC is not the optimal solution. As already discussed, the execution time is not near real-time, and for future work, the MSAM part of the algorithm can be sped up. As discussed previously, this can be achieved by using parallel programming to find the inter-frame motion segments for different frame pairs. Additionally, the number of frame pairs can be reduced by forming frame pairs with every second or third frame rather than the consecutive frame. Another way to improve the algorithm accuracy is to use an improved SSC algorithm. There are many variations of the classic SSC algorithms which have greater accuracy and improved speed. These include ASSA and LNRSI. By swapping out the classic SSC algorithm with one of these variations, the accuracy and speed of the MSAM-SSC algorithm can be improved. A major drawback of the MSAM-SSC algorithm is that the number of moving objects within the scene must be known beforehand since this information is required by the classic SSC algorithm. In most real-world applications, the number of motions is not available and must be inferred from the data. This can be achieved by using methods such as rank estimation.

It is clear that by adopting a frame-to-frame analysis, challenges such as occlusions and missing data can be handled with great accuracy. Since frame-to-frame analysis as a pre-processing step improves the performance of SSC, it must be investigated further to determine if it can be used to improve the performance of other manifold clustering-based approaches such as ALC and LRR.

REFERENCES

[1] S. Khan and M. Shah, "Object Based Segmentation of Video Using Color, Motion and Spatial Information," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Oct. 2001, pp. 746 – 751.

[2] J. Mattheus, H. Grobler, and A. Abu-Mahfouz, "A Review of Motion Segmentation: Approaches and Major Challenges," in *2nd International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2020*, 2020, pp. 1 – 8.

[3] A. Colombari, A. Fusiello, and V. Murino, "Segmentation and tracking of multiple video objects," *Pattern Recognition*, vol. 40, pp. 1307 – 1317, Apr. 2007.

[4] Y. Zhang, B. Luo, and L. Zhang, "Permutation Preference Based Alternate Sampling and Clustering for Motion Segmentation," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 432 – 436, 2018.

[5] A. Kushwaha, C. Sharma, M. Khare, O. Prakash, and A. Khare, "Adaptive Real-time Motion Segmentation Technique Based on Statistical Background Model," *The Imaging Science Journal*, vol. 62, pp. 285 – 302, Jun. 2014.

[6] L. Xu, J. Chen, and J. Jia, "A Segmentation Based Variational Model for Accurate Optical Flow Estimation," in *ECCV 2008: 10th European Conference on Computer Vision*, Oct. 2008, pp. 671 – 684.

[7] A. Delong, A. Osokin, H. Isack, and Y. Boykov, "Fast Approximate Energy Minimization with Label Costs," *International Journal of Computer Vision*, vol. 96, pp. 2173 – 2180, Jun. 2010.

[8] M. Khare, R. Srivastava, and A. Khare, "Moving object segmentation in Daubechies complex wavelet domain," *Signal Image and Video Processing*, vol. 9, pp. 635 — 650, Mar. 2015.

[9] F. Shi, Z. Zhou, J. Xiao, and W. Wu, "Robust Trajectory Clustering for Motion Segmentation," in *2013 IEEE International Conference on Computer Vision*, Dec. 2013, pp. 3088 – 3095.

[10] Y. Zhang and Z. He, "Video Object Segmentation of Dynamic Scenes with Large Displacements," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 9, pp. 1719 – 1723, 2015.

[11] M. Cai, X. Lu, X. Wu, and Y. Feng, "Intelligent Video Analysis-based Forest Fires Smoke Detection Algorithms," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Aug. 2016, pp. 1504 – 1508.

[12] A. B. Chan and N. Vasconcelos, "Modeling, Clustering, and Segmenting Video with Mixtures of Dynamic Textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909 – 926, 2008.

[13] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi, "Tracking Deforming Objects Using Particle Filtering for Geometric Active Contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1470 – 1475, 2007.

[14] M. Pawan Kumar, P. H. S. Torr, and A. Zisserman, "Learning Layered Motion Segmentations of Video," *International Journal of Computer Vision*, vol. 76, no. 3, pp. 301 – 319, Mar 2008.

[15] L. Ge, C. Zhang, Z. Chen, and M. Li, "Optical Flow Estimation from Layered Nearest Neighbor Flow Fields," in *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2018, pp. 1 – 6.

