

Robust Motion Segmentation with Subspace Constraints

Pan Ji

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

November 2016

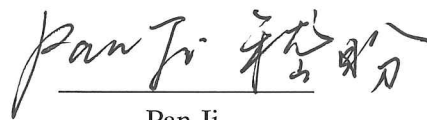
© Copyright by Pan Ji 2016

All Rights Reserved

Declaration

I hereby declare that this thesis is my own original work, except where due references are made in the text. This thesis has not been previously submitted by me in whole or part for a degree or diploma in any university or other tertiary education institution. The content of this thesis is mainly based on the publications during my PhD as listed below.

1. Pan Ji, Hongdong Li, Mathieu Salzmann, and Yiran Zhong. Robust multi-body feature tracker: A segmentation-free approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Spotlight (short oral) (spotlight acceptance rate: 9.7%).
2. Pan Ji, Mathieu Salzmann, and Hongdong Li. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. Oral (oral acceptance rate: 3.3%).
3. Pan Ji, Hongdong Li, Mathieu Salzmann, and Yuchao Dai. Robust motion segmentation with unknown correspondences. In *European Conference on Computer Vision (ECCV)*, 2014. Poster (acceptance rate: 29.0%).
4. Pan Ji, Yiran Zhong, Hongdong Li, and Mathieu Salzmann. Null space clustering with applications to motion segmentation and face clustering. In *IEEE International Conference on Image Processing (ICIP)*, 2014. Best Student Paper Award (3 out of over 2600).
5. Pan Ji, Mathieu Salzmann, and Hongdong Li. Efficient dense subspace clustering. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014. Short oral and poster (acceptance rate: 40.0%).



Pan Ji

23 November 2016

to my family

Acknowledgments

This thesis would never have been possible without the support of the ANU University Research Scholarship (International) and the ANU HDR Merit Scholarship that were awarded to me in 2013. I would like to thank ANU for creating a tremendous place to study, and for funding me to attend several international conferences. I also wish to express my gratitude to the computer vision groups at ANU and NICTA (or now Data61) for regularly holding wonderful academic seminars, which benefitted my researches in many ways.

I owe special thanks to my co-supervisors As/Pro. Hongdong Li and Dr. Mathieu Salzmann, for guiding me with patience and wisdom throughout this PhD journey. Over the past 3.5 years, they have always been generous with their time, knowledge and ideas, and allowed me great freedom in my research.

I would like to thank Dr. Xuming He, my advisor on the supervisory panel, for always being approachable and offering helpful advice and encouragements. I am also grateful to Dr. Yuchao Dai for having a continuous exchange of ideas with me, and for kindly sharing a room in ICCV15 Santiago for free.

Last but not least, I want to thank my family in China for all the support over the years, and my wife for her accompany in this largely solitary journey.

Abstract

Motion segmentation is an important task in computer vision with many applications such as dynamic scene understanding and multi-body structure from motion. When the point correspondences across frames are given, motion segmentation can be addressed as a subspace clustering problem under an affine camera model. In the first two parts of this thesis, we target the general subspace clustering problem and propose two novel methods, namely Efficient Dense Subspace Clustering (EDSC) and the Robust Shape Interaction Matrix (RSIM) method.

Instead of following the standard compressive sensing approach, in EDSC we formulate subspace clustering as a Frobenius norm minimization problem, which inherently yields denser connections between data points. While in the noise-free case we rely on the self-expressiveness of the observations, in the presence of noise we recover a clean dictionary to represent the data. Our formulation lets us solve the subspace clustering problem efficiently. More specifically, for outlier-free observations, the solution can be obtained in closed-form, and in the presence of outliers, we solve the problem by performing a series of linear operations. Furthermore, we show that our Frobenius norm formulation shares the same solution as the popular nuclear norm minimization approach when the data is free of any noise.

In RSIM, we revisit the Shape Interaction Matrix (SIM) method, one of the earliest approaches for motion segmentation (or subspace clustering), and reveal its connections to several recent subspace clustering methods. We derive a simple, yet effective algorithm to robustify the SIM method and make it applicable to real-world scenarios where the data is corrupted by noise. We validate the proposed method by intuitive examples and justify it with the matrix perturbation theory. Moreover, we show that RSIM can be extended to handle missing data with a Grassmannian gradient descent method.

The above subspace clustering methods work well for motion segmentation, yet they require that point trajectories across frames are known *a priori*. However, finding point correspondences is in itself a challenging task. Existing approaches tackle the correspondence estimation and motion segmentation problems separately. In the third part of this thesis, given a set of feature points detected in each frame of the sequence, we develop an approach which simultaneously performs motion segmentation and finds point correspondences across the frames. We formulate this problem in terms of Partial Permutation Matrices (PPMs) and aim to match feature descriptors while simultaneously encouraging point trajectories to satisfy subspace constraints. This lets us handle outliers in both point locations and feature appearance. The resulting optimization problem is solved via the Alternating Direction Method of Multipliers

(ADMM), where each subproblem has an efficient solution. In particular, we show that most of the subproblems can be solved in closed-form, and one binary assignment subproblem can be solved by the Hungarian algorithm.

Obtaining reliable feature tracks in a frame-by-frame manner is desirable in applications such as online motion segmentation. In the final part of the thesis, we introduce a novel multi-body feature tracker that exploits a multi-body rigidity assumption to improve tracking robustness under a general perspective camera model. A conventional approach to addressing this problem would consist of alternating between solving two subtasks: motion segmentation and feature tracking under rigidity constraints for each segment. This approach, however, requires knowing the number of motions, as well as assigning points to motion groups, which is typically sensitive to motion estimates. By contrast, we introduce a segmentation-free solution to multi-body feature tracking that bypasses the motion assignment step and reduces to solving a series of subproblems with closed-form solutions.

In summary, in this thesis, we exploit the powerful subspace constraints and develop robust motion segmentation methods in different challenging scenarios where the trajectories are either given as input, or unknown beforehand. We also present a general robust multi-body feature tracker which can be used as the first step of motion segmentation to get reliable trajectories.

Keywords: Motion segmentation, subspace clustering, feature tracking

Contents

Acknowledgments	vii
Abstract	ix
1 Introduction	1
1.1 Related Work	2
1.1.1 Two-frame Motion Segmentation Methods	3
1.1.2 Multi-frame Motion Segmentation Methods	4
1.2 Contributions	9
1.3 Thesis Structure	11
2 Preliminaries	13
2.1 Basic Notations	13
2.2 Basic Concepts	14
2.2.1 Subspace	14
2.2.1.1 Linear Subspace and Affine Subspace	14
2.2.1.2 Null Space, Row Space and Column Space	15
2.2.1.3 Subspace Clustering	16
2.2.2 Grassmann Manifold	16
2.3 Subspace Formulation for Motion Segmentation	17
2.3.1 Affine Camera Model	17
2.3.2 Factorization	18
2.3.3 Self-expressiveness	20
2.3.4 Spectral Clustering	22
2.3.4.1 Unnormalized Spectral Clustering	22
2.3.4.2 Normalized Spectral Clustering	24
2.4 Optimization Methods	25
2.4.1 Alternating Direction Method of Multipliers	25
2.4.1.1 Lagrangian Dual Ascent Method	25
2.4.1.2 Augmented Lagrangian	26
2.4.1.3 A Concrete Example	27
2.4.2 Hungarian Algorithm	29

3	Efficient Dense Subspace Clustering	33
3.1	Problem Analysis	34
3.2	Frobenius Norm Formulation	36
3.2.1	A Null Space Perspective	39
3.2.2	Dealing with Noise	40
3.2.3	Dealing with Outliers	42
3.3	Affinity Matrix and Normalized Cuts	43
3.4	Experiments	44
3.4.1	Hopkins155: Motion Segmentation	44
3.4.2	Extended Yale B: Face Clustering	46
3.5	Summary	47
4	The Robust Shape Interaction Matrix Method	49
4.1	SIM Revisited: Review and Analysis	50
4.1.1	The Pre-Spectral-Clustering Era	52
4.1.2	The Post-Spectral-Clustering Era	52
4.2	SIM Robustified: Corrupted Data	53
4.2.1	Row Normalization	54
4.2.2	Elementwise Powering	55
4.2.3	Rank Determination	56
4.2.4	Robust Shape Interaction Matrix	58
4.3	SIM Robustified: Missing Data	58
4.4	Experimental Evaluation	61
4.4.1	Hopkins155: Complete Data with Noise	63
4.4.2	Extended Yale B: Complete Data with Outliers	63
4.4.3	Hopkins12Real: Incomplete Data with Noise	64
4.4.4	Hopkins Outdoor: Semi-dense, Incomplete Data with Outliers	65
4.5	Summary	66
5	Robust Motion Segmentation with Unknown Correspondences	73
5.1	Background	74
5.2	Problem Formulation with PPMs	76
5.2.1	Solving (5.6) via ADMM	78
5.3	Experimental Evaluation	81
5.3.1	Experiment-1: Synthetic Data, Noise-free Case	82
5.3.2	Experiment-2: Synthetic data, with Noise and Outliers	83
5.3.3	Experiment-3: Real images, Hopkins155 dataset	84
5.3.4	Experiment-4: Real images, Other Real Sequences	86
5.4	Summary	87

6	Robust Multi-body Feature Tracker: A Segmentation-free Approach	89
6.1	Background	90
6.2	Multi-body Feature Tracker	92
6.2.1	A First Attempt: Segmentation-based Tracking	92
6.2.2	Our Segmentation-free Approach	93
6.2.2.1	Epipolar Subspace Constraints	93
6.2.2.2	Approximation and Problem Reformulation	95
6.2.2.3	ADMM Solution	97
6.2.2.4	Our Complete Multi-body Feature Tracker	100
6.3	Experiments	100
6.3.1	Hopkins Checkerboard Sequences	102
6.3.2	Hopkins Car-and-People Sequences	103
6.3.3	MTPV Sequences	104
6.3.4	KITTI Sequences	104
6.3.5	Frame-by-Frame Motion Segmentation	105
6.4	Summary	107
7	Conclusions and Future Work	109
7.1	Conclusions	109
7.2	Future Work	110

List of Figures

1.1	Left (a) Human can effortlessly tell which parts of the image belong to the same object. Middle (b) Superpixels generated by the method of Achanta et al. [2012]. Right (c) Motion vectors of a car sequence from Tron and Vidal [2007]: The red points denote the current positions of the feature points, and green lines the motion since the previous frame.	2
1.2	Multi-frame motion segmentation with feature tracks over frames. . .	5
2.1	A toy example of subspace clustering. Three subspaces are contained in this example: two lines (L_1 , L_2), and one plane (P_1).	15
3.1	A toy example for subspace clustering: two lines L_1 (with a point p) and L_2 passing through the origin o form two independent linear subspaces.	36
3.2	Recovering a clean dictionary from noisy data with EDSC. (a) Data corrupted by Gaussian noise; (b) Optimal solution $\mathbf{C}^*$ of (3.14); (c) Recovered clean dictionary $\mathbf{D}$	41
3.3	Recovering a clean dictionary in the presence of outliers with EDSC. (a) Data corrupted by noise and outliers; (b) Optimal solution $\mathbf{C}^*$ of (3.16); (c) Recovered clean dictionary $\mathbf{D}$	44
3.4	Sample images from the Hopkin155 dataset including indoor checker-board sequences, outdoor traffic sequences, non-rigid and articulated sequences.	45
3.5	Sample face images from the extended yale B face dataset: each row contains aligned face images of the same subject under different illumination conditions	47
3.6	Average runtimes of the algorithms on Extended Yale B as a function of the number of subjects.	48
4.1	Subspace clustering example: (a) Two motions, each forming one subspace; (b) Shape Interaction Matrix of the trajectories in (a), which is sensitive to noise; (c) Affinity matrix obtained by our method: a much clearer block-diagonal structure.	50

4.2	SIM for clustering two lines in 3D: (a) Two lines (each forming one subspace) with an arbitrary angle; (b) New data representation with $\mathbf{V}_r$. Note that the lines have become orthogonal; (c) SIM (absolute value) normalized by its maximum value; the darker the SIM image, the greater the value.	51
4.3	Clustering two lines in 3D: Row normalization (a) Two lines (each forming one subspace) with an arbitrary angle; (b) New data representation with row-normalized $\mathbf{V}_r$. Note that the lines collapse four points on the unit circle, corresponding to orthogonal vectors; (c) New SIM (absolute value) without magnitude bias.	55
4.4	Clustering two lines with noise in 3D: (a) Two lines with Gaussian noise; (b) New SIM after row normalization, with noise in the off-diagonal blocks; (c) Affinity matrix after elementwise powering. Note that the block-diagonal structure is much cleaner.	57
4.5	Comparison of SSC-R and RSIM-M on semi-dense data: While SSC-R removes the trajectories with missing entries, and thus gets less dense results, our method can handle missing data robustly. Each image is a frame sampled from one of the video sequences. The points marked with the same color are clustered into the same group by the respective methods. Best viewed in color.	66
4.6	Typical behavior of SSC-O on semi-dense data: By treating missing entries as outliers, SSC-O tends to cluster the trajectories with missing entries into same group. The points marked with the same color are clustered into the same group by SSC-O. Best viewed in color.	66
4.7	Failure cases of SSC-R and of RSIM-M: We conjecture that these failures are due to tracking failures (e.g., very few trajectories), or to high dependence between motions. Best viewed in color.	67
4.8	Semi-dense motion segmentation results for sequences 1-4. Our method (RSIM-M) uses all the available tracks (3026 on average) with an average runtime of 5.22 seconds per sequence ; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.	68
4.9	Semi-dense motion segmentation results for sequences 5-10. Our method (RSIM-M) uses all the available tracks (3026 on average) with an average runtime of 5.22 seconds per sequence ; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.	69

4.10	Semi-dense motion segmentation results for sequences 11-16. Our method (RSIM-M) uses all the available tracks (3026 on average) with an average runtime of 5.22 seconds per sequence ; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.	70
4.11	Semi-dense motion segmentation results for sequences 17-18. Our method (RSIM-M) uses all the available tracks (3026 on average) with an average runtime of 5.22 seconds per sequence ; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.	71
5.1	Motion segmentation with unknown correspondences: given a video sequence and feature points in each frame, our goal is to simultaneously estimate motion clusters and feature matching.	74
5.2	(a) Convergence curves for the objective function value and the primal residuals. Blue curves: initialization (V1); red curves: initialization (V2). (b) Accuracy of motion segmentation by adding different amounts of outliers and Gaussian noise.	84
5.3	Motion segmentation results of different algorithms for the cars2_07_g12 sequence. Points marked with the same color and marker ($\circ$ or $\times$) are from the same motion: (a) SIFT + SSC; (b) RE + SSC (RE: ROML with embedded feature); (c) RS + SSC (RS: ROML with SIFT feature); (d) Our algorithm. Best viewed in color.	85
5.4	Motion segmentation results of the airport sequence: Points marked with the same color and marker ($\circ$ or $\times$) are from the same motion. Best viewed in color.	87
6.1	Performance of different trackers on the 1RT2TC checkerboard sequence: The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. Best viewed zoomed-in on screen.	102
6.2	Performance of different trackers on the car1 sequence: The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. Best viewed in color. . . .	103
6.3	Tracking error as a function of the frame number: In these two typical sequences, our method consistently outperforms the baselines, and is less prone to tracking drift over time.	104

6.4	The MAN and MONK sequences of the MTPV dataset: The feature points are marked as red. Note that the number of points on the walking man and monk is much smaller than on the background. . . .	105
6.5	Performance of different trackers on the KITTI sequence: The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. As evidenced by the regions highlighted with a blue rectangle, L1-KLT and BFT make more tracking errors than our approach. Best viewed on screen with zoom-in.	106

List of Tables

3.1	Clustering error (in %) on Hopkins155.	46
3.2	Clustering error (in %) on Extended Yale B.	48
4.1	Clustering error (in %) on Hopkins 155.	63
4.2	Ablation study on Hopkins 155.	63
4.3	Clustering error (in %) on Extended Yale B.	64
4.4	Clustering error (in %) on Hopkins 12 Real Motion Sequences with Incomplete Data.	65
5.1	Average motion segmentation and point correspondence accuracies (%) on the Hopkins155 27 non-checkerboard sequences.	86
5.2	Average motion segmentation and point correspondence accuracies (%) on the Hopkins155 checkerboard sequences with at most 200 trajectories.	86
6.1	Average number of tracking errors ($\epsilon = 10$) on the Hopkins checkerboard sequences with noise of different variances σ^2 . The lower, the better.	103
6.2	Average number of tracking errors ($\epsilon = 5$) on the Hopkins Car-and-People sequences with noise of different variances σ^2 . The lower, the better.	103
6.3	Average number of tracking errors ($\epsilon = 5$) on the MTPV sequences with noise of different variances σ^2 . The lower, the better.	105
6.4	Average number of tracking errors ($\epsilon = 5$) on the KITTI sequences with different noise variances σ^2 . The lower, the better.	105
6.5	Average error rate (in %) of two-frame motion segmentation on the 22 Hopkins sequences with noise of different variances σ^2 . The lower, the better.	107

Introduction

Making a computer see as a human is the ultimate goal of computer vision research. Consider Figure 1.1(a) for the task of *object-level segmentation*. It takes no effort for us humans to tell which parts of the image belong to the same object. However, this remains a difficult task for a computer. Top-down learning based methods can get object-level segmentation, but the training phase requires numerous manually annotated images, which are normally expensive to acquire. Moreover, from the perspective of human vision, infants can learn to see without supplying them large amounts of carefully annotated images. In the same spirit, Geoffrey Hinton, a famous professor of machine learning, once said:

When we're learning to see, nobody's telling us what the right answers are – we just look. Every so often, your mother says "that's a dog", but that's very little information. You'd be lucky if you got a few bit of information – even one bit per second – that way. The brain's visual system has 10^{14} neural connections. And you only live for 10^9 seconds. So it's no use learning one bit per second. You need more like 10^5 bits per second. And there's only one place you can get that much information: from the input itself. – Geoffrey Hinton, 1996 (quoted in Murphy [2012]).

Modern bottom-up (unsupervised) segmentation methods rely on the information of the input itself, or more specifically the color and texture of the input, but only generate over-segmented small homogeneous regions, a.k.a. superpixels. For example, in Figure 1.1(b), with one of the state-of-the-art bottom-up segmentation methods, the car is segmented into multiple non-overlapping superpixels rather than a single object.

We come into a dilemma here: on one hand, we do not want to rely on costly annotated training data for the task of object segmentation; on the other hand, unsupervised bottom-up methods such as superpixel segmentation methods, in most cases, cannot produce satisfactory object-level segmentation due to intrinsic ambiguities.

However, if we are given a video sequence and the objects in the sequence move for a period of time, these ambiguities can be easily resolved by leveraging constraints of temporal trajectories. Indeed, motion vectors are typically known to be more consistent than color and texture within an object region (Ochs et al. [2014]), especially



Figure 1.1: **Left** (a) Human can effortlessly tell which parts of the image belong to the same object. **Middle** (b) Superpixels generated by the method of Achanta et al. [2012]. **Right** (c) Motion vectors of a car sequence from Tron and Vidal [2007]: The red points denote the current positions of the feature points, and green lines the motion since the previous frame.

for rigid motions. For instance, in Figure 1.1(c), while color and texture are rather diverse within the car, the motion vectors of the feature points on the moving car have similar patterns.

Therefore, in this thesis, we focus on the study of motion segmentation, an important step towards object segmentation using unsupervised learning. In particular, the goal of motion segmentation is to group the pixels (or feature points) of a dynamic video sequence according to their motions. This is an important problem because its outcomes can be beneficial for many related areas in computer vision, such as scene understanding, video analysis, and multi-body structure from motion.

Generally, motion segmentation methods deal with feature tracks obtained by some feature tracking methods such as the Kanade-Lucas-Tomasi (KLT) feature tracker and optical flow. However, due to the drifting and occlusion problems in feature tracking, these feature tracks inevitably contain noise, outliers, or missing entries. One of our goals then is to design novel motion segmentation algorithms that are not only robust to these data corruptions, but also efficient in computations. We also consider the scenarios where the feature tracks are not available and aim to design algorithms that are able to solve for motion segmentation with unknown feature correspondences. In addition, we explore the possibility of improving current feature tracking methods by using additional scene constraints.

Before presenting the contributions of this thesis, we first review the related work on motion segmentation in literature.

1.1 Related Work

Over the years, many methods have been devoted to segmenting dynamic objects from images. For example, background subtraction (Piccardi [2004]) can be thought of as motion segmentation. However, with only a few exceptions (e.g. Sheikh et al. [2009]), they often assume a stationary camera and aim to identify the moving foreground

objects. By contrast, no special constraint on the camera motion is enforced for motion segmentation. Object tracking (Yilmaz et al. [2006]) is another related research area which tracks moving objects in the video with bounding boxes. In most cases, however, it requires manually specifying the objects to be tracked in the first frame. With a focus on motion segmentation in this thesis, we, therefore, do not discuss background subtraction and object tracking in more details.

Motion segmentation methods naturally fall into two categories: two-frame motion segmentation methods, and multi-frame motion segmentation methods.

1.1.1 Two-frame Motion Segmentation Methods

Most methods in this category rely on optical flow (Horn and Schunck [1981]; Brox et al. [2004]; Amiaz and Kiryati [2006]), which provides dense point correspondences between two consecutive frames. Early attempts at motion segmentation searched for discontinuities in the optical flow field (Thompson et al. [1985]; Black and Anandan [1990]), or piecewise affine partitions of the flow field (Adiv [1985]; Nagel et al. [1994]). These methods suffered from the fact that the optical flow at discontinuities is hard to recover due to occlusion.

Weber and Malik [1997] used the fact that each independently moving object corresponds to a unique epipolar constraint, or equivalently a unique fundamental matrix. The motion segmentation problem was formulated as minimizing a cost functional consisting of a data term that fits to the fundamental matrix and a discontinuity penalty term for spatial smoothness. The segmentation problem was solved via region-growing.

Klappstein et al. [2009] discussed the detection of moving objects in the context of driver assistance systems and road safety. Multiple constraints (i.e., epipolar constraints, positive depth constraints, positive height constraints, and trifocal constraints) were used to compute a motion metric reflecting how likely a feature point to be moving.

Narayana et al. [2013] assumed that only translational motions were involved. Instead of using the complete flow vectors, they leveraged optical flow orientations for dense motion segmentation. However, in practice, the pure translational motion assumption is often too restrictive and does not hold as long as either the object or the camera motions are rotational.

Other methods (Brox et al. [2006]; Amiaz and Kiryati [2005]; Cremers and Soatto [2005a]; Mémín and Pérez [2002]; Brox and Weickert [2006]) tried to simultaneously solve for optical flow and segmentation with the variational method and the level-set based motion competition technique (Zhu and Yuille [1996]). The flow was estimated independently for each segmented region, and the region segmentation represented by level sets was driven by the fitting error of the optical flow.

It is often the case that optical flow based methods lead to dense object segmenta-

tion. However, they also suffer from the difficulties in estimating very accurate optical flow, especially in texture-less and occluded regions.

A few other two-frame motion segmentation methods rely on sparse feature correspondences between two views. For example, Li [2007] used a mixture-of-fundamental-matrices model to describe the multi-body motions from two views, and with a random sampling scheme (Fischler and Bolles [1981]), solved the problem via a linear program relaxation. In essence, this method tried to solve a chicken-and-egg problem, and alternated between solving for multiple fundamental matrices and updating the motion assignment in an Expectation Maximization (EM) manner, which is sensitive to initialization.

Vidal et al. [2006] derived a multi-body epipolar constraint and its associated multi-body fundamental matrix from noise-free sparse point correspondences of two perspective views. The epipolar lines and epipoles were then computed using an algebraic geometric approach. Given the epipolar lines and the associated multiple fundamental matrices, the clustering was obtained by assigning each point to its nearest fundamental matrix based on the Sampson distance. As pointed out by the authors, this approach was mostly designed for noise-free correspondences and is thus sensitive to noise and outliers.

On a frame-to-frame level, Dragon et al. [2012] employed a motion-split-and-merge approach that consists of splitting the tracklets with J-Linkage (Toldo and Fusiello [2008]) until the segments are consistent and merging with neighbouring segments until the algorithm converges. This method was improved for real time performance by Dragon et al. [2013]. Elqursh and Elgammal [2013] formulated motion segmentation as a manifold separation problem and presented an online method using label propagation on a dynamically changing graph.

All in all, two-frame methods suffer from the fact that motions between consecutive frames are relatively small and thus motion segmentation from two frames is often ambiguous.

1.1.2 Multi-frame Motion Segmentation Methods

Multi-frame motion segmentation methods are generally more robust than two-frame methods. Most of the recent works on motion segmentation belong to this class. Typically, point trajectories over frames are assumed to be given *a priori* from the KLT tracker or more recent feature tracking methods (e.g., Sundaram et al. [2010]). See Figure 1.2 for an example. Then the affinities between trajectories are constructed in a way that those from the same motion have high affinity and those from different motions have low affinity (or ideally zero affinity). With the affinity matrix, the final motion segmentation is obtained by the normalized cuts (Shi and Malik [2000]) or spectral clustering (Ng et al. [2002]).

One line of research exploited the affinities from long term point trajectories of pos-

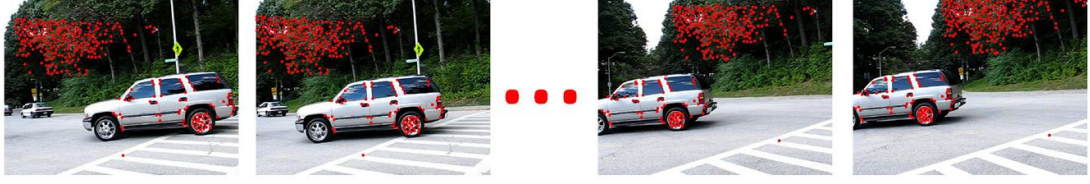


Figure 1.2: Multi-frame motion segmentation with feature tracks over frames.

sibly different length. Brox and Malik [2010] and Ochs et al. [2014] defined affinities with pairwise distances between trajectories, and used a spatially regularized spectral clustering for segmentation. However, the pairwise distance only accounts for the translational model, and thus cannot handle other motion models such as rotation and scaling. A follow-up work by Ochs and Brox [2012] addressed this limitation by using higher order tuples of trajectories to form hypergraphs. To apply spectral clustering, the hypergraph was flattened into an ordinary graph via a nonlinear projection, i.e., a regularized maximum operator. Unfortunately, although sampling techniques were applied, the high order motion models still resulted in high computational complexity.

Other works leveraged explicit camera models, e.g., the perspective camera model or affine camera model, to derive trajectory affinities. Under the perspective camera model, Jung et al. [2014] iteratively computed fundamental matrices and motion labels. Evidence of segmentation was then accumulated across frames via randomized voting based on the Sampson distance between the point and the epipolar line. Li et al. [2013] derived a subspace method from the two-view epipolar constraint, and formulated the motion segmentation from multiple frames as a mixed norm optimization problem. A model selection method was also proposed based on an over-segment-and-merge procedure.

The affine camera model is commonly used as an approximation of the perspective camera model where depth variation of the scene is negligible compared to the overall scene depth. It has been proven that, under the affine camera model, trajectories from the same motion lie in a linear subspace with dimension up to four and trajectories from different motions correspond to different subspaces (Yan and Pollefeys [2006]). Under the affine camera model assumption, the motion segmentation problem can be solved as a subspace clustering problem. Thanks to the useful property of this subspace formulation, a large amount of work follows this direction.

The following paragraphs focus on the literature of *subspace clustering* (Vidal [2011]). These methods can be roughly divided into five categories: factorization based algorithms, algebraic methods, expectation maximization based approaches, high-order model based methods, and self-expressiveness based algorithms.

Factorization Based Algorithms

Factorization based algorithms (Boult and Brown [1991]; Costeira and Kanade [1998]; Gear [1998]), initially introduced for motion segmentation, achieve segmentation by a low-rank factorization of the data matrix $\mathbf{X}$ whose columns represent the data points. To group data into corresponding subspaces, Costeira and Kanade [1998] relied on the Shape Interaction Matrix (SIM), which was defined as $\mathbf{Q} = \mathbf{V}\mathbf{V}^T \in \mathbb{R}^{N \times N}$, with $\mathbf{V}$ the matrix consisting of right singular vectors of $\mathbf{X}$ and N the number of trajectories. The matrix $\mathbf{Q}$ has the property that $\mathbf{Q}_{ij} = 0$ if points i and j lie in different subspaces. Segmentation was achieved by simply block-diagonalizing $\mathbf{Q}$, or by applying spectral clustering.

Kanatani [2001] later analysed motion segmentation as a subspace segmentation problem, and showed that under the condition that the subspaces are linearly independent, the SIM is block-diagonal up to a permutation. Many other methods (Gear [1998]; Ichimura [1999]; Kanatani [2002]; Wu et al. [2001]; Zelnik-Manor and Irani [2003]) were proposed to improve the robustness of the SIM method. Unfortunately, they are sensitive to noise and outliers, and require a reliable estimation of the intrinsic dimension of the data matrix.

Inspired by the theory of non-negative matrix factorization (Lee and Seung [1999]), Cheriadat and Radke [2009] decomposed the velocity profiles of point trajectories into different motion components and corresponding non-negative weights, and then segmented the motions using spectral clustering on the derived weights. Similarly, Mo and Draper [2012] modeled dense point trajectories with a semi-non-negative matrix factorization method to get semantically meaningful motion components.

Algebraic Methods

Algebraic methods, such as Generalized Principal Component Analysis (GPCA) (Vidal et al. [2005a]), fit a set of homogeneous polynomials to the data. The gradient at a data point of such polynomials gives a vector normal to the subspace corresponding to that point. Subspace clustering is then reduced to fitting, differentiating and clustering a set of homogeneous polynomials. GPCA can deal with subspaces of different dimensions, and does not impose any restriction on the relative orientation of the subspaces. However, determining the number of subspaces K and the subspace dimension d_i assumes noise-free and outlier-free data. This drawback was overcome in the more recent Robust Algebraic Segmentation (RAS) (Rao et al. [2010b]) using robust statistics. However, those algebraic methods suffer from the fact that the complexity of fitting polynomials increases exponentially with K and d_i .

Expectation Maximization Based Approaches

Expectation Maximization (EM) based approaches estimate subspace models and assignments iteratively. For example, K -planes (Bradley and Mangasarian [2000]) and K -subspaces (Tseng [2000]) algorithms extended the K -means method (MacQueen et al. [1967]; Duda et al. [2012]) (which is an EM based method) to clustering data sampled from multiple planes or subspaces. The two algorithms iterated over the following two steps until converge: 1) given an initial cluster assignment, fit a subspace/plane for each cluster with PCA; 2) with the PCA basis for each subspace/plane, assign each data point to its nearest subspace/plane. These methods suffer from two drawbacks: first, the problems they solved are nonconvex, so the solution depends on initialization; second, the PCA subspace/plane fitting step is sensitive to outliers.

A few other methods (e.g., Tipping and Bishop [1999]; Ma and Derksen [2007]) addressed the subspace clustering problem within a probabilistic framework. In statistical learning, mixed data are often modeled as samples drawn from mixtures of Gaussian distributions. In this context, subspace clustering becomes a model fitting problem. A major drawback of this approach, however, is that it often results in non-convex optimization problems. One therefore has to resort to an EM method, whose solution strongly depends on initialization.

High-order Model Based Methods

High-order model based methods try to fit a pre-defined model with multiple trajectories (or points). In particular, Local Subspace Affinity (LSA) (Yan and Pollefeys [2006]) fitted a linear subspace at each trajectory with local adjacent trajectory sampling, and then derived the affinity matrix based on the principal subspace angles between these estimated subspaces. Alternatively, LSA can also be interpreted as using the projection distance of points on the Grassmann manifold because a subspace can be viewed as a point on the corresponding Grassmann manifold. However, this local subspace fitting can be unreliable due to various data corruptions, especially for points lying close to subspace intersections.

Chen and Lerman [2009] proposed a Spectral Curvature Clustering (SCC) method using the polar curvature of $d + 2$ distinct points (with d the subspace dimension) to form the affinity tensor. To make it computationally tractable, an iterative sampling procedure was suggested to reduce the affinity tensor to an affinity matrix.

Chin et al. [2009] proposed a new kernel, called the Ordered Residual Kernel (ORK), to elicit the potential of two point trajectories to have emerged from the same subspace. Subspace hypotheses were first obtained by randomly sampling trajectories for subspace model fitting. The subspace fitting residuals were then encapsulated into the ordered residual kernel, which is a valid Mercer kernel derived from the intersection kernel. This method is able to handle severe data corruption of outliers.

Jain and Govindu [2013] presented Sparse Grassmann Clustering (SGC) that utilized the residual error of fitting a high dimensional model (with at least $d + 2$ points) to construct affinities. The efficiency of the algorithm was gained by updating the eigenspace of the affinity matrix via an efficient Grassmannian gradient descent method and iterative sampling.

Similarly, Purkait et al. [2014] also used model fitting residuals to build affinities, but with more points (much more than $d + 2$), thus leading to large hyperedges of hypergraphs. A novel guided sampling strategy for large hyperedges was proposed based on the concept of random cluster models to reduce the sampling costs.

High-order model based methods can be applied to solve general model fitting and clustering problems. However, in many cases, explicit models are required to know *a priori*, and the computational complexity still remains to be a limiting factor of this type of methods.

Self-expressiveness Based Algorithms

Recently, self-expressiveness based algorithms (Elhamifar and Vidal [2009]; Liu et al. [2010]; Favaro et al. [2011]; Elhamifar and Vidal [2010]; Liu et al. [2013]; Elhamifar and Vidal [2013]) have received much attention in the research community. The data drawn from multiple subspaces are deemed self-expressive in the sense that a data point in a subspace can be represented as a linear combination of the other points in the same subspace. The affinity matrix can then be constructed by the linear combination coefficient matrix. Relating back to the high-order model based methods, by self-expressiveness, a point (or trajectory) is in fact connected to all the other points simultaneously, which can be thought of as virtually applying the highest-possible-order model. This, to some extent, explains the recent popularity and success of this kind of methods.

The most popular approaches in this category were inspired by the theory of compressive sensing. In particular, Sparse Subspace Clustering (SSC) (Elhamifar and Vidal [2009]) employed the data matrix as a dictionary to reconstruct the data itself. It then utilized the coefficients recovered by an ℓ_1 -norm minimization problem to construct an affinity matrix whose non-zero entries correspond to points in the same subspace. Similarly, Low Rank Representation (LRR) (Liu et al. [2010]) exploited the self-expressiveness of the data, but, instead of sparsity, assumed that the coefficient matrix has low rank.

Peng et al. [2013] made SSC scalable with a sampling-clustering-coding-and-classifying strategy. Spatial constraints were incorporated by Pham et al. [2012] to improve the performance of SSC. Tierney et al. [2014] considered the ordered subspace clustering for sequential data. Gao et al. [2015] dealt with the multi-view scenario and used a common cluster structure to guarantee coherence among different views. Wang et al. [2013] combined SSC and LRR, and presented a low-rank sparse subspace

clustering method. There are also several non-linear extensions of SSC, such as (Patel et al. [2013]), (Patel and Vidal [2014]) and (Yin et al. [2016]).

While both SSC and LRR solved the problem formulations in an iterative manner, a closed-form solution was introduced in Low-Rank Subspace Clustering (LRSC) (Favaro et al. [2011]), which, for noisy data, involved performing a series of singular value thresholding operations. To derive this closed-form solution, LRSC relied on recovering a clean dictionary instead of directly making use of the noisy data.

Other types of regularization on the self-expressiveness include using the quadratic programming (Wang et al. [2011b]), the latent low rank representation (Liu and Yan [2011]), the least square regression (Lu et al. [2012]), the trace lasso (Lu et al. [2013]), the half-quadratic minimization (Zhang et al. [2013]), the block-diagonal prior (Feng et al. [2014]), the smooth representation (Hu et al. [2014]), a structured norm (Li and Vidal [2015]), the Gaussian mixture regression (Li et al. [2015]) and the elastic net (You et al. [2016]).

The nice property of this class of methods is that in most cases, the resulting problem formulations are convex, so global optimal solutions can be achieved. However, to apply self-expressiveness for motion segmentation, these methods require the point trajectories to be of the same length and thus cannot naturally handle the case of missing data, whose presence is inevitable in real-world scenarios.

1.2 Contributions

This thesis focuses on *multi-frame* motion segmentation with subspace constraints and aims to handle various types of data corruption in a robust and efficient manner. The contributions are three-fold: first, we present novel subspace clustering algorithms that can be directly applied to motion segmentation; second, we tackle the motion segmentation problem under unknown feature correspondences and give a solution that simultaneously performs motion segmentation and feature matching; third, we develop a segmentation-free multi-body feature tracking method using the subspace constraint derived from epipolar geometry.

Novel Subspace Clustering Algorithms

Two novel subspace clustering methods are presented in this thesis, namely Efficient Dense Subspace Clustering (EDSC), and the Robust Shape Interaction Matrix (RSIM) method.

Instead of following the standard compressive sensing approach, in EDSC, we formulate subspace clustering as a Frobenius norm minimization problem, which inherently yields denser connections between the data points. While in the noise-free case we rely on the self-expressiveness of the observations, in the presence of noise we recover a clean dictionary to represent the data. Our formulation lets us solve

the subspace clustering problem efficiently. More specifically, the solution can be obtained in closed-form for outlier-free observations, and by performing a series of linear operations in the presence of outliers. Furthermore, we show that our Frobenius norm formulation shares the same solution as the popular nuclear norm minimization approach when the data is free of any noise.

In RSIM, we revisit the Shape Interaction Matrix (SIM) method, one of the earliest approaches for motion segmentation (or subspace clustering), and reveal its connections to several recent subspace clustering methods. We derive a simple, yet effective algorithm to robustify the SIM method and make it applicable to real-world scenarios where the data is corrupted by noise. We validate the proposed method by intuitive examples and justify it with the matrix perturbation theory. Moreover, we show that RSIM can be extended to handle missing data with a Grassmannian gradient descent method.

Both the above-mentioned subspace clustering algorithms (i.e., EDSC and RSIM) can be applied to motion segmentation with robustness to data noise, outliers, and missing data, which is evidenced by our experiments on standard motion segmentation datasets, such as Hopkins 155 (Tron and Vidal [2007]).

Motion Segmentation with Unknown Correspondences

Motion segmentation can be addressed as a subspace clustering problem, assuming that the trajectories of interest points are known. However, establishing point correspondences is in itself a challenging task. Most existing approaches tackle the correspondence estimation and motion segmentation problems separately. In this part, we develop an approach to performing motion segmentation without any prior knowledge of point correspondences. We formulate this problem in terms of Partial Permutation Matrices (PPMs) and aim to match feature descriptors while simultaneously encouraging point trajectories to satisfy subspace constraints. This lets us handle outliers in both point locations and feature appearance. The resulting optimization problem is solved via the Alternating Direction Method of Multipliers (ADMM), where each subproblem has an efficient solution. In particular, we show that most of the subproblems can be solved in closed-form, and one binary assignment subproblem can be solved by the Hungarian algorithm.

Robust Multi-body Feature Tracker

Obtaining reliable feature tracks in a frame-by-frame manner is desirable in applications such as online motion segmentation. In this part, we introduce a novel multi-body feature tracker that exploits a multi-body rigidity assumption to improve tracking robustness under a general perspective camera model. A conventional approach to addressing this problem would consist of alternating between solving two subtasks: motion segmentation and feature tracking under rigidity constraints for each segment.

This approach, however, requires knowing the number of motions, as well as assigning points to motion groups, which is typically sensitive to motion estimates. By contrast, we introduce a segmentation-free solution to multi-body feature tracking that bypasses the motion assignment step and reduces to solving a series of subproblems with closed-form solutions.

Summary of Contributions

In summary, in this thesis, we exploit the powerful subspace constraints and present motion segmentation methods that are robust to various types of data corruptions in the cases of known or unknown feature correspondences. In addition, we also propose a robust feature tracker for dynamic scenes, which can be applied to get reliable trajectories for motion segmentation.

1.3 Thesis Structure

Following this introduction chapter, in Chapter 2, we give the preliminaries of some mathematical concepts and optimization methods that are used in our work. Readers who are familiar with related concepts and methods may skip this chapter. After that, in Chapter 3, we develop an efficient dense subspace clustering method based on the self-expressiveness property of the data. In Chapter 4, we revisit the 20-year-old shape interaction matrix method and derive a simple method to make it robust to various types of data corruption, and both empirical and theoretical justifications are provided. Chapter 5 presents an approach for motion segmentation without any prior knowledge of point correspondences, and an ADMM solution is provided. In Chapter 6, we consider the prerequisite step of motion segmentation, i.e., feature tracking, and propose a robust segmentation-free feature tracker for the multi-body motion case. Finally, in Chapter 7, we draw conclusions based on the observations made in the previous chapters, and provide future research directions.

Preliminaries

2.1 Basic Notations

Throughout this thesis, we denote matrices by bold upper case letters, vectors by bold lower case letters, and scalars by non-bold letters. Upper case non-bold letters are also used to denote a space, subspace or field. $\mathbb{R}$, $\mathbb{R}^N$, and $\mathbb{R}^{D \times N}$ are respectively the set of real numbers, the set of real N -dimensional vectors, and the set of real D -by- N matrices.

We organize the point trajectories as the columns of the data matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$, and the i^{th} column (or trajectory) of $\mathbf{X}$ is represented by $\mathbf{x}_i$. The transpose of $\mathbf{X}$ is denoted by $\mathbf{X}^T$. The vectorization of $\mathbf{X}$, denoted by $\text{vec}(\mathbf{X})$, is obtained by stacking the columns of $\mathbf{X}$, i.e., $\text{vec}(\mathbf{X}) = [\mathbf{x}_1^T \cdots \mathbf{x}_N^T]^T$. The identity matrix is denoted by $\mathbf{I}$, and the all-one column vector is denoted by $\mathbf{1}$. The full Singular Value Decomposition (SVD) of $\mathbf{X}$ is expressed as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ with $\mathbf{U} \in \mathbb{R}^{D \times p}$ ($p = \min(D, N)$) and $\mathbf{V} \in \mathbb{R}^{p \times N}$ being orthonormal matrices, and $\Sigma \in \mathbb{R}^{p \times p}$ the diagonal matrix whose diagonal elements are the singular values of $\mathbf{X}$. If $\mathbf{X}$ is low-rank with the rank $r < \min(D, N)$, its compact SVD can be written as $\mathbf{X} = \mathbf{U}_r \Sigma_r \mathbf{V}_r^T$, where $\mathbf{U}_r$ contains the first r columns of $\mathbf{U}$, $\mathbf{V}_r$ the first r columns of $\mathbf{V}$, and Σ_r the first r singular values on its diagonal.

Vector or matrix norms are commonly used. For a vector $\mathbf{x}$, its ℓ_0 norm $\|\mathbf{x}\|_0$ is defined as the number of non-zero elements of $\mathbf{x}$; its ℓ_p norm $\|\mathbf{x}\|_p$ (for $p = 1, 2, \dots$) is defined as $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{1/p}$, e.g. $\|\mathbf{x}\|_1 = \sum_i |x_i|$; its ℓ_∞ norm is defined as the maximal element of $|\mathbf{x}|$, i.e. $\|\mathbf{x}\|_\infty = \max_i |x_i|$. By default, $\|\mathbf{x}\|$ denotes the ℓ_2 norm of $\mathbf{x}$. A matrix norm is a natural extension of a vector norm. Specifically, for element-wise matrix norms, $\|\mathbf{X}\|_1 = \sum_{i,j} |x_{ij}|$, the Frobenius norm $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j} x_{ij}^2} = \sqrt{\text{trace}(\mathbf{X}^T \mathbf{X})}$, the ℓ_p norm $\|\mathbf{X}\|_p = (\sum_{i,j} |x_{ij}|^p)^{1/p}$, and the $\ell_{2,1}$ norm $\|\mathbf{X}\|_{2,1} = \sum_j \|\mathbf{x}_j\| = \sum_j \sqrt{\sum_i x_{ij}^2}$. Another widely used norm is the nuclear norm (also known as trace norm), which is defined as $\|\mathbf{X}\|_* = \sum_i \sigma_i$ with σ_i the i^{th} singular value of $\mathbf{X}$.

The pseudoinverse (also known as Moore-Penrose pseudoinverse) of $\mathbf{X}$ is denoted by $\mathbf{X}^\dagger$. It has the following properties: 1) $\mathbf{X}\mathbf{X}^\dagger\mathbf{X} = \mathbf{X}$; 2) $\mathbf{X}^\dagger\mathbf{X}\mathbf{X}^\dagger = \mathbf{X}^\dagger$; 3) $(\mathbf{X}\mathbf{X}^\dagger)^T =$

$\mathbf{X}\mathbf{X}^\dagger$; 4) $(\mathbf{X}^\dagger\mathbf{X})^T = \mathbf{X}^\dagger\mathbf{X}$. If the columns of $\mathbf{X}$ are linearly independent, then $\mathbf{X}^T\mathbf{X}$ is invertible. In this case, the pseudoinverse of $\mathbf{X}$ can be explicitly written as $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$. Similarly, if the rows of $\mathbf{X}$ are linearly independent, $\mathbf{X}^\dagger = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$. A simple way to compute the pseudoinverse is by the SVD. Given the SVD of $\mathbf{X}$ as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{X}^\dagger = \mathbf{V}\Sigma^{-1}\mathbf{U}^T$. The pseudoinverse can be used to solve for a least squares solution to a system of linear equations. For example, given a system $\mathbf{A}\mathbf{y} = \mathbf{b}$ with $\mathbf{y}$ the unknown, the best solution in the sense of the least squares error can be computed as $\mathbf{y}^* = \mathbf{A}^\dagger\mathbf{b}$.

The Kronecker product is denoted by $\otimes$. For two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$, their Kronecker product is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mp \times nq}. \quad (2.1)$$

The Kronecker product is useful to solve some linear systems in matrix form. For example, consider the equation $\mathbf{A}\mathbf{Y}\mathbf{B} = \mathbf{C}$, where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are given, and $\mathbf{Y}$ is unknown. We can vectorize both sides of the equation and get the following equation

$$\text{vec}(\mathbf{A}\mathbf{Y}\mathbf{B}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{Y}) = \text{vec}(\mathbf{C}). \quad (2.2)$$

2.2 Basic Concepts

2.2.1 Subspace

2.2.1.1 Linear Subspace and Affine Subspace

In linear algebra, the concept of subspace¹ is directly related to that of a vector space. A vector space is a mathematical structure that is *closed* under *addition* and *scalar multiplication*. A common example of a vector space is the Euclidian space. A subspace, which in itself is a vector space, is a subset of some higher-dimensional space. More formally, given a vector space V over a field F , it is often possible to form another vector space by taking a subset W of V using the vector space operations (i.e., *addition* and *scalar multiplication*) of V . To show whether a subset of a vector space is a subspace, we have the following theorem:

Theorem 2.1 (Leon [2010]) *Let V be a vector space over the field F , and let W be a nonempty subset of V . Then W is a subspace of V if and only if W satisfies the following three conditions*

1. *The zero vector $\mathbf{0}$ is in W .*
2. *If $\mathbf{u}$ and $\mathbf{v}$ are the elements of W , then the sum $\mathbf{u} + \mathbf{v}$ is the element of W .*

¹Here, a subspace, by default, denotes a linear subspace.

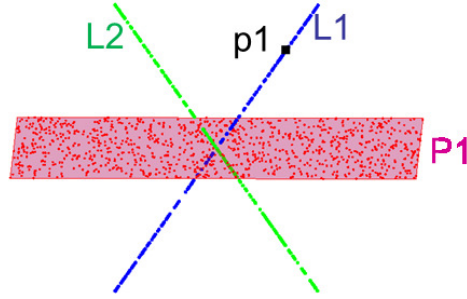


Figure 2.1: A toy example of subspace clustering. Three subspaces are contained in this example: two lines ($L1$, $L2$), and one plane ($P1$).

3. If $\mathbf{u}$ is an element of W and c is a scalar from F , then the product $c\mathbf{u}$ is an element of W .

Condition 1 is one of the axioms that define the vector space and should be the first condition to verify when defining the subspace. Condition 2 states that W is closed under addition, which means that the sum of two elements of W is always an element of W . Condition 3 says that W is closed under scalar multiplication. That is, whenever an element of W is multiplied by a scalar, the result is still an element of W . Therefore, if we use the operations from V and the elements of W to do arithmetic, the results will always be in W . Another way to characterize subspaces is that they are also closed under linear combinations.

For example, in Figure 2.1, the lines $L1$ and $L2$, and the plane $P1$ (which intersect at the origin) are three subspaces of the 3D Euclidean space: the zero vector (the origin) is included in $L1$, $L2$ and $P1$; and it can be easily verified that they are all closed under the operations of addition and scalar multiplication.

An *affine subspace*, also known as a *flat*, is similar to a linear subspace except that it does not need to pass through the origin. For example, any line in the 2D Euclidean space forms an affine subspace or flat, but only those going through the origin are linear subspaces.

2.2.1.2 Null Space, Row Space and Column Space

Consider a matrix $\mathbf{X} \in \mathbb{R}^{D \times N}$. Let $N(\mathbf{X})$ be the set of the solutions of the homogeneous equation $\mathbf{X}\mathbf{c} = \mathbf{0}$. We have

$$N(\mathbf{X}) = \{\mathbf{c} \in \mathbb{R}^N | \mathbf{X}\mathbf{c} = \mathbf{0}\} . \quad (2.3)$$

We now verify that $N(\mathbf{X})$ is a subspace of $\mathbb{R}^N$. Clearly, the equality holds when $\mathbf{c} = \mathbf{0}$, so $\mathbf{0} \in N(\mathbf{X})$. If $\mathbf{c} \in N(\mathbf{X})$ and α is a scalar, then $\mathbf{X}(\alpha\mathbf{c}) = \alpha\mathbf{X}\mathbf{c} = \alpha\mathbf{0} = \mathbf{0}$

and hence $\alpha \mathbf{c} \in N(\mathbf{X})$. If $\mathbf{c}_1, \mathbf{c}_2 \in N(\mathbf{X})$, then

$$\mathbf{X}(\mathbf{c}_1 + \mathbf{c}_2) = \mathbf{X}\mathbf{c}_1 + \mathbf{X}\mathbf{c}_2 = \mathbf{0},$$

which indicates that $\mathbf{c}_1 + \mathbf{c}_2 \in N(\mathbf{X})$. It then follows that $N(\mathbf{X})$ is a subspace of $\mathbb{R}^N$, and we call $N(\mathbf{X})$ the *null space* of $\mathbf{X}$.

If we consider each row of $\mathbf{X}$ as a vector in $\mathbb{R}^{1 \times N}$, the subspace of $\mathbb{R}^{1 \times N}$ spanned by the row vectors of $\mathbf{X}$ is called the *row space* of $\mathbf{X}$. Similarly, the subspace of $\mathbb{R}^D$ spanned by the column vectors of $\mathbf{X}$ is called the *column space* of $\mathbf{X}$. Note that the dimension of the row space of $\mathbf{X}$ is equal to the dimension of the column space of $\mathbf{X}$.

A subspace can be represented by a set of orthogonal basis with the number of basis equal to the dimension of the subspace. So given a rank- r matrix $\mathbf{X}$ (where $r \leq \min(D, N)$) and its full SVD $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, the column space of $\mathbf{X}$ can be represented by $\mathbf{U}_r$ (i.e., the first r columns of $\mathbf{U}$), the row space of $\mathbf{X}$ by $\mathbf{V}_r$ (i.e., the first r columns of $\mathbf{V}$), and the null space of $\mathbf{X}$ by $\mathbf{V}_{r+1:end}$ (i.e., the last $N - r$ columns of $\mathbf{V}$).

2.2.1.3 Subspace Clustering

With the definition of a subspace, the problem of subspace clustering is somewhat self-explanatory. Formally, suppose that we are given a set of N data points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in \mathbb{R}^D$ and that these data points are drawn from a union of K subspaces $\{S_i\}_{i=1}^K$ of unknown dimensions. Then the goal of subspace clustering is to cluster the data points into their respective subspaces, or in plain words, to find which data point lies on which subspace. For example in Figure 2.1, we aim to cluster the points according to three subspaces $L1$, $L2$ and $P1$, and different subspaces are plotted with different colors. In practice, the data points are often contaminated by noise and/or outliers, and some entries can even be unobserved. Thus, a practical subspace clustering method should be able to handle these data corruptions.

Affine subspaces are also within the scope of discussion of subspace clustering. In fact, an affine subspace can be thought of as lying on a higher dimensional linear subspace. For example, a line not going through the origin, which is an affine subspace, also lies on the plane going through the line and origin, which is a linear subspace. In literature (Chen and Lerman [2009]; Zhang et al. [2010]), subspace clustering is also called *Hybrid Linear Modeling* (HLM) when the data can be well approximated by a mixture of affine subspaces, or equivalently flats.

2.2.2 Grassmann Manifold

Linear subspaces of a fixed rank form a Riemannian manifold known as the *Grassmann manifold*, or equivalently the *Grassmannian*. More formally, let $\mathcal{G}(N, r)$ denote the Grassmann manifold that parameterizes all r -dimensional linear subspaces of $\mathbb{R}^N$. For example, $\mathcal{G}(3, 1)$ denotes the set of all lines passing through the origin in $\mathbb{R}^3$.

A point S on the Grassmann manifold $\mathcal{G}(N, r)$ is an r -dimensional linear subspace, which can be specified by an arbitrary orthogonal matrix $\mathbf{U} \in \mathbb{R}^{N \times r}$ whose columns span the subspace S . Note that the matrix representation for a point on $\mathcal{G}(N, r)$ is not unique. For example, if the orthogonal matrix $\mathbf{U} \in \mathbb{R}^{N \times r}$ represents a point on the Grassmannian $\mathcal{G}(N, r)$, then $\mathbf{U}\mathbf{Q}$ (with $\mathbf{Q}$ an arbitrary orthogonal r -by- r matrix) denotes the same point on $\mathcal{G}(N, r)$.

On a Grassmann manifold, the shortest distance between two points is along the *geodesic*, which is the curve of shortest length between two points on a manifold. To develop optimization algorithms (such as gradient methods), it is essential to compute a gradient step along the geodesic on the Grassmann manifold. To this end, we have the following theorem:

Theorem 2.2 (Edelman et al. [1998]) *Given an optimization problem*

$$\min_{\mathbf{Y} \in \mathcal{G}(N, r)} f(\mathbf{Y}), \quad (2.4)$$

and an initialization $\mathbf{Y}(0) = \mathbf{Y}$ and its regular gradient $\frac{df}{d\mathbf{Y}} = \mathbf{H}$, an update of $\mathbf{Y}$ along the geodesic with a step size η is given by

$$\mathbf{Y}(\eta) = [\mathbf{Y}\mathbf{V} \quad \mathbf{U}] \begin{bmatrix} \cos(\Sigma\eta) \\ \sin(\Sigma\eta) \end{bmatrix} \mathbf{V}^T, \quad (2.5)$$

where $\mathbf{U}\Sigma\mathbf{V}^T$ is the SVD of $\mathbf{H}$.

The proof of Theorem 2.2 is provided in (Edelman et al. [1998]). Various gradient methods (such as conjugate gradient and Newton's method) on the Grassmann manifold can be developed with this theorem. Interested readers may refer to Edelman et al. [1998] for more details.

2.3 Subspace Formulation for Motion Segmentation

2.3.1 Affine Camera Model

According to Hartley and Zisserman [2004], an *affine camera* is one that has a camera matrix $\mathbf{P}$ in which the last row is of the form $(0, 0, 0, 1)$; it is called an affine camera because points at infinity are mapped to points at infinity. The affine camera model is widely used as an approximation to the perspective camera model when the variation of the depth of the scene is much smaller than the distance of the points to the image plane (Ma et al. [2012]).

Specifically, under the affine camera model which includes the orthographic, scaled

orthographic, weak perspective and paraperspective models, a 3D point $[X, Y, Z]^T$ in space is projected to a 2D point $[x, y]^T$ on the image plane as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.6)$$

where $\mathbf{K} \in \mathbb{R}^{3 \times 3}$ is the matrix of intrinsic camera parameters, and $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{t} \in \mathbb{R}^3$ are camera rotation and translation, respectively. So we can write the camera matrix of an affine camera as

$$\mathbf{P} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}. \quad (2.7)$$

It is easy to see that $\mathbf{P}$ has the form

$$\mathbf{P} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & t_1 \\ m_{21} & m_{22} & m_{23} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.8)$$

where we can define the affine motion matrix $\mathbf{M} \in \mathbb{R}^{2 \times 4}$ as

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & t_1 \\ m_{21} & m_{22} & m_{23} & t_2 \end{bmatrix}. \quad (2.9)$$

Thus, projection under an affine camera model can be expressed as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{M} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.10)$$

2.3.2 Factorization

If we assume that a moving affine camera observes the 3D point $[X, Y, Z]^T$ over F frames², then we have F camera motion matrices $\mathbf{M}_1, \dots, \mathbf{M}_F$. Stacking the image

²Motion is relative. If the point is moving and the camera is static, we can always fix the coordinate system on that point so that the camera becomes moving and the point becomes static in that coordinate system.

points $[x_i, y_i]^T$ ($i = 1, \dots, F$) into a trajectory vector leads to

$$\begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_F \\ y_F \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_F \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (2.11)$$

where we can define the total camera motion matrix $\mathbf{W}$ as

$$\mathbf{W} = \begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_F \end{bmatrix} \in \mathbb{R}^{2F \times 4}. \quad (2.12)$$

Now, we consider N image points in the i^{th} frame projected from N points $[X_j, Y_j, Z_j]^T$ of a rigid body over an affine camera $\mathbf{M}_i$

$$\begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{iN} \\ y_{i1} & y_{i2} & \cdots & y_{iN} \end{bmatrix} = \mathbf{M}_i \begin{bmatrix} X_1 & \cdots & X_N \\ Y_1 & \cdots & Y_N \\ Z_1 & \cdots & Z_N \\ 1 & \cdots & 1 \end{bmatrix}, \quad (2.13)$$

where we define the shape matrix $\mathbf{S}$ as

$$\mathbf{S} = \begin{bmatrix} X_1 & \cdots & X_N \\ Y_1 & \cdots & Y_N \\ Z_1 & \cdots & Z_N \\ 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{4 \times N}. \quad (2.14)$$

If we stack all the N feature points in F frames together, we have

$$\underbrace{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ y_{11} & y_{12} & \cdots & y_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{F1} & x_{F2} & \cdots & x_{FN} \\ y_{F1} & y_{F2} & \cdots & y_{FN} \end{bmatrix}}_{\mathbf{X} \in \mathbb{R}^{2F \times N}} = \underbrace{\begin{bmatrix} \mathbf{M}_1 \\ \vdots \\ \mathbf{M}_F \end{bmatrix}}_{\mathbf{W} \in \mathbb{R}^{2F \times 4}} \underbrace{\begin{bmatrix} X_1 & \cdots & X_N \\ Y_1 & \cdots & Y_N \\ Z_1 & \cdots & Z_N \\ 1 & \cdots & 1 \end{bmatrix}}_{\mathbf{S} \in \mathbb{R}^{4 \times N}}, \quad (2.15)$$

where $\mathbf{X}$ is the data matrix with each column a trajectory, $\mathbf{W}$ the motion matrix containing all the motion information, and $\mathbf{S}$ the shape matrix containing the homogeneous 3D coordinates of all feature points.

Thus, the trajectory matrix $\mathbf{X}$ can be factorized into a bilinear form as

$$\mathbf{X} = \mathbf{W}\mathbf{S}. \quad (2.16)$$

Note that the factorization of $\mathbf{X}$ is unique since for any invertible matrix $\mathbf{Q} \in \mathbb{R}^{4 \times 4}$, $\mathbf{X} = (\mathbf{W}\mathbf{Q})(\mathbf{Q}^{-1}\mathbf{S})$. For Eq. (2.16), if we take the four columns of $\mathbf{W}$ as the four basis of a linear subspace with intrinsic dimension 4 and ambient dimension $2F$, then the elements of $\mathbf{S}$ are the coefficients, and the columns of $\mathbf{X}$ become points in this subspace. So, under an affine camera model, one motion corresponds to one linear subspace with dimension at most four. Alternatively, since the last row of $\mathbf{S}$ is $\mathbf{1}^T$, the trajectories on the image planes of a rigid motion lie in an affine subspace of $\mathbb{R}^{2F}$ with dimension at most three (Elhamifar and Vidal [2009]).

Naturally, for K rigid motions, we can still factorize the data matrix as

$$\begin{aligned} \mathbf{X} &= [\mathbf{X}_1 \ \cdots \ \mathbf{X}_K] \\ &= [\mathbf{W}_1\mathbf{S}_1 \ \cdots \ \mathbf{W}_K\mathbf{S}_K] \\ &= [\mathbf{W}_1 \ \cdots \ \mathbf{W}_K] \begin{bmatrix} \mathbf{S}_1 & & \\ & \ddots & \\ & & \mathbf{S}_K \end{bmatrix}, \end{aligned} \quad (2.17)$$

where $\mathbf{W}_i$ is the motion matrix for the i^{th} motion, and $\mathbf{S}_i$ is the shape matrix for the i^{th} rigid body. Note that, in practice, the trajectories are not sorted according to their motions, so this factorization works up to certain permutations of the columns of $\mathbf{X}$.

We can see that the trajectories of K motions lie in a union of linear subspaces of dimension up to $4K$. Thus under the affine camera model, the motion segmentation problem can be addressed as a subspace clustering problem.

2.3.3 Self-expressiveness

In subspace clustering, an important step is to construct an affinity matrix from the data points. This affinity matrix should build connections between data points in a way that points in the same subspace have strong connections (or equivalently high affinity) and points from different subspaces have weak connections or no connection.

To this end, in SSC (Elhamifar and Vidal [2009]), the interesting concept of *self-expressiveness* was introduced, i.e., the data points drawn from a union of subspaces are *self-expressive* in the sense that a data point from a subspace can be expressed as a linear combination of other data points from the same subspace. To be more specific, let's look at the example in Fig 2.1. By self-expressiveness, in this example, we mean that the point $\mathbf{p}_1$ on line $L1$ can be expressed as a linear combination of other points on $L1$. A connectivity, or affinity, matrix, can then be built by making use of the coefficients of these linear combinations.

Mathematically, the property of self-expressiveness can be summarized as one single equation

$$\mathbf{X} = \mathbf{XC}, \quad (2.18)$$

where $\mathbf{X}$ is the data matrix with each column being one data point, and $\mathbf{C}$ is the coefficient matrix.

To employ this idea to construct an affinity matrix, one has to ensure that the coefficient matrix $\mathbf{C}$ has nonzero entries only for the points in the same subspaces. In other words, $\mathbf{C}_{ij} = 0$ if points i and j belong to different subspaces, and $\mathbf{C}_{ij} \neq 0$ otherwise. This can be achieved by minimizing certain norms of $\mathbf{C}$. In particular, inspired by the compressive sensing theory, SSC (Elhamifar and Vidal [2013]) minimizes the ℓ_1 norm of $\mathbf{C}$ to get sparse representation of the data:

$$\begin{aligned} \min_{\mathbf{C}} \|\mathbf{C}\|_1 \\ \text{s.t. } \mathbf{X} = \mathbf{XC}, \\ \text{diag}(\mathbf{C}) = \mathbf{0}, \end{aligned} \quad (2.19)$$

where the diagonal constraint is enforced to avoid the trivial solution (i.e., the identity matrix).

For affine subspaces, an additional constraint on $\mathbf{C}$ can be incorporated to enforce the sum of each column of $\mathbf{C}$ to be 1, i.e.,

$$\begin{aligned} \min_{\mathbf{C}} \|\mathbf{C}\|_1 \\ \text{s.t. } \mathbf{X} = \mathbf{XC}, \\ \mathbf{1}^T \mathbf{C} = \mathbf{1}^T, \\ \text{diag}(\mathbf{C}) = \mathbf{0}. \end{aligned} \quad (2.20)$$

Alternatively, one could also add an extra all-one row to $\mathbf{X}$ so that the affine constraint is implicitly enforced.

When the data contain noise and some sparse outliers, the equality constraint $\mathbf{X} = \mathbf{XC}$ does not hold. This case can be handled by adding extra regularization terms in the cost function (Elhamifar and Vidal [2013]),

$$\begin{aligned} \min_{\mathbf{C}} \|\mathbf{C}\|_1 + \frac{\lambda_1}{2} \|\mathbf{E}_1\|_F^2 + \lambda_2 \|\mathbf{E}_2\|_1 \\ \text{s.t. } \mathbf{X} = \mathbf{XC} + \mathbf{E}_1 + \mathbf{E}_2, \\ \mathbf{1}^T \mathbf{C} = \mathbf{1}^T, \\ \text{diag}(\mathbf{C}) = \mathbf{0}, \end{aligned} \quad (2.21)$$

where $\mathbf{E}_1$ accounts for the data noise, $\mathbf{E}_2$ accounts for the sparse outliers, and λ_1, λ_2

are parameters.

To get low-rank representation, LRR (Liu et al. [2013]) minimizes the nuclear norm of $\mathbf{C}$:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_* \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C} . \quad (2.22)$$

A slightly different way to handle data corruptions is by modeling them with a structured norm on the noise term (Liu et al. [2013]):

$$\begin{aligned} \min_{\mathbf{C}, \mathbf{E}} \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} \\ \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{E} , \end{aligned} \quad (2.23)$$

where $\mathbf{E}$ models the data corruptions. Since each column of $\mathbf{X}$ represents a data point, the $\ell_{2,1}$ norm regularization on the noise term $\mathbf{E}$ then assumes that only a few data points (or columns of $\mathbf{X}$) are grossly corrupted.

2.3.4 Spectral Clustering

Most motion segmentation methods consist of first building an affinity matrix in some way (Yan and Pollefeys [2006]; Chen and Lerman [2009]; Elhamifar and Vidal [2009]; Liu et al. [2010]) and then applying spectral clustering with the affinity matrix to get the final segmentation results. So spectral clustering is an important tool for the motion segmentation task. Here, we follow von Luxburg [2007] to give a brief introduction to spectral clustering.

The spectral clustering algorithms rely on the construction of a Laplacian matrix. In literature, there are two types of Laplacian matrices, i.e., unnormalized Laplacian and normalized Laplacian.

2.3.4.1 Unnormalized Spectral Clustering

Suppose we have a nonnegative symmetric affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ where $a_{ij} = a_{ji} \geq 0$. Equivalently, $\mathbf{A}$ also describes an undirected, weighted graph G with weight matrix $\mathbf{A}$. The *unnormalized Laplacian matrix* is defined as

$$\mathbf{L}_{\text{un}} = \mathbf{D} - \mathbf{A} , \quad (2.24)$$

where $\mathbf{D}$ is the diagonal degree matrix defined as

$$\mathbf{D} = \begin{bmatrix} \sum_j a_{1j} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_j a_{Nj} \end{bmatrix} , \quad (2.25)$$

or in other words, the i^{th} element d_i on the diagonal of $\mathbf{D}$ is the sum of the i^{th} row of $\mathbf{A}$. $\mathbf{L}_{\text{un}}$ is *positive semi-definite* because, for every vector $\mathbf{f} \in \mathbb{R}^N$, we have

$$\mathbf{f}^T \mathbf{L}_{\text{un}} \mathbf{f} = \mathbf{f}^T \mathbf{D} \mathbf{f} - \mathbf{f}^T \mathbf{A} \mathbf{f} \quad (2.26)$$

$$= \sum_{i=1}^N d_i f_i^2 - \sum_{i,j=1}^N f_i f_j w_{ij} \quad (2.27)$$

$$= \frac{1}{2} \left(\sum_{i=1}^N d_i f_i^2 - 2 \sum_{i,j=1}^N f_i f_j w_{ij} + \sum_{j=1}^N d_j f_j^2 \right) \quad (2.28)$$

$$= \frac{1}{2} \left(\sum_{i,j=1}^N w_{ij} f_i^2 - 2 \sum_{i,j=1}^N f_i f_j w_{ij} + \sum_{j,i=1}^N w_{ji} f_j^2 \right) \quad (2.29)$$

$$= \frac{1}{2} \left(\sum_{i,j=1}^N w_{ij} f_i^2 - 2 \sum_{i,j=1}^N f_i f_j w_{ij} + \sum_{i,j=1}^N w_{ij} f_j^2 \right) \quad (2.30)$$

$$= \frac{1}{2} \sum_{i,j=1}^N w_{ij} (f_i - f_j)^2 \geq 0. \quad (2.31)$$

By observation, it is obvious that the smallest eigenvalue of $\mathbf{L}_{\text{un}}$ is 0 and the corresponding eigenvector is the constant one vector $\mathbf{1}$.

One of the important properties of the unnormalized Laplacian is summarized in the following proposition.

Proposition 2.1 *Let G be an undirected graph with non-negative weights $\mathbf{A}$. Then the multiplicity k of the eigenvalue 0 of $\mathbf{L}_{\text{un}}$ equals the number of connected components $A_1, \dots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$ of those components.*

The proof is given in (von Luxburg [2007]). This proposition says that the number of clusters is equal to the multiplicity of the eigenvalue 0 of the unnormalized Laplacian $\mathbf{L}_{\text{un}}$ and the clustering is indicated by the eigenvectors of eigenvalue 0.

With this proposition, the unnormalized spectral clustering algorithm can be written as

Algorithm 2.1 Unnormalized spectral clustering

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, number of clusters k

1. Compute the unnormalized Laplacian $\mathbf{L}_{\text{un}}$ as in (2.24).
2. Compute the smallest k eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_k$ of $\mathbf{L}_{\text{un}}$, and form $\mathbf{U} \in \mathbb{R}^{N \times k}$ with $\mathbf{u}_1, \dots, \mathbf{u}_k$ as columns. Each row of $\mathbf{U}$ is denoted by $\mathbf{y}_i \in \mathbb{R}^k, i = 1, \dots, N$.
3. Cluster the points $\{\mathbf{y}_i\}_{i=1, \dots, N}$ with the k -means algorithm and get the cluster label vector $\mathbf{s} \in \{1, \dots, k\}^N$.

Output: The cluster label $\mathbf{s}$

2.3.4.2 Normalized Spectral Clustering

In literature, there are two ways to normalize the Laplacian matrix, i.e.,

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2} \mathbf{L}_{\text{un}} \mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{L}_{\text{un}} \mathbf{W}^{-1/2} \quad (2.32)$$

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1} \mathbf{L}_{\text{un}} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{W}, \quad (2.33)$$

where $\mathbf{L}_{\text{sym}}$ is a symmetric matrix, and $\mathbf{L}_{\text{rw}}$ is closely related to a random walk.

Similarly to the unnormalized Laplacian $\mathbf{L}_{\text{un}}$, the normalized Laplacians $\mathbf{L}_{\text{sym}}$ and $\mathbf{L}_{\text{rw}}$ are both positive semi-definite and have 0 as the smallest eigenvalue. The corresponding eigenvector of $\mathbf{L}_{\text{rw}}$ is the constant one vector $\mathbf{1}$, and that of $\mathbf{L}_{\text{sym}}$ is $\mathbf{D}^{-1/2} \mathbf{1}$.

As in the unnormalized Laplacian, the multiplicity of the eigenvalue 0 of the normalized Laplacians $\mathbf{L}_{\text{sym}}$ and $\mathbf{L}_{\text{rw}}$ is also related to the number of clusters.

Proposition 2.2 *Let G be an undirected graph with non-negative weights $\mathbf{A}$. Then the multiplicity k of the eigenvalue 0 of both $\mathbf{L}_{\text{rw}}$ and $\mathbf{L}_{\text{sym}}$ equals the number of connected components $A_1, \dots, A_k$ in the graph. For $\mathbf{L}_{\text{rw}}$, the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$ of those components. For $\mathbf{L}_{\text{sym}}$, the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbf{D}^{-1/2} \mathbf{1}_{A_1}, \dots, \mathbf{D}^{-1/2} \mathbf{1}_{A_k}$.*

This leads to two versions of the normalized spectral clustering algorithm.

Algorithm 2.2 Normalized spectral clustering according to Shi and Malik [2000]

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, number of clusters k

1. Compute the unnormalized Laplacian $\mathbf{L}_{\text{un}}$ as in (2.24).
2. Compute the smallest k generalized eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_k$ of $\mathbf{L}_{\text{un}}$ of the generalize eigenproblem $\mathbf{L}_{\text{un}} \mathbf{u} = \lambda \mathbf{D} \mathbf{u}$, and form $\mathbf{U} \in \mathbb{R}^{N \times k}$ with $\mathbf{u}_1, \dots, \mathbf{u}_k$ as columns. Each row of $\mathbf{U}$ is denoted by $\mathbf{y}_i \in \mathbb{R}^k, i = 1, \dots, N$.
3. Cluster the points $\{\mathbf{y}_i\}_{i=1, \dots, N}$ with the k -means algorithm and get the cluster label vector $\mathbf{s} \in \{1, \dots, k\}^N$.

Output: The cluster label $\mathbf{s}$

Note that in algorithm 2.2 the generalized eigenvectors of $\mathbf{L}_{\text{un}}$ are equivalent to the eigenvectors of $\mathbf{L}_{\text{rw}}$. This algorithm can be interpreted from a perspective of normalized graph cuts, so it is also called the normalized cuts algorithm (Shi and Malik [2000]).

The next algorithm uses the symmetric normalized Laplacian $\mathbf{L}_{\text{sym}}$ instead of $\mathbf{L}_{\text{rw}}$. Apart from this, it also introduces an additional row normalization step to the eigenvectors of $\mathbf{L}_{\text{sym}}$.

Algorithm 2.3 Normalized spectral clustering according to Ng et al. [2002]

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, number of clusters k

1. Compute the normalized Laplacian $\mathbf{L}_{\text{sym}}$ as in (2.32).
2. Compute the smallest k eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_k$ of $\mathbf{L}_{\text{sym}}$, and form $\mathbf{U} \in \mathbb{R}^{N \times k}$ with $\mathbf{u}_1, \dots, \mathbf{u}_k$ as columns. Each row of $\mathbf{U}$ is denoted by $\mathbf{y}_i \in \mathbb{R}^k, i = 1, \dots, N$.
3. Normalize $\mathbf{y}_i$ to get $\mathbf{t}_i = \mathbf{y}_i / \|\mathbf{y}_i\|$ for $i = 1, \dots, N$.
4. Cluster the points $\{\mathbf{t}_i\}_{i=1, \dots, N}$ with the k -means algorithm and get the cluster label vector $\mathbf{s} \in \{1, \dots, k\}^N$.

Output: The cluster label $\mathbf{s}$

The algorithms 2.1, 2.2 and 2.3 all look rather similar, except that they use different Laplacian matrices. We can see that the main trick of all three algorithms is to change the representation of the data points $\mathbf{x}_i$ to points $\mathbf{y}_i \in \mathbb{R}^k$ which are the eigenvectors of the Laplacian matrices, which enhances the cluster-properties in the new representation. With that, we simply apply the k -means clustering algorithm to detect the clusters in this new data representation. It has been shown in this literature that the normalized spectral clustering algorithms generally perform better than the unnormalized one (von Luxburg [2007]). So in the remainder of this thesis, when mentioning spectral clustering, we always refer to normalized spectral clustering.

2.4 Optimization Methods

2.4.1 Alternating Direction Method of Multipliers

We follow Boyd et al. [2011] to introduce the Alternating Direction Method of Multipliers (ADMM), a widely used method for solving convex optimization problems with a large number of variables. ADMM works by decomposing a large optimization problem into a set of smaller subproblems which are easier to handle. Before introducing ADMM, we first review a precursor method, the *Lagrangian dual ascent method*.

2.4.1.1 Lagrangian Dual Ascent Method

Consider an equality-constrained convex problem of the form

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{Ax} + \mathbf{Bz} = \mathbf{c} \end{aligned} \tag{2.34}$$

with variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, where $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{B} \in \mathbb{R}^{p \times m}$, $\mathbf{c} \in \mathbb{R}^p$, and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^m \rightarrow \mathbb{R}$ are convex. A conventional way to solve

Problem (2.34) is via the *Lagrangian dual ascent method*.

The Lagrangian of Problem (2.34) can be written as

$$L(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) , \quad (2.35)$$

and the Lagrangian dual function is

$$g(\mathbf{y}) = \inf_{\mathbf{x}, \mathbf{z}} L(\mathbf{x}, \mathbf{z}, \mathbf{y}) , \quad (2.36)$$

where $\mathbf{y} \in \mathbb{R}^p$ is the dual variable or Lagrange multiplier. The dual problem becomes

$$\max_{\mathbf{y}} g(\mathbf{y}) . \quad (2.37)$$

For convex problems, strong duality holds, i.e., the optimal values of the primal problem (2.34) and dual problem (2.37) are the same. Under strict convexity, we can recover a primal optimal point $(\mathbf{x}^*, \mathbf{z}^*)$ from a dual optimal point $\mathbf{y}^*$ as

$$(\mathbf{x}^*, \mathbf{z}^*) = \operatorname{argmin}_{\mathbf{x}, \mathbf{z}} L(\mathbf{x}, \mathbf{z}, \mathbf{y}^*) . \quad (2.38)$$

In the *Lagrangian dual ascent method*, the dual problem is solved using gradient ascent. The gradient of $g(\mathbf{y})$ w.r.t. $\mathbf{y}$ can be computed as follows. First find $(\mathbf{x}^*, \mathbf{z}^*) = \operatorname{argmin}_{\mathbf{x}, \mathbf{z}} L(\mathbf{x}, \mathbf{z}, \mathbf{y})$; then compute the gradient as $\nabla g = \mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{z}^* - \mathbf{c}$, which is just the residual of the equality constraint. The dual ascent method works by iterating over the following steps:

$$(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}) = \operatorname{argmin}_{\mathbf{x}, \mathbf{z}} L(\mathbf{x}, \mathbf{z}, \mathbf{y}^k) \quad (2.39)$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha^k (\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}) , \quad (2.40)$$

where $\alpha^k > 0$ is a step size, and k is the iteration counter. Here, the Lagrangian is minimized jointly w.r.t. $\mathbf{x}$ and $\mathbf{z}$, and the dual function value is increased in each step via the $\mathbf{y}$ -update.