[16] K. Wattanachote and T. K. Shih, "Automatic Dynamic Texture Transformation Based on a New Motion Coherence Metric," *IEEE Transactions on Circuits and Systems for Video Technology*,

vol. 26, no. 10, pp. 1805 – 1820, Oct. 2016.

[17] D. Anguelov, D. Koller, H. C. Pang, P. Srinivasan, and S. Thrun, "Recovering Articulated Object Models from 3D Range Data," in *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, Jul. 2012, pp. 18 – 26.

[18] H. J. Chang and Y. Demiris, "Unsupervised learning of complex articulated kinematic structures combining motion and skeleton information," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3138 – 3146.

[19] Y. Wang, P. Wang, Z. Yang, C. Luo, Y. Yang, and W. Xu, "UnOS: Unified Unsupervised Optical-Flow and Stereo-Depth Estimation by Watching Videos," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8063 – 8073.

[20] P. Tzirakis, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Time-series Clustering with Jointly Learning Deep Representations, Clusters and Temporal Boundaries," in *2019 14th IEEE International Conference on Automatic Face Gesture Recognition (FG 2019)*, 2019, pp. 1 – 5.

[21] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas, "Deep Part Induction from Articulated Object Pairs," *ACM Transactions on Graphics*, vol. 37, pp. 1 – 18, Sept. 2018.

[22] J. Fayad, C. Russell, and L. Agapito, "Automated articulated structure and 3D shape recovery from point correspondences," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 431 – 438, Nov. 2011.

[23] K. Yücer, O. Wang, A. Sorkine-Hornung, and O. Sorkine-Hornung, "Reconstruction of Articulated Objects from a Moving Camera," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Dec. 2015, pp. 823 – 831.

[24] R. Vidal, R. Tron, and R. Hartley, "Multiframe Motion Segmentation with Missing Data Using PowerFactorization and GPCA," *International Journal of Computer Vision*, vol. 79, pp. 85 – 105, Aug. 2008.

[25] J. Yan and M. Pollefeys, "A Factorization-Based Approach for Articulated Nonrigid Shape, Motion and Kinematic Chain Recovery From Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 865 – 877, May 2008.

[26] M. Paladini, A. Del Bue, M. Stosic, M. Dodig, J. Xavier, and L. Agapito, "Factorization for Non-rigid and Articulated Structure using Metric Projections," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2898 – 2905.

[27] L. Zappella, X. Llado, E. Provenzi, and J. Salvi, "Enhanced Local Subspace Affinity for Feature-based Motion Segmentation," *Pattern Recognition*, vol. 44, pp. 454 – 470, Feb. 2011.

[28] L. Zappella, X. Llado, and J. Salvi, "Motion Segmentation: a Review," in *Frontiers in Artificial Intelligence and Applications*, vol. 184, Jan. 2008, pp. 398 – 407.

[29] P. Tokmakov, K. Alahari, and C. Schmid, "Learning Motion Patterns in Videos," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 531 – 539.

[30] C. Zheng and H. Yao, "Segmentation for Remote-sensing Imagery Using the Object-based Gaussian-Markov Random Field Model with Region Coefficients," *International Journal of Remote Sensing*, vol. 40, pp. 1 – 32, Jan. 2019.

[31] J. H. Hammer, M. Voit, and J. Beyerer, "Motion Segmentation and Appearance Change Detection-based 2D Hand Tracking," in *2016 19th International Conference on Information Fusion (FUSION)*, 2016, pp. 1743 – 1750.

[32] R. Vidal, Y. Ma, and S. Sastry, "Generalized Principal Component Analysis (GPCA)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 12, pp. 1945 – 1959, 2005.

[33] S. Rao, R. Tron, R. Vidal, and L. Yu, "Motion Segmentation in the Presence of Outlying, Incomplete, or Corrupted Trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1832 – 45, Oct. 2010.

[34] E. Elhamifar and R. Vidal, "Sparse Subspace Clustering," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2790 – 2797.

[35] J. Yan and M. Pollefeys, "Articulated Motion Segmentation Using RANSAC with Priors," *Proceedings of the 2005/2006 International Conference on Dynamical vision*, vol. 4358, pp. 75 – 85, May 2006.

[36] X. Deng, T. Sun, P. Du, and D. Li, "A Nonconvex Implementation of Sparse Subspace Clustering: Algorithm and Convergence Analysis," *IEEE Access*, vol. 8, pp. 54 741 – 54 750, 2020.