Under some assumptions, including f and g being strictly convex and finite, the dual ascent method converges to an optimal solution. However, in practice, these assumptions do not always hold. For example, if f or g is a nonzero affine function, then the update step (2.39) fails because L is not bounded from below. So in many cases where these assumptions do not hold, the dual ascent method cannot be used.

2.4.1.2 Augmented Lagrangian

ADMM differs from the dual ascent method mostly in two ways: first, the *augmented Lagrangian* is introduced with an additional quadratic penalty term to the Lagrangian;

second, the primal variables are not updated jointly, but in an alternating manner.

Specifically, the augmented Lagrangian of Problem (2.34) is written as

$$\mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}) + \frac{\rho}{2} \|\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}\|_2^2, \quad (2.41)$$

where $\rho > 0$ is the penalty parameter. Note that $\mathcal{L}_0$ (i.e., $\rho = 0$) is the standard Lagrangian. The augmented Lagrangian can be thought of as the standard Lagrangian of the following problem

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} \quad & f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}\|_2^2 \\ \text{s.t.} \quad & \mathbf{Ax} + \mathbf{Bz} = \mathbf{c} \end{aligned} \quad (2.42)$$

It is easy to see that Problem (2.42) is equivalent to Problem (2.34), because any feasible $\mathbf{x}$ and $\mathbf{z}$ make the last term zero in the objective. The benefit of adding this augmented term is that this method has far better convergence properties and is able to handle cases when f or g are not strictly convex or unbounded from above (Boyd et al. [2011]).

With the augmented Lagrangian, the ADMM for solving Problem (2.34) consists of the following iterations:

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}} L_\rho(\mathbf{x}, \mathbf{z}^k, \mathbf{y}^k) \quad (2.43)$$

$$\mathbf{z}^{k+1} = \underset{\mathbf{z}}{\operatorname{argmin}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{y}^k) \quad (2.44)$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}). \quad (2.45)$$

The iterations are stopped when certain stopping criteria are met, e.g., the primal and dual residuals are below a small threshold. This sequential update method is also called the inexact Augmented Lagrange Multipliers (ALM) method in (Lin et al. [2009]).

Recently, the ADMM has also been used to solve certain types of non-convex problems (Shen et al. [2014]; Zhang [2010]; Zeng et al. [2012]; Jia et al. [2015]), for example the problems with bilinear terms. This proves the practical usefulness of ADMM in handling non-convex problems, and recent works (Li and Pong [2015]; Wang et al. [2015]; Zhang et al. [2016]) have provided some theoretical analysis on it.

2.4.1.3 A Concrete Example

To get a comprehensive understanding of ADMM, we give a concrete example of using ADMM to solve a convex problem. We consider the optimization problem (2.23) for LRR (Liu et al. [2010]). For ease of solving the nuclear norm problem, we first

introduce an auxiliary variable $\mathbf{Z}$

$$\begin{aligned} \min_{\mathbf{C}, \mathbf{Z}, \mathbf{E}} \quad & \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} \\ \text{s.t.} \quad & \mathbf{X} = \mathbf{XZ} + \mathbf{E}, \\ & \mathbf{C} = \mathbf{Z}. \end{aligned} \quad (2.46)$$

Clearly, this problem is equivalent to problem (2.23). We now derive the augmented Lagrangian as

$$\begin{aligned} \mathcal{L}_\rho(\mathbf{C}, \mathbf{Z}, \mathbf{E}, \mathbf{Y}_1, \mathbf{Y}_2) = & \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} + \\ & \text{tr}[\mathbf{Y}_1^T(\mathbf{X} - \mathbf{XZ} - \mathbf{E})] + \text{tr}[\mathbf{Y}_2^T(\mathbf{C} - \mathbf{Z})] + \\ & \frac{\rho}{2}(\|\mathbf{X} - \mathbf{XZ} - \mathbf{E}\|_F^2 + \|\mathbf{C} - \mathbf{Z}\|_F^2), \end{aligned} \quad (2.47)$$

where $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are the Lagrange multipliers.

We then minimize $\mathcal{L}_\rho$ over one of the primal variables $\mathbf{C}$, $\mathbf{Z}$ and $\mathbf{E}$ while fixing the others. In particular, the subproblem for updating $\mathbf{C}$ is

$$\mathbf{C}^* = \underset{\mathbf{C}}{\text{argmin}} \frac{1}{\rho} \|\mathbf{C}\|_* + \frac{1}{2} \|\mathbf{C} - (\mathbf{Z} - \mathbf{Y}_2/\rho)\|_F^2, \quad (2.48)$$

which can be solved in closed form via the singular value thresholding operator.

For $\mathbf{Z}$, we simply take the derivative of $\mathcal{L}_\rho$ and set it to zero, which leads to

$$\mathbf{Z}^* = (\mathbf{X}^T \mathbf{X} + \mathbf{I})^{-1} (\mathbf{X}^T (\mathbf{X} - \mathbf{E}) + \mathbf{C} + (\mathbf{X}^T \mathbf{Y}_1 + \mathbf{Y}_2)/\rho), \quad (2.49)$$

and $\mathbf{E}$ is updated by solving the subproblem

$$\mathbf{E}^* = \underset{\mathbf{E}}{\text{argmin}} \frac{\lambda}{\rho} \|\mathbf{E}\|_{2,1} + \frac{1}{2} \|\mathbf{E} - (\mathbf{X} - \mathbf{XZ} + \mathbf{Y}_1/\rho)\|_F^2, \quad (2.50)$$

which also leads to a closed form solution given in (Liu et al. [2010]).

After the primal variables are updated, the dual variables $\mathbf{Y}_1$ and $\mathbf{Y}_2$ can be updated as

$$\mathbf{Y}_1 := \mathbf{Y}_1 + \rho(\mathbf{X} - \mathbf{XZ} - \mathbf{E}) \quad (2.51)$$

$$\mathbf{Y}_2 := \mathbf{Y}_2 + \rho(\mathbf{C} - \mathbf{Z}). \quad (2.52)$$

If a varying penalty parameter ρ is used, then ρ is updated after all the primal and dual variables are updated. These update steps are repeated sequentially until the ADMM algorithm converges.

2.4.2 Hungarian Algorithm

The *Hungarian algorithm*, also known as the *Munkres algorithm*, is a combinatorial optimization method that solves the assignment problem in polynomial time. In plain words, the assignment problem can be stated as follows: Suppose we have N agents to which N tasks are assigned on a one-to-one basis, and suppose we know the cost of assigning each task to each agent. Then the assignment problem is to find the optimal assignment that minimizes the total assignment cost.

In mathematical words, given an $N \times N$ cost matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix},$$

where a_{ij} is the cost of assigning the i^{th} task to the j^{th} agent, the assignment problem aims to find a permutation matrix $\mathbf{P} \in \{0, 1\}^{N \times N}$ that reorders the column index as $\mathbf{b} = [1 \ 2 \ \cdots \ N]\mathbf{P}$ such that the total assignment cost $\sum_{i=1}^N a_{ib_i}$ (with b_i the i^{th} element of $\mathbf{b}$) is minimized. For example, for a 3×3 cost matrix

$$\mathbf{A} = \begin{bmatrix} 7 & 5 & 11.2 \\ 5 & 4 & 1 \\ 9.3 & 3 & 2 \end{bmatrix},$$

there are six possible permutations for which the associated sums are

$\mathbf{b}$	$\sum_{i=1}^N a_{ib_i}$
[1 2 3]	13.0
[1 3 2]	11.0
[2 1 3]	12.0
[2 3 1]	15.3
[3 1 2]	19.2
[3 2 1]	24.5

Obviously, the second permutation with permutation matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

provides the minimal sum $a_{11} + a_{23} + a_{32} = 11.0$ and is the solution to the assignment problem with this cost matrix.

However, as the size of the problem grows, it quickly becomes impossible to

enumerate all the possible ($N!$) permutations. Then we may resort to the Hungarian algorithm to get the solution in $O(N^3)$ running time.

The Hungarian algorithm relies on the following two theorems.

Theorem 2.3 (Bourgeois and Lassalle [1971]) *If a number is added to or subtracted from all the entries of any one row or column of a cost matrix, then the optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix.*

Theorem 2.4 (Bourgeois and Lassalle [1971]) *Given a cost matrix $\mathbf{A}$, if m is the maximum number of independent zeros elements in the matrix, then there are m lines (rows or columns or both) which cover all the zero elements of $\mathbf{A}$.*

With the two theorems, the Hungarian algorithm finds an optimal assignment for a given $N \times N$ cost matrix with the following steps:

1. Subtract the smallest entry in each row from all the entries of its row;
2. Subtract the smallest entry in each column from all the entries of its column;
3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used;
4. *Test for Optimality:* (i) If the minimum number of covering lines is N , an optimal assignment of zeros is possible and we are finished. (ii) If the minimum number of covering lines is less than N , an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5;
5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

As an example,

$$\begin{aligned}
 & \begin{bmatrix} 7 & 5 & 11.2 \\ 5 & 4 & 1 \\ 9.3 & 3 & 2 \end{bmatrix} \xrightarrow{\text{step 1}} \begin{bmatrix} 2 & 0 & 6.2 \\ 4 & 3 & 0 \\ 7.3 & 1 & 0 \end{bmatrix} \xrightarrow{\text{step 2}} \begin{bmatrix} 0 & 0 & 6.2 \\ 2 & 3 & 0 \\ 5.3 & 1 & 0 \end{bmatrix} \xrightarrow{\text{step 3}} \\
 & \begin{bmatrix} 0 & 0 & 6.2 \\ 2 & 3 & 0 \\ 5.3 & 1 & 0 \end{bmatrix} \xrightarrow{\text{step 5}} \begin{bmatrix} 0 & 0 & 7.2 \\ 1 & 2 & 0 \\ 4.3 & 0 & 0 \end{bmatrix} \xrightarrow{\text{step 3}} \begin{bmatrix} 0^* & 0 & 7.2 \\ 1 & 2 & 0^* \\ 4.3 & 0^* & 0 \end{bmatrix} .
 \end{aligned}$$

If the number of agents and the number of tasks are different, we end up with a non-square cost matrix (Bourgeois and Lassalle [1971]). In this case, we can add dummy rows or columns to ensure that the cost matrix is a square matrix without changing the

assignment result. A conventional way to accomplish this is to make each element in the dummy rows/columns equal to (or larger than) the largest number in the matrix, so that the costs in the dummy rows/columns will never be selected.

There are many implementations of the Hungarian algorithm in different programming languages³. So for practical use, one only has to come up with the cost matrix for the task at hand.

³For example, a C implementation is given in <https://github.com/maandree/hungarian-algorithm-n3/blob/master/hungarian.c>, a C++ implementation in <http://dlib.net/optimization.html>, and a C and Matlab implementation in <http://www.mathworks.com/matlabcentral/fileexchange/6543-functions-for-the-rectangular-assignment-problem>.

Efficient Dense Subspace Clustering

As shown in the previous chapter, the motion segmentation problem can be addressed as a subspace clustering problem. So in this chapter, we consider the more general subspace clustering case.

While traditional techniques, such as Principal Component Analysis (PCA) (Jolliffe [1986]), represent high-dimensional data with a single low-dimensional subspace, recent research has shown that, in many situations, the data points lie in the union of multiple subspaces. This representation has proven beneficial for various computer vision problems, such as image segmentation (Yang et al. [2008]), motion segmentation (Vidal [2011]), face clustering (Ho et al. [2003]), and image compression (Hong et al. [2006]).

Over the years, many different approaches to identifying the multiple subspaces from which data points are drawn have been proposed (Vidal [2011]). Recent subspace clustering methods mostly rely on the self-expressiveness of the observations (Liu et al. [2013]; Elhamifar and Vidal [2013]). As a consequence, these techniques follow a two-step procedure: First, they establish the connections between the data points based on the self-expressiveness assumption. Second, they perform clustering using an affinity matrix built from these connections. In the first step, the data connections are typically estimated by solving a convex optimization problem involving a regularizer inspired by compressive sensing, i.e., the ℓ_1 -norm or the nuclear norm. Unfortunately, the ℓ_1 -norm yields sparse connections, which results in poor graph connectivity and may be poorly-suited for the latter clustering step (Nasihatkon and Hartley [2011]). In contrast, the nuclear norm yields dense connections. However, the computational efficiency of the resulting algorithms remains a limiting factor in the presence of noisy observations.

In this chapter, we introduce an algorithm called Efficient Dense Subspace Clustering (EDSC) that lets us learn dense connections between the data points. To this end, we model each observation as a linear combination of the elements of a clean dictionary, and employ an ℓ_2 -norm regularizer on the combination coefficients. Whereas with noise-free data we directly use the observations as dictionary elements, in the presence of noise we learn the clean dictionary simultaneously as the combination

weights. When the data contains no outliers, our formulation yields a closed-form solution to subspace clustering. With outliers, our solution boils down to performing a series of linear operations, which can be achieved efficiently.

In addition to deriving an efficient algorithm, we provide a theoretical analysis of our formulation which shows that, in the absence of noise and with independent subspaces, the observations will be correctly clustered according to the subspaces. Furthermore, our analysis reveals that our formulation yields the same solution as a nuclear norm minimization approach in the noise-free case.

We demonstrate the effectiveness and efficiency of our approach on motion segmentation and face clustering. In particular, we achieve state-of-the-art results on the Hopkins 155 motion segmentation dataset and on the Extended Yale B face recognition dataset (Georghiades et al. [2001]).

The use of the ℓ_2 -norm was also advocated in LSR (Lu et al. [2012]) and DSC (Dyer et al. [2013]). Our research significantly differs from these two works in the following ways. First, DSC does not handle noise explicitly and is therefore sensitive to it. While, in contrast, LSR does model the noise, it directly expresses each point as a linear combination of the noisy data; Self-expressiveness of the data is only truly meaningful in the absence of noise, which motivates our use of a clean dictionary. Second, LSR and DSC do not consider the case of data contaminated with outliers. Third, we provide a theoretical proof that any ℓ_p ($0 \leq p < \infty$) norm regularizer yields perfect subspace clustering results in the presence of clean data. Finally, we show interesting connections between our formulation and LRR and SIM.

3.1 Problem Analysis

In this section, we address the following subspace clustering problem:

Problem: Given a data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ ($\mathbf{X} \in \mathbb{R}^{D \times N}$ with each column a data point) drawn from a union of k subspaces $\{S_i\}_{i=1}^k$ of unknown dimensions, cluster the columns of $\mathbf{X}$ into their respective subspaces.

Assumption: The subspaces $\{S_i\}_{i=1}^k$ in which the data lies are independent. In other words, $\dim(\oplus_{i=1}^k S_i) = \sum_{i=1}^k \dim(S_i)$, where $\oplus$ is the direct sum operator.

With this assumption, subspace clustering can be formulated by first solving for each data point $\mathbf{x}_i$ the optimization problem

$$\min_{\mathbf{c}_i} \|\mathbf{c}_i\|_p \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{X}_{-i} \mathbf{c}_i, \quad (3.1)$$

where $\|\cdot\|_p$ is any p vector norm with $p \geq 0$, and $\mathbf{X}_{-i}$ is the data matrix from which point $\mathbf{x}_i$ has been removed. Given the coefficients $\{\mathbf{c}_i\}$ for each data point, we can form the coefficient matrix $\mathbf{C} = [\mathbf{c}_1 \cdots \mathbf{c}_N]$, from which the affinity matrix $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}^T|$ can be built. The clusters corresponding to the subspaces can then

be obtained by applying spectral clustering.

The idea behind (3.1) is that the data is self-expressive, and thus a data point can be represented as the linear combination of all the other data points. To make this representation tight, the current data point should only use the points in the same subspace, which is achieved by minimizing the ℓ_p norm of coefficients $\mathbf{c}_i$. The following theorem shows that, when the subspaces are independent, the solution to (3.1) is block-diagonal with non-zero blocks corresponding to points in the same subspaces.

Theorem 3.1 *Let $\mathbf{X} \in \mathbb{R}^{D \times N}$ be the data matrix whose columns are drawn from a union of k independent linear subspaces. Let us assume that $\mathbf{X}$ has been sorted according to the subspaces, i.e., $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N]\Gamma$, where $\Gamma \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix that specifies the clusters of the data. Then the solution to (3.1) $\mathbf{C}^* = [\mathbf{c}_1^* \cdots \mathbf{c}_N^*]\Gamma$ is block diagonal, i.e., there are only connections within clusters and no connection between clusters.*

Proof. If $\mathbf{c}_i^*$ is a solution of the ℓ_p problem in (3.1), then $\mathbf{c}_i^*$ is a vector of minimum ℓ_p norm satisfying $\mathbf{x}_i = \mathbf{X}_{-i}\mathbf{c}_i^*$. Let us decompose $\mathbf{c}_i^*$ as $\mathbf{c}_i^* = \mathbf{c}_i + \mathbf{h}_i$, where $\mathbf{c}_i$ corresponds to the coefficient recovered from the true subspace of $\mathbf{x}_i$ and $\mathbf{h}_i$ corresponds to the coefficient recovered from the other subspaces. Now we only need to show that $\mathbf{h}_i = 0$. Since $\mathbf{c}_i^* = \mathbf{c}_i + \mathbf{h}_i$, then we have $\mathbf{x}_i = \mathbf{X}_{-i}\mathbf{c}_i^* = \mathbf{X}_{-i}(\mathbf{c}_i + \mathbf{h}_i) = \mathbf{X}_{-i}\mathbf{c}_i + \mathbf{X}_{-i}\mathbf{h}_i$. Furthermore, since $\mathbf{x}_i \in S_i$ and $\mathbf{X}_{-i}\mathbf{c}_i \in S_i$ and, from the independence assumption, $\mathbf{X}_{-i}\mathbf{h}_i \notin S_i$, we have $\mathbf{X}_{-i}\mathbf{h}_i = 0$. If we assume that $\mathbf{h}_i \neq 0$, and from the fact that $\mathbf{c}_i$ and $\mathbf{h}_i$ have support on disjoint subset of indices, we have $\|\mathbf{c}_i\|_p < \|\mathbf{c}_i + \mathbf{h}_i\|_p = \|\mathbf{c}_i^*\|_p$, which contradicts the optimality of $\mathbf{c}_i^*$. Therefore, $\mathbf{h}_i = 0$, which concludes the proof. $\square$

Intuitively, consider a toy example in Figure 3.1 where we have two lines, each of which forms one linear subspace. The two subspaces are independent because the dimension of the space spanned by line $L1$ and $L2$ equals the dimension of $L1$ plus the dimension of $L2$. Suppose we want to represent the point $\mathbf{x}_p$ on $L1$ with other points, and let $\mathbf{X}_{\text{in}}$ be the matrix containing points on $L1$ (excluding point p) as columns and $\mathbf{X}_{\text{out}}$ be the matrix containing points on $L2$. We have

$$\mathbf{x}_p = \mathbf{X}_{\text{in}}\mathbf{c}_{\text{in}} + \mathbf{X}_{\text{out}}\mathbf{c}_{\text{out}}, \quad (3.2)$$

where $\mathbf{c}_{\text{in}}$ and $\mathbf{c}_{\text{out}}$ are the corresponding representation coefficients.

Note that any combinations of points on line $L2$ still lie on $L2$, so $\mathbf{X}_{\text{out}}$ should have no contribution in representing $\mathbf{x}_p$, i.e., $\mathbf{X}_{\text{out}}\mathbf{c}_{\text{out}} = \mathbf{0}$. There are two cases for $\mathbf{c}_{\text{out}}$: first, $\mathbf{c}_{\text{out}} = \mathbf{0}$; second, $\mathbf{c}_{\text{out}} \neq \mathbf{0}$ and the sum of points on $L2$ cancels out by certain nonzero coefficients $\mathbf{c}_{\text{out}}$. By theorem 3.1, we are guaranteed to have the first case, i.e. $\mathbf{c}_{\text{out}} = \mathbf{0}$, if we minimize $\|\mathbf{c}\|_p = \|[\mathbf{c}_{\text{in}}^T \mathbf{c}_{\text{out}}^T]^T\|_p$. This makes sense because we always have $\|[\mathbf{c}_{\text{in}}^T \mathbf{0}^T]^T\|_p \leq \|[\mathbf{c}_{\text{in}}^T \mathbf{c}_{\text{out}}^T]^T\|_p$.

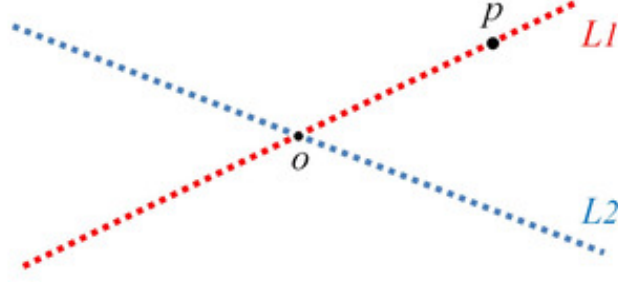


Figure 3.1: A toy example for subspace clustering: two lines $L1$ (with a point $\mathbf{p}$) and $L2$ passing through the origin o form two independent linear subspaces.

Instead of solving individual optimization problems for each data point, a global minimization problem over the matrix $\mathbf{C}$ can be formulated. For instance, SSC (Elhamifar and Vidal [2013]) and LRR (Liu et al. [2013]) are special cases of this formulation. For the sake of clustering, we want the intra-cluster connectivity to be as dense as possible. The Frobenius (or ℓ_2) norm provides a good regularizer to encourage this property. This motivated the formulation of this chapter discussed in the next section.

3.2 Frobenius Norm Formulation

We now present our EDSC algorithm. We start by considering the noise-free case, and then tackle the more realistic scenarios where noise and outliers are present.

To encourage denser connections between points belonging to the same clusters, we propose to make use of a Frobenius norm regularizer. In this setting, when the data points are free of noise and outliers, the first step of the subspace clustering problem can be formulated as

$$\min_{\mathbf{C}} \frac{1}{2} \|\mathbf{C}\|_F^2 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C}. \quad (3.3)$$

We now introduce a corollary to Theorem 3.1 for Problem (3.3) where $p = 2$. Note that, in (3.3), we do not enforce $\text{diag}(\mathbf{C}) = 0$. However, we will see that (3.3) will not result in trivial solutions even without this constraint.

Corollary 3.1 *If the subspaces are independent and the data samples are sorted according to the clusters, then the solution to Problem (3.3) is block-diagonal, i.e.,*

$$\mathbf{C}^* = \begin{bmatrix} \mathbf{C}_1^* & 0 & 0 & 0 \\ 0 & \mathbf{C}_2^* & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{C}_k^* \end{bmatrix}. \quad (3.4)$$

To prove Corollary 3.1, we follow a slightly different path than for Theorem 3.1,

and rely more on the algebraic form of the Frobenius norm. To this end, we first introduce the following lemma:

Lemma 3.1 *Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be square matrices. Then for any matrices $\mathbf{A}$ and $\mathbf{B}$ with compatible dimensions,*

$$\left\| \begin{pmatrix} \mathbf{C}_1 & \mathbf{A} \\ \mathbf{B} & \mathbf{C}_2 \end{pmatrix} \right\|_F \geq \left\| \begin{pmatrix} \mathbf{C}_1 & 0 \\ 0 & \mathbf{C}_2 \end{pmatrix} \right\|_F. \quad (3.5)$$

Proof. According to the definition of Frobenius norm,

$$\left\| \begin{pmatrix} \mathbf{C}_1 & \mathbf{A} \\ \mathbf{B} & \mathbf{C}_2 \end{pmatrix} \right\|_F^2 = \text{tr} \begin{pmatrix} \mathbf{C}_1^T \mathbf{C}_1 + \mathbf{B}^T \mathbf{B} & \mathbf{C}_1^T \mathbf{A} + \mathbf{B}^T \mathbf{C}_2 \\ \mathbf{A}^T \mathbf{C}_1 + \mathbf{C}_2^T \mathbf{B} & \mathbf{A}^T \mathbf{A} + \mathbf{C}_2^T \mathbf{C}_2 \end{pmatrix},$$

and

$$\left\| \begin{pmatrix} \mathbf{C}_1 & 0 \\ 0 & \mathbf{C}_2 \end{pmatrix} \right\|_F^2 = \text{tr} \begin{pmatrix} \mathbf{C}_1^T \mathbf{C}_1 & 0 \\ 0 & \mathbf{C}_2^T \mathbf{C}_2 \end{pmatrix}. \quad (3.6)$$

It can then easily be seen that

$$\text{tr} \begin{pmatrix} \mathbf{C}_1^T \mathbf{C}_1 + \mathbf{B}^T \mathbf{B} & \mathbf{C}_1^T \mathbf{A} + \mathbf{B}^T \mathbf{C}_2 \\ \mathbf{A}^T \mathbf{C}_1 + \mathbf{C}_2^T \mathbf{B} & \mathbf{A}^T \mathbf{A} + \mathbf{C}_2^T \mathbf{C}_2 \end{pmatrix} \geq \text{tr} \begin{pmatrix} \mathbf{C}_1^T \mathbf{C}_1 & 0 \\ 0 & \mathbf{C}_2^T \mathbf{C}_2 \end{pmatrix},$$

where the equality holds when $\mathbf{A} = \mathbf{0}$ and $\mathbf{B} = \mathbf{0}$. This directly concludes the proof of Lemma 3.1. $\square$

From Lemma 3.1, we can easily derive the proof of Corollary 3.1 as follows:

Proof. (Corollary 3.1) Let $\mathbf{C}$ be any minimizer of Problem (3.3). From $\mathbf{C}$, we can construct a block-diagonal matrix $\mathbf{C}^*$ of the form

$$\mathbf{C}_{ij}^* = \begin{cases} \mathbf{C}_{ij} & \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to the same subspace,} \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the i^{th} , j^{th} columns of the data matrix $\mathbf{X}$, respectively.

Now let $\mathbf{Q} = \mathbf{C} - \mathbf{C}^*$ be the off-block-diagonal part of $\mathbf{C}$. Without loss of generality, let us assume that $\mathbf{x}_j$ lies in the l^{th} subspace S_l , i.e., $[\mathbf{X}\mathbf{C}]_{:,j} \in S_l$. From the definition of $\mathbf{C}^*$ and $\mathbf{Q}$, $[\mathbf{X}\mathbf{C}^*]_{:,j} \in S_l$ and $[\mathbf{X}\mathbf{Q}]_{:,j} \in \bigcup_{m \neq l} S_m$. However, $[\mathbf{X}\mathbf{Q}]_{:,j} = [\mathbf{X}\mathbf{C}]_{:,j} - [\mathbf{X}\mathbf{C}^*]_{:,j} = [\mathbf{X}]_{:,j} - [\mathbf{X}\mathbf{C}^*]_{:,j} \in S_l$. Under the assumption that subspaces are independent, we have $S_l \cap \bigoplus_{m \neq l} S_m = \{0\}$. So, $[\mathbf{X}\mathbf{Q}]_{:,j} = 0$, $\forall j$, i.e., $\mathbf{X}\mathbf{Q} = 0$.

Therefore, $\mathbf{C}^*$ is also a feasible solution to (3.3). From Lemma 1, we have $\|\mathbf{C}\|_F \geq \|\mathbf{C}^*\|_F$, where the equality holds iff $\mathbf{Q} = 0$. Since, by hypothesis, $\mathbf{C}$ is a minimizer of (3.3), it must therefore be block-diagonal, which concludes the proof. $\square$

Now that Corollary 3.1 guarantees that our Frobenius norm formulation (3.3) will result in a block-diagonal matrix, we show in the following theorem that this solution is unique.

Theorem 3.2 *Problem (3.3) has a unique solution given by*

$$\mathbf{C}^* = \mathbf{X}^\dagger \mathbf{X}, \quad (3.7)$$

where $\mathbf{X}^\dagger$ is the pseudoinverse of $\mathbf{X}$.

Proof. It can easily be seen that $\mathbf{Z} = \mathbf{X}^\dagger \mathbf{X}$ is a solution of the linear equation $\mathbf{X} = \mathbf{X}\mathbf{C}$ (with unknown $\mathbf{C}$). However, in general, it is not necessarily the unique solution. Now, note that $\mathbf{Z}\mathbf{Z} = \mathbf{X}^\dagger \mathbf{X} \mathbf{X}^\dagger \mathbf{X} = \mathbf{X}^\dagger \mathbf{X} = \mathbf{Z}$ and $\mathbf{Z}^T = \mathbf{Z}$. Therefore, under the constraint $\mathbf{X} = \mathbf{X}\mathbf{C}$, we have $\mathbf{Z}^T(\mathbf{C} - \mathbf{Z}) = (\mathbf{Z}\mathbf{Z})^T(\mathbf{C} - \mathbf{Z}) = \mathbf{Z}^T \mathbf{Z}(\mathbf{C} - \mathbf{Z}) = \mathbf{Z}^T(\mathbf{X}^\dagger \mathbf{X}\mathbf{C} - \mathbf{Z}) = \mathbf{Z}^T(\mathbf{X}^\dagger \mathbf{X} - \mathbf{Z}) = 0$. Thus, $\frac{1}{2}\|\mathbf{C}\|_F^2 = \frac{1}{2}(\|\mathbf{Z}\|_F^2 + 2\text{tr}[\mathbf{Z}^T(\mathbf{C} - \mathbf{Z})] + \|\mathbf{C} - \mathbf{Z}\|_F^2) = \frac{1}{2}(\|\mathbf{Z}\|_F^2 + \|\mathbf{C} - \mathbf{Z}\|_F^2) \geq \frac{1}{2}\|\mathbf{Z}\|_F^2$, where the equality holds if and only if $\mathbf{C} = \mathbf{Z}$, which concludes the proof. $\square$

Theorem 3.2 builds an interesting connection between our formulation and LRR (Liu et al. [2010]). For clean data, LRR solves the optimization problem

$$\min_{\mathbf{C}} \|\mathbf{C}\|_* \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C}. \quad (3.8)$$

In (Liu et al. [2013]), it was shown that Problem (3.8) has a unique solution corresponding to $\mathbf{C}^* = \mathbf{X}^\dagger \mathbf{X}$. Therefore, Problem (3.3) and Problem (3.8) have the same unique solution.

To get a better understanding of the meaning of this solution, let us first compute the reduced Singular Value Decomposition (SVD) of $\mathbf{X}$, i.e., $\mathbf{X} = \mathbf{U}_r \Sigma_r \mathbf{V}_r^T$, where r is the rank of $\mathbf{X}$ and $\mathbf{U}_r^T \mathbf{U}_r = \mathbf{V}_r^T \mathbf{V}_r = \mathbf{I}$. We have $\mathbf{X}^\dagger = \mathbf{V}_r \Sigma_r^{-1} \mathbf{U}_r^T$. Thus, $\mathbf{C}^* = \mathbf{X}^\dagger \mathbf{X} = \mathbf{V}_r \Sigma_r^{-1} \mathbf{U}_r^T \mathbf{U}_r \Sigma_r \mathbf{V}_r^T = \mathbf{V}_r \mathbf{V}_r^T$, which is exactly the Shape Interaction Matrix (SIM) introduced in (Costeira and Kanade [1998]). Mathematically, it corresponds to the orthogonal operator that projects the N dimensional vectors to the subspace spanned by the columns of $\mathbf{V}$. As was shown in (Costeira and Kanade [1998]), the SIM has a block-diagonal structure. To summarize, we have the following lemma

Lemma 3.2 *For subspace clustering with noise free data, SIM (Costeira and Kanade [1998]), LRR (Liu et al. [2010]) and our method EDSC are equivalent.*

Proof. The optimization problem (3.3) for EDSC and the optimization problem (3.8) for LRR have been proven to have the same solution $\mathbf{X}^\dagger \mathbf{X}$, which has again been shown to be equal to $\mathbf{V}_r \mathbf{V}_r^T$, i.e., the SIM. $\square$

Specifically, the optimal objective value for problem (3.3) is

$$\begin{aligned}
 \frac{1}{2}\|\mathbf{C}^*\|_F^2 &= \frac{1}{2}\text{tr}(\mathbf{C}^{*T}\mathbf{C}^*) \\
 &= \frac{1}{2}\text{tr}(\mathbf{V}_r\mathbf{V}_r^T\mathbf{V}_r\mathbf{V}_r^T) \\
 &= \frac{1}{2}\text{tr}(\mathbf{V}_r\mathbf{V}_r^T) \\
 &= \frac{1}{2}\text{tr}(\mathbf{V}_r^T\mathbf{V}_r) \\
 &= \frac{1}{2}\text{tr}(\mathbf{I}_{r \times r}) = \frac{1}{2}r,
 \end{aligned}$$

which has a direct connection to the rank of $\mathbf{X}$ and Low Rank Representation (LRR) (Liu et al. [2010]). We now can see why Problem (3.3) will not result in the trivial solution, i.e., the identity matrix. If $\mathbf{C} = \mathbf{I}$, then we have $\frac{1}{2}\|\mathbf{C}\|_F^2 = \frac{1}{2}\|\mathbf{I}\|_F^2 = \frac{1}{2}N > \frac{1}{2}r$. So the identity matrix is not a solution to Problem (3.3).

3.2.1 A Null Space Perspective

From Problem (3.3), if we let $\mathbf{Z} = \mathbf{I} - \mathbf{C}$, we have

$$\min_{\mathbf{Z}} \frac{1}{2}\|\mathbf{I} - \mathbf{Z}\|_F^2 \quad \text{s.t.} \quad \mathbf{X}\mathbf{Z} = \mathbf{0}. \quad (3.9)$$

From the equality constraint in (3.9), $\mathbf{Z} \in \mathbb{R}^{N \times N}$ must lie in the *null space* of $\mathbf{X}$, which has rank $N - r$ and can be represented by $\mathbf{V}_n \in \mathbb{R}^{N \times (N-r)}$ consisting of the last $N - r$ right singular vectors of $\mathbf{X}$. Note that $\mathbf{V}_n$ has orthogonal columns, so $\mathbf{V}_n^T\mathbf{V}_n = \mathbf{I}$.

Lemma 3.3 *The solution to (3.9) is: $\mathbf{Z}^* = \mathbf{V}_n\mathbf{V}_n^T$.*

Proof. The linear constraints in (3.9) can be eliminated by expressing $\mathbf{Z} = \mathbf{V}_n\Phi$, where $\Phi \in \mathbb{R}^{N \times (N-r)}$ is the new variable. Then

$$\begin{aligned}
 \Phi^* &= \underset{\Phi}{\text{argmin}} \|\mathbf{I} - \mathbf{V}_n\Phi\|_F^2 \\
 &= \underset{\Phi}{\text{argmin}} \text{tr}(-2\mathbf{V}_n\Phi + \Phi^T\mathbf{V}_n^T\mathbf{V}_n\Phi) \\
 &= \underset{\Phi}{\text{argmin}} \text{tr}(-2\mathbf{V}_n\Phi + \Phi^T\Phi)
 \end{aligned}$$

Taking the derivative over Φ and setting it to zero yields $\Phi^* = \mathbf{V}_n^T$. So $\mathbf{Z} = \mathbf{V}_n\Phi^* = \mathbf{V}_n\mathbf{V}_n^T$, which concludes the proof. $\square$

Let us define the null space projection matrix $\mathbf{P}_{\text{null}} = \mathbf{V}_n \mathbf{V}_n^T$.¹ Then the solution to our Frobenius norm formulation (3.3) is related to the null space of $\mathbf{X}$ by

$$\mathbf{C}^* = \mathbf{I} - \mathbf{P}_{\text{null}}. \quad (3.10)$$

3.2.2 Dealing with Noise

In practice, the input data points are often contaminated with noise (i.e., Gaussian noise). To better deal with noise, we relax the constraints of (3.3). This yields the optimization problem

$$\min_{\mathbf{C}} \frac{1}{2} \|\mathbf{C}\|_F^2 + \frac{\lambda}{2} \|\mathbf{X}\mathbf{C} - \mathbf{X}\|_F^2 \quad (\text{s.t. } \mathbf{1}^T \mathbf{C} = \mathbf{1}^T). \quad (3.11)$$

Note that, here, we also have introduced optional constraints to handle the case of affine subspaces. The following lemma gives the closed-form solutions to (3.11) with and without constraints.

Lemma 3.4 (a) *Without affine constraints, the closed-form solution to (3.11) is given by*

$$\mathbf{C} = \lambda(\mathbf{I} + \lambda \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X}. \quad (3.12)$$

(b) *With affine constraints, the closed-form solution to (3.11) is given by*

$$\varphi = -(\mathbf{v}^T \mathbf{v} + \lambda \mathbf{v}^T \mathbf{X}^T \mathbf{X} \mathbf{v})^{-1} \mathbf{v}^T, \quad (3.13)$$

where $\mathbf{v}$ is the basis of the null space of $\mathbf{1}^T$, and φ is the new variable to estimate. Given φ , the coefficient matrix $\mathbf{C}$ is computed as $\mathbf{C} = \mathbf{I} + \mathbf{v}\varphi$.

Proof. Without affine constraints, the proof of Lemma 3.4 can be derived by simply taking the derivative of the objective function in (3.11) w.r.t. $\mathbf{C}$. When the constraints are enforced, they can be eliminated by expressing $\mathbf{C} = \mathbf{I} + \mathbf{v}\varphi$. Taking the derivatives of the objective function w.r.t. the new variable φ directly gives the solution. $\square$

In the following, we ignore the affine constraints, since they can easily be handled in a similar manner as above.

Intuitively, it seems more reasonable to represent the data with a clean dictionary rather than as linear combinations of noisy observations. Problem (3.11) can be extended to model this clean dictionary. This yields

$$\begin{aligned} \min_{\mathbf{C}, \mathbf{D}} \quad & \frac{1}{2} \|\mathbf{C}\|_F^2 + \frac{\lambda}{2} \|\mathbf{X} - \mathbf{D}\|_F^2 \\ \text{s.t.} \quad & \mathbf{D} = \mathbf{D}\mathbf{C} \end{aligned} \quad (3.14)$$

¹ $\mathbf{P}_{\text{null}}$ is a projection matrix because $\mathbf{P}_{\text{null}}^2 = \mathbf{V}_n \mathbf{V}_n^T \mathbf{V}_n \mathbf{V}_n^T = \mathbf{V}_n \mathbf{V}_n^T = \mathbf{P}_{\text{null}}$.

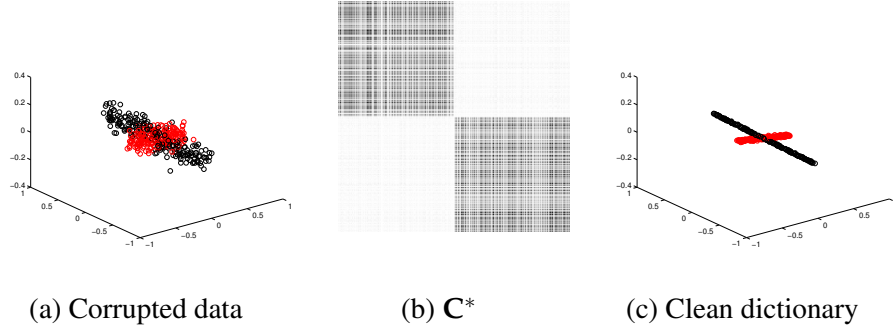


Figure 3.2: **Recovering a clean dictionary from noisy data with EDSC.** (a) Data corrupted by Gaussian noise; (b) Optimal solution $\mathbf{C}^*$ of (3.14); (c) Recovered clean dictionary $\mathbf{D}$.

where λ balances the influence of both terms. The following lemma gives the closed-form solution to (3.14).

Lemma 3.5 *Let $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ be the SVD of the data matrix $\mathbf{X}$. Then the optimal solution to (3.14) is given by $\mathbf{D} = \mathbf{U}_r\Sigma_r\mathbf{V}_r^T$ and $\mathbf{C} = \mathbf{V}_r\mathbf{V}_r^T$, where r is the number of singular values that are greater than $\sqrt{2/\lambda}$ and $\Sigma_r, \mathbf{U}_r, \mathbf{V}_r$ are the top r singular values and singular vectors of $\mathbf{X}$, respectively.*

Proof. Without loss of generality, let us decompose $\mathbf{D}$ as its rank- r SVD, i.e., $\mathbf{D} = \tilde{\mathbf{U}}_r\Lambda_r\tilde{\mathbf{V}}_r^T$, where r is arbitrary. From Theorem 3.2, given $\mathbf{D}$, the optimal solution for $\mathbf{C}$ is $\mathbf{C} = \tilde{\mathbf{V}}_r\tilde{\mathbf{V}}_r^T$. With this result, the proof of our lemma can directly be derived from that of Lemma 3 in (Favaro et al. [2011]), which considers the problem

$$\|\mathbf{C}\|_* + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{D}\|_F^2 \quad \text{s.t.} \quad \mathbf{D} = \mathbf{DC}. \quad (3.15)$$

We refer the readers to (Favaro et al. [2011]) for a detailed proof of the closed-form solution of (3.15). In short, this proof makes use of the solution to the noise-free problem (3.8). Since the terms involving $\mathbf{D}$ are the same in (3.14) and (3.15), and since the solution to the noise-free problems (3.3) and (3.8) are the same, it can be seen that the optimal solutions of (3.14) and (3.15) coincide. $\square$

To show the effectiveness of our formulation (3.14), we consider a toy example where the input data points are sampled from two one-dimensional linear subspaces embedded in $\mathbb{R}^3$, and corrupted by Gaussian noise. Figure 3.2 depicts the data and the solution of (3.14). Note that our formulation lets us recover a clean dictionary from the noisy data and yields dense intra-cluster connectivity.

3.2.3 Dealing with Outliers

We now turn to the case where the data is corrupted by gross errors (i.e., outliers). In this scenario, we formulate the solution of the first step of subspace clustering as

$$\begin{aligned} \min_{\mathbf{C}, \mathbf{D}, \mathbf{E}} \quad & \frac{1}{2} \|\mathbf{C}\|_F^2 + \frac{\lambda_1}{2} \|\mathbf{X} - \mathbf{D} - \mathbf{E}\|_F^2 + \lambda_2 \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{D} = \mathbf{D}\mathbf{C} \end{aligned} \quad (3.16)$$

where the ℓ_1 norm regularizer on the error $\mathbf{E}$ indicates that we expect only a few outlier data points.

We solve Problem (3.16) via ADMM (Lin et al. [2009]; Boyd et al. [2011]). In particular, the augmented Lagrangian of (3.16) can be expressed as

$$\begin{aligned} L(\mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{Y}) = & \frac{1}{2} \|\mathbf{C}\|_F^2 + \frac{\lambda_1}{2} \|\mathbf{X} - \mathbf{D} - \mathbf{E}\|_F^2 + \\ & \lambda_2 \|\mathbf{E}\|_1 + \text{tr}[\mathbf{Y}^T(\mathbf{D} - \mathbf{D}\mathbf{C})] + \frac{\rho}{2} \|\mathbf{D} - \mathbf{D}\mathbf{C}\|_F^2, \end{aligned} \quad (3.17)$$

where $\mathbf{Y}$ contains the Lagrange multipliers corresponding to the constraints. Estimating the coefficient matrix $\mathbf{C}$, the clean dictionary $\mathbf{D}$ and the outliers $\mathbf{E}$ can then be achieved following the steps detailed in Algorithm 3.1. More specifically, the updates for $\mathbf{D}$ (step 1 in Algorithm 3.1) and $\mathbf{C}$ (step 2 in Algorithm 3.1) can be obtained by simply solving linear equations in closed form

$$\mathbf{D} = (\lambda_1(\mathbf{X} - \mathbf{E}) - \mathbf{Y}(\mathbf{I} - \mathbf{C}^T))(\lambda_1\mathbf{I} + \rho(\mathbf{I} - \mathbf{C})(\mathbf{I} - \mathbf{C}^T))^{-1} \quad (3.18)$$

$$\mathbf{C} = (\mathbf{I} + \rho\mathbf{D}^T\mathbf{D})^{-1}(\mathbf{D}^T\mathbf{Y} + \rho\mathbf{D}^T\mathbf{D}). \quad (3.19)$$

The update for $\mathbf{E}$ (step 3 in Algorithm 3.1) can be performed via a soft-thresholding (shrinkage) operator (Lin et al. [2009]) defined as

$$S_\varepsilon(x) = \begin{cases} x - \varepsilon & \text{if } x > \varepsilon, \\ x + \varepsilon & \text{if } x < -\varepsilon, \\ 0 & \text{otherwise.} \end{cases} \quad (3.20)$$

This is really reduced to element-wise thresholding and can thus be computed efficiently. The update for $\mathbf{E}$ is then given by $\mathbf{E} = S_{\lambda_2/\lambda_1}(\mathbf{X} - \mathbf{D})$. Note that LRSC also leverages ADMM to handle outliers. However, the solution for LRSC involves a singular value hard thresholding operation at every iteration, which is computationally more expensive.

In Fig 3.3, we illustrate the effectiveness of our formulation via a toy example. As in the noisy case, we are able to automatically recover the clean dictionary from data corrupted by both Gaussian noise and sparse outliers.

Algorithm 3.1 Solving (3.16) by ADMM**Input:**Data matrix $\mathbf{X}$, parameters λ_1, λ_2 ;**Initialization:** $\mathbf{C} = \mathbf{0}, \mathbf{Y} = \mathbf{0}, \rho = 10^{-6}, \eta = 3, \rho_m = 10^{10}, \varepsilon = 10^{-8}$ **while** not converged **do**

1. Update $\mathbf{D}$ by solving the linear equation

$$\mathbf{D}(\lambda_1 \mathbf{I} + \rho(\mathbf{I} - \mathbf{C})(\mathbf{I} - \mathbf{C}^T)) = \lambda_1(\mathbf{X} - \mathbf{E}) - \mathbf{Y}(\mathbf{I} - \mathbf{C}^T) \quad (3.21)$$

2. Update $\mathbf{C}$ by solving the linear equation

$$(\mathbf{I} + \rho \mathbf{D}^T \mathbf{D}) \mathbf{C} = \mathbf{D}^T \mathbf{Y} + \rho \mathbf{D}^T \mathbf{D} . \quad (3.22)$$

3. Update $\mathbf{E}$ as

$$\mathbf{E} = \underset{\tilde{\mathbf{E}}}{\operatorname{argmin}} \frac{\lambda_2}{\lambda_1} \|\tilde{\mathbf{E}}\|_1 + \frac{1}{2} \|\tilde{\mathbf{E}} - (\mathbf{X} - \mathbf{D})\|_F^2 . \quad (3.23)$$

4. Update the Lagrange multipliers $\mathbf{Y}$ and the penalty parameter ρ as

$$\begin{aligned} \mathbf{Y} &= \mathbf{Y} + \rho(\mathbf{D} - \mathbf{D}\mathbf{C}) , \\ \rho &= \min(\eta\rho, \rho_m) . \end{aligned}$$

5. Check convergence

$$\|\mathbf{D} - \mathbf{D}\mathbf{C}\|_\infty < \varepsilon .$$

end while**Output:** Coefficient matrix $\mathbf{C}$

3.3 Affinity Matrix and Normalized Cuts

Having computed the coefficient matrix $\mathbf{C}$, we need to build an affinity matrix to perform the clustering step. Affinity matrices are typically built as $|\mathbf{C}| + |\mathbf{C}^T|$, or $|\mathbf{C}|$. In LRR (Liu et al. [2013]), a different heuristic was employed to further enhance the block-structure, which proved beneficial for clustering accuracy. Motivated by this, here, we follow a similar approach to building the affinity matrix. In particular, we make use of a variation of the SIM that lets us deal with noise. This affinity matrix can be obtained via the following steps:

1. Compute the SVD of $\mathbf{C}$, i.e., $\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^T$, and let $m = kd + 1$, where d is the maximal intrinsic dimension of the subspaces;
2. Let $\mathbf{Z} = \mathbf{U}_m \Sigma_m^{\frac{1}{2}}$, where $\mathbf{U}_m$ is the $N \times m$ matrix containing the first m columns of $\mathbf{U}$, and normalize each row of $\mathbf{Z}$ to have unit norm;

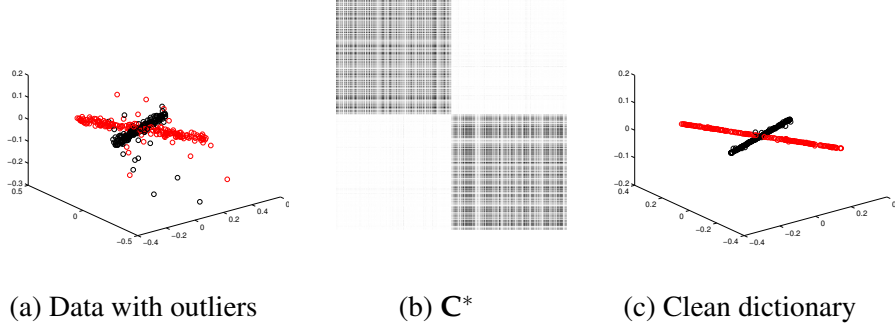


Figure 3.3: **Recovering a clean dictionary in the presence of outliers with EDSC.** (a) Data corrupted by noise and outliers; (b) Optimal solution C^* of (3.16); (c) Recovered clean dictionary D

3. Build the affinity matrix A , such that $A_{ij} = [ZZ^T]_{ij}^\alpha$, where α is empirically selected according to the level of noise. In our experiments, we set $\alpha = 4$.

We then obtain the clusters via normalized cuts (Shi and Malik [2000]) on A .

3.4 Experiments

We evaluate the performance of our EDSC algorithm on two standard datasets: Hopkins155 for motion segmentation, and Extended Yale B for face clustering. We compare EDSC to state-of-the-art algorithms, such as LSA (Yan and Pollefeys [2006]), SCC (Chen and Lerman [2009]), SSC (Elhamifar and Vidal [2013]), LRR (Liu et al. [2013]), and LRSC (Vidal and Favaro [2014]). For all competing methods, we report their clustering error in percentage, which is defined as

$$\text{Err} = \frac{\# \text{ wrongly clustered trajectories}}{\text{total \# trajectories}} \times 100\% . \quad (3.24)$$

And we give the number of motions as an input for all baselines.

3.4.1 Hopkins155: Motion Segmentation

The Hopkins155 dataset consists of 155 sequences, each of which contains between 39 and 550 data samples from two or three motions. The sequences can be roughly divided into three categories: i) *checkerboard sequences*, which include 104 sequences of indoor scenes taken with a handheld camera under controlled conditions, ii) *traffic sequences*, which consist of 38 sequences of outdoor traffic scenes taken by a moving handheld camera, iii) *other articulated/non-rigid sequences*, which contain 13 sequences displaying motions constrained by joints, face motions, people walking, etc..



Figure 3.4: Sample images from the Hopkin155 dataset including indoor checkerboard sequences, outdoor traffic sequences, non-rigid and articulated sequences.

See Figure 3.4 for sample images from this dataset. For each sequence, the 2D trajectories in F frames were extracted automatically with a tracker and manually cleaned up, so this dataset contains no outliers. As we discussed in Chapter 2, each motion corresponds to an affine subspace of dimension three. The data therefore corresponds to multiple affine subspaces contaminated by noise, which lets us use our formulation in (3.11) with affine constraints. For EDSC, we set $\lambda = 120$. For the baselines, we tuned their respective parameters to achieve the best results. Table 3.1 depicts the results of the different algorithms without (a) and with (b) PCA pre-processing of the data. Note that, in both cases, EDSC outperforms all the baselines. In terms of runtime, closed-form methods (EDSC and LRSC) are much faster (typically hundreds of times faster) than other iterative algorithms (LSA, SCC, SSC and LRR). On Hopkins 155, the average runtime of our algorithm for computing the coefficient matrix $\mathbf{C}$ is only

around 0.0055 seconds.

Table 3.1: Clustering error (in %) on Hopkins155.

Method	LSA	SCC	SSC	LRR	LRSC	EDSC
(a) 2F-dimensional data points						
2 motions						
Mean	4.23	2.89	1.52	2.13	2.57	0.86
Median	0.56	0.00	0.00	0.00	0.00	0.00
3 motions						
Mean	7.02	8.25	4.40	4.03	6.64	2.49
Median	1.45	0.24	0.56	1.43	1.76	0.21
All						
Mean	4.86	4.10	2.18	2.56	3.47	1.23
Median	0.89	0.00	0.00	0.00	0.09	0.00
(b) $4k$ -dimensional data points by applying PCA						
2 motions						
Mean	3.61	3.04	1.83	3.41	2.57	0.85
Median	0.51	0.00	0.00	0.00	0.00	0.00
3 motions						
Mean	7.65	7.91	4.40	4.86	6.62	2.51
Median	1.27	1.14	0.56	1.47	1.76	0.21
All						
Mean	4.52	4.14	2.41	3.74	3.47	1.23
Median	0.57	0.00	0.00	0.00	0.00	0.00

3.4.2 Extended Yale B: Face Clustering

Since in this chapter we consider the general subspace clustering problem, we further evaluate our method for face clustering in addition to motion segmentation. The Extended Yale B dataset consists of 192×168 pixels cropped face images of 38 individuals. Each class (individual) contains 64 frontal face images acquired under different illumination conditions. We follow the same experimental setting as in (Elhamifar and Vidal [2013]), which divides the 38 subjects into 4 groups (1 to 10, 11 to 20, 21 to 30, and 31 to 38) to avoid testing all combinations of n subjects out of 38. To reduce the computational cost for all algorithms, we downsampled the images to 48×42 pixels and stacked them into 2016-dimensional column vectors. Face images of k subjects under varying illuminations are known to correspond to corrupted data points lying close to a union of 9-dimensional subspaces. Since the

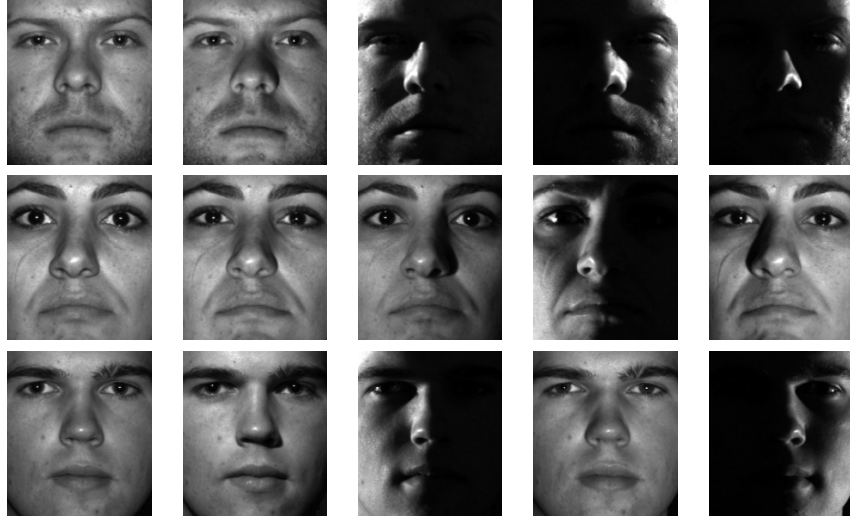


Figure 3.5: Sample face images from the extended yale B face dataset: each row contains aligned face images of the same subject under different illumination conditions

face data is corrupted by noise and outliers, we used the formulation in (3.16). We set $\lambda_1 = 0.06$ and $\lambda_2 = 0.01$ for EDSC, and tuned the parameters to obtain the best results for the baselines. The results for different number of subjects are reported in Table 3.2. Note that, for 2, 3, 5 and 8 subjects, we obtain comparable results to SSC, which significantly outperforms the other baselines. Importantly, when 10 subjects are used, our algorithm yields the best results, which shows the better robustness of EDSC to the number of clusters. In Figure 3.6, we show the average runtimes of the algorithms for computing the affinity matrix. Our method is at least one order of magnitude faster than the baselines. These results clearly evidence the benefits of exploiting the Frobenius norm, which yields efficient and effective algorithms.

3.5 Summary

In this chapter, we have introduced an efficient dense subspace clustering algorithm that effectively handles noise and outliers. To this end, we have relied on a Frobenius norm formulation, which yields dense connections within each cluster. When the data is corrupted by noise, our formulation has a closed-form solution. With noise and outliers, ADMM still yields an efficient solution. Furthermore, we have shown the connection between Frobenius norm minimization, nuclear norm minimization, the shape interaction matrix, and the null space projection matrix. Our experimental evaluation on motion segmentation and face clustering has demonstrated that our approach outperforms the state-of-art algorithms both in terms of accuracy and efficiency.

Table 3.2: Clustering error (in %) on Extended Yale B.

Method	LSA	SCC	SSC	LRR	LRSC	EDSC
2 subjects						
Mean	32.80	16.62	1.86	2.54	5.32	2.65
Median	47.66	7.82	0.00	0.78	4.69	1.56
3 subjects						
Mean	52.29	38.16	3.10	4.21	8.47	3.86
Median	50.00	39.06	1.04	2.60	7.86	3.13
5 subjects						
Mean	58.02	58.90	4.31	6.90	12.24	5.11
Median	56.87	59.38	2.50	5.63	11.25	3.75
8 subjects						
Mean	59.19	66.11	5.85	14.34	23.72	6.07
Median	58.59	66.11	4.49	14.34	28.03	4.88
10 subjects						
Mean	60.42	73.02	10.94	22.92	30.36	7.24
Median	57.50	75.78	5.63	23.59	28.75	6.09

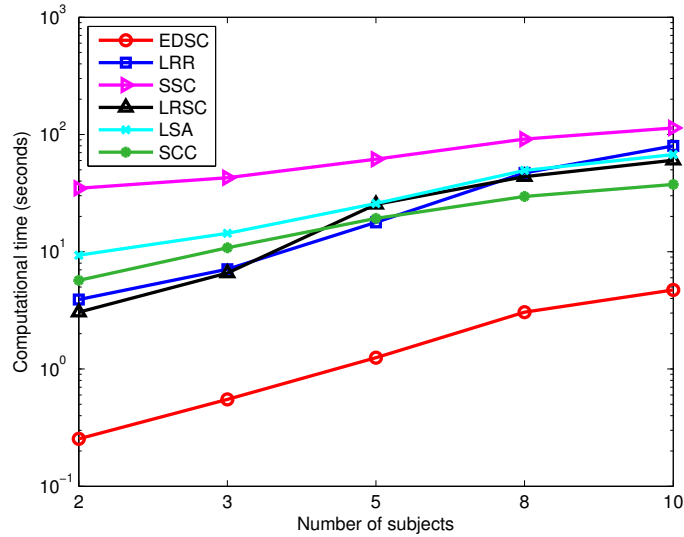


Figure 3.6: Average runtimes of the algorithms on Extended Yale B as a function of the number of subjects.

The Robust Shape Interaction Matrix Method

In Chapter 3, we have seen that motion segmentation (or subspace clustering) can be formulated as a Frobenius norm minimization problem which has an efficient solution. Importantly, we have found that our EDSC is equivalent to the SIM (Costeira and Kanade [1995]) for noise-free data. Indeed, most of the research in motion segmentation (or subspace clustering) takes its roots in the pioneering work of Costeira and Kanade [1995], which introduced the SIM to solve the motion segmentation problem.

While the SIM provably yields perfect clusters given ideal measurements from independent subspaces, the quality of the clusters quickly degrades in the presence of noise, as illustrated by Figure 4.1. As a consequence, many algorithms have been proposed to improve the robustness of subspace clustering. However, these methods typically work either by using discriminant criteria to reduce the effects of noise (Ichimura [1999]; Wu et al. [2001]), which may be sensitive to the noise level, or by formulating subspace clustering as a regularized optimization problem (Elhamifar and Vidal [2009, 2013]; Liu et al. [2010, 2013]), thus requiring to tune the regularization weight to the data at hand. This is also the case of the method introduced in the previous chapter. Furthermore, little work has been done to address the missing data scenario, for which, to the best of our knowledge, expensive two-step methods (i.e., data completion followed by clustering) are typically employed (Vidal et al. [2008]; Rao et al. [2010a]).

In this chapter, we revisit the use of the SIM for subspace clustering and study its connections to several recent algorithms. Based on our analysis, we show that simple, yet effective modifications of the SIM can significantly improve its robustness to data corruptions. This, in turn, lets us introduce an efficient approach to handling missing data, whose presence is inevitable in real-world scenarios.

More specifically, we derive an algorithm that robustifies the SIM via a series of simple operations on the right singular vectors of the data matrix and on the SIM itself. Furthermore, we draw inspirations from the perturbation theory and introduce a parameter-free criterion that lets us select the correct number of singular vectors automatically, thus making our algorithm robust to degenerate cases (e.g., planar mo-

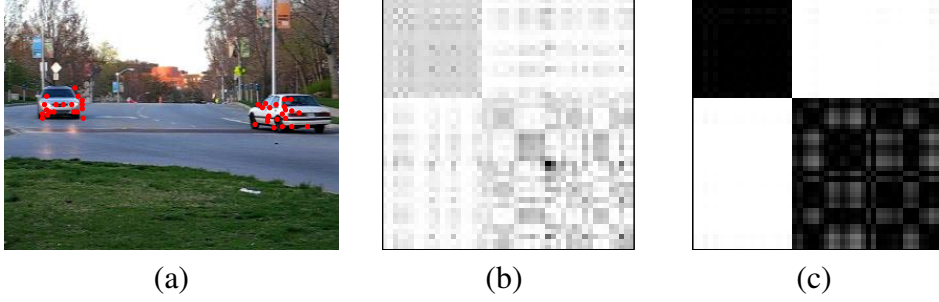


Figure 4.1: **Subspace clustering example:** (a) Two motions, each forming one subspace; (b) Shape Interaction Matrix of the trajectories in (a), which is sensitive to noise; (c) Affinity matrix obtained by our method: a much clearer block-diagonal structure.

tion). While effective, this algorithm still assumes complete data. To overcome this limitation, we then introduce an approach to estimating the row space of the data matrix in the presence of missing data via an efficient iterative update on the Grassmann manifold.

We demonstrate the effectiveness of our algorithms on motion segmentation and face clustering in different scenarios, including the presence of noise, outliers and missing data. Our experiments evidence the benefits of our approach over existing methods in all these scenarios.

4.1 SIM Revisited: Review and Analysis

The Shape Interaction Matrix (SIM) was originally introduced by Costeira and Kanade [1995] to extend Tomasi and Kanade [1992]’s groundbreaking work on factorization-based structure-from-motion from a single motion to the multi-body case. In the single-motion scenario, the trajectory matrix $\mathbf{X} \in \mathbb{R}^{2F \times N}$ (for N points in F frames) can be factorized into the product of a motion matrix $\mathbf{M} \in \mathbb{R}^{2F \times 4}$ and a shape matrix $\mathbf{S} \in \mathbb{R}^{4 \times N}$ with metric (rotation and translation) constraints. However, for multi-body motions, the metric constraints no longer directly apply, but require the knowledge of the membership of each point to each motion.

In their work, Costeira and Kanade showed that these motion memberships could be obtained from the data itself. To this end, they introduced the SIM, defined as

$$\mathbf{Q} = \mathbf{V}_r \mathbf{V}_r^T, \quad (4.1)$$

where $\mathbf{V}_r \in \mathbb{R}^{N \times r}$ is the matrix containing the first r right singular vectors of $\mathbf{X}$, with $r = 4K$ in the case of K non-degenerate motions. Mathematically, the SIM is the *orthogonal projection matrix* onto the *column space* of $\mathbf{V}_r$, or, equivalently, onto the *row space* of $\mathbf{X}$. Importantly, it can be shown that $\mathbf{Q}_{ij} = 0$ if points i and j belong to

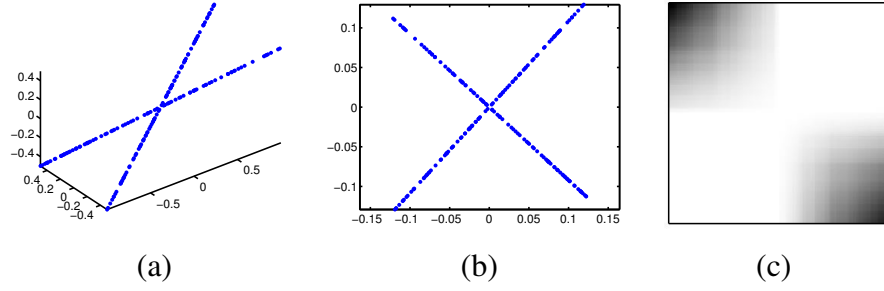


Figure 4.2: **SIM for clustering two lines in 3D:** (a) Two lines (each forming one subspace) with an arbitrary angle; (b) New data representation with $\mathbf{V}_r$. Note that the lines have become orthogonal; (c) SIM (absolute value) normalized by its maximum value; the darker the SIM image, the greater the value.

different motions, and $\mathbf{Q}_{ij} \neq 0$ if points i and j belong to the same motion. Therefore, in (Costeira and Kanade [1995]), segmentation was achieved by block-diagonalizing $\mathbf{Q}$, which at the time involved an expensive operation.

Intuitively, we can think of $\mathbf{V}_r$ as a new data representation of the original $\mathbf{X}$, with each row of $\mathbf{V}_r$ a data point. Then, the theory of SIM shows that different independent subspaces become *orthogonal* to each other in the new representation. In Figure 4.2, we demonstrate this via a toy example.

Theoretically, consider the data matrix $\mathbf{X}_k$ for the k^{th} motion (or subspace) with $\text{rank}(\mathbf{X}_k) = r_k$ and compute its compact SVD $\mathbf{X}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T$. Then we have

$$\begin{aligned} \mathbf{X} &= [\mathbf{X}_1 \quad \cdots \quad \mathbf{X}_K] \\ &= [\mathbf{U}_1 \Sigma_1 \quad \cdots \quad \mathbf{U}_K \Sigma_K] \begin{bmatrix} \mathbf{V}_1^T & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{V}_K^T \end{bmatrix}. \end{aligned} \quad (4.2)$$

When the K subspaces are independent, the columns of $[\mathbf{U}_1 \Sigma_1 \quad \cdots \quad \mathbf{U}_K \Sigma_K]$ are linearly independent so that $r = r_1 + \cdots + r_K$. Therefore, the columns of $\text{diag}(\mathbf{V}_1, \dots, \mathbf{V}_K)$ span the row space of $\mathbf{X}$. The SIM, defined as the row space projection matrix of $\mathbf{X}$, can then be written as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{V}_1 \mathbf{V}_1^T & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{V}_K \mathbf{V}_K^T \end{bmatrix}, \quad (4.3)$$

which has block-diagonal structure¹.

The main drawback of the SIM arises from the fact that, while it yields provably

¹This is true when the data points are sorted according to the subspaces. When they are not, $\mathbf{Q}$ can then be block-diagonalized with certain permutations.

correct clusters for independent motions and noise-free measurements, its accuracy decreases in the presence of noise, outliers, or degenerate motions. Over the years, many methods have therefore been proposed to improve the SIM. In the remainder of this section, we review these methods in a rough chronological order.

4.1.1 The Pre-Spectral-Clustering Era

Earlier approaches to accounting for noise, outliers and degeneracies (Gear [1994]; Costeira and Kanade [1998]; Gear [1998]; Ichimura [1999]; Kanatani [2001, 2002]; Wu et al. [2001]; Zelnik-Manor and Irani [2003]) were mostly focused on modifications of the SIM itself, or on directly related formulations. For instance, Gear [1998] advocated the use of the reduced row echelon method instead of the SVD to better account for noise and automatically find the rank of the trajectory matrix. Wu et al. [2001] presented an orthogonal subspace decomposition method to make the SIM more robust to noise by reasoning at group-level instead of considering individual point trajectories.

From a more general perspective, Kanatani [2001, 2002] reformulated motion segmentation as a subspace separation problem, and showed that, under the condition that the subspaces are linearly independent, the SIM is block-diagonal (up to a permutation of the data). Later, Zelnik-Manor and Irani [2003] considered the degenerate cases of the motion segmentation problem when the motions are not independent. They analyzed the causes of these degeneracies and proposed to overcome some of them by using the eigenvectors $\mathbf{E} = [\mathbf{e}_1^T \cdots \mathbf{e}_N^T]^T$ of the row-normalized matrix $\mathbf{X}^T \mathbf{X}$ and constructing a new shape interaction matrix as $\mathbf{Q}_{ij} = \sum_{k=1}^r \exp((\mathbf{e}_i(k) - \mathbf{e}_j(k))^2)$.

4.1.2 The Post-Spectral-Clustering Era

An important advance in the subspace clustering research was achieved by Park et al. [2004], who, based on the then recent success of spectral clustering methods (Shi and Malik [2000]; Ng et al. [2002]), showed that the absolute value of the SIM could be employed as an affinity matrix in spectral clustering, thus yielding more accurate results than much more sophisticated methods, such as (Kanatani and Sugaya [2003]). This then moved the focus of the subspace clustering community away from the SIM (at least in appearance, as discussed below) and towards designing better affinity matrices for spectral clustering.

In this context, Yan and Pollefeys [2006] introduced a Local Subspace Affinity (LSA) measure to build affinity matrices. LSA measures the affinity between two points as the principal angle between their local subspaces. Instead of using the original data points $\mathbf{X}$, LSA represents the data with the row-normalized singular vectors $\mathbf{V}$ of $\mathbf{X}$. More recently, Lauer and Schnorr [2009] proposed a spectral-clustering-based method that directly relies on the angles between the data points. As in LSA, instead

Algorithm 4.1 The Shape Interaction Matrix Method

Input: The data matrix $\mathbf{X} \in \mathbb{R}^{2F \times N}$, with F the number of frames and N the number of points

1. Construct the matrix $\mathbf{V}$ whose columns are the first r right singular vectors of $\mathbf{X}$.
2. Construct the Shape Interaction Matrix as $\mathbf{Q} = \mathbf{V}\mathbf{V}^T$.
3. Block diagonalize $\mathbf{Q}$ to get the final segmentation, or, nowadays, apply normalized cuts or spectral clustering using $|\mathbf{Q}|$ as an affinity matrix.

Output: The cluster labels $\mathbf{s}$

of computing the angles from the original data, they also represented the data with its normalized singular vectors.

The recent trends in the subspace clustering literature exploit the notion of self-expressiveness of the data to build affinity matrices (Elhamifar and Vidal [2009, 2013]; Liu et al. [2010, 2013]). In particular, SSC (Elhamifar and Vidal [2009, 2013]) minimized the ℓ_1 norm of self-expression coefficient matrix $\mathbf{C}$; Low Rank Representation (LRR) (Liu et al. [2010, 2013]) the nuclear norm of $\mathbf{C}$; and EDSC, the method presented the previous chapter, the Frobenius norm of $\mathbf{C}$.

Interestingly, in the previous chapter, we have shown that LRR and EDSC are equivalent to the SIM in the noise-free case. The difference lies in their ability to handle noise and outliers via additional regularization terms in their objective functions. Note that, even in the noisy case, it was shown (Peng et al. [2015]) that the optimal solutions of LRR and EDSC take the form $\mathbf{V}\mathcal{P}(\Sigma)\mathbf{V}^T$, where $\mathcal{P}(\cdot)$ denotes the shrinkage-thresholding operator, and therefore essentially correspond to a modified version of the SIM. More importantly, the effect of these regularizers is sensitive to their weights, which therefore need to be tuned for the data at hand.

By contrast, in this chapter, we introduce an efficient subspace clustering method that is directly motivated by the SIM, but does not require additional regularization terms to handle data corruptions. More specifically, we show how the SIM can be robustified to data corruptions via three simple steps. We then further introduce an algorithm that robustly recovers the row-space of the data from incomplete measurements, thus effectively making the powerful SIM representation applicable to the missing data scenario.

4.2 SIM Robustified: Corrupted Data

In this section, we introduce a robust subspace clustering method inspired by the SIM, but that lets us handle corrupted measurements. Before presenting our contribution, we first summarize the original SIM method in Algorithm 4.1.

Interestingly, as pointed out by Weiss [1999], a closer look at the SIM method

Algorithm 4.2 The Scott and Longuet-Higgins Algorithm

Input: An affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$

- Step 2-1. Construct the matrix $\mathbf{V}$ whose columns are the first K eigenvectors of $\mathbf{A}$.
- Step 2-2. Normalize each row of $\mathbf{V}$ so that they have unit norm, $\mathbf{V}(i,:) = \mathbf{V}(i,:) / \|\mathbf{V}(i,:)\|$.
- Step 2-3. Construct the matrix $\mathbf{Q} = \mathbf{V}\mathbf{V}^T$, and set the negative elements of $\mathbf{Q}$ to zero if necessary.
- Step 2-4. Cluster the points by examining the elements of $\mathbf{Q}$. Ideally, $\mathbf{Q}(i,j) = 1$ if i and j belong to the same cluster, and $\mathbf{Q}(i,j) = 0$ if they belong to different clusters.

Output: The cluster labels $\mathbf{s}$

shows that it bears strong similarities with the well-known Scott and Longuet-Higgins (SLH) relocalization algorithm (Scott and Longuet-Higgins [1990]), which is a widely applicable method for feature clustering using the eigenvectors of an affinity matrix. As shown in Algorithm 4.2, given an affinity matrix $\mathbf{A}$ and the number of clusters K , the SLH algorithm builds a new matrix $\mathbf{Q}$ to perform clustering.

Apart from the fact that SLH utilizes an affinity matrix as input instead of the data matrix, a comparison of the two algorithms reveals that they are identical up to two additional steps performed by SLH: (i) row-wise normalization of $\mathbf{V}$; and (ii) truncation of the negative elements. These differences motivate us to modify the SIM to potentially make it more robust.

To make the SIM method robust to data corruptions, we design a series of three steps: (i) row normalization of $\mathbf{V}_r$, (ii) elementwise powering of the new SIM, and (iii) determining the best rank r . While the first two steps aim at making direct modifications to the SIM, the third step is designed to account for degenerate cases, e.g., planar motions. In the remainder of this section, we present these steps and explain the rationale behind them.

4.2.1 Row Normalization

A closer look at the SIM method reveals that there is a *magnitude bias* within it, i.e., although the inter-cluster (subspace) affinities are guaranteed to be zero, the intra-cluster (subspace) affinities depend on the magnitude of the data points. More specifically, for points drawn from the same subspace, the affinities between those that are closer to the origin will be smaller than between those that are further away. For example, in Fig 4.2(c), the affinity values are much smaller in the center (i.e., points close to the origin) than in the corners. However, *ideally all points on the same subspace should be treated equally, since they belong to the same class*. Moreover, this magnitude bias is also undesirable because it makes the points close to the origin more sensitive to noise.

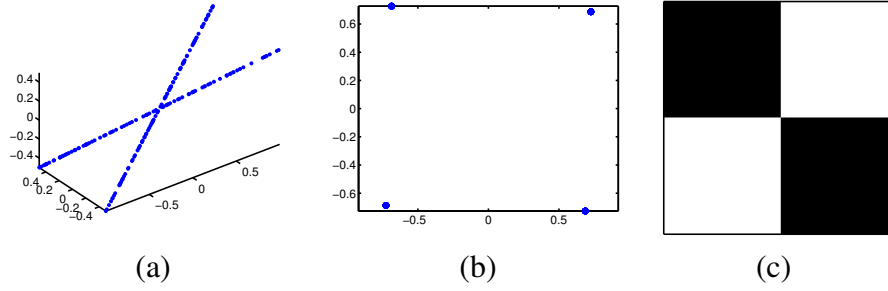


Figure 4.3: **Clustering two lines in 3D: Row normalization** (a) Two lines (each forming one subspace) with an arbitrary angle; (b) New data representation with row-normalized $\mathbf{V}_r$. Note that the lines collapse four points on the unit circle, corresponding to orthogonal vectors; (c) New SIM (absolute value) without magnitude bias.

To avoid the magnitude bias, we introduce an extra step, row normalization of $\mathbf{V}_r$, so that all data points in the new representation have the same magnitude 1. As a consequence, in an ideal scenario, the new SIM will become uniform within each subspace, as illustrated in Figure 4.3.

4.2.2 Elementwise Powering

In an ideal scenario (i.e., without noise), after row normalization the inter-cluster affinities are all zero and the intra-cluster affinities all one. However, in noisy cases, the elements of the affinity matrix (i.e., the absolute value of the new SIM) lie in the interval $[0, 1]$, and the inter-cluster affinities are often nonzero, but have rather small values. *Elementwise powering of the new SIM will thus virtually suppress these small values while keeping the large affinities mostly unaffected.* This operation is quite intuitive and just aims to denoise the SIM (affinity matrix). It was first used by Lauer and Schnorr [2009] to increase the gap between inter-cluster and intra-cluster affinities. The result of this step is illustrated in Figure 4.4. Since, after row normalization, the data in $\mathbf{V}_r$ always have a similar magnitude, independently of the problem of interest, the same powering factor can always be employed, thus preventing the need to tune a parameter for the data at hand. From our experiments, we observed that values in $[3, 5]$ generally yield good results. Note that when a non-integer powering factor is used, we take the real part of the resultant affinities, which implicitly removes the negative elements of the original SIM.