[37] J. Yan and M. Pollefeys, "A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate," in *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision*, May 2006.

[38] U. Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, pp. 395 – 416, Jan. 2004.

[39] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of Multivariate Mixed Data via Lossy Data Coding and Compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546 – 1562, 2007.

[40] G. Liu, Z. Lin, and Y. Yu, "Robust Subspace Segmentation by Low-Rank Representation," in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, Aug. 2010, pp. 663 – 670.

[41] V. M. Patel, H. V. Nguyen, and R. Vidal, "Latent Space Sparse Subspace Clustering," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 225 – 232.

[42] J. Carvalho, M. Marques, and J. P. Costeira, "Recovery of Subspace Structure from High-Rank Data with Missing Entries," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 2010 – 2014.

[43] B. Li, C. Lu, Z. Wen, C. Leng, and X. Liu, "Locality-constrained Nonnegative Robust Shape Interaction Subspace Clustering and Its Applications," *Digital Signal Processing*, vol. 60, pp. 113 – 121, Sept. 2016.

[44] K. Guo, L. Liu, X. Xu, D. Xu, and D. Tao, "GoDec+: Fast and Robust Low-Rank Matrix Decomposition Based on Maximum Correntropy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2323 – 2336, 2018.

[45] R. Dragon, B. Rosenhahn, and J. Ostermann, "Multi-scale Clustering of Frame-to-Frame Correspondences for Motion Segmentation," in *Computer Vision – ECCV 2012*, 2012, pp. 445 – 458.

[46] R. Jain, R. Kasturi, and B. Schunck, *Machine Vision*. McGraw-Hill, 1995, ch. 3, pp. 96 – 104.

[47] Z. Li, J. Guo, L. Cheong, and S. Z. Zhou, "Perspective Motion Segmentation via Collaborative Clustering," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1369 – 1376.

[48] R. Tron and R. Vidal, "A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1 – 8.

[49] Y. Sugaya and K. Kanatani, "Geometric Structure of Degeneracy for Multi-body Motion Segmentation," *Statistical Methods in Video Processing. SMVP 2004. Lecture Notes in Computer Science*, vol. 3247, pp. 13 – 25, May 2004.

[50] R. Vidal and R. Hartley, "Two-View Multibody Structure from Motion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, pp. 214 – 227, Mar. 2008.

[51] OpenCV. (Accessed: 2021, Apr 13). [Online]. Available: https://opencv.org/releases/

[52] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: the KITTI Dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231 – 1237, Sept. 2013.

[53] X. Xu, L. Cheong, and Z. Li, "Motion Segmentation by Exploiting Complementary Geometric Models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 2859 – 2867.

[54] N. Sundaram, T. Brox, and K. Keutzer, *Dense Point Trajectories by GPU-Accelerated Large Displacement Optical Flow*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 438 – 451.

[55] UdG-MS19. (Accessed: 28 Mar. 2020). [Online]. Available: http://dixie.udg.edu/udgms/UdG-MS29/

[56] Visionlab. Code. (Accessed: 1 Jun. 2020). [Online]. Available: http://vision.jhu.edu/code/

[57] K. Karsch, C. Liu, and S. B. Kang, "Depth Transfer: Depth Extraction from Video Using Non-Parametric Sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2144 – 2158, 2014.

[58] L. Song and W. Luo, "Self-Supervised Learning of Visual Odometry," in *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, 2020, pp. 5 – 9.

[59] J. Hur and S. Roth, "Self-Supervised Monocular Scene Flow Estimation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7394–7403.

[60] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society B*, vol. 58, no. 1, pp. 267 – 288, 1996.

[61] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. (Accessed: 10 Jul. 2020). [Online]. Available: http://cvxr.com/cvx

[62] Michael Grant and Stephen Boyd, "Graph Implementations for Nonsmooth Convex Programs," in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer-Verlag Limited, 2008, pp. 95 – 110.

[63] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[64] B. Finkston. Mean Shift Clustering, MATLAB Central File Exchange. (2020, October 26). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering

[65] E. Schubert, J. Sander, M. Ester, H. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, pp. 1 – 21, Jul. 2017.

[66] R. Toldo and A. Fusiello, "Robust Multiple Structures Estimation with J-Linkage," in *Proceedings of the 10th European Conference on Computer Vision*, vol. 5302, Oct. 2008, pp. 537 – 547.

[67] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1 – 122, Jan. 2011.