Remark *The first two steps are simple and intuitive. However, we can also interpret them in a more theoretical way with kernel methods. Note that each row of $\mathbf{V}_r$ is a point in the new data representation, so the SIM $\mathbf{Q} = \mathbf{V}_r \mathbf{V}_r^T$ indeed consists of the inner product of every pair of these points. Now we show that the first two steps, i.e. row normalization and elementwise powering, are equivalent to applying*

a normalized polynomial kernel. Given a polynomial kernel $\kappa(\mathbf{v}_i, \mathbf{v}_j) = (\mathbf{v}_i^T \mathbf{v}_j)^\alpha$ and its corresponding feature mapping ϕ , the normalized polynomial kernel $\hat{\kappa}(\mathbf{v}_i, \mathbf{v}_j)$ corresponds to the feature map

$$\mathbf{v}_i \longrightarrow \frac{\phi(\mathbf{v}_i)}{\|\phi(\mathbf{v}_i)\|}. \quad (4.4)$$

Then the kernel $\hat{\kappa}$ can be expressed as

$$\hat{\kappa}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\phi(\mathbf{v}_i^T) \cdot \phi(\mathbf{v}_j)}{\|\phi(\mathbf{v}_i)\| \|\phi(\mathbf{v}_j)\|} \quad (4.5)$$

$$= \frac{\kappa(\mathbf{v}_i, \mathbf{v}_j)}{\sqrt{\kappa(\mathbf{v}_i, \mathbf{v}_i) \kappa(\mathbf{v}_j, \mathbf{v}_j)}} \quad (4.6)$$

$$= \frac{(\mathbf{v}_i^T \mathbf{v}_j)^\alpha}{\sqrt{(\mathbf{v}_i^T \mathbf{v}_i)^\alpha (\mathbf{v}_j^T \mathbf{v}_j)^\alpha}} \quad (4.7)$$

$$= \left(\frac{\mathbf{v}_i^T \mathbf{v}_j}{\sqrt{(\mathbf{v}_i^T \mathbf{v}_i) (\mathbf{v}_j^T \mathbf{v}_j)}} \right)^\alpha \quad (4.8)$$

$$= \left(\frac{\mathbf{v}_i^T}{\|\mathbf{v}_i\|} \frac{\mathbf{v}_j}{\|\mathbf{v}_j\|} \right)^\alpha, \quad (4.9)$$

where the last equality indicates two steps, i.e. normalization and powering.

4.2.3 Rank Determination

Determining the correct rank r is crucial for the success of the SIM method. As early as in (Gear [1998]), it was shown that the SIM yields poor results if the incorrect rank is employed. Therefore, several approaches to determining the correct rank have been studied. In (Costeira and Kanade [1998]) the rank was obtained by examining the gaps in the singular values, which is typically sensitive to the level of noise. Inspired by the sparse representation community, ALC (Ma et al. [2007]) uses the sparsity-preserving dimension $d_{\text{sp}} = \min d \text{ s.t. } d \geq 2D \log(2F/d)$, where D is the estimated intrinsic dimensionality of each subspace. SC (Lauer and Schnorr [2009]) estimates the rank by looking at the relative eigenvalue gaps of the Laplacian matrix.

Here, we draw inspiration from the matrix perturbation theory and introduce a simple, yet effective method to detect the correct rank of the SIM. In general, one can easily define a range of possible ranks $[r_{\min}, r_{\max}]$. Our rank selection method then works by simply exhaustively searching over all possible rank values, and selecting

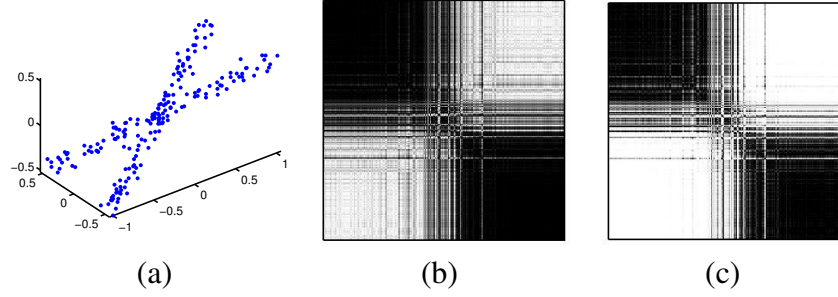


Figure 4.4: **Clustering two lines with noise in 3D:** (a) Two lines with Gaussian noise; (b) New SIM after row normalization, with noise in the off-diagonal blocks; (c) Affinity matrix after elementwise powering. Note that the block-diagonal structure is much cleaner.

the r which minimizes

$$C(r) = \frac{\text{minCut}(A_1^r, \dots, A_K^r)}{|\lambda_K - \lambda_{K+1}|}, \quad (4.10)$$

where A_i^r is the i^{th} cluster of the graph defined by the affinity matrix $\mathbf{A}^r$, λ_i is the i^{th} largest eigenvalue of the Laplacian matrix $\mathbf{L}^r = \mathbf{D}^{-1}\mathbf{A}^r$ (where $\mathbf{D}$ is the degree matrix of $\mathbf{A}^r$), and the minimal cut $\text{minCut}(A_1^r, \dots, A_K^r)$ can be obtained via the Ncuts algorithm (Shi and Malik [2000]). Intuitively, the smaller the cost of minCut and the larger the eigengap, the better the segmentation.

Our rank selection criterion can be justified by the Davis-Kahan Theorem from the matrix perturbation theory, which provides an upper bound on the distance between the eigenspaces of two Hermitian matrices that differ by some perturbations. This theorem is stated below.

Theorem 4.1 (Davis-Kahan Theorem) *Let $\mathbf{L}$ and $\tilde{\mathbf{L}}$ be two N -by- N Hermitian matrices. Let $\{\lambda_1, \dots, \lambda_k, \lambda_{k+1}, \dots, \lambda_N\}$ ($\lambda_i \geq \lambda_j, i < j$) denote the eigenvalues of $\mathbf{L}$, and $\mathbf{U}_1$ the matrix containing its first k eigenvectors. Let $\{\tilde{\lambda}_1, \dots, \tilde{\lambda}_k, \tilde{\lambda}_{k+1}, \dots, \tilde{\lambda}_N\}$ and $\tilde{\mathbf{U}}_1$ be the analogous quantities for $\tilde{\mathbf{L}}$. Then, by defining $\sigma := \min_{1 \leq i \leq k, 1 \leq j \leq n-k} |\lambda_i - \tilde{\lambda}_{k+j}|$, we have*

$$\|\sin \Theta(\mathbf{U}_1, \tilde{\mathbf{U}}_1)\|_F \leq \frac{\|\tilde{\mathbf{L}} - \mathbf{L}\|_F}{\sigma}, \quad (4.11)$$

where $\Theta(\mathbf{U}_1, \tilde{\mathbf{U}}_1)$ is the vector of principal angles between $\mathbf{U}_1$ and $\tilde{\mathbf{U}}_1$.

Proof. The proof of this theorem is given in (Davis and Kahan [1970]). $\square$

The Davis-Kahan Theorem states that the distance between the eigenspaces of two Hermitian matrices that differ by some perturbations is bounded by the ratio between the perturbation level and their eigengap. In our case, since we do not have access to

Algorithm 4.3 Robust Shape Interaction Matrix (RSIM)**Input:** Data matrix $\mathbf{X}$, minimum rank $r_{\min}$, and maximum rank $r_{\max}$ **for** $r := r_{\min}$ **to** $r_{\max}$ **do**

1. *SVD*: Compute the SVD of the data matrix $\mathbf{X}$, i.e., $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, and take the first r right singular vectors $\mathbf{V}_r$.
2. *Normalization*: Normalize each row of $\mathbf{V}_r$ to have unit norm $\rightarrow \tilde{\mathbf{V}}_r$.
3. *New SIM*: Build the new Shape Interaction Matrix as $\mathbf{Q} = \tilde{\mathbf{V}}_r \tilde{\mathbf{V}}_r^T$.
4. *Powering*: Take the elementwise power of $\mathbf{Q}$, i.e., $\mathbf{A}_{ij} = (\mathbf{Q}_{ij})^\gamma$.
5. *Rank Determination*: Apply the normalized cuts algorithm to get the cluster labels, and compute the value $C(r)$ as in Eq. (4.10).

end for $r_{\text{best}} = \underset{r}{\operatorname{argmin}} C(r).$ **Output:** The cluster labels $\mathbf{s}$, the best rank r_{best} .

the true Laplacian, we make use of the eigenvalues of the noisy Laplacian to estimate the eigengap σ , which will then occur between the K^{th} and $K + 1^{\text{th}}$ eigenvalues for K clusters. Furthermore, we rely on minCut to approximate the noise level of the Laplacian matrix $\mathbf{L}$. This approximation is reasonable because $\mathbf{L}$ is nothing but a normalized version of the affinity matrix. So by minimizing $C(r)$, we aim to find the lowest upper bound of the distance between the noisy Laplacian and the true one. This minimum should correspond to the optimal rank.

4.2.4 Robust Shape Interaction Matrix

Our complete Robust Shape Interaction Matrix (RSIM) algorithm is outlined in Algorithm 4.3. Note that, while its steps are simple, to the best of our knowledge, it is the first time that such an algorithm is proposed. Furthermore, our experiments clearly evidence the effectiveness of RSIM and its benefits over more sophisticated methods, such as SSC and LRR.

4.3 SIM Robustified: Missing Data

Our previous solution to handling data corruption relies on the computation of the row space $\mathbf{V}$ of the data $\mathbf{X}$. When the data contains missing entries, computing the row-space cannot simply be achieved by SVD. Here, we exploit the idea that our goal truly is to estimate the subspace on which the data lies (which $\mathbf{V}$ is an orthogonal basis of). Linear subspaces of a fixed dimension form a Riemannian manifold known as the Grassmannian. Therefore, we propose to make use of an optimization technique on the Grassmann manifold to obtain an estimate of $\mathbf{V}$ in the presence of missing data.

More formally, let $\mathcal{G}(N, r)$ denote the Grassmann manifold of r -dimensional linear subspaces of $\mathbb{R}^N$ (Chikuse [2003]). A point $\mathbf{Y} \in \mathcal{G}(N, r)$, i.e., an r -dimensional subspace of $\mathbb{R}^N$, can be represented by any orthogonal matrix $\mathbf{V} \in \mathbb{R}^{N \times r}$ whose columns span the r -dimensional subspace $\mathbf{Y}$. Estimating the row space $\mathbf{V}$ (an orthogonal matrix) of the data matrix can then be thought of as finding the corresponding linear subspace on $\mathcal{G}(N, r)$.

To estimate $\mathbf{V}$, we utilize the GROUSE (Grassmannian Rank-One Update Subspace Estimation) algorithm (Balzano et al. [2010]). GROUSE is an efficient online algorithm that recovers the column space of a highly incomplete observation matrix. To this end, it utilizes a gradient descent method on the Grassmannian to incrementally update the subspace by considering one column of the observation matrix at a time.

More specifically, in our context, at each iteration t , we take as input a vector $\mathbf{x}_{\Omega_t} \in \mathbb{R}^{N_t}$, which corresponds to the partial observation of a single vector $\mathbf{x}_t \in \mathbb{R}^N$ in the data matrix $\mathbf{X}$,² with observed indices defined by $\Omega_t \subset \{1, \dots, N\}$. Let $\mathbf{V}_{\Omega_t}$ be the submatrix of $\mathbf{V}$ consisting of the rows indexed by Ω_t . Following the GROUSE formalism, which relies on the least-squares reconstruction of the data, we can formulate the update at iteration t as the solution to the optimization problem

$$\min_{\mathbf{V} \in \mathcal{G}(N, r), \mathbf{a}} \frac{1}{2} \|\mathbf{V}_{\Omega_t} \mathbf{a} - \mathbf{x}_{\Omega_t}\|_2^2 \quad (4.12)$$

where $\mathbf{a}$ corresponds to the representation (or weights) of the data $\mathbf{x}_{\Omega_t}$ in the current estimate of the subspace.

Since (4.12) is not jointly convex in $\mathbf{a}$ and $\mathbf{V}$, the two variables are obtained in a sequential manner: First, the optimal weights $\mathbf{w} = \mathbf{a}^*$ are computed for the current subspace, and then the subspace is updated given those weights. Due to the least-squares form of the objective function, the solution for the weights can be obtained in closed-form as $\mathbf{w} = \mathbf{V}_{\Omega_t}^\dagger \mathbf{x}_{\Omega_t}$, where $\mathbf{V}_{\Omega_t}^\dagger$ is the pseudoinverse of $\mathbf{V}_{\Omega_t}$. To update the subspace, i.e., the orthogonal basis matrix $\mathbf{V}$, GROUSE exploits an incremental gradient descent method on the Grassmann manifold, which we describe below.

Let $\mathbf{I}_{\Omega_t} \in \mathbb{R}^{N \times N_t}$ be the N_t columns of the $N \times N$ identity matrix indexed by Ω_t . Then, the objective function of (4.12) can be rewritten as

$$E_t = \frac{1}{2} \|\mathbf{I}_{\Omega_t} (\mathbf{V}_{\Omega_t} \mathbf{w} - \mathbf{x}_{\Omega_t})\|_2^2. \quad (4.13)$$

The update of the subspace is achieved by taking a step in the direction of the gradient of this objective function on the Grassmannian, i.e., moving along the geodesic defined by the negative Grassmannian gradient. To this end, we first need to compute the regular gradient of the objective function with respect to $\mathbf{V}$. This gradient can be

²Note that even though we consider $\mathbf{x}_t$ to be a column vector, it really corresponds to one row of the data matrix $\mathbf{X}$.

written as

$$\frac{\partial E_t}{\partial \mathbf{V}} = -(\mathbf{I}_{\Omega_t}(\mathbf{x}_{\Omega_t} - \Omega_t \mathbf{w})) \mathbf{w}^T \quad (4.14)$$

$$= -\mathbf{r} \mathbf{w}^T, \quad (4.15)$$

where $\mathbf{r} = \mathbf{I}_{\Omega_t}(\mathbf{x}_{\Omega_t} - \mathbf{V}_{\Omega_t} \mathbf{w})$ denotes the (zero-padded) vector of residuals.

The gradient on the Grassmannian can then be obtained by projecting the regular gradient on the tangent space of the Grassmannian at the current point. Following Edelman et al. [1998], this can be written as

$$\nabla E_t = (\mathbf{I} - \mathbf{V} \mathbf{V}^T) \frac{\partial E_t}{\partial \mathbf{V}} \quad (4.16)$$

$$= -(\mathbf{I} - \mathbf{V} \mathbf{V}^T) \mathbf{r} \mathbf{w}^T \quad (4.17)$$

$$= -\mathbf{r} \mathbf{w}^T. \quad (4.18)$$

As shown in (Edelman et al. [1998]) (or Theorem 2.2 in Chapter 2), a gradient step along the geodesic with tangent vector $-\nabla E_t$ is defined as a function of the singular values and vectors of ∇E_t . Since ∇E_t has rank one, its SVD is trivial to compute. The compact SVD of $-\nabla E_t$ can be written as

$$-E_t = \frac{\mathbf{r}}{\|\mathbf{r}\|} \times \|\mathbf{r}\| \|\mathbf{w}\| \times \left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \right)^T \quad (4.19)$$

$$= \mathbf{p}_1 \sigma \mathbf{q}_1^T, \quad (4.20)$$

with $\mathbf{p}_1 = \frac{\mathbf{r}}{\|\mathbf{r}\|}$, $\sigma = \|\mathbf{r}\| \|\mathbf{w}\|$ and $\mathbf{q}_1 = \frac{\mathbf{w}}{\|\mathbf{w}\|}$.

Let $\mathbf{p}_2, \dots, \mathbf{p}_N$ be the orthonormal set orthogonal to $\mathbf{p}_1$, and $\mathbf{q}_2, \dots, \mathbf{q}_N$ be the orthonormal set orthogonal to $\mathbf{q}_1$. Then the full SVD of $-E_t$ can be written as

$$-E_t = \mathbf{P} \Sigma \mathbf{Q}^T \quad (4.21)$$

$$= [\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_N] \times \text{diag}(\sigma, 0, \dots, 0) \times [\mathbf{q}_1 \ \mathbf{q}_2 \ \dots \ \mathbf{q}_r]^T, \quad (4.22)$$

with $\mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_N] \in \mathbb{R}^{N \times r}$, $\Sigma = \text{diag}(\sigma, 0, \dots, 0) \in \mathbb{R}^{r \times r}$, and $\mathbf{Q} = [\mathbf{q}_1 \ \mathbf{q}_2 \ \dots \ \mathbf{q}_r] \in \mathbb{R}^{r \times r}$. Clearly, $\mathbf{P}^T \mathbf{P} = \mathbf{I}_{r \times r}$, $\mathbf{P} \mathbf{P}^T = \mathbf{I}_{N \times N}$, and $\mathbf{Q}^T \mathbf{Q} = \mathbf{Q} \mathbf{Q}^T = \mathbf{I}_{r \times r}$.

Following Eq. (2.5), we update $\mathbf{V}$ with a step size η as

$$\begin{aligned} \mathbf{V}(\eta) &= [\mathbf{V} \mathbf{Q} \ \mathbf{P}] \begin{bmatrix} \cos(\Sigma \eta) \\ \sin(\Sigma \eta) \end{bmatrix} \mathbf{Q}^T \\ &= \mathbf{V} \mathbf{Q} \cos(\Sigma \eta) \mathbf{Q}^T + \mathbf{P} \sin(\Sigma \eta) \mathbf{Q}^T \end{aligned}$$

$$\begin{aligned}
&= \mathbf{V}\mathbf{Q}\cos(\text{diag}(\sigma\eta, 0, \dots, 0))\mathbf{Q}^T + \\
&\quad \mathbf{P}\sin(\text{diag}(\sigma\eta, 0, \dots, 0))\mathbf{Q}^T \\
&= \mathbf{V}\mathbf{Q}\text{diag}(1, 1, \dots, 1)\mathbf{Q}^T + \\
&\quad \mathbf{V}\mathbf{Q}\text{diag}(\cos(\sigma\eta) - 1, 0, \dots, 0)\mathbf{Q}^T + \\
&\quad \mathbf{P}\text{diag}(\sin(\sigma\eta), 0, \dots, 0)\mathbf{Q}^T \\
&= \mathbf{V}\mathbf{Q}\mathbf{I}\mathbf{Q}^T + (\cos(\sigma\eta) - 1)\mathbf{V}\mathbf{q}_1\mathbf{q}_1^T + \sin(\sigma\eta)\mathbf{p}_1\mathbf{q}_1^T \\
&= \mathbf{V} + (\cos(\sigma\eta) - 1)\mathbf{V}\mathbf{q}_1\mathbf{q}_1^T + \sin(\sigma\eta)\mathbf{p}_1\mathbf{q}_1^T \\
&= \mathbf{V} + \frac{(\cos(\sigma\eta) - 1)}{\|\mathbf{w}\|^2}\mathbf{V}\mathbf{w}\mathbf{w}^T + \sin(\sigma\eta)\frac{\mathbf{r}}{\|\mathbf{r}\|}\frac{\mathbf{w}^T}{\|\mathbf{w}\|}.
\end{aligned}$$

In short, the update of $\mathbf{V}$ at time t is given by

$$\mathbf{V}_{t+1} = \mathbf{V}_t + \frac{(\cos(\sigma\eta) - 1)}{\|\mathbf{w}\|^2}\mathbf{V}_t\mathbf{w}\mathbf{w}^T + \sin(\sigma\eta)\frac{\mathbf{r}}{\|\mathbf{r}\|}\frac{\mathbf{w}^T}{\|\mathbf{w}\|}. \quad (4.23)$$

The Grassmannian update is very efficient since each subspace update only involves linear operations. Furthermore, for a specific diminishing step-size η , it is guaranteed to converge to a locally optimal estimate of $\mathbf{V}$ (Balzano et al. [2010]). After getting an estimate of $\mathbf{V}$ using this method, we can directly apply the RSIM to perform subspace clustering.

The pseudocode of our robust SIM with missing data (RSIM-M) algorithm is given in Algorithm 4.4. Note that:

1. Stochastic gradient descent may require a relatively large number of steps to be stable. With small amounts of data, we run multiple passes over the data. For example, in our experiments on motion segmentation with incomplete trajectories, we iterated over all the frames 100 times. Thanks to the high efficiency of rank-one Grassmannian update, RSIM-M remains very efficient.
2. Due to the non-convexity of this problem, initialization is important for convergence speed and optimality. In practice, we start with the subspace spanned by the most complete r rows of $\mathbf{X}$.

4.4 Experimental Evaluation

We evaluate the performance of our algorithms with four sets of experiments that represent different scenarios: (i) Hopkins155 for motion segmentation; (ii) Extended Yale Face B for face clustering; (iii) Hopkins12Real: 12 additional real-world sequences with missing data; (iv) Hopkins outdoor sequences for semi-dense motion segmentation. We compare the results of our algorithms with the following baselines: SIM (followed by spectral clustering) (Park et al. [2004]), SSC (Elhamifar and Vidal

Algorithm 4.4 RSIM with Missing Data (RSIM-M)

Input: An incomplete data matrix $\mathbf{X}$, a subspace initialization $\mathbf{V}_0$, a step size η , bounds $r_{\min}, r_{\max}$

for $t = 1, \dots, T$ **do**

1. Take the t^{th} row of $\mathbf{X}$ with observed entry Ω_t ;
2. Update the current $\mathbf{V}_t$ via Eq. (4.23);

end for

Run Algorithm 4.3 to perform robust subspace clustering.

Output: The cluster labels $\mathbf{s}$, the best rank r_{best} .

[2013]), LRSC (Vidal and Favaro [2014]), LRR (Liu et al. [2013]), and EDSC in Chapter 3. Note that the last two methods have proposed to make use of an additional post-processing step (called a heuristic in Elhamifar and Vidal [2013]), which yields the additional baselines LRR-H and EDSC-H. For the case of LRR, this heuristic relies on the following steps:

1. Solve the optimization problem

$$\min_{\mathbf{C}, \mathbf{E}} \|\mathbf{C}\|_* + \lambda \|\mathbf{E}\|_{2,1} \quad \text{s.t.} \quad \mathbf{X} = \mathbf{XC} + \mathbf{E}. \quad (4.24)$$

2. Compute the SVD of $\mathbf{C}$, i.e. $\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^T$, and take the first r singular vectors $\mathbf{V}_r$.
3. Construct $\mathbf{Z} = \mathbf{V}_r \Sigma_r^{\frac{1}{2}}$, and normalize each row of $\mathbf{Z}$.
4. Build the affinity matrix $\mathbf{A}$ as $\mathbf{ZZ}^T$ with elementwise powering such that $\mathbf{A}_{ij} = [\mathbf{ZZ}^T]_{ij}^4$.

Interestingly, this post-processing is nothing else but another way to build an improved SIM. Indeed, $\mathbf{Z}$ can be thought of approximately as the row space of the denoised data $\mathbf{X} - \mathbf{E}$ from the equality constraint in (4.24). In other words, one can also think of LRR (and EDSC) as a pre-processing step to denoise the data before computing the SIM. In contrast, the proposed method does not require any pre-processing step and, as evidenced below, achieves much better results.

The parameters of the baselines are tuned to the best results for each experiment. For our method, we report the results of all the four sets of experiments with the same powering factor $\gamma = 4.5$. Note that we could potentially get better results if we fine-tuned the parameter γ . For motion segmentation, the rank is selected iteratively from the integers in $[K, 4K]$; for the face clustering experiment, the rank is in $[4K, 6K]$ with K the number of clusters. The step size η of the Grassmannian gradient method is set as 0.01 in our experiments.

Table 4.1: Clustering error (in %) on Hopkins 155.

Methods	SIM	SSC	LRR	LRR-H	EDSC	EDSC-H	RSIM
2 motions							
Mean	6.50	1.53	4.10	2.13	2.67	0.86	0.65
Median	1.14	0.00	0.22	0.00	0.00	0.00	0.00
3 motions							
Mean	12.26	4.40	9.89	4.03	8.06	2.49	1.71
Median	6.12	6.22	0.56	1.43	2.53	0.21	0.28
Overall							
Mean	7.80	2.18	5.41	2.56	4.04	1.23	0.89
Median	1.53	0.00	0.53	0.00	0.30	0.00	0.00

4.4.1 Hopkins155: Complete Data with Noise

The results of our RSIM algorithm and of the baselines on Hopkins155 (Tron and Vidal [2007]) are reported in Table 4.1. Note that our method achieves the lowest overall average clustering error.

We also performed an ablation study on this dataset to see the contributions of the proposed steps. We denote the SIM with our first two steps (i.e., normalization and polynomial kernel) by SIM+1&2, and denote the SIM with our third step (i.e., rank determination) by SIM+3. The results are shown in Table 4.2. Note that the proposed first two steps improve the motion segmentation accuracy over the original SIM, the proposed third step boosts the segmentation results with a big margin, and our complete robust shape interaction matrix method achieves the best results.

Table 4.2: Ablation study on Hopkins 155.

Methods	SIM	SIM+1&2	SIM+3	RSIM
Mean	7.80	5.77	3.17	0.89
Median	1.53	0.24	0.31	0.00

4.4.2 Extended Yale B: Complete Data with Outliers

We follow exactly the same experimental settings as in the previous chapter. Note that, since this data is grossly corrupted, the baselines (Elhamifar and Vidal [2013]; Liu et al. [2013]; Ji et al. [2014]) use an additional regularizer to account for outliers, with weight specifically tuned for this dataset. In contrast, our method doesn't have this extra term and parameter. The results are presented in Table 4.3. Interestingly, although our method does not handle the outliers explicitly, it achieves the comparable accuracies for 2 and 3 subjects, and get far better accuracies for 5, 8 and 10 subjects. In contrast to the baselines, our method remains stable as the number of

Table 4.3: Clustering error (in %) on Extended Yale B.

Methods	SIM	SSC	LRR	LRR-H	EDSC	EDSC-H	RSIM
2 subjects							
Mean	8.10	1.86	9.52	2.54	5.42	2.65	2.36
Median	6.25	0.00	5.47	0.78	4.69	1.56	1.56
3 subjects							
Mean	24.64	3.10	19.52	4.21	14.05	3.86	3.21
Median	16.67	1.04	14.58	2.60	8.33	3.13	2.60
5 subjects							
Mean	45.62	4.31	34.16	6.90	36.99	5.11	3.56
Median	48.13	2.50	35.00	5.63	30.63	3.75	3.13
8 subjects							
Mean	57.05	5.85	41.19	14.34	54.24	6.07	3.60
Median	55.96	4.49	43.75	14.34	48.73	4.88	3.32
10 subjects							
Mean	65.10	10.94	38.85	22.92	59.58	7.24	3.70
Median	64.06	5.63	41.09	23.59	50.47	6.09	3.44

subjects increases. From a different perspective, this dataset can be thought of as being contaminated with both Gaussian noise and Laplacian noise, so the baseline methods (SSC, LRR and EDSC) all have two regularization terms, one for the Gaussian noise and the other for the Laplacian one, and their weight parameters, therefore, need to be tuned for the data at hand. In contrast, our method relies on no specific assumptions about the distributions of the noise, and is thus robust to a mixture of different types of noise.

4.4.3 Hopkins12Real: Incomplete Data with Noise

To demonstrate that our method can handle missing data gracefully, we employed the Hopkins 12 additional sequences containing incomplete data and noise. Most of the baselines used previously cannot deal with missing data. Therefore, we only compare our method with those that have proposed to tackle this challenging scenario. In particular, we compare our results against those published by Rao et al. [2008], where ALC was employed after filling in the missing entries of the data matrix with a matrix completion method, e.g., Power Factorization (PF) (Hartley and Schaffalitzky [2003]), Robust Principal Component Analysis (RPCA) (Candès et al. [2011]), and ℓ_1 sparse representation (Rao et al. [2008]). We also evaluate SSC (Elhamifar and Vidal [2009, 2013]), which works with missing data by either removing the trajectories with missing entries (SSC-R), or treating the missing entries as outliers (SSC-O). In contrast, our method doesn't require any matrix completion or trajectory removal. The results in Table 4.4 clearly evidence the benefits of our method in the presence of

Table 4.4: Clustering error (in %) on Hopkins 12 Real Motion Sequences with Incomplete Data.

%	PF+ALC	RPCA+ALC	ℓ_1 +ALC	SSC-R	SSC-O	RSIM-M
Mean	10.81	13.78	1.28	3.82	8.78	0.61
Median	7.85	8.27	1.07	0.31	4.80	0.61
Max	34.57	41.36	4.35	20.25	26.34	1.64
Std	0.04	12.25	1.29	6.80	8.79	0.53

missing data.

4.4.4 Hopkins Outdoor: Semi-dense, Incomplete Data with Outliers

To study a more realistic scenario, where outliers and missing data are ubiquitous due to occlusions and tracking failures, we took 18 outdoor sequences from the Hopkins155 dataset and obtained semi-dense trajectories by applying the tracking method of Wang et al. [2011a]³. For the 18 sequences, the tracking method found an average of 3026 trajectories per sequence, among which 16.66% (684 out of 3026) on average contained missing entries, which were set to zero. We compare our results to those of the same SSC-O and SSC-R baselines used previously.

Since there is no ground-truth for this data, we can only provide a qualitative comparison. In particular, we observed that our method performed either better, or on par with SSC-R, and consistently outperformed SSC-O. We found that SSC-O tends to group the trajectories with missing entries in a single cluster. This is mainly due to the fact that, according to the self-expressiveness criterion, incomplete trajectories are poorly represented by complete ones, and thus end up being grouped together. Figure 4.5 shows some typical behaviors of SSC-R and of our approach. It can easily be checked that our approach yields better clusters on average. The results of SSC-O are shown in Figure 4.6, where the behavior described above can be observed. Finally, in Figure 4.7, we show some failure cases where both SSC-R and our approach were unable to find the right clusters. The results for all the sequences are provided in the appendix of this chapter. Since SSC-R removes the missing trajectories, it utilized only 2522 trajectories on average out of the original average of 3026. In contrast, our method makes use of all the available trajectories. Nonetheless, while SSC-R takes 150.48 seconds per sequence on average, our method only takes about 5.22 seconds on average.

³While there are 21 outdoor videos in Hopkins155, the tracking code that we used was unable to read the 3 Kanatani videos.

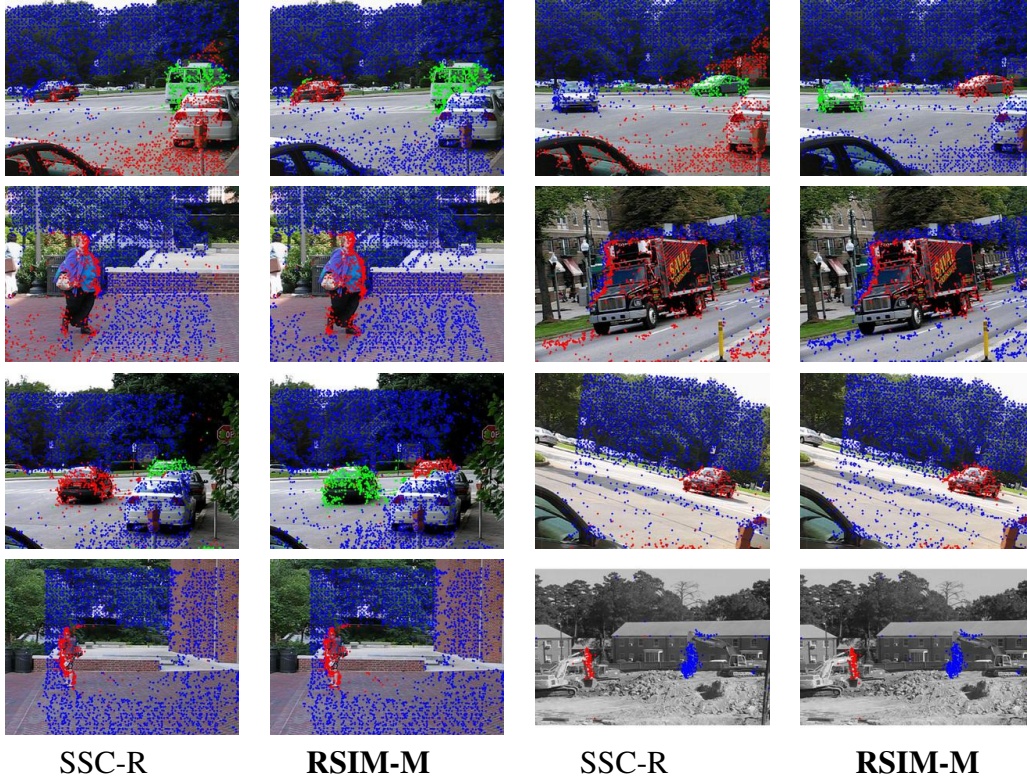


Figure 4.5: **Comparison of SSC-R and RSIM-M on semi-dense data:** While SSC-R removes the trajectories with missing entries, and thus gets less dense results, our method can handle missing data robustly. Each image is a frame sampled from one of the video sequences. The points marked with the same color are clustered into the same group by the respective methods. Best viewed in color.

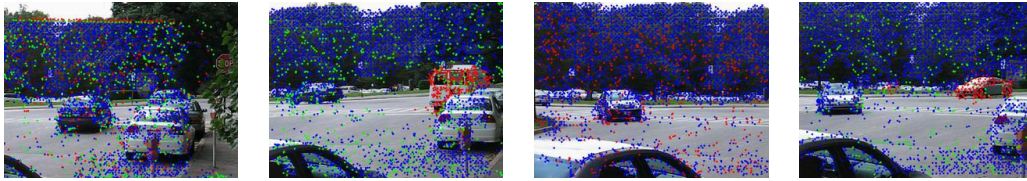


Figure 4.6: **Typical behavior of SSC-O on semi-dense data:** By treating missing entries as outliers, SSC-O tends to cluster the trajectories with missing entries into same group. The points marked with the same color are clustered into the same group by SSC-O. Best viewed in color.

4.5 Summary

In this chapter, we have revealed that many recent subspace clustering methods actually did not go far beyond the 20-year-old SIM method, but rather had indirect connections to it. While recent methods exploit notions of compressed sensing and self-expressiveness, our method performs simple and direct modifications of the SIM

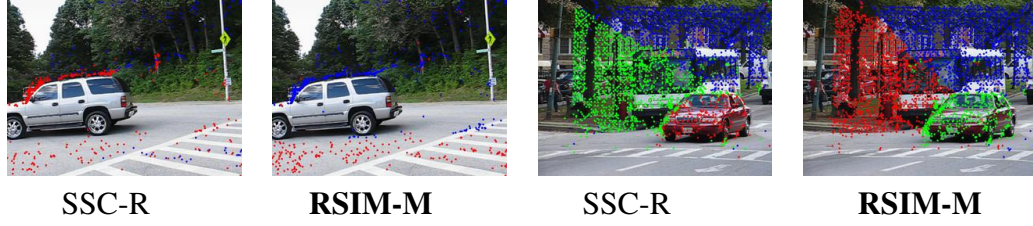


Figure 4.7: **Failure cases of SSC-R and of RSIM-M:** We conjecture that these failures are due to tracking failures (e.g., very few trajectories), or to high dependence between motions. Best viewed in color.

itself and makes it robust to corruptions. Furthermore, we have extended our method to the case of missing data. Our experimental evaluation has demonstrated that our algorithms are not only efficient, but also generally applicable to subspace segmentation in realistic scenarios. Compared to EDSC in the previous chapter, the RSIM method in this chapter can handle data corruptions of more general forms including the case of missing data. Moreover, the RSIM method does not have the weight parameters as in the self-expressiveness based algorithms (e.g., EDSC, LRR and SSC), which need to be tuned for the data at hand.

Appendix – Hopkins Outdoor: Semi-dense, Incomplete Data with Outliers

Here, we show the results of our algorithm (RSIM-M) and of the baselines SSC-O and SSC-R on all the 18 sequences described in Section 4.4.4 of this chapter.



Figure 4.8: Semi-dense motion segmentation results for sequences 1-4. Our method (RSIM-M) uses all the available tracks (**3026 on average**) with an **average runtime of 5.22 seconds per sequence**; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.



Figure 4.9: Semi-dense motion segmentation results for sequences 5-10. Our method (RSIM-M) uses all the available tracks (**3026 on average**) with an **average runtime of 5.22 seconds per sequence**; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.



Figure 4.10: Semi-dense motion segmentation results for sequences 11-16. Our method (RSIM-M) uses all the available tracks (**3026 on average**) with an **average runtime of 5.22 seconds per sequence**; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.

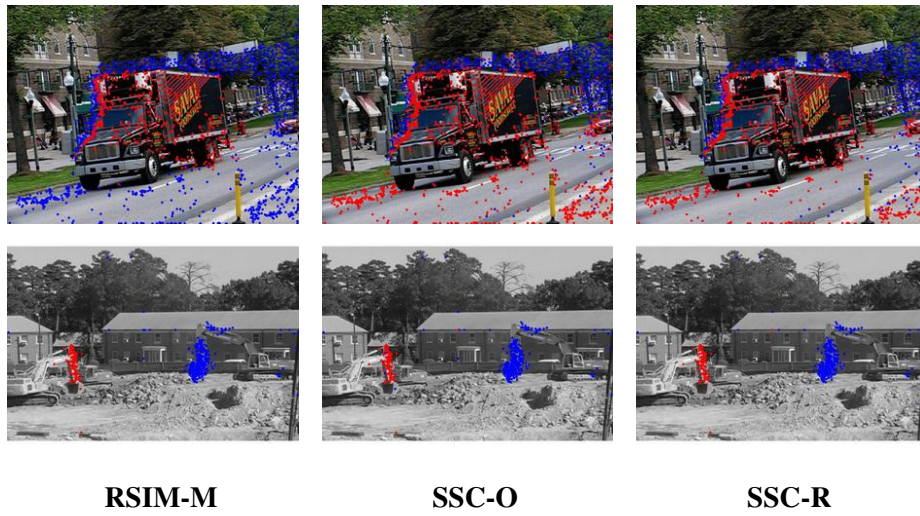


Figure 4.11: Semi-dense motion segmentation results for sequences 17-18. Our method (RSIM-M) uses all the available tracks (**3026 on average**) with an **average runtime of 5.22 seconds per sequence**; SSC-O tends to group the trajectories with missing entries in the same cluster; SSC-R takes 150.48 seconds on average and only makes use of 2522 points on average after removing the incomplete trajectories. Best viewed in color.

Robust Motion Segmentation with Unknown Correspondences

In Chapter 3 and Chapter 4, we have presented two efficient subspace clustering methods, i.e., EDSC and RSIM, which can be applied to solve the motion segmentation problem given the point trajectories over the frames. However, motion segmentation is still far from being a solved problem. Indeed, most existing methods (Elhamifar and Vidal [2013]; Liu et al. [2013]; Vidal and Favaro [2014]; Lu et al. [2012]) including EDSC and RSIM in previous two chapters, assume that point trajectories are available as input. For example, in the Hopkins155 dataset (Tron and Vidal [2007]), perfect trajectories were obtained by manually cleaning the trajectories throughout the sequences. Such manual intervention is, of course, impractical in many scenarios. While some methods are robust to a small amount of outliers (e.g., (Elhamifar and Vidal [2013]; Liu et al. [2013])), their performance quickly degrades as the number of mismatches increases. In practice, interest point detection and correspondence estimation are challenging tasks. Inevitable outliers and missing data make the problem even harder.

While research in the area of point correspondence estimation has also been progressing (Oliveira et al. [2005]; Zeng et al. [2012]; Jia et al. [2015]), existing methods are not being considered in the context of motion segmentation. Therefore, they cannot benefit from constraints associated with the problem. In particular, when multiple motions are observed, the underlying point trajectories should lie in a union of subspaces.

In this chapter, we introduce an approach for motion segmentation with unknown correspondences (see Figure 5.1). In contrast to existing techniques that proceed in two stages (i.e., first correspondence estimation and then motion segmentation), this allows us to **(i)** benefit from the motion segmentation constraints throughout the entire process; and **(ii)** not require any pre-processing stage to clean up the trajectories used for motion segmentation, and thus be robust to outliers and missing observations. This, we believe, is a crucial step towards making motion segmentation applicable to more realistic scenarios.

More specifically, given interest points extracted independently in all the frames



Figure 5.1: Motion segmentation with unknown correspondences: given a video sequence and feature points in each frame, our goal is to simultaneously estimate motion clusters and feature matching.

of a video sequence comprising both inliers and outliers, we exploit the constraint that data lying in the union of subspaces should be self-expressive. In other words, a trajectory lying in a subspace can be expressed as a linear combination of the other trajectories in the same subspace. We therefore search for Partial Permutation Matrices (PPMs) and combination coefficients that automatically select and reorder the inlier points so as to make them self-expressive. Furthermore, we make use of the fact that matched feature descriptors have a similar appearance across the frames and thus, when correctly arranged, should form a low-rank matrix. To obtain a solution to the resulting optimization problem, we employ the Alternating Direction Method of Multipliers (ADMM), and show that each subproblem can be solved efficiently. Given the combination coefficients, we can then separate the different motions by normalized cuts (Shi and Malik [2000]) or spectral clustering, as in (Elhamifar and Vidal [2013]; Liu et al. [2013]; Vidal and Favaro [2014]; Lu et al. [2012]).

We demonstrate the robustness and effectiveness of our method on several real sequences. Our experimental evaluation evidences the benefits of our formulation over sequentially solving correspondence estimation and motion segmentation, as is done by existing approaches. Importantly, on Hopkins155, our formulation with unknown correspondences achieves competitive results with the state-of-the-art motion segmentation methods that exploit perfect trajectories as input.

5.1 Background

Over the years, many techniques (Amiaz and Kiryati [2006]; Cremers and Soatto [2005b]; Li [2007]; Volz et al. [2011]; Sun et al. [2012]; Ochs et al. [2014]; Elhamifar and Vidal [2013]; Liu et al. [2013]; Lu et al. [2012]) have been proposed to tackle the problem of motion segmentation. These techniques can be roughly categorized into those working with dense observations, and those tackling the sparse points case.

In the dense scenario, the use of optical flow has been investigated to separate the motion of different objects observed in two frames (Cremers and Soatto [2005b]; Amiaz and Kiryati [2006]), or in very short sequences (Volz et al. [2011]; Sun et al. [2012]). Dense point trajectories were also employed for motion segmentation in

longer videos (Ochs et al. [2014]).

Our work is more directly related to methods that perform motion segmentation of sparse trajectories (Costeira and Kanade [1995]; Zelnik-Manor and Irani [2003]; Yan and Pollefeys [2006]). In particular, our approach in this chapter draws inspiration from the recent subspace clustering literature (Elhamifar and Vidal [2013]; Liu et al. [2013]; Lu et al. [2012]) and our method EDSC in Chapter 3, and relies on the self-expressiveness property of the point trajectories.

Whether dense or sparse, trajectory-based methods all assume that the correspondence problem has been solved beforehand, and that the trajectories are thus given as input. While some advances have been made towards handling outliers (Elhamifar and Vidal [2013]; Liu et al. [2013]) and incomplete point tracks (Sivic et al. [2006]; Cheriadat and Radke [2009]), the resulting techniques still require relatively clean data to yield good accuracy.

Ultimately, motion segmentation methods strongly rely on the accuracy of point correspondences. These correspondences can typically be obtained by independently matching local feature descriptors (Lowe [2004]; Mikolajczyk and Schmid [2004]), or by making use of the temporal nature of the data to track sparse (Shi and Tomasi [1994]) or dense (Brox and Malik [2011]) image points. Rather than treating each point independently, several methods have been proposed to jointly find nonrigid correspondences between two sets of points (Leordeanu and Hebert [2005]; Torresani et al. [2008]; Duchenne et al. [2009]). In (Torki and Elgammal [2010]), this is achieved by combining point location and appearance.

In the case of a single rigid motion, it was shown that correspondence estimation can be expressed as a rank-minimization problem in terms of PPMs (Oliveira et al. [2005]). Indeed, when correctly organized, the trajectories of rigidly moving points form a rank 4 matrix. While, in (Oliveira et al. [2005]), this was achieved by incrementally incorporating one frame at a time, which is subject to error propagation, this idea was pursued for Robust Object Matching using low-rank and sparse constraints (ROML) (Zeng et al. [2012]; Jia et al. [2015]), where a whole sequence was treated at once. Furthermore, in (Zeng et al. [2012]; Jia et al. [2015]), this framework was extended to minimizing the rank of a matrix built from feature descriptors, thus making the approach applicable to more general correspondence problems.

However, while attractive, general solutions to the correspondence problem, such as (Torki and Elgammal [2010]; Zeng et al. [2012]; Jia et al. [2015]) do not permit taking into account the specific constraints of the task at hand. Here, in contrast, we introduce an approach that jointly performs correspondence estimation and motion segmentation, and can thus incorporate the subspace constraints in motion segmentation throughout the whole process. As a result, not only does it yield high segmentation accuracy, but it also improves correspondence estimation.

5.2 Problem Formulation with PPMs

We now present our approach to robust motion segmentation with unknown correspondences. Intuitively, we seek to select the inlier input points and reorder them such that they satisfy the subspace constraints (i.e., the point trajectories lie in a union of subspaces) and appearance constraints (i.e., the matched feature points have similar appearance across the images).

More specifically, let $\mathbf{w}_{fi} \in \mathbb{R}^{2 \times 1}$ be the 2D location of point i detected in frame f of an F -frame sequence depicting multiple motions. Furthermore, let $\mathbf{t}_{fi} \in \mathbb{R}^{d \times 1}$ be the appearance descriptor of the same point in the same frame. The locations of all points in frame f can be concatenated in a $2 \times N_f$ position matrix $\mathbf{W}_f = [\mathbf{w}_{f1}, \dots, \mathbf{w}_{fN_f}]$. Similarly, we can group all feature descriptors in frame f in a $d \times N_f$ matrix $\mathbf{T}_f = [\mathbf{t}_{f1}, \dots, \mathbf{t}_{fN_f}]$.

In the absence of point correspondence across the frames, and even if we assume $N_f = N$, $\forall f$, simply stacking up of all the position matrices $\{\mathbf{W}_f\}_{f=1}^F$ does not yield valid point trajectories. However, there exists a reordering of the columns of the position matrix in each frame that yields coherent point trajectories¹. Furthermore, when applied to the descriptor matrices $\{\mathbf{T}_f\}_{f=1}^F$, this ordering should also make the appearance of corresponding features coherent across the frames. In the presence of outliers, i.e., $N_f \neq N_{f'}$, $f \neq f'$, this process should not only reorder the points, but also select the inliers.

Following (Oliveira et al. [2005]; Cheriyyadat and Radke [2009]; Zeng et al. [2012]), we utilize Partial Permutation Matrices (PPMs) to model this reordering. Let $\mathbf{P}_f \in \{0, 1\}^{N_f \times N}$ denote the PPM that selects and reorders the N inlier point coordinates in frame f . Given the F PPMs $\{\mathbf{P}_f\}_{f=1}^F$, we define the trajectory matrix as

$$\mathbf{D}_c = [(\mathbf{W}_1 \mathbf{P}_1)^T | \dots | (\mathbf{W}_F \mathbf{P}_F)^T]^T. \quad (5.1)$$

In an ideal, noise-free scenario, there exist PPMs such that the trajectory matrix $\mathbf{D}_c$ satisfies the subspace constraints. In practice, to account for the noise of the measurements, we decompose $\mathbf{D}_c$ into a clean measurement matrix $\mathbf{L}_c$ and a noise matrix $\mathbf{E}_c$. This can be written as

$$\mathbf{D}_c = \mathbf{L}_c + \mathbf{E}_c. \quad (5.2)$$

As shown in (Elhamifar and Vidal [2013]; Liu et al. [2013]), the fact that trajectories lie in a union of subspaces can equivalently be formulated in terms of self-expressiveness of the data. Note, however, that self-expressiveness only holds in the noise-free case. In our formulation, we therefore make use of the clean measurement

¹Note that this global reordering is subject to an ambiguity, since the order of the trajectories themselves is irrelevant. This, however, can easily be solved by fixing the order in the first frame of the sequence.

matrix $\mathbf{L}_c$ to encode self-expressiveness. This yields

$$\mathbf{L}_c = \mathbf{L}_c \mathbf{C}, \quad (5.3)$$

where each clean trajectory is represented as a linear combination of the other trajectories, with $\mathbf{C}$ storing the (unknown) combination coefficients. In the presence of affine subspaces, an additional constraint of the form $\mathbf{1}_N^T \mathbf{C} = \mathbf{1}_N^T$, where $\mathbf{1}_N$ is a column vector of 1s, can be further imposed.

For feature appearance, we define the $Nd \times F$ descriptor matrix obtained from the PPMs $\{\mathbf{P}_f\}_{f=1}^F$ as

$$\mathbf{D}_d = [\text{vec}(\mathbf{T}_1 \mathbf{P}_1) | \cdots | \text{vec}(\mathbf{T}_F \mathbf{P}_F)], \quad (5.4)$$

where $\text{vec}(\cdot)$ vectorizes its matrix argument in a columnwise manner.

Since, in a noise-free scenario, each specific feature point should have the same appearance in all the frames, $\mathbf{D}_d$ should have low rank (ideally rank one). To tackle the more realistic case of noisy measurements, however, we decompose $\mathbf{D}_d$ into a clean low rank component $\mathbf{L}_d$ and a noise component $\mathbf{E}_d$. Therefore, we have

$$\mathbf{D}_d = \mathbf{L}_d + \mathbf{E}_d. \quad (5.5)$$

In this formalism, our goal is to propose a formulation to motion segmentation with unknown correspondences that satisfies the following requirements:

1. The matrix of clean inlier point trajectories $\mathbf{L}_c$ should be self-expressive.
2. The matrix of clean inlier feature descriptors $\mathbf{L}_d$ should have low-rank.
3. Noise and outliers in both point locations and feature descriptors must be accounted for.

While Point 1 is partially accounted for by the constraint in Eq. 5.3, it is crucial to prevent non-zero coefficients in $\mathbf{C}$ for any two trajectories belonging to different motions. Indeed, to perform motion segmentation, $\mathbf{C}$ needs to reflect the membership of the trajectories to their respective subspace. As shown in Chapter 3, this can be achieved with any p -norm regularizer on $\mathbf{C}$. Here, in particular, we make use of the Frobenius norm, which is convex and easy to minimize. To address Point 2 in our requirements, we propose to search for the $\mathbf{L}_d$ with minimum rank. To this end, we employ a nuclear norm regularizer on $\mathbf{L}_d$, which is a convex surrogate to the rank function. Finally, Point 3 is addressed in two different ways. First, outliers in the point locations are accounted for by the PPMs. Second, to model further noise in the locations and in the descriptors, which we expect to be sparse, we make use of ℓ_1 regularizers on $\mathbf{E}_c$ and $\mathbf{E}_d$ as convex surrogates to the ℓ_0 norm.

Integrating all these constraints, we express motion segmentation with unknown correspondences as the optimization problem

$$\begin{aligned}
& \min_{\{\mathbf{P}_f\}_{f=1}^F, \mathbf{C}, \mathbf{L}_{c,d}, \mathbf{E}_{c,d}} \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda_1 \|\mathbf{L}_d\|_* + \lambda_2 \|\mathbf{E}_c\|_1 + \lambda_3 \|\mathbf{E}_d\|_1 \quad (5.6) \\
& \text{s.t. } \mathbf{L}_c = \mathbf{L}_c \mathbf{C}, \quad \left(\mathbf{1}_N^T \mathbf{C} = \mathbf{1}_N^T \right), \\
& \quad \mathbf{D}_c = \mathbf{L}_c + \mathbf{E}_c, \quad \mathbf{D}_d = \mathbf{L}_d + \mathbf{E}_d, \\
& \quad \mathbf{1}_{N_f}^T \mathbf{P}_f = \mathbf{1}_N^T, \quad \mathbf{P}_f \mathbf{1}_N \leq \mathbf{1}_{N_f}, \quad \mathbf{P}_f \in \{0, 1\}^{N_f \times N},
\end{aligned}$$

where λ_1 , λ_2 , and λ_3 balance the different terms in the objective function, and where the constraints on $\{\mathbf{P}_f\}_{f=1}^F$ enforce these matrices to be PPMs. Note that these PPMs appear in (5.6) via the matrices $\mathbf{D}_c$ and $\mathbf{D}_d$ as can be seen from their definitions in Eqs. (5.1) and (5.4), respectively.

Note that, as shown in Chapter 3, under the self-expressiveness constraint $\mathbf{L} = \mathbf{L}\mathbf{C}$, $\min_{\mathbf{C}} \frac{1}{2} \|\mathbf{C}\|_F^2$ and $\min_{\mathbf{C}} \|\mathbf{C}\|_*$ are equivalent. With this observation, the Frobenius norm comes as a natural choice over the nuclear norm in (5.6) due to its computational simplicity.

Importantly, note that, in (5.6), the trajectory matrix $\mathbf{D}_c$ and the descriptor matrix $\mathbf{D}_d$ share the same PPMs $\{\mathbf{P}_f\}_{f=1}^F$. This induces a connection between motion segmentation and point correspondence, and thus makes the two problems work in a cooperative manner and help each other during the optimization procedure. As will be shown in our experiments, this collaboration not only yields accurate motion segmentation, but also improves the point correspondence results over methods dedicated to this task only.

5.2.1 Solving (5.6) via ADMM

Due to the discrete nature of PPMs, (5.6) is non-convex. Here, we propose to solve it via the ADMM, which has proven effective for many non-convex problems such as matrix separation (Shen et al. [2014]), non-negative matrix factorization (Zhang [2010]), and correspondence estimation (Zeng et al. [2012]). The ADMM works by decomposing the original optimization problem into several smaller subproblems, each of which can be solved efficiently.

We therefore seek to decompose (5.6) into several subproblems. With the ADMM, this is achieved by first deriving the augmented Lagrangian of (5.6), which can be expressed as

$$\begin{aligned}
\mathcal{L}_\rho(\{\mathbf{P}_f\}_{f=1}^F, \mathbf{C}, \mathbf{L}_{c,d}, \mathbf{E}_{c,d}, \{\mathbf{Y}_i\}_{i=1}^4) = & \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda_1 \|\mathbf{L}_d\|_* + \lambda_2 \|\mathbf{E}_c\|_1 + \lambda_3 \|\mathbf{E}_d\|_1 + \\
& \text{tr} \left(\mathbf{Y}_1^T (\mathbf{L}_c - \mathbf{L}_c \mathbf{C}) \right) + \text{tr} \left(\mathbf{Y}_2^T (\mathbf{1}_N^T \mathbf{C} - \mathbf{1}_N^T) \right) +
\end{aligned}$$

$$\begin{aligned} & \text{tr} \left(\mathbf{Y}_3^T (\mathbf{D}_c - \mathbf{L}_c - \mathbf{E}_c) \right) + \text{tr} \left(\mathbf{Y}_4^T (\mathbf{D}_d - \mathbf{L}_d - \mathbf{E}_d) \right) + \\ & \frac{\rho}{2} (\|\mathbf{D}_c - \mathbf{L}_c - \mathbf{E}_c\|_F^2 + \|\mathbf{L}_c - \mathbf{L}_c \mathbf{C}\|_F^2 + \\ & \|\mathbf{D}_d - \mathbf{L}_d - \mathbf{E}_d\|_F^2 + \|\mathbf{1}^T \mathbf{C} - \mathbf{1}^T\|_2^2), \end{aligned} \quad (5.7)$$

where $\{\mathbf{Y}_i\}_{i=1}^4$ are the matrices of Lagrange multipliers corresponding to the four constraints in (5.6), and ρ is the penalty parameter². Note that, although not explicitly written here, the constraints on the PPMs are maintained.

The ADMM then consists of iteratively updating the individual variables so as to minimize $\mathcal{L}_\rho$ while the other variables are fixed. In the following, we derive the update for each of our variables. We denote by a superscript t the current value of the variables and by a superscript $t + 1$ the new values.

Updating $\mathbf{L}_c$ and $\mathbf{C}$ in closed form

From Eq. (5.7), it can be seen that $\mathbf{L}_c$ only appears in linear and least-squares terms. Therefore, its update can easily be obtained in closed-form. More specifically, it is given by

$$\mathbf{L}_c^{t+1} = \left[\left(\mathbf{Y}_3^t + \mathbf{Y}_1^t (\mathbf{C}^{tT} - \mathbf{I}) \right) / \rho^t + \mathbf{D}_c^t - \mathbf{E}_c^t \right] \left[\mathbf{I} + (\mathbf{I} - \mathbf{C}^t)(\mathbf{I} - \mathbf{C}^{tT}) \right]^{-1}. \quad (5.8)$$

Similarly, all the terms involving $\mathbf{C}$ are simple linear and quadratic terms. This yields the closed-form update

$$\mathbf{C}^{t+1} = \left[\mathbf{I} + \rho^t (\mathbf{L}_c^{t+1T} \mathbf{L}_c^{t+1} + \mathbf{1} \mathbf{1}^T) \right]^{-1} \left(\mathbf{L}_c^{t+1T} \mathbf{Y}_1^t - \mathbf{1} \mathbf{Y}_2^t + \rho^t (\mathbf{L}_c^{t+1T} \mathbf{L}_c^{t+1} + \mathbf{1} \mathbf{1}^T) \right). \quad (5.9)$$

Updating $\mathbf{L}_d$, $\mathbf{E}_c$ and $\mathbf{E}_d$ by elementwise thresholding operators

Although not as straightforward, the updates for $\mathbf{L}_d$, $\mathbf{E}_c$ and $\mathbf{E}_d$ can still be computed efficiently. To this end, we note that these updates correspond to the solutions of the following optimization problems:

$$\mathbf{L}_d^{t+1} = \underset{\mathbf{L}_d}{\text{argmin}} \lambda_1 / \rho^t \|\mathbf{L}_d\|_* + 1/2 \|\mathbf{L}_d - (\mathbf{D}_d^t - \mathbf{E}_d^t + \mathbf{Y}_4^t / \rho^t)\|_F^2, \quad (5.10)$$

$$\mathbf{E}_c^{t+1} = \underset{\mathbf{E}_c}{\text{argmin}} \lambda_2 / \rho^t \|\mathbf{E}_c\|_1 + 1/2 \|\mathbf{E}_c - (\mathbf{D}_c^t - \mathbf{L}_c^{t+1} + \mathbf{Y}_3^t / \rho^t)\|_F^2, \quad (5.11)$$

$$\mathbf{E}_d^{t+1} = \underset{\mathbf{E}_d}{\text{argmin}} \lambda_3 / \rho^t \|\mathbf{E}_d\|_1 + 1/2 \|\mathbf{E}_d - (\mathbf{D}_d^t - \mathbf{L}_d^{t+1} + \mathbf{Y}_4^t / \rho^t)\|_F^2. \quad (5.12)$$

²Note that, for ease of notation, we have omitted explicitly writing the dimension of the vectors of all 1s, now all denoted by $\mathbf{1}$.

Problems (5.10), (5.11) and (5.12) are convex programs whose solutions can be obtained in closed-form. To this end, let us define the soft-thresholding operator (Cai et al. [2010]) $\mathcal{T}_\tau[x] = \text{sign}(x) \cdot \max(|x| - \tau, 0)$, which operates elementwise on scalars or matrices. The optimal solution to (5.10) can then be obtained as

$$\mathbf{L}_d^{t+1} = \mathbf{U} \mathcal{T}_{\lambda_1/\rho^t}(\Sigma) \mathbf{V}^T, \quad (5.13)$$

where $[\mathbf{U}, \Sigma, \mathbf{V}] = \text{svd}(\mathbf{D}_d^t - \mathbf{E}_d^t + \mathbf{Y}_4^t/\rho^t)$. The updates for $\mathbf{E}_c$ and $\mathbf{E}_d$ can be written as

$$\mathbf{E}_c^{t+1} = \mathcal{T}_{\lambda_2/\rho^t}(\mathbf{D}_c^t - \mathbf{L}_c^{t+1} + \mathbf{Y}_3^t/\rho^t). \quad (5.14)$$

$$\mathbf{E}_d^{t+1} = \mathcal{T}_{\lambda_3/\rho^t}(\mathbf{D}_d^t - \mathbf{L}_d^{t+1} + \mathbf{Y}_4^t/\rho^t). \quad (5.15)$$

Updating $\{\mathbf{P}_f\}_{f=1}^F$ with the Hungarian algorithm

The PPMs $\{\mathbf{P}_f\}_{f=1}^F$ are binary matrices, and thus updating them is non-trivial. Recall that, in Eq. (5.7), the PPMs appear via $\mathbf{D}_c$ and $\mathbf{D}_d$ only. Therefore, $\{\mathbf{P}_f\}_{f=1}^F$ can be updated by solving the problem

$$\begin{aligned} \min_{\{\mathbf{P}_f\}_{f=1}^F} & \|\mathbf{D}_c - (\mathbf{L}_c^{t+1} + \mathbf{E}_c^{t+1} - \mathbf{Y}_3^t/\rho^t)\|_F^2 + \|\mathbf{D}_d - (\mathbf{L}_d^{t+1} + \mathbf{E}_d^{t+1} - \mathbf{Y}_4^t/\rho^t)\|_F^2 \\ \text{s.t. } & \mathbf{1}^T \mathbf{P}_f = \mathbf{1}, \mathbf{P}_f \mathbf{1} \leq \mathbf{1}, \mathbf{P}_f \in \{0, 1\}^{N_f \times N}, \end{aligned} \quad (5.16)$$

where $\mathbf{D}_c = [(\mathbf{W}_1 \mathbf{P}_1)^T | \dots | (\mathbf{W}_F \mathbf{P}_F)^T]^T$, $\mathbf{D}_d = [\text{vec}(\mathbf{T}_1 \mathbf{P}_1) | \dots | \text{vec}(\mathbf{T}_F \mathbf{P}_F)]$ (as defined in Eqs. (5.1) and (5.4), respectively).

Problem (5.16) can then be decomposed into F independent subproblems, each of which only involves one PPM. The subproblem for frame f can be written as

$$\begin{aligned} \min_{\mathbf{P}_f} & \|\mathbf{W}_f \mathbf{P}_f - \alpha_f^T (\mathbf{L}_c^{t+1} + \mathbf{E}_c^{t+1} - \mathbf{Y}_3^t/\rho^t)\|_F^2 + \\ & \|\text{vec}(\mathbf{T}_f \mathbf{P}_f) - (\mathbf{L}_d^{t+1} + \mathbf{E}_d^{t+1} - \mathbf{Y}_4^t/\rho^t) \mathbf{e}_f\|_2^2 \\ \text{s.t. } & \mathbf{1}^T \mathbf{P}_f = \mathbf{1}, \mathbf{P}_f \mathbf{1} \leq \mathbf{1}, \mathbf{P}_f \in \{0, 1\}^{N_f \times N}, \end{aligned} \quad (5.17)$$

where $\mathbf{e}_f$ is a binary column vector with only the f^{th} element set to 1, and $\alpha_f = [\mathbf{e}_{2f-1} | \mathbf{e}_{2f}]$. Problem (5.17) turns out to be a binary assignment problem, which can be solved by the Hungarian algorithm in polynomial time (Munkres [1957]).

From Problem (5.17), it is not straightforward to get the cost matrix for the Hungarian algorithm. Note, however, that we have $\text{vec}(\mathbf{W}_f \mathbf{P}_f) = (\mathbf{I} \otimes \mathbf{W}_f) \text{vec}(\mathbf{P}_f)$, $\text{vec}(\mathbf{T}_f \mathbf{P}_f) = (\mathbf{I} \otimes \mathbf{T}_f) \text{vec}(\mathbf{P}_f)$, where $\otimes$ is the Kronecker product. Let us define $\mathbf{G}_f = \mathbf{I} \otimes \mathbf{W}_f$, $\mathbf{a}_f = \text{vec}[\alpha_f^T (\mathbf{L}_c^{t+1} + \mathbf{E}_c^{t+1} - \mathbf{Y}_3^t/\rho^t)]$, $\mathbf{J}_f = \mathbf{I} \otimes \mathbf{T}_f$, and $\mathbf{b}_f =$

$(\mathbf{M}^{t+1} + \mathbf{E}_2^{t+1} - \mathbf{Y}_3^t / \rho^t) \mathbf{e}_f$. Then Problem (5.17) becomes

$$\begin{aligned} \min_{\mathbf{P}_f} & \|\mathbf{G}_f \text{vec}(\mathbf{P}_f) - \mathbf{a}_f\|_2^2 + \|\mathbf{J}_f \text{vec}(\mathbf{P}_f) - \mathbf{b}_f\|_2^2 \\ \text{s.t. } & \mathbf{1}^T \mathbf{P}_f = \mathbf{1}, \mathbf{P}_f \mathbf{1} \leq \mathbf{1}, \mathbf{P}_f \in \{0, 1\}^{N_f \times N}. \end{aligned} \quad (5.18)$$

Therefore the assignment cost $\mathbf{Q}_f^1$ (or $\mathbf{Q}_f^2$) corresponding to the first (or second) term in (5.18) is the squared Euclidean distance between each column of $\mathbf{G}_f$ (or $\mathbf{J}_f$) and $\mathbf{a}_f$ (or $\mathbf{b}_f$). The total assignment cost is then $\mathbf{Q}_f = \mathbf{Q}_f^1 + \mathbf{Q}_f^2$. With this assignment cost, Problem (5.17) can be solved via the Hungarian algorithm.

Once we have computed the updates $\{\mathbf{P}_f^{t+1}\}_{f=1}^F$, $\mathbf{D}_c$ and $\mathbf{D}_d$ can be updated accordingly.

Updating the dual variables and the penalty parameter

Finally, the Lagrange multipliers $\{\mathbf{Y}_i\}_{i=1}^4$ and ρ can be updated as

$$\mathbf{Y}_1^{t+1} = \mathbf{Y}_1^t + \rho(\mathbf{L}_c - \mathbf{L}_c \mathbf{C}), \quad (5.19)$$

$$\mathbf{Y}_2^{t+1} = \mathbf{Y}_2^t + \rho(\mathbf{1}^T \mathbf{C} - \mathbf{1}^T), \quad (5.20)$$

$$\mathbf{Y}_3^{t+1} = \mathbf{Y}_3^t + \rho(\mathbf{D}_c - \mathbf{L}_c - \mathbf{E}_c), \quad (5.21)$$

$$\mathbf{Y}_4^{t+1} = \mathbf{Y}_4^t + \rho(\mathbf{D}_d - \mathbf{L}_d - \mathbf{E}_d). \quad (5.22)$$

$$\rho^{t+1} = \min(\eta \rho^t, \rho_m), \quad (5.23)$$

where $\eta > 1$ and ρ_m is the predefined maximum for ρ .

The process of iteratively updating all the variables is repeated until convergence, or until a maximum number of iterations is reached. Note that, while the ADMM does not have theoretical guarantee of global convergence for non-convex problems, in our experiments, we find it always converges to the correct solution for our problem. The empirical convergence of our algorithm will be discussed in Section 5.3. Our algorithm for motion segmentation with unknown correspondences is summarized in Algorithm 5.1.

5.3 Experimental Evaluation

To evaluate the effectiveness of our approach for motion segmentation with unknown point correspondences, we conducted extensive experiments on both synthetic data and real images. In total, we performed four different sets of experiments, which we discuss below. To measure/compare the performance of different algorithms, we use

Algorithm 5.1 Motion segmentation without correspondences via the ADMM**Input:**

Position matrices $\{\mathbf{W}_f\}_{f=1}^F$, descriptor matrices $\{\mathbf{T}_f\}_{f=1}^F$,
 $\lambda_1, \lambda_2, \lambda_3, \eta > 1, \rho_m, \epsilon$;

Initialize: $\mathbf{C}^0, \{\mathbf{P}_f^0\}_{f=1}^F, \mathbf{E}_c^0 = \mathbf{0}, \mathbf{E}_d^0 = \mathbf{0}, \{\mathbf{Y}_i^0\}_{i=1}^4 = \mathbf{0}, \rho^0$;

while not converged **do**

1. Update $(\mathbf{L}_c, \mathbf{C}, \mathbf{L}_d, \mathbf{E}_c, \mathbf{E}_d)$ by Eq. (5.8), Eq. (5.9), Eq. (5.13), Eq. (5.14) and Eq. (5.15);
2. Update $\{\mathbf{P}_f\}_{f=1}^F$ by solving F binary assignment problems (5.17) using the Hungarian algorithm, and then update $\mathbf{D}_c$ and $\mathbf{D}_d$ accordingly;
3. Update $\{\mathbf{Y}_i\}_{i=1}^4$ and ρ by Eq. (5.19)-Eq. (5.23);
4. Check the convergence conditions $\|\mathbf{L}_c - \mathbf{L}_c \mathbf{C}\|_\infty \leq \epsilon, \|\mathbf{1}^T \mathbf{C} - \mathbf{1}^T\|_\infty \leq \epsilon, \|\mathbf{D}_c - \mathbf{L}_c - \mathbf{E}_c\|_\infty \leq \epsilon$ and $\|\mathbf{D}_d - \mathbf{L}_d - \mathbf{E}_d\|_\infty \leq \epsilon$;

end while

Output: Coefficient matrix $\mathbf{C}$, PPMs $\{\mathbf{P}_f\}_{f=1}^F$.

the following criteria: (i) Accuracy in motion segmentation, expressed as

$$\text{ACC}_{\text{ms}} = \frac{\# \text{ correctly segmented trajectories}}{\text{total \# trajectories}} \times 100\%, \quad (5.24)$$

and (ii) Accuracy in point correspondences, computed as

$$\text{ACC}_{\text{pc}} = \frac{1}{FN^2} \sum_{f=1}^F \|\mathbf{P}_f \circ \mathbf{P}_f^*\|_0 \times 100\%, \quad (5.25)$$

where $\{\mathbf{P}_f^*\}_{f=1}^F$ are the ground truth correspondences (in PPM matrix), $\circ$ denotes the element-wise (Hadamard) product, and $\|\cdot\|_0$ is the ℓ_0 norm which counts the number of non-zeros entries.

We tuned the respective parameters of baselines to achieve the best results, and empirically set $\lambda_1 = 1, \lambda_2 = 0.05, \lambda_3 = 5/\sqrt{N}$ in our algorithm.

5.3.1 Experiment-1: Synthetic Data, Noise-free Case

In this first set of experiments, we aim to study the convergence of our algorithm. In other words, we want to understand, with perfectly controlled inputs, whether or not the proposed algorithm converges; and if so, whether it converges to the correct solution.

To this end, we synthesized two motion matrices $\mathbf{M}_1, \mathbf{M}_2 \in \mathbb{R}^{2F \times 4}$ by simulating F random rotation and translations, and two independently moving objects with shape

matrices $\mathbf{S}_1, \mathbf{S}_2 \in \mathbb{R}^{4 \times N/2}$. This yields a total of N 3D points in motion. Under an affine camera model, the measurement matrix of the synthesized sequence can be computed as $\mathbf{X} = [\mathbf{M}_1 \mathbf{S}_1 | \mathbf{M}_2 \mathbf{S}_2] \in \mathbb{R}^{2F \times N}$. We chose $F = 25$ and $N = 40$ in all our synthetic experiments. In addition to $\mathbf{X}$, which contains point locations, we synthesized a 128-dimensional appearance vector (i.e., feature descriptor) for each feature point. The choice of 128D is only to conform to the convention of SIFT feature descriptors which will be used in all our real image experiments. We randomly generated 128-dimensional random vectors and required that the same feature point across multiple images has identical descriptors. Given this synthesized sequence, we randomly permuted the points, so that all the correspondence information is lost.

With this data, we tested our algorithm starting from two different initialization conditions (V1 and V2): (V1) Initialize the unknown permutation matrices $\{\mathbf{P}_f\}_{f=1}^F$ as the identity matrices (generally only 0-2% accuracy); (V2) Initialize the unknown permutation matrices $\{\mathbf{P}_f\}_{f=1}^F$ by a set of random permutations, under the constraint that 60% of the point correspondences are correct. The parameter of our method were set to (V1) $\rho_0 = 10^{-6}$, $\eta = 1.01$; (V2) $\rho_0 = 10^{-2}$, $\eta = 1.01$, which reflects the better initialization of (V2).

Figure 5.2(a) depicts typical convergence curves (observed over 10 random trials) corresponding to the two initialization cases. We report the objective function value ($\frac{1}{2}\|\mathbf{C}\|_F^2 + \lambda_1\|\mathbf{L}_d\|_* + \lambda_2\|\mathbf{E}_c\|_1 + \lambda_3\|\mathbf{E}_d\|_1$) and the primal residuals³ ($\|\mathbf{D}_c - \mathbf{L}_c - \mathbf{E}_c\|_F, \|\mathbf{L}_c - \mathbf{L}_c \mathbf{C}\|_F, \|\mathbf{D}_d - \mathbf{L}_d - \mathbf{E}_d\|_F, \|\mathbf{1}^T \mathbf{C} - \mathbf{1}^T\|_F$). Note that our algorithm converges to the same objective function value independently of initialization. Note also that better initialization leads to faster convergence. Here, the number of iterations reduces from 1800 to about 700. In terms of wall-clock time, this corresponds to a reduction from about 2 minutes to 46 seconds on a regular Core-i7 PC with 8GB of memory. In this experiment, the average ACC_{ms} and ACC_{pc} over the ten random tests were all 100% for both initializations. This shows that, starting from virtually no point correspondence information, our algorithm successfully recovers both the correct point matches (with 100% accuracy) and the correct motion segmentation results.

5.3.2 Experiment-2: Synthetic data, with Noise and Outliers

We then investigated the robustness of our algorithm to different amounts of outliers and noise, using synthetic data. In the same manner as above, we generated a 25-frame sequence of 40 points sampled from two independent motions, and drew feature vectors from i.i.d. Gaussian distributions. We then randomly permuted the point correspondences. We incrementally increased the number of outliers from 0 to 20 by adding gross errors to both the point coordinates and the feature descriptors. In addition to outliers, we also added zero-mean Gaussian noise with standard deviation σ to the inliers.

³Here we plot the maximum value of the four primal residuals.

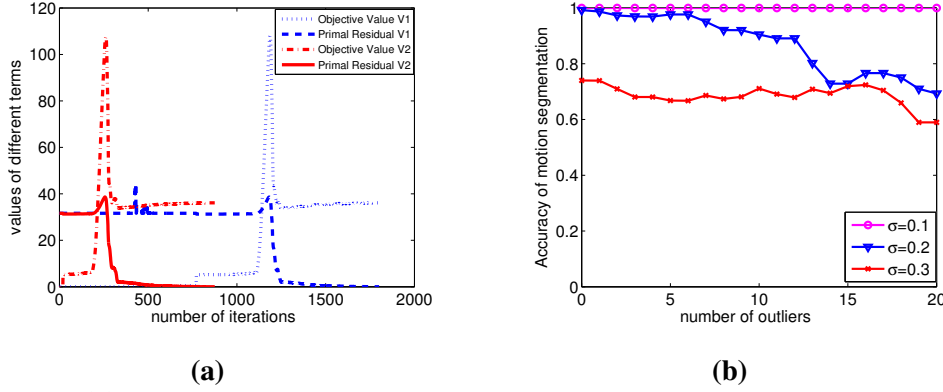


Figure 5.2: **(a)** Convergence curves for the objective function value and the primal residuals. Blue curves: initialization (V1); red curves: initialization (V2). **(b)** Accuracy of motion segmentation by adding different amounts of outliers and Gaussian noise.

Figure 5.2(b) depicts the performance (averaged over 5 random trials) of our algorithm under different amounts of outliers and at fixed inlier noise level. In particular, we show the motion segmentation accuracy as a function of the number of outliers for 3 different levels of Gaussian noise. Note that the performance of our algorithm degrades gracefully as the amount of noise and outliers increases. Overall, our algorithm is rather robust to these adverse yet realistic conditions. For example, when the amount of measurement noise is moderate (i.e., σ from 0 to 0.1), our algorithm can almost achieve perfect motion segmentation results even for large number of outliers.

5.3.3 Experiment-3: Real images, Hopkins155 dataset

The Hopkins155 contains 155 video sequences, which are however all generated from 49 source video sequences. Among the 49 sequences, 28 are indoor scenes containing some checkerboard patterns, and the remaining 21 are outdoor natural scenes with no checkerboard pattern. While having a checkerboard pattern simplifies manual feature point matching, the repetitive pattern can actually confuse an appearance-based automatic feature matching algorithm. For example, SIFT descriptors computed at different corner points in a checkerboard are very similar. Since our algorithm makes use of appearance information (i.e., SIFT), we tested it on the 21 outdoor sequences first. More importantly, these 21 sequences are also much more realistic than the checkerboard ones. By dividing some of the sequences with three motions (g1g2g3) into subsequences with two motions (g1g2, g1g3, and g2g3), we obtained 27 sequences which cover most of the outdoor scenes in Hopkins155. The objective for this set of experiments on the real images of Hopkins155 is to verify the practical usefulness of our algorithm.

To create appearance observations, we computed a 128D SIFT descriptor at each one of the feature point locations provided with the Hopkins155 dataset. We then

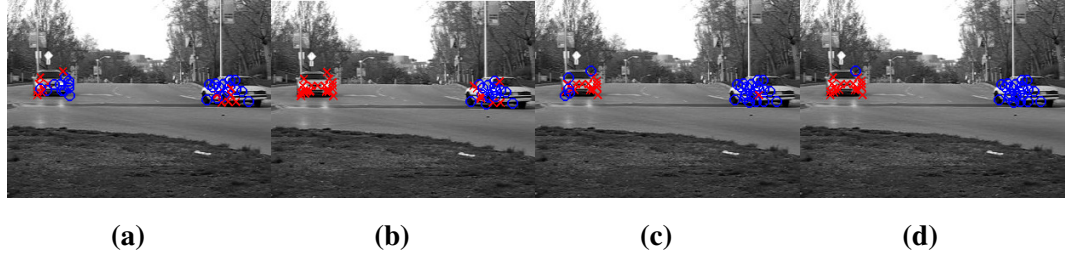


Figure 5.3: Motion segmentation results of different algorithms for the cars2_07_g12 sequence. Points marked with the same color and marker ($\circ$ or $\times$) are from the same motion: (a) SIFT + SSC; (b) RE + SSC (RE: ROML with embedded feature); (c) RS + SSC (RS: ROML with SIFT feature); (d) Our algorithm. Best viewed in color.

deliberately threw away all feature correspondence information. Given this input, our goal is to recover the missing point correspondences, and at the same time, to segment (cluster) all the feature points into correct motion groups.

Conventionally, when no point correspondences are given, motion segmentation is performed in a two-stage approach: (i) Point correspondences are estimated by, e.g., matching feature appearance or SIFT descriptors, and (ii) Subspace segmentation methods such as SSC (Elhamifar and Vidal [2013]) or LRR (Liu et al. [2013]) are applied to the estimated point correspondences. Therefore, we employed the following methods as baselines: (1) SIFT matching followed by SSC, denoted by SIFT+SSC; (2) SIFT matching followed by LRR, denoted by SIFT+LRR; (3) ROML matching using embedded features (Torki and Elgammal [2010]; Jia et al. [2015]) followed by SSC, denoted by RE+SSC; (4) ROML matching using embedded features (Torki and Elgammal [2010]; Jia et al. [2015]) followed by LRR, denoted by RE+LRR; (5) ROML matching using SIFT feature (Jia et al. [2015]; Zeng et al. [2012]) followed by SSC, denoted by RS+SSC; (6) ROML matching using SIFT feature followed by LRR, denoted by RS+LRR. Note that the embedded features combine the SIFT features and the point coordinates by manifold learning (Torki and Elgammal [2010]), and were used for ROML in (Jia et al. [2015]).

For ROML-based methods ((3)–(6)) and our algorithm, we initialized $\{\mathbf{P}_f\}_{f=1}^F$ with the PPMs recovered from SIFT matching, and set $\rho_0 = 10^{-2}$, $\eta = 1.01$.

In Table 5.1, we summarize the results in terms of ACC_{ms} and ACC_{pc} . From Table 5.1(a), we can see that our algorithm outperforms the baselines in terms of both correspondence and motion segmentation accuracies. Moreover, when given 100% complete trajectories, SSC and LRR achieve ACC_{ms} of 99.31% and 97.16%, respectively. This means that our algorithm, while not requiring any point correspondence as input, achieves motion segmentation results comparable to SSC and better than LRR.

Furthermore, when looking at the sequences whose ACC_{pc} of SIFT matching is less than 75% (see Table 5.1(b)), we find that our algorithm outperforms the baselines significantly in terms of both motion segmentation and point correspondence. For

Table 5.1: Average motion segmentation and point correspondence accuracies (%) on the Hopkins155 27 non-checkerboard sequences.

Methods	SIFT+SSC	SIFT+LRR	RE+SSC	RE+LRR	RS+SSC	RS+LRR	Ours
(a) All 27 non-checkerboard sequences							
ACC_{ms}	84.83	80.10	88.47	88.47	93.03	91.30	97.29
ACC_{pc}	84.86	84.86	87.86	87.86	95.46	95.46	98.03
(b) The 6 sequences whose ACC_{PC} of SIFT matching is less than 75%							
ACC_{ms}	75.01	75.57	82.35	79.47	91.60	84.34	99.59
ACC_{pc}	64.09	64.09	75.73	75.73	86.70	86.70	95.35

Table 5.2: Average motion segmentation and point correspondence accuracies (%) on the Hopkins155 checkerboard sequences with at most 200 trajectories.

Methods	SIFT+SSC	SIFT+LRR	RE+SSC	RE+LRR	RS+SSC	RS+LRR	Ours
ACC_{ms}	60.86	60.25	75.68	63.14	78.68	66.07	83.24
ACC_{pc}	32.27	32.37	46.85	46.85	53.76	53.76	65.02

visual comparison, in Fig 5.3 we show the motion segmentation results of different algorithms on one sequence of Hopkins155.

Out of curiosity, we also performed experiments on the Hopkins155 checkerboard sequences. Due to the algorithmic complexity of ROML and of our method (mainly in the step solving the binary assignment problems), we selected the sequences with at most 200 trajectories. From the results in Table 5.2, we can see that our approach also outperforms the baselines on the checkerboard sequences. This shows that jointly solving motion segmentation and feature correspondence indeed helps compensating for the lack of discriminative appearance.

5.3.4 Experiment-4: Real images, Other Real Sequences

Here, we show that our algorithm can be applied to perform motion segmentation in more realistic scenarios, where neither interest points nor point correspondences are provided.

Given video sequences with multiple motions, we first ran a SIFT detector over the frames to get the locations and descriptors of the detected interest points. Note that these interest points contain both inliers and outliers. The number of inlier points can be empirically approximated as $N = N_{sift} - 10$, where N_{sift} is the minimum number of SIFT matches from the first frame to any other frame. We then ran our algorithm to automatically select the inlier points, establish correspondences between them, and

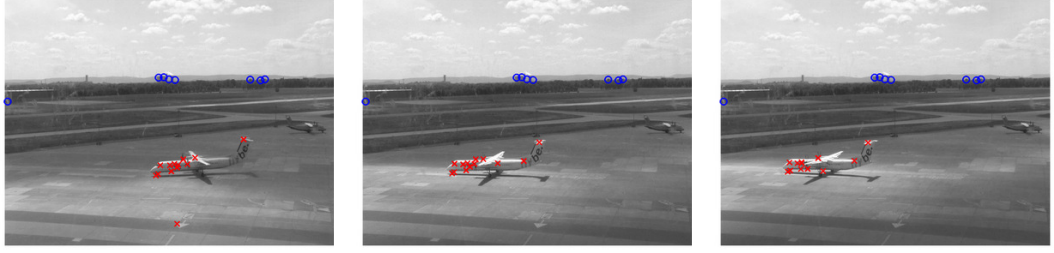


Figure 5.4: Motion segmentation results of the airport sequence: Points marked with the same color and marker ($\circ$ or $\times$) are from the same motion. Best viewed in color.

segment the trajectories into their respective motions.

We tested our algorithm on the airport sequence taken from the airport motion segmentation dataset (Dragon et al. [2013]). Figure 5.4 shows the results on three frames sampled from the 40-frame sequence. The inlier points are marked differently and each type of marker corresponds to one motion. Note that in each frame, 75-125 interest points were automatically extracted by the SIFT detector, and only 21 points are set as inliers, i.e., the number of outliers is 2-4 times as large as that of inliers. Moreover, the measurements of coordinates and descriptors are contaminated by noise due to the illumination variations across the sequence. As challenging as this sequence is for the task of motion segmentation, after manually labeling the ground-truth correspondences for evaluation purpose, we found that our algorithm yields $ACC_{ms} = 96.43\%$ and $ACC_{pc} = 94.17\%$.

5.4 Summary

Different from the methods in Chapter 3 and 4 where the point trajectories are given as input, in this section, we have proposed a unified framework to solve the problem of motion segmentation with unknown point correspondences. Our problem formulation is based on two important constraints: First, the recovered inlier point trajectories should satisfy the subspace constraints; Second, the matching feature descriptors should be low-rank, ideally rank one. With these two constraints, we have formulated our problem in terms of PPMs, which simultaneously select and reorder the inlier points. We have shown that our problem formulation can be solved via the ADMM. We have verified the effectiveness and robustness of our algorithm on both synthetic and real-world data and showed that it outperforms the existing two-step methods in terms of both motion segmentation and point correspondence accuracies.

The computational bottleneck of our method lies in the Hungarian algorithm for solving the binary assignment problem, which has complexity of $O(N^3)$. A direction of future work could, therefore, focus on speeding up this step by exploiting the inherent spatial constraints of motion segmentation.

Robust Multi-body Feature Tracker: A Segmentation-free Approach

In the previous chapter, we pointed out that the current feature tracking methods may not be ideal for the purpose of motion segmentation, and thus proposed a method that jointly finds the motion clusters and feature correspondences. However, another direction of interest could be to improve the feature tracking itself so that we can get more reliable point trajectories.

Feature tracking is a prerequisite for most motion segmentation methods, as well as for many other computer vision tasks such as visual SLAM and action recognition. Among all the feature tracking methods, the Kanade-Lucas-Tomasi (KLT) tracker (Lucas et al. [1981]; Tomasi and Kanade [1991]; Shi and Tomasi [1994]; Baker and Matthews [2004]), although developed 30 years ago, still remains one of the most widely used techniques. One of the reasons for this popularity is its computational efficiency; the KLT tracker is local, in the sense that it treats each local region independently of the others, which makes it highly parallelizable. This locality, however, comes at a cost in tracking robustness: the tracking of each feature cannot benefit from intrinsic scene constraints, and thus often suffers from drift.

Real-world scenes, however, are often strongly constrained. For example, in autonomous driving, most of the moving objects (cars, vehicles, pedestrian) are rigid, or quasi-rigid if seen from afar. Several methods have therefore been proposed to exploit this scene rigidity to improve feature tracking (Torresani and Bregler [2002]; Buchanan and Fitzgibbon [2007]; Poling et al. [2014]). Unfortunately, these methods all assume an affine camera model and are thus ill-suited to handle strong perspective effects. More importantly, they work either as a post-processing step on an entire sequence (Torresani and Bregler [2002]), which is sensitive to initial tracking results and does not apply to online feature tracking, or within a temporal sliding window (Buchanan and Fitzgibbon [2007]; Poling et al. [2014]), which is sensitive to initialization in the first few frames.

By contrast, in this chapter, we introduce a novel feature tracker that takes advantage of *multi-body scene rigidity* to improve tracking robustness under a general

perspective camera model. A conventional approach to addressing this problem would consist of alternating between two subtasks: motion segmentation and feature tracking under rigidity constraints for each segment. This, however, suffers from the following drawbacks: First, it requires knowing the number of observed motions; and, second, it relies on assigning points to individual motions, which is very sensitive to the initial motion estimates.

Here, we introduce a *segmentation-free* multi-body feature tracker that overcomes these drawbacks. Specifically, our approach bypasses the motion assignment step by making use of subspace constraints derived directly from the epipolar constraints of multiple motions. As a result, our algorithm does not require prior knowledge of the number of motions. Furthermore, this allows us to formulate tracking as an optimization problem whose subproblems all have closed-form solutions.

We demonstrate the effectiveness of our method on both feature point tracking and frame-by-frame motion segmentation on real world sequences. Our experiments show that, by incorporating multi-motion constraints, our tracker yields better accuracies and is more robust to noise than the standard KLT tracker and the state-of-the-art tracking algorithm of Poling et al. [2014].

6.1 Background

The KLT tracker (Tomasi and Kanade [1991]; Shi and Tomasi [1994]) was derived from the Lucas-Kanade algorithm for image alignment (Lucas et al. [1981]). Feature tracking was achieved by optimizing the sum of squared differences between a template patch and an image patch with the Gauss-Newton method. It was later extended to handle relatively large displacements by the use of image pyramids (Bouguet [2001]).

Global rigidity constraints have been incorporated in feature point tracking to improve robustness. For instance, Torresani and Bregler [2002] proposed to regularize tracking with a global low-rank constraint on the trajectory matrix of the whole sequence. They relied on the original KLT tracker to get a set of reliable tracks, and explicitly factorized the reliable trajectory matrix into two low-rank matrices with the rank given *a priori*. One of the low-rank matrices, called the motion parameter matrix, was then used to rectify the unreliable tracks. In short, this method can be viewed as a post-processing step on the results of the KLT tracker, and is therefore not suitable for online frame-to-frame tracking.

Instead of using the whole sequence, low-rank constraints (Buchanan and Fitzgibbon [2007]) and similar subspace priors (Poling et al. [2014]) were applied within a temporal sliding window. Specifically, Buchanan and Fitzgibbon [2007] exploited the low-rank constraints within a Bayesian tracking framework, making predictions of the new location of a particular point using a low rank approximation obtained from the previous frames. Recently, Poling et al. [2014] proposed a better feature tracker by

adding soft subspace constraints to the original KLT tracker and jointly solving for the displacement vectors of all feature points. These methods, however, assume an affine camera model within a temporal window, and are therefore ill-suited to handle strong perspective effects. Moreover, since the low-rank constraints are enforced in a temporal sliding window, these methods are sensitive to initialization in the first few frames.

By contrast, Piccini et al. [2014] exploited perspective projection by making use of epipolar constraints to track edgels in two consecutive frames. This method, however, was specifically designed to model a single motion, and thus does not easily extend to the multi-body case.

In the closely related optical flow literature (Bruhn et al. [2005]), several methods have been devoted to improving robustness via rigidity constraints. For instance, Valgaerts et al. [2008] introduced a variational model to jointly recover the fundamental matrix and the optical flow; Wedel et al. [2008, 2009] leveraged the fundamental matrix prior as an additional weak prior within a variational framework. These methods, however, assume that the scene is mostly stationary (and thus a single fundamental matrix is estimated), and treat the dynamic parts as outliers (Wedel et al. [2009]). Garg et al. [2010, 2013] proposed to make use of subspace constraints to regularize the multi-frame optical flow within a variational approach. This approach, however, assumes an affine camera model and works over entire sequences.

While, to the best of our knowledge, explicitly modeling multi-body motion has not been investigated in the context of feature tracking and optical flow estimation, a large body of work (Costeira and Kanade [1998]; Vidal et al. [2005b]; Yan and Pollefeys [2006]; Li [2007]; Li et al. [2013]; Vidal et al. [2002]; Elhamifar and Vidal [2013]) including our work in Chapter 3 and 4, has been devoted to multi-body motion segmentation given good point trajectories in relatively long sequences. Typically, these tracks are first obtained with the KLT tracker, and then manually cleaned up, e.g., the Hopkins155 dataset (Tron and Vidal [2007]). In a sense, the lack of better tracking algorithms that can incorporate the intrinsic constraints of dynamic scenes prevents the practical use of these motion segmentation algorithms.

In this chapter, we seek to track feature points in dynamic scenes where multiple motions are present. In this scenario, a single fundamental matrix is not sufficient to express the epipolar constraints any more. While one could think of alternating between estimating multiple fundamental matrices, motion assignments and displacement vectors, the resulting algorithm would typically be very sensitive to initialization, since the motion assignments strongly depend on the motion estimates. By contrast, we introduce a segmentation-free approach that bypasses the motion assignment problem by exploiting subspace constraints derived from epipolar geometry. This yields a robust multi-body tracking algorithm that, as demonstrated by our experiments, opens up the possibility to perform motion segmentation in realistic scenarios.

6.2 Multi-body Feature Tracker

We now introduce our approach to multi-body feature tracking. Formally, let $I(\mathbf{x})$ denote the current image, $T(\mathbf{x})$ the previous image (or template image), and $\mathbf{x}_{ij} = [x_{ij}, y_{ij}]^T$ the j^{th} image point in the i^{th} patch Ω_i of the template image. Our goal is to estimate the displacement vector $\mathbf{u} = [\mathbf{u}_1^T, \dots, \mathbf{u}_N^T]^T \in \mathbb{R}^{2 \times N}$ for all N tracked feature points. To this end, we rely on the standard brightness constancy assumption (Szeliski [2010]), which lets us derive the data term

$$\mathcal{D}(\mathbf{u}) = \sum_{i=1}^N \sum_{\mathbf{x}_{ij} \in \Omega_i} \psi(I(\mathbf{x}_{ij} + \mathbf{u}_i) - T(\mathbf{x}_{ij})) , \quad (6.1)$$

where, typically, $\psi(x) = x^2$ or $\psi(x) = |x|$. In particular, we use the ℓ_1 norm, which provides robustness to outliers.

Estimating the displacements from this data term only is typically sensitive to noise and may be subject to drift. A general approach to making the process more robust consists of introducing a regularizer $\mathcal{R}(\mathbf{u})$ to form an energy function of the form

$$\mathcal{F}(\mathbf{u}) = \gamma \mathcal{D}(\mathbf{u}) + \mathcal{R}(\mathbf{u}) . \quad (6.2)$$

As mentioned above, several attempts at designing such a regularizer have been proposed. For example, under an affine camera model, $\mathcal{R}(\mathbf{u})$ can encode a low-rank prior (Torresani and Bregler [2002]; Poling et al. [2014]); with a general projective camera model, $\mathcal{R}(\mathbf{u})$ can represent epipolar constraints (i.e., a fundamental matrix prior) (Wedel et al. [2008]; Valgaerts et al. [2008]; Piccini et al. [2014]). In the latter case, the fundamental matrix can be either pre-computed via an existing feature matching method (Piccini et al. [2014]), or re-computed iteratively.

When multiple motions are present, however, a single epipolar constraint is not sufficient. Instead, multiple fundamental matrices should be estimated so as to respect the assignments of the tracked points to individual motions. A straightforward way to address this problem consists of adding a motion segmentation step in the tracking algorithm, so that the fundamental matrices can be iteratively re-estimated. This leads to the simple *segmentation-based* approach to multi-body feature tracking described below.

6.2.1 A First Attempt: Segmentation-based Tracking

To derive a segmentation-based approach, we rely on epipolar constraints. Recall that, in epipolar geometry (Hartley and Zisserman [2004]), the homogeneous coordinates $\bar{\mathbf{x}}'_i = (x'_i, y'_i, 1)^T$ and $\bar{\mathbf{x}}_i = (x_i, y_i, 1)^T$ of two corresponding image points in two

frames are related by a fundamental matrix $\mathbf{F}$, such that

$$\bar{\mathbf{x}}_i'^T \mathbf{F} \bar{\mathbf{x}}_i = 0. \quad (6.3)$$

It is therefore natural to exploit these constraints to regularize tracking according to the motion assignments of the different points.

More specifically, in a segmentation-based approach, three types of variables must be estimated: the displacement vector $\mathbf{u}$, the fundamental matrices $\{\mathbf{F}^k\}_{k=1,\dots,K}$ (where K is the number of motions), and the motion label of each tracked point. Let us denote by $\bar{\mathbf{x}}_i^k$ the homogeneous coordinate of the i^{th} feature point (i.e., the center of the patch Ω_i) assigned to motion k . We can define a multi-body regularization term as

$$\mathcal{R}_1(\mathbf{u}, \mathbf{F}^k) = \sum_k \sum_i [(\bar{\mathbf{x}}_i^k + \bar{\mathbf{u}}_i)^T \mathbf{F}^k \bar{\mathbf{x}}_i^k]^2, \quad (6.4)$$

where $\bar{\mathbf{u}}_i = [\mathbf{u}_i^T, 0]^T$.

The energy function can then be approximately minimized by iterating over the three following steps:

1. Update $\mathbf{u}$ by first-order gradient descent (Poling et al. [2014]);
2. Estimate $\mathbf{F}^k$ for each motion given the current point assignments;
3. Re-assign the motion labels of the feature points to the nearest $\mathbf{F}^k$.

This segmentation-based approach suffers from several drawbacks. First, the number of motions needs to be known *a priori*, which is typically hard for general-purpose tracking. Second, and more importantly, the quality of the solution obtained with this approach will strongly depend on the initializations of $\mathbf{F}^k$ and of the motion labels. This, in a sense, is a chicken-and-egg problem, since good initialization for these variables can only be obtained from good motion estimates. Instead, in the remainder of this section, we introduce a new segmentation-free approach that bypasses the need to explicitly compute the fundamental matrices and the motion assignments.

6.2.2 Our Segmentation-free Approach

In this section, we introduce our segmentation-free multi-body feature tracker, which is the key contribution of this chapter. We first show how the epipolar constraints can be converted to subspace constraints, and incorporated into our tracking formalism. We then derive the solution to the resulting optimization problem by decomposing it into several convex subproblems all with closed-form solutions.

6.2.2.1 Epipolar Subspace Constraints

As in the segmentation-based approach, we seek to rely on epipolar geometry. To this end, we make use of the constraint expressed in Eq. (6.3). We first note that this

constraint can be re-written as

$$\mathbf{f}^T \text{vec}(\bar{\mathbf{x}}'_i \bar{\mathbf{x}}_i^T) = 0, \quad (6.5)$$

where $\mathbf{f} \in \mathbb{R}^9$ is the vectorized fundamental matrix $\mathbf{F}$, and

$$\text{vec}(\bar{\mathbf{x}}'_i \bar{\mathbf{x}}_i^T) = (x_i x'_i, x_i y'_i, x_i, y_i x'_i, y_i y'_i, y_i, x'_i, y'_i, 1)^T. \quad (6.6)$$

Let us define $\mathbf{w}_i = \text{vec}(\bar{\mathbf{x}}'_i \bar{\mathbf{x}}_i^T)$. Then, $\mathbf{w}_i$ lies in the orthogonal complement of $\mathbf{f}^T$, which is a subspace of dimension up to eight¹, and which we call the *epipolar subspace*. Since image points undergoing the same motion share the same fundamental matrix, all $\mathbf{w}_i$ s corresponding to points belonging to the same rigid motion lie on the same subspace (Li et al. [2013]).

Therefore, in our multi-body feature tracking scenario, if the feature points are correctly tracked, the data vectors defined as

$$\mathbf{w}_i = \text{vec}((\bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i) \bar{\mathbf{x}}_i^T), \quad \forall 1 \leq i \leq N, \quad (6.7)$$

should lie in a union of linear subspaces. This subspace constraint can be characterized by the self-expressiveness property (Elhamifar and Vidal [2013]), i.e., a data point drawn from one subspace in a union of subspaces can be represented as a linear combination of the points lying in the same subspace.

In our case, this self-expressiveness property can be expressed as

$$\mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C}, \quad (6.8)$$

where² $\mathbf{W}_{(\mathbf{u})} = [\mathbf{w}_1 \cdots \mathbf{w}_N] \in \mathbb{R}^{9 \times N}$, and $\mathbf{C}$ is the coefficient matrix encoding the linear combinations. On its own, this term has a trivial solution for $\mathbf{C}$ (i.e., the identity matrix). To avoid this solution, $\mathbf{C}$ needs to be regularized. In the subspace clustering literature, $\mathbf{C}$ is encouraged to be either sparse (Elhamifar and Vidal [2013]) by minimizing $\|\mathbf{C}\|_1$, low rank (Liu et al. [2013]) by minimizing $\|\mathbf{C}\|_*$, or dense block diagonal in Chapter 3 by minimizing $\|\mathbf{C}\|_F^2$. Here, we choose the Frobenius norm, which has proven effective and is easy to optimize. Furthermore, we explicitly model noise and outliers, which are inevitable in real-world sequences.

More specifically, we write our regularization term for multi-body tracking as

$$\mathcal{R}_2(\mathbf{u}, \mathbf{C}) = \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1, \text{ s.t. } \mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C} + \mathbf{E}, \quad (6.9)$$

¹Note that, in practice, this dimension is typically smaller than 8, since, in real scenes, the motion of objects, such as cars or people, is not arbitrary, and thus corresponds to degenerate (i.e., low-rank) motion (Li et al. [2013]).

²In the following, we make use of subscript $(\mathbf{u})$, i.e., $\mathbf{W}_{(\mathbf{u})}$, to indicate that $\mathbf{W}$ depends on the variable $\mathbf{u}$. For compactness, and without causing confusion, we drop this explicit dependency in Section 6.2.2.3.

where $\mathbf{E}$ accounts for noise and outliers, and is thus encouraged to be sparse. Note that, for a given displacement $\mathbf{u}$, and ignoring noise, the optimal value of this regularizer depends on the intrinsic dimension of the motion (see Chapter 3). Since here we optimize $\mathbf{u}$, this regularizer therefore tends to favor degenerate rigid motions over purely arbitrary rigid motions. This actually reflects reality, since, in real scenes, cars, people and other objects typically move in a well-constrained manner.

Importantly, this regularization term requires explicitly computing neither the fundamental matrices, nor the motion assignments. As such, it therefore yields a *segmentation-free* approach.

Altogether, the energy function of our multi-body tracking framework can be written as

$$\mathcal{F}(\mathbf{u}, \mathbf{C}) = \gamma \mathcal{D}(\mathbf{u}) + \mathcal{R}_2(\mathbf{u}, \mathbf{C}) . \quad (6.10)$$

Our goal is to minimize $\mathcal{F}(\mathbf{u}, \mathbf{C})$ w.r.t. $\mathbf{u}$ and $\mathbf{C}$. We next show how to solve this optimization problem.

6.2.2.2 Approximation and Problem Reformulation

To optimize Eq. (6.10), we first approximate the data term in the same manner as the original KLT. In other words, given an initial displacement $\mathbf{u}_i^0$ for patch i , we approximate the intensity values $I(\mathbf{x}_{ij} + \mathbf{u}_i)$ with their first-order Taylor expansion at $\mathbf{x}_{ij} + \mathbf{u}_i^0$. This can be written as

$$I(\mathbf{x}_{ij} + \mathbf{u}_i) \approx I(\mathbf{x}_{ij} + \mathbf{u}_i^0) + \nabla I(\mathbf{x}_{ij} + \mathbf{u}_i^0)(\mathbf{u}_i - \mathbf{u}_i^0) . \quad (6.11)$$

For notational convenience, let $\nabla I_{ij} = \nabla I(\mathbf{x}_{ij} + \mathbf{u}_i^0) \in \mathbb{R}^{1 \times 2}$, and $\tau_{ij} = \nabla I_{ij} \mathbf{u}_i^0 + T(\mathbf{x}_{ij}) - I(\mathbf{x}_{ij} + \mathbf{u}_i^0)$. Then, the data term can be expressed as

$$\mathcal{D}(\mathbf{u}) = \sum_{i,j} |\nabla I_{ij} \mathbf{u}_i - \tau_{ij}| . \quad (6.12)$$

By combining this data term with our regularizer, we get the optimization problem

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{C}, \mathbf{E}} \quad & \gamma \|\mathbf{A}_{(\mathbf{u})}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C} + \mathbf{E} , \end{aligned} \quad (6.13)$$

where $\mathbf{A}_{ij} = \nabla I_{ij} \mathbf{u}_i - \tau_{ij}$, i.e.,

$$\mathbf{A}_{(\mathbf{u})} = \begin{bmatrix} \nabla I_{11} \mathbf{u}_1 - \tau_{11} & \cdots & \nabla I_{1m} \mathbf{u}_1 - \tau_{1m} \\ \vdots & & \vdots \\ \nabla I_{N1} \mathbf{u}_N - \tau_{N1} & \cdots & \nabla I_{Nm} \mathbf{u}_N - \tau_{Nm} \end{bmatrix} , \quad (6.14)$$

where m is the number of pixels in a patch, e.g. $m = 49$ if the patch size is 7×7 .

For convenience of optimization, we introduce an auxiliary variable $\mathbf{Z} = \mathbf{A}_{(\mathbf{u})}$. Then, (6.13) can be equivalently written as

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}} \quad & \gamma \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_{(\mathbf{u})}, \mathbf{W}_{(\mathbf{u})} = \mathbf{W}_{(\mathbf{u})} \mathbf{C} + \mathbf{E}. \end{aligned} \quad (6.15)$$

The main hurdle in optimizing (6.15) now lies in the term with $\mathbf{W}_{(\mathbf{u})}$ due to its seemingly complicated dependency on $\mathbf{u}$. However, we show below that this term can be simplified by a few matrix derivations.

First, note that, by definition, we have

$$\text{vec}(\mathbf{W}_{(\mathbf{u})}) = \begin{bmatrix} \text{vec}((\bar{\mathbf{x}}_1 + \bar{\mathbf{u}}_1)\bar{\mathbf{x}}_1^T) \\ \vdots \\ \text{vec}((\bar{\mathbf{x}}_N + \bar{\mathbf{u}}_N)\bar{\mathbf{x}}_N^T) \end{bmatrix} \quad (6.16)$$

$$= \underbrace{\begin{bmatrix} \bar{\mathbf{x}}_1 \otimes \mathbf{I}_{3 \times 3} & & \\ & \ddots & \\ & & \bar{\mathbf{x}}_N \otimes \mathbf{I}_{3 \times 3} \end{bmatrix}}_{\bar{\mathbf{P}} \in \mathbb{R}^{9N \times 3N}} (\bar{\mathbf{x}} + \bar{\mathbf{u}}), \quad (6.17)$$

where $\bar{\mathbf{x}} = [\bar{\mathbf{x}}_1^T \cdots \bar{\mathbf{x}}_N^T]^T$, $\bar{\mathbf{u}} = [\bar{\mathbf{u}}_1^T \cdots \bar{\mathbf{u}}_N^T]^T$, $\mathbf{I}_{3 \times 3}$ is the 3-by-3 identity matrix and $\otimes$ denotes the Kronecker product. Let us define $\mathbf{b} = \bar{\mathbf{P}}\bar{\mathbf{x}}$ (or equivalently $\mathbf{b}_i = \text{vec}(\bar{\mathbf{x}}_i\bar{\mathbf{x}}_i^T)$) and introduce another auxiliary variable $\mathbf{m} = \mathbf{P}\mathbf{u}$ (where $\mathbf{P} \in \mathbb{R}^{9N \times 2N}$ is obtained by removing every $3i^{\text{th}}$ column of $\bar{\mathbf{P}}$)³. Our optimization problem then becomes

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}, \mathbf{m}} \quad & \gamma \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{A}_{(\mathbf{u})}, \mathbf{W}_{(\mathbf{m})} = \mathbf{W}_{(\mathbf{m})} \mathbf{C} + \mathbf{E}, \mathbf{m} = \mathbf{P}\mathbf{u}, \end{aligned} \quad (6.18)$$

where now $\text{vec}(\mathbf{W}_{(\mathbf{m})}) = \mathbf{b} + \mathbf{m}$.

The above optimization problem involves a large number of variables. We propose to solve it via the Alternating Direction Method of Multipliers (ADMM) (Boyd et al. [2011]), which decomposes a big optimization problem into several small subproblems. Below, we show how this can be achieved for our problem.

³Note that $\mathbf{P}\mathbf{u} = \bar{\mathbf{P}}\bar{\mathbf{u}}$, since $\bar{\mathbf{u}}_i = [\mathbf{u}_i^T, 0]^T$.

6.2.2.3 ADMM Solution

To apply the ADMM, we first need to derive the augmented Lagrangian of (6.18), which can be expressed as

$$\begin{aligned} \mathcal{L}_\rho(\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}, \mathbf{m}, \mathbf{y}, \mathbf{Y}_1, \mathbf{Y}_2) = & \gamma \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{C}\|_F^2 + \lambda \|\mathbf{E}\|_1 + \mathbf{y}^T (\mathbf{m} - \mathbf{P}\mathbf{u}) + \\ & \langle \mathbf{Y}_1, \mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} - \mathbf{E} \rangle + \langle \mathbf{Y}_2, \mathbf{Z} - \mathbf{A}_{(\mathbf{u})} \rangle + \\ & \frac{\rho}{2} (\|\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} - \mathbf{E}\|_F^2 + \|\mathbf{Z} - \mathbf{A}_{(\mathbf{u})}\|_F^2 + \|\mathbf{m} - \mathbf{P}\mathbf{u}\|_2^2), \end{aligned} \quad (6.19)$$

where $\langle \cdot, \cdot \rangle$ denotes the matrix inner product, $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{y}$ are Lagrange multipliers, and ρ is the penalty parameter. The ADMM then works by alternatively minimizing $\mathcal{L}_\rho$ w.r.t. one of the five variables $\mathbf{u}, \mathbf{C}, \mathbf{E}, \mathbf{Z}, \mathbf{m}$ while keeping the remaining four fixed. This yields the following steps:

(1) Computing $\mathbf{Z}$ can be expressed as the convex program

$$\min_{\mathbf{Z}} \frac{\gamma}{\rho} \|\mathbf{Z}\|_1 + \frac{1}{2} \|\mathbf{Z} - (\mathbf{A}_{(\mathbf{u})} - \mathbf{Y}_2/\rho)\|_F^2, \quad (6.20)$$

which can be solved in closed-form by element-wise thresholding (Cai et al. [2010]). Let $\mathcal{T}_\alpha[x] = \text{sign}(x) \cdot \max(|x| - \alpha, 0)$ be the soft-thresholding operator. Then, the solution to (6.20) is given by

$$\mathbf{Z}^* = \mathcal{T}_{\frac{\gamma}{\rho}}[\mathbf{A}_{(\mathbf{u})} - \mathbf{Y}_2/\rho]. \quad (6.21)$$

(2) Similarly, computing $\mathbf{E}$ translates to

$$\min_{\mathbf{E}} \frac{\lambda}{\rho} \|\mathbf{E}\|_1 + \frac{1}{2} \|\mathbf{E} - (\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} + \mathbf{Y}_1/\rho)\|_F^2, \quad (6.22)$$

which again can be solved in closed-form as

$$\mathbf{E}^* = \mathcal{T}_{\frac{\lambda}{\rho}}[\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} + \mathbf{Y}_1/\rho]. \quad (6.23)$$

(3) To compute $\mathbf{C}$, we need to solve the least-squares problem

$$\min_{\mathbf{C}} \frac{1}{2} \|\mathbf{C}\|_F^2 + \frac{\rho}{2} \|\mathbf{W}_{(\mathbf{m})}\mathbf{C} - \mathbf{W}_{(\mathbf{m})} + \mathbf{E} - \mathbf{Y}_1/\rho\|_F^2, \quad (6.24)$$

which can easily be achieved in closed-form as

$$\mathbf{C}^* = \left(\mathbf{I} + \rho \mathbf{W}_{(\mathbf{m})}^T \mathbf{W}_{(\mathbf{m})} \right)^{-1} \left(\rho \mathbf{W}_{(\mathbf{m})}^T (\mathbf{W}_{(\mathbf{m})} - \mathbf{E} + \mathbf{Y}_1/\rho) \right). \quad (6.25)$$

Note that here naively inverting the N -by- N matrix $\mathbf{I} + \rho \mathbf{W}_{(\mathbf{m})}^T \mathbf{W}_{(\mathbf{m})}$ is computationally expensive when the number of feature points N is large. However, we can apply the matrix inversion lemma such that

$$\begin{aligned} & \left(\mathbf{I} + \rho \mathbf{W}_{(\mathbf{m})}^T \mathbf{W}_{(\mathbf{m})} \right)^{-1} \\ &= \mathbf{I} - \rho \mathbf{W}_{(\mathbf{m})}^T \left(\mathbf{I} + \rho \mathbf{W}_{(\mathbf{m})} \mathbf{W}_{(\mathbf{m})}^T \right)^{-1} \mathbf{W}_{(\mathbf{m})}, \end{aligned} \quad (6.26)$$

where we only need to invert⁴ the 9-by-9 matrix $\mathbf{I} + \rho \mathbf{W}_{(\mathbf{m})} \mathbf{W}_{(\mathbf{m})}^T$, which is computationally much cheaper.

(4) Computing $\mathbf{u}$ requires solving the problem

$$\min_{\mathbf{u}} \frac{\rho}{2} \|\mathbf{P}\mathbf{u} - \mathbf{m} - \mathbf{y}/\rho\|_2^2 - \text{tr} \mathbf{Y}_2^T \mathbf{A}_{(\mathbf{u})} + \frac{\rho}{2} \|\mathbf{A}_{(\mathbf{u})} - \mathbf{Z}\|_F^2, \quad (6.27)$$

where $\mathbf{A}_{(\mathbf{u})}$ is defined in (6.14).

Expanding the definition of $\mathbf{A}_{(\mathbf{u})}$ in (6.27) and rearranging the terms yields:

$$\min_{\mathbf{u}} \frac{\rho}{2} \|\mathbf{P}\mathbf{u} - \mathbf{m} - \mathbf{y}/\rho\|_2^2 - \mathbf{g}^T \mathbf{u} + \frac{\rho}{2} \mathbf{u}^T \mathbf{H} \mathbf{u}, \quad (6.28)$$

where $\mathbf{g}$ is a column vector defined as

$$\mathbf{g} = \left(\cdots, \sum_j (\mathbf{Y}_{2ij} + \rho(\tau_{ij} + \mathbf{Z}_{ij})) \nabla I_{ij}, \cdots \right)^T \in \mathbb{R}^{2N},$$

and $\mathbf{H}$ is a sparse block-diagonal matrix expressed as

$$\mathbf{H} = \begin{bmatrix} \ddots & & \\ & \sum_j \nabla I_{ij}^T \nabla I_{ij} & \\ & & \ddots \end{bmatrix} \in \mathbb{R}^{2N \times 2N}.$$

This subproblem has again a closed-form solution given by

$$\mathbf{u}^* = \left(\rho \mathbf{P}^T \mathbf{P} + \rho \mathbf{H} \right)^{-1} \left(\mathbf{g} + \mathbf{P}^T \mathbf{y} + \rho \mathbf{P}^T \mathbf{m} \right). \quad (6.29)$$

Note that $\mathbf{P}$ and $\mathbf{H}$ are sparse matrices, so $\mathbf{u}^*$ can be computed efficiently by sparse matrix techniques.

⁴In practice, we don't need to invert the matrix explicitly. For example, in Matlab, we can use the backslash operator, which produces the solution using Gaussian elimination.

(5) While solving for $\mathbf{m}$ may not seem straightforward, we show below that it is nothing but a least-squares problem. The subproblem w.r.t $\mathbf{m}$ can be written as

$$\min_{\mathbf{m}} \frac{\lambda}{2} \|\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C}\|_F^2 + \frac{\rho}{2} \|\mathbf{m} - \mathbf{P}\mathbf{u} + \mathbf{y}/\rho\|_2^2. \quad (6.30)$$

Let $\mathbf{M}, \mathbf{B}, \mathbf{G} \in \mathbb{R}^{9 \times N}$ be the matrix forms of $\mathbf{m}, \mathbf{b}, \mathbf{y}/\rho - \mathbf{P}\mathbf{u}$, respectively. Then, (6.30) can be equivalently written as

$$\min_{\mathbf{M}} \frac{\lambda}{2} \|\mathbf{M}(\mathbf{I} - \mathbf{C}) + \mathbf{B}(\mathbf{I} - \mathbf{C})\|_F^2 + \frac{\rho}{2} \|\mathbf{M} + \mathbf{G}\|_F^2. \quad (6.31)$$

This again leads to a closed-form solution for $\mathbf{M}$ given by

$$\mathbf{M}^* = -\left(\rho\mathbf{G} + \mathbf{B}\mathbf{Q} + \mathbf{T}\right)\left(\lambda\mathbf{Q} + \rho\mathbf{I}\right)^{-1}, \quad (6.32)$$

where

$$\begin{aligned} \mathbf{Q} &= (\mathbf{I} - \mathbf{C})(\mathbf{I} - \mathbf{C})^T \in \mathbb{R}^{N \times N}, \\ \mathbf{T} &= (\mathbf{Y}_1/\rho - \mathbf{E})(\mathbf{I} - \mathbf{C}^T) \in \mathbb{R}^{9 \times N}. \end{aligned}$$

In short, the five subproblems derived from the augmented Lagrangian are all convex problems that can be solved efficiently in closed-form as follows:

$$\mathbf{Z} = \mathcal{T}_{\frac{\lambda}{\rho}}[\mathbf{A}_{(\mathbf{u})} - \mathbf{Y}_2/\rho], \quad (6.33)$$

$$\mathbf{E} = \mathcal{T}_{\frac{\lambda}{\rho}}[\mathbf{W} - \mathbf{W}\mathbf{C} + \mathbf{Y}_1/\rho], \quad (6.34)$$

$$\mathbf{C} = \left(\mathbf{I} + \rho\mathbf{W}^T\mathbf{W}\right)^{-1} [\rho\mathbf{W}^T(\mathbf{W} - \mathbf{E} + \mathbf{Y}_1/\rho)], \quad (6.35)$$

$$\mathbf{u} = \left(\rho\mathbf{P}^T\mathbf{P} + \rho\mathbf{H}\right)^{-1} \left(\mathbf{g} + \mathbf{P}^T\mathbf{y} + \rho\mathbf{P}^T\mathbf{m}\right), \quad (6.36)$$

$$\mathbf{M} = -\left(\rho\mathbf{G} + \mathbf{B}\mathbf{Q} + \mathbf{T}\right)\left(\lambda\mathbf{Q} + \rho\mathbf{I}\right)^{-1}. \quad (6.37)$$

Finally, the Lagrange multipliers and penalty parameter can be updated as

$$\mathbf{Y}_1 := \mathbf{Y}_1 + \rho(\mathbf{W} - \mathbf{W}\mathbf{C} - \mathbf{E}), \quad (6.38)$$

$$\mathbf{Y}_2 := \mathbf{Y}_2 + \rho(\mathbf{Z} - \mathbf{A}_{(\mathbf{u})}), \quad (6.39)$$

$$\mathbf{y} := \mathbf{y} + \rho(\mathbf{m} - \mathbf{P}\mathbf{u}) \quad (6.40)$$

$$\rho := \min(\eta\rho, \rho_m), \quad (6.41)$$

where $\eta > 1$, and ρ_m is the predefined maximum of ρ .

Algorithm 6.1 Solving (6.18) via the ADMM

Input:

Image I and template T , positions of the feature points $\mathbf{x}$ in T , initial displacement vector $\mathbf{u}^0$, parameters γ, λ .

Initialize: $\mathbf{C} = 0, \mathbf{Y}_1 = 0, \mathbf{Y}_2 = 0, \mathbf{y} = 0, \rho_0, \rho_m, \eta, \epsilon$

while not converged **do**

1. Update $\mathbf{Z}, \mathbf{E}, \mathbf{C}, \mathbf{u}$ and $\mathbf{m}$ in close-form via Eqs. (6.33)- (6.37), respectively;
2. Update $\mathbf{A}_{(\mathbf{u})}$ and $\mathbf{W}_{(\mathbf{m})}$ with updated $\mathbf{u}$ and $\mathbf{m}$;
3. Update the Lagrange multipliers and penalty parameter via Eqs. (6.38)- (6.41);
4. Check the convergence conditions $\|\mathbf{m} - \mathbf{P}\mathbf{u}\|_\infty \leq \epsilon$, $\|\mathbf{W}_{(\mathbf{m})} - \mathbf{W}_{(\mathbf{m})}\mathbf{C} - \mathbf{E}\|_\infty \leq \epsilon$, and $\|\mathbf{Z} - \mathbf{A}_{(\mathbf{u})}\|_\infty \leq \epsilon$;

end while

Output: Displacement vector $\mathbf{u}$, coefficient matrix $\mathbf{C}$.

Our approach to solving (6.18) is outlined in Algorithm 6.1. Note that the problem we are trying to solve is non-convex in that (i) the intensity function $I(\mathbf{x}; \mathbf{u})$ is non-convex w.r.t. $\mathbf{u}$; (ii) the optimization problem (6.18) involves a bilinear term in an equality constraint. While the ADMM does not guarantee convergence, it has proven effective in previous chapters.

6.2.2.4 Our Complete Multi-body Feature Tracker

In the same spirit as Bouguet [2001], we make use of an image pyramid to handle large displacements and avoid local optima. The results obtained at a coarser level ℓ of the pyramid are used as initialization for the next ($\ell - 1$, finer) level. Within each pyramid level, the initial displacement $\mathbf{u}^0$, where the first-order Taylor approximation is performed, is updated with the displacement vector of the previous iteration. We iterate over successive Taylor approximations until the displacement vector does not change significantly. Our complete segmentation-free multi-body feature tracker is outlined in Algorithm 6.2.

6.3 Experiments

To show the benefits of our multi-body feature tracker, we performed extensive experiments on different sequences. In the remainder of this section, we present both qualitative and quantitative results.

In these experiments, we compare our approach with the following baselines: the original KLT tracker (KLT), the L1-norm KLT tracker (L1-KLT), and the more recent Better Feature Tracker (BFT) through Subspace Constraints (Poling et al. [2014]).

Algorithm 6.2 Our Multi-body Feature Tracker**Input:**

Image I and template T , positions of the feature points $\mathbf{x}$ in T , initial displacement vector $\mathbf{u}^0$, number of pyramid levels L , parameters $\gamma, \lambda, \rho, \rho_m, \max_i, \epsilon$

for $\ell = L - 1 : 0$ **do**

Update $\mathbf{u}^0 \leftarrow \mathbf{u}^0 / 2^\ell$, $\mathbf{x} \leftarrow \mathbf{x}^0 / 2^\ell$ and compute ∇I at current image pyramid level;

for $i = 1 : \max_i$ **do**

1. Approximate the image intensities with Eq. (6.11), and compute $\tau, \mathbf{P}, \mathbf{H}$ according to their definitions;
2. Update $\mathbf{u}$ with Algorithm 6.1;
3. Check the convergence condition $\|\mathbf{u} - \mathbf{u}^0\| < \epsilon$;
4. If not converged, update $\mathbf{u}^0 = \mathbf{u}$.

end for

Update $\mathbf{u} \leftarrow 2^\ell \mathbf{u}$, $\mathbf{u}^0 \leftarrow 2^\ell \mathbf{u}^0$, and $\mathbf{x} \leftarrow 2^\ell \mathbf{x}$.

end for

Output: Displacement vector $\mathbf{u}$, coefficient matrix $\mathbf{C}$.

For the original KLT, we used the Matlab built-in vision toolbox *vision.PointTracker*; we implemented the L1-norm KLT tracker using the same framework as our method by just disabling the regularization term; for BFT, we used the code released by the authors.

Due to the lack of benchmark datasets for feature tracking, we make use of motion segmentation datasets where both the ground-truth tracks and the original videos are available. Since those videos are typically only provided for illustration purpose, they are generally highly compressed and not ideal for reliable feature tracking. This, however, is not really a problem when one seeks to evaluate feature tracking methods, since (i) it essentially represents a challenging scenario; and (ii) all algorithms are evaluated on the same data. In particular, here, we employed 10 checkerboard (indoor) sequences and 12 cars-and-people (outdoor) sequences from the well-known Hopkins155 dataset (Tron and Vidal [2007]). Moreover, we used another 8 outdoor sequences from the more recent MTPV dataset (Li et al. [2013]). To test the robustness of the different methods, we added different levels of Gaussian noise (with variance $\sigma^2 = 0.01, 0.02, 0.03$, or 0.04)⁵ to the images. Altogether, this results in 150 evaluation sequences.

To compare the algorithms, we measure the number of tracking errors, i.e., the number of points that drift from the ground-truth by more than a certain error tolerance ϵ . Note that, in the sequences that we use, the ground-truth was obtained by the

⁵Note that the intensities of the images are normalized to $[0, 1]$. So $\sigma^2 = 0.04$ already represents more noise than would typically occur in practice.

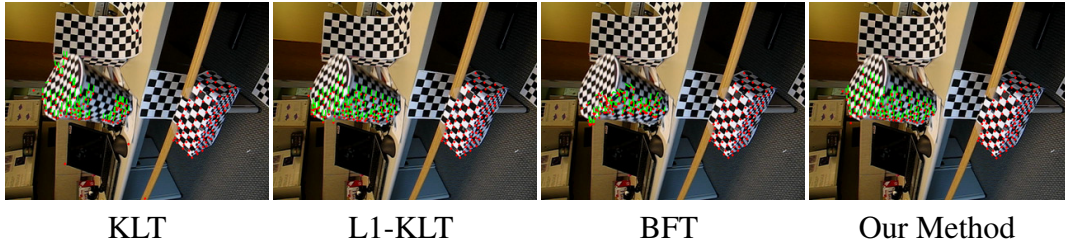


Figure 6.1: **Performance of different trackers on the 1RT2TC checkerboard sequence:** The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. Best viewed zoomed-in on screen.

standard KLT tracker and then manually cleaned up, so the ground-truth itself contains some noise whose level depends on the scene itself. In particular, we observed that the ground-truth of the indoor checkerboard sequences generally has more noise than that of the outdoor sequences. Therefore, we set a larger error tolerance for the checkerboard sequences ($\varepsilon = 10$) than for the outdoor ones ($\varepsilon = 5$). For every sequence, we compute the average number of incorrectly tracked feature points over all the frames, and then average this number over the sequences.

6.3.1 Hopkins Checkerboard Sequences

We first evaluated our method and the baselines on the Hopkins checkerboard sequences, which depict controlled indoor scenes with multiple rigidly moving objects. Generally, the repetitive texture in these sequences makes feature tracking more ambiguous and thus harder. However, in this experiment, we show that our multi-body feature tracker is more robust to this ambiguity. To provide a fair comparison, we used the same patch size (7×7) and the same number of image pyramid levels (4) for all the methods. Furthermore, we initialized all the tracking methods with the ground-truth locations of the feature points in the first frame.

From Table 6.1, we can see that the L1-KLT tracker consistently achieves better results than the original KLT tracker and than BFT. Our algorithm, however, consistently outperforms L1-KLT, which clearly evidences the benefits of incorporating our multi-body prior. We observed that BFT generally fails to track moving objects, as illustrated in Figure 6.1. This is mainly because BFT heavily relies on a good estimate of the global motion, obtained by registering the entire current image to the previous one. For scenes with multiple motions, however, global motion estimation becomes unreliable, thus causing BFT to fail to track the moving objects. Note that the performance of all the trackers remain relatively unaffected as the noise level increases. This is mainly due to the fact that the corners in the checkerboard, while resembling each other, are very strong features that are robust to noise.

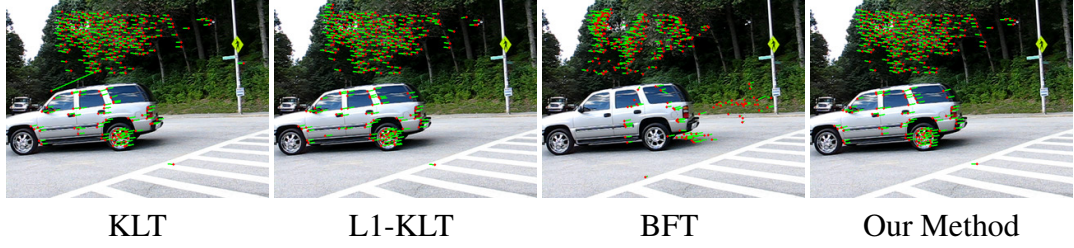


Figure 6.2: **Performance of different trackers on the car1 sequence:** The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. Best viewed in color.

Table 6.1: Average number of tracking errors ($\epsilon = 10$) on the Hopkins checkerboard sequences with noise of different variances σ^2 . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.00$	47.63	34.69	39.68	27.77
$\sigma^2 = 0.01$	46.92	30.86	39.30	27.32
$\sigma^2 = 0.02$	45.95	29.69	38.84	27.13
$\sigma^2 = 0.03$	46.59	30.16	39.16	28.18
$\sigma^2 = 0.04$	47.19	31.16	39.35	27.21

6.3.2 Hopkins Car-and-People Sequences

We then evaluated the algorithms on the Hopkins Car-and-People sequences, depicting real-world outdoor scenes with multiple rigid motions. The number of tracks provided by the ground-truth ranges from 147 to 548 with an average of 369. Here, for all the methods, we used the same patch size and image pyramid levels as in the previous experiment, and initialized the feature points with their ground-truth locations in the first frame. The average number of tracking errors for the different methods under different image noise level is reported in Table 6.2. Again, our multi-body feature tracker achieves the lowest tracking error compared to the baselines, which confirms the robustness of our method. A typical example of the results of all the methods on these sequences is shown in Figure 6.2.

Table 6.2: Average number of tracking errors ($\epsilon = 5$) on the Hopkins Car-and-People sequences with noise of different variances σ^2 . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.00$	21.71	24.28	49.13	16.14
$\sigma^2 = 0.01$	34.59	29.31	51.69	18.82
$\sigma^2 = 0.02$	54.95	36.32	54.63	26.56
$\sigma^2 = 0.03$	76.02	46.49	57.57	33.80
$\sigma^2 = 0.04$	95.17	56.92	58.36	42.43

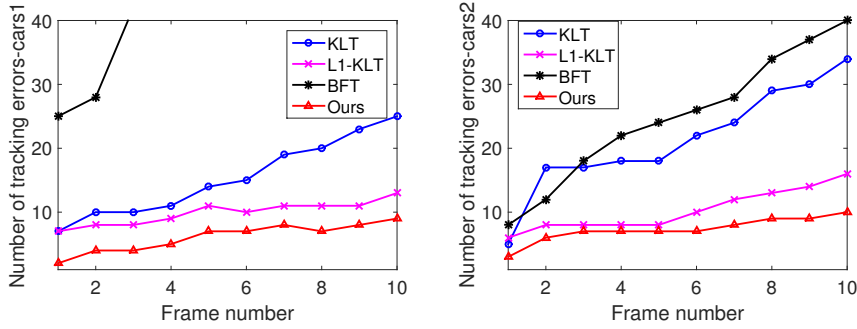


Figure 6.3: **Tracking error as a function of the frame number:** In these two typical sequences, our method consistently outperforms the baselines, and is less prone to tracking drift over time.

To give a better idea of the behavior of the methods over time, in Figure 6.3, we show the tracking error as a function of the frame number for two typical sequences (cars1 and cars2 with $\sigma^2=0.01$).

6.3.3 MTPV Sequences

We further tested our method on the MTPV sequences, which provide images of higher quality and resolution⁶ than the Hopkins dataset. By contrast, however, this dataset contains some outliers and missing data. For evaluation purpose, i.e., to create a complete and accurate ground-truth, we discarded the outliers and missing data. Since the image resolution is higher in this dataset, we used a larger patch size of 13×13 for all the methods. The results of all the algorithms are provided in Table 6.3. Note that we still outperform all the baselines for most noise levels, with the exception of BFT for $\sigma^2 = 0.04$. We believe that the slightly less impressive gap between our approach and the baselines, in particular BFT, is due to the fact that the feature points in this dataset are often dominated by the background. See Figure 6.4 for typical examples of this dataset.

6.3.4 KITTI Sequences

To evaluate the algorithms on realistic, high-quality images, we employed four sequences⁷ from KITTI (Geiger et al. [2012]), depicting street/traffic scenes with multiple motions. Since no ground-truth trajectories are provided with this data, to obtain quantitative results, we took 10 consecutive frames from each sequence, applied the

⁶Note, however, that they are still highly compressed and not well-suited for tracking, as pointed out in the readme file of the dataset.

⁷2011_09_26_drive_0018, 2011_09_26_drive_0051, 2011_09_26_drive_0056, and 2011_09_28_drive_0016.

Table 6.3: Average number of tracking errors ($\varepsilon = 5$) on the MTPV sequences with noise of different variances σ^2 . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.00$	3.07	13.34	6.83	2.34
$\sigma^2 = 0.01$	17.76	22.12	8.84	3.87
$\sigma^2 = 0.02$	28.39	27.26	11.17	6.94
$\sigma^2 = 0.03$	40.61	35.53	11.26	9.92
$\sigma^2 = 0.04$	47.69	38.93	12.34	13.22



Figure 6.4: **The MAN and MONK sequences of the MTPV dataset:** The feature points are marked as red. Note that the number of points on the walking man and monk is much smaller than on the background.

KLT tracker to them, and manually cleaned up the results to get ground-truth trajectories with an average of 177 points per sequence. The results of this experiment for different levels of noise added to the input are reported in Table 6.4, and Figure 6.5 shows a qualitative comparison of the algorithms. Note that our method also outperforms the baselines on this data.

Table 6.4: Average number of tracking errors ($\varepsilon = 5$) on the KITTI sequences with different noise variances σ^2 . The lower, the better.

Methods	KLT	L1-KLT	BFT	Ours
$\sigma^2 = 0.01$	21.43	22.05	27.48	14.18
$\sigma^2 = 0.02$	24.35	22.85	27.80	16.70
$\sigma^2 = 0.03$	31.15	26.88	27.85	17.70
$\sigma^2 = 0.04$	34.43	29.23	27.75	20.33

6.3.5 Frame-by-Frame Motion Segmentation

In our formulation, we optimize our energy function w.r.t. two variables: the displacement vector $\mathbf{u}$ and the self-expressiveness coefficients $\mathbf{C}$. While the vector $\mathbf{u}$

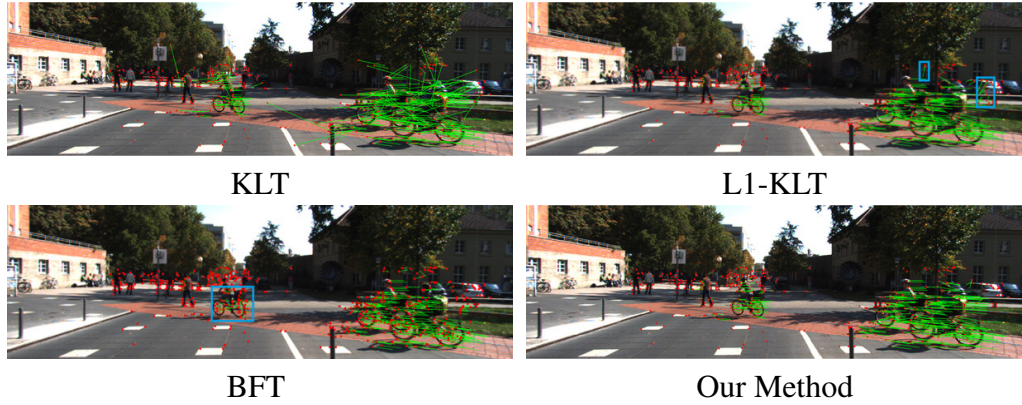


Figure 6.5: **Performance of different trackers on the KITTI sequence:** The red points denote the current positions of the feature points, and the green lines the motion since the previous frame. As evidenced by the regions highlighted with a blue rectangle, L1-KLT and BFT make more tracking errors than our approach. Best viewed on screen with zoom-in.

provides the tracking results, the matrix $\mathbf{C}$, as in the subspace clustering literature, can be used to build an affinity matrix for spectral clustering, and thus, if we assume that the number of motions is known *a priori*, lets us perform motion segmentation. In other words, our method can also be interpreted as simultaneous feature tracking and frame-by-frame motion segmentation. In this experiment, we therefore aim to evaluate the frame-by-frame motion segmentation accuracy of our method. Since, to the best of our knowledge, no existing motion segmentation methods perform feature tracking and frame-by-frame motion segmentation jointly⁸, we compare our results with the following two-steps baselines: first, we find the tracks by KLT or L1-KLT and form the epipolar subspaces as in Eq. (6.7); second, we apply a subspace clustering method, i.e., Sparse Subspace Clustering (SSC) or Efficient Dense Subspace Clustering (EDSC), to perform motion segmentation. This results in four baselines denoted by KLT+SSC (Li et al. [2013]), KLT+EDSC, L1+SSC and L1+EDSC. The results of motion segmentation on the 22 Hopkins sequences used previously are shown in Table 6.5. These results clearly evidence that our method outperforms the baselines significantly in terms of motion segmentation.

Runtimes: Typical runtimes (e.g., car8 of Hopkins155 which contains 192 trajectories) are: KLT – 0.002 sec, L1-KLT – 1.1 sec, BFT – 0.6 sec, Ours – 0.9 sec. This indicates that, while much slower than the original KLT, L1-KLT, BFT and our method are on par in terms of runtimes.

⁸Note that our method in Chapter 5 works in batch rather than in a frame-by-frame (online) manner.

Table 6.5: Average error rate (in %) of two-frame motion segmentation on the 22 Hopkins sequences with noise of different variances σ^2 . The lower, the better.

Methods	KLT+SSC	KLT+EDSC	L1+SSC	L1+EDSC	Ours
$\sigma^2 = 0.00$	19.76	20.57	18.71	19.11	8.97
$\sigma^2 = 0.01$	19.76	20.61	19.61	20.41	9.35
$\sigma^2 = 0.02$	19.21	20.99	21.02	21.92	9.33
$\sigma^2 = 0.03$	20.63	20.69	22.21	20.48	9.89
$\sigma^2 = 0.04$	20.38	19.82	21.35	20.80	11.26

6.4 Summary

In this chapter, we have introduced a novel feature tracker that incorporates a multi-body rigidity prior into feature tracking. To this end, we have derived epipolar subspace constraints that prevent us from having to compute fundamental matrices and motion assignments explicitly. Our formulation only involves a series of convex subproblems, all of which have closed-form solutions. We have demonstrated the effectiveness of our method via extensive experiments on indoor and outdoor sequences. Importantly, this feature tracker provides an alternative way to get reliable feature trajectories for motion segmentation methods in Chapter 3 and 4.

While adding global rigidity constraints (be it the low-rank or the epipolar subspace constraints) to the local KLT tracker improves robustness, it comes with some computational overhead. These methods could be further sped up, for instance by exploiting the GPU. Furthermore, our current model assumes that each patch undergoes only translation between consecutive frames. One could also investigate the use of more accurate models, such as affine transformations.

Conclusions and Future Work

7.1 Conclusions

In this thesis, we have addressed the motion segmentation problem with subspace constraints. The contributions of this thesis are three-fold: first, when the trajectories across the frames are given, we have proposed robust and efficient subspace clustering methods for motion segmentation that can handle various types of data corruption including noise, outliers and missing entries (Chapter 3 and 4); second, with unknown feature correspondences, we have formulated the problem in terms of partial permutation matrices and subspace self-expressiveness constraints, and have jointly solved for motion segmentation and feature matching (Chapter 5); third, for the prerequisite step (i.e., frame-by-frame feature tracking) of many applications such as the motion segmentation methods in Chapter 3 and 4, we have presented a robust multi-body feature tracker that regularizes the feature tracking with the epipolar subspace constraints (Chapter 6).

To be specific, in Chapter 3, we have introduced an efficient dense subspace clustering algorithm that effectively handles noise and outliers. To this end, we have relied on a Frobenius norm formulation, which yields dense connections within each cluster. When the data is corrupted by noise, our formulation has a closed-form solution. With noise and outliers, ADMM still yields an efficient solution. Furthermore, we have shown the connections between Frobenius norm minimization, nuclear norm minimization, the Shape Interaction Matrix, and the null space projection matrix.

In Chapter 4, we have revisited the 20-year-old Shape Interaction Matrix method and revealed its connections to several recent subspace clustering methods. Based on our analysis, we have made the Shape Interaction Matrix method robust via three simple steps, i.e., normalization, element-wise powering, and rank selection. The first two steps can be equivalently interpreted as applying a normalized polynomial kernel, which makes the data more separable, and the third step is justified by the matrix perturbation theory. We have then further introduced an algorithm that robustly recovers the row-space of the data from incomplete measurements, thus effectively making the powerful Shape Interaction Matrix representation applicable to the missing

data scenario.

In Chapter 5, we have tackled the motion segmentation problem with unknown correspondences. Our problem formulation is based on two important constraints: First, the recovered inlier point trajectories should satisfy subspace constraints; Second, the matching feature descriptors should be low-rank, ideally rank one. With these two constraints, we have formulated our problem in terms of PPMs, which simultaneously select and reorder the inlier points. The resulting optimization problem is solved via ADMM, where five out of the six subproblems are solved in closed form and the remaining binary assignment subproblem is solved via the Hungarian algorithm.

In Chapter 6, we have presented a novel feature tracker that incorporates a multi-body rigidity prior into feature tracking. To this end, we have derived the epipolar subspace constraints that free us from having to compute fundamental matrices and motion assignments explicitly. Our formulation only involves a series of convex subproblems, all of which have closed-form solutions.

All in all, we have addressed the motion segmentation problem under various challenging scenarios and have presented robust motion segmentation methods with subspace constraints. In general, the feature tracking method presented in Chapter 6 can be used in methods of Chapter 3 and 4 to get reliable feature trajectories, and Chapter 5 provides an alternative motion segmentation method in scenarios where obtaining reliable feature trajectories beforehand is not possible. The source codes for all the proposed methods in this thesis have been released in my personal webpage¹.

7.2 Future Work

In most of the previous chapters (except in Chapter 6), we have assumed an affine camera model, i.e., the depth fluctuations of the scene are negligible compared to the distance to the camera. This model leads to decent formulations where each motion can be represented by a linear subspace with dimension up to four. In practice, this camera model has a considerable amount of limitations. For example, when the scene contains strong perspective effects (which is quite common in traffic scenes), the affine camera model no longer applies. One of the future research directions, therefore, is to extend the applicability of the present motion segmentation methods to a more general camera model, i.e., the perspective camera model. We know that given the projective depth of each point, the data matrix can still be factorized into the product of two low-dimensional matrices (with dimension up to four) (Hartley and Vidal [2008]; Dai et al. [2010]). One direct extension would be to jointly recover the projective depths and motion clusters with our subspace formulations. However, this will result in a non-convex optimization problem, so the main hurdle will be to design algorithms that can effectively solve this problem.

¹<https://sites.google.com/site/peterji1990/>

Up to now, we have only considered the cases of rigid motion segmentation. However, non-rigid motions are also quite common. We know that non-rigid body motions lie in low-dimensional manifolds (Goh and Vidal [2007]). Therefore, another possible direction would be to design specific kernels that project points into high dimensional Hilbert space where data points become more separable. Simple kernels such as the polynomial kernel and the Gaussian RBF kernel may not suffice for this purpose, but one could use the multiple kernel learning technique (Gönen and Alpaydm [2011]) to learn a kernel from a set of base kernels. Besides, if a set of pairwise points is identified to be on the same non-rigid motion, one could also derive projected kernels as in (Tuzel et al. [2009]) to project the trajectories of the same motion into the same point in the high-dimensional Hilbert space, where the segmentation/clustering becomes much easier.

Moreover, the aforementioned methods are mostly based on sparse feature trajectories. While sparse feature tracks are often more reliable, dense segmentation is also attractive for practical use. In the case of dense motion segmentation, we may consider formulating the problem within a framework of optical flow (e.g., the total variation method) and jointly solving the motion segmentation and optical flow problems. Under certain assumptions (e.g., an affine camera model and small displacements), Irani [2002] has shown that the flow vectors of multiple independent motions lie in multiple independent subspaces. We may resort to the image pyramid technique to meet these assumptions, especially the assumption that the displacements are small.

Another possible research topic could be to combine geometric context and semantic understanding with motion segmentation. High-level semantic understanding is often more informative, and thus should be helpful for motion segmentation in difficult cases, such as objects moving out and reoccurring. Furthermore, good motion segmentation results should help disambiguating semantic segmentation patches with similar appearance but coming from different motions. By combining these two problems into a unified framework, we expect to get better results for both motion segmentation and semantic understanding.

Bibliography

- ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FUA, P.; AND SUSSTRUNK, S., 2012. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34, 11 (2012), 2274–2282. (cited on pages xv and 2)
- ADIV, G., 1985. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 4 (1985), 384–401. (cited on page 3)
- AMIAZ, T. AND KIRYATI, N., 2005. Dense discontinuous optical flow via contour-based segmentation. In *IEEE International Conference on Image Processing 2005*, vol. 3, III–1264. IEEE. (cited on page 3)
- AMIAZ, T. AND KIRYATI, N., 2006. Piecewise-smooth dense optical flow via level sets. *International Journal of Computer Vision*, 68, 2 (2006), 111–124. (cited on pages 3 and 74)
- BAKER, S. AND MATTHEWS, I., 2004. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56, 3 (2004), 221–255. (cited on page 89)
- BALZANO, L.; NOWAK, R.; AND RECHT, B., 2010. Online identification and tracking of subspaces from highly incomplete information. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 704–711. IEEE. (cited on pages 59 and 61)
- BLACK, M. J. AND ANANDAN, P., 1990. Constraints for the early detection of discontinuity from motion. In *Proceedings of the eighth National conference on Artificial intelligence-Volume 2*, 1060–1066. AAAI Press. (cited on page 3)
- BOUGUET, J.-Y., 2001. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Technical Report, Intel Microprocessor Research Labs*, (2001). (cited on pages 90 and 100)
- BOULT, T. AND BROWN, L., 1991. Factorization-based segmentation of motions. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, 179–186. (cited on page 6)

- BOURGEOIS, F. AND LASSALLE, J.-C., 1971. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14, 12 (1971), 802–804. (cited on page 30)
- BOYD, S.; PARIKH, N.; CHU, E.; PELEATO, B.; AND ECKSTEIN, J., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3, 1 (2011), 1–122. (cited on pages 25, 27, 42, and 96)
- BRADLEY, P. AND MANGASARIAN, O., 2000. k-plane clustering. *J. of Global Optimization*, 16, 1 (2000), 23–32. (cited on page 7)
- BROX, T.; BRUHN, A.; PAPENBERG, N.; AND WEICKERT, J., 2004. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004*. (cited on page 3)
- BROX, T.; BRUHN, A.; AND WEICKERT, J., 2006. Variational motion segmentation with level sets. In *European conference on computer vision*, 471–483. Springer. (cited on page 3)
- BROX, T. AND MALIK, J., 2010. Object segmentation by long term analysis of point trajectories. In *Computer Vision-ECCV 2010*, 282–295. Springer. (cited on page 5)
- BROX, T. AND MALIK, J., 2011. Large displacement optical flow: Descriptor matching in variational motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33, 3 (2011), 500–513. (cited on page 75)
- BROX, T. AND WEICKERT, J., 2006. Level set segmentation with multiple regions. *IEEE Transactions on Image Processing*, 15, 10 (2006), 3213–3218. (cited on page 3)
- BRUHN, A.; WEICKERT, J.; AND SCHNÖRR, C., 2005. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61, 3 (2005), 211–231. (cited on page 91)
- BUCHANAN, A. AND FITZGIBBON, A., 2007. Combining local and global motion models for feature point tracking. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. (cited on pages 89 and 90)
- CAI, J.-F.; CANDÈS, E. J.; AND SHEN, Z., 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20, 4 (2010), 1956–1982. (cited on pages 80 and 97)
- CANDÈS, E. J.; LI, X.; MA, Y.; AND WRIGHT, J., 2011. Robust principal component analysis? *Journal of the ACM (JACM)*, 58, 3 (2011), 11. (cited on page 64)

-
- CHEN, G. AND LERMAN, G., 2009. Spectral curvature clustering (SCC). *International Journal of Computer Vision*, 81, 9 (2009), 317–330. (cited on pages 7, 16, 22, and 44)
- CHERIYADAT, A. M. AND RADKE, R. J., 2009. Non-negative matrix factorization of partial track data for motion segmentation. In *Computer vision, 2009. The proceedings of the seventh IEEE international conference on*, 865–872. (cited on pages 6, 75, and 76)
- CHIKUSE, Y., 2003. *Statistics on Special Manifolds*. Springer. (cited on page 59)
- CHIN, T.-J.; WANG, H.; AND SUTER, D., 2009. The ordered residual kernel for robust motion subspace clustering. In *Advances in Neural Information Processing Systems*, 333–341. (cited on page 7)
- COSTEIRA, J. AND KANADE, T., 1995. A multi-body factorization method for motion analysis. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, 1071–1076. (cited on pages 49, 50, 51, and 75)
- COSTEIRA, J. AND KANADE, T., 1998. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29, 3 (1998), 159–179. (cited on pages 6, 38, 52, 56, and 91)
- CREMERS, D. AND SOATTO, S., 2005a. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62, 3 (2005), 249–265. (cited on page 3)
- CREMERS, D. AND SOATTO, S., 2005b. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62, 3 (2005), 249–265. (cited on page 74)
- DAI, Y.; LI, H.; AND HE, M., 2010. Element-wise factorization for n-view projective reconstruction. In *Computer Vision – ECCV 2010*, vol. 6314 of *Lecture Notes in Computer Science*, 396–409. Springer Berlin Heidelberg. (cited on page 110)
- DAVIS, C. AND KAHAN, W. M., 1970. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7, 1 (1970), 1–46. (cited on page 57)
- DRAGON, R.; OSTERMANN, J.; AND VAN GOOL, L., 2013. Robust realtime motion-split-and-merge for motion segmentation. In *Pattern Recognition*, 425–434. Springer. (cited on pages 4 and 87)
- DRAGON, R.; ROSENHAHN, B.; AND OSTERMANN, J., 2012. Multi-scale clustering of frame-to-frame correspondences for motion segmentation. In *Computer Vision–ECCV 2012*, 445–458. Springer. (cited on page 4)

- DUCHENNE, O.; BACH, F.; KWEON, I.; AND PONCE, J., 2009. A tensor-based algorithm for high-order graph matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 1980–1987. (cited on page 75)
- DUDA, R. O.; HART, P. E.; AND STORK, D. G., 2012. *Pattern classification*. John Wiley & Sons. (cited on page 7)
- DYER, E. L.; STUDER, C.; AND BARANIUK, R. G., 2013. Subspace clustering with dense representations. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. (cited on page 34)
- EDELMAN, A.; ARIAS, T.; AND SMITH, S., 1998. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20, 2 (1998), 303–353. (cited on pages 17 and 60)
- ELHAMIFAR, E. AND VIDAL, R., 2009. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2790–2797. IEEE. (cited on pages 8, 20, 22, 49, 53, and 64)
- ELHAMIFAR, E. AND VIDAL, R., 2010. Clustering disjoint subspaces via sparse representation. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 1926–1929. IEEE. (cited on page 8)
- ELHAMIFAR, E. AND VIDAL, R., 2013. Sparse subspace clustering: Algorithm, theory, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35, 11 (2013), 2765–2781. (cited on pages 8, 21, 33, 36, 44, 46, 49, 53, 61, 62, 63, 64, 73, 74, 75, 76, 85, 91, and 94)
- ELQURSH, A. AND ELGAMMAL, A., 2013. Online motion segmentation using dynamic label propagation. In *Proceedings of the IEEE International Conference on Computer Vision, 2008–2015*. (cited on page 4)
- FAVARO, P.; VIDAL, R.; AND RAVICHANDRAN, A., 2011. A closed form solution to robust subspace estimation and clustering. In *Computer Vision and Pattern Recognition, 2011. CVPR 2011. IEEE Conference on*. (cited on pages 8, 9, and 41)
- FENG, J.; LIN, Z.; XU, H.; AND YAN, S., 2014. Robust subspace segmentation with block-diagonal prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3818–3825. (cited on page 9)
- FISCHLER, M. A. AND BOLLES, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, 6 (1981), 381–395. (cited on page 4)

-
- GAO, H.; NIE, F.; LI, X.; AND HUANG, H., 2015. Multi-view subspace clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, 4238–4246. (cited on page 8)
- GARG, R.; PIZARRO, L.; RUECKERT, D.; AND AGAPITO, L., 2010. Dense multi-frame optic flow for non-rigid objects using subspace constraints. In *Computer Vision—ACCV 2010*. (cited on page 91)
- GARG, R.; ROUSSOS, A.; AND AGAPITO, L., 2013. A variational approach to video registration with subspace constraints. *International Journal of Computer Vision*, 104, 3 (2013), 286–314. (cited on page 91)
- GEAR, C., 1994. Feature grouping in moving objects. In *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*, 214–219. (cited on page 52)
- GEAR, C., 1998. Multibody grouping from motion images. *International Journal of Computer Vision*, 29, 2 (1998), 133–150. (cited on pages 6, 52, and 56)
- GEIGER, A.; LENZ, P.; AND URTASUN, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. (cited on page 104)
- GEORGHIADES, A.; BELHUMEUR, P.; AND KRIEGMAN, D., 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23, 6 (2001), 643–660. (cited on page 34)
- GOH, A. AND VIDAL, R., 2007. Segmenting motions of different types by unsupervised manifold clustering. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–6. IEEE. (cited on page 111)
- GÖNEN, M. AND ALPAYDIN, E., 2011. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12, Jul (2011), 2211–2268. (cited on page 111)
- HARTLEY, R. AND SCHAFFALITZKY, F., 2003. Powerfactorization: 3d reconstruction with missing or uncertain data. In *Australia-Japan advanced workshop on computer vision*, vol. 74, 76–85. (cited on page 64)
- HARTLEY, R. AND VIDAL, R., 2008. Perspective nonrigid shape and motion recovery. In *Computer Vision—ECCV 2008*, 276–289. Springer. (cited on page 110)
- HARTLEY, R. I. AND ZISSERMAN, A., 2004. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edn. (cited on pages 17 and 92)

- HO, J.; YANG, M.; LIM, J.; LEE, K.; AND KRIEGMAN, D., 2003. Clustering appearances of objects under varying illumination conditions. In *Computer Vision and Pattern Recognition, 2003. CVPR 2003. IEEE Conference on*. (cited on page 33)
- HONG, W.; WRIGHT, J.; HUANG, K.; AND MA, Y., 2006. Multi-scale hybrid linear models for lossy image representation. *Image Processing, IEEE Transactions on*, 15, 12 (2006), 3655–3671. (cited on page 33)
- HORN, B. K. AND SCHUNCK, B. G., 1981. Determining optical flow. In *Technical symposium east*, 319–331. International Society for Optics and Photonics. (cited on page 3)
- HU, H.; LIN, Z.; FENG, J.; AND ZHOU, J., 2014. Smooth representation clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3834–3841. (cited on page 9)
- ICHIMURA, N., 1999. Motion segmentation based on factorization method and discriminant criterion. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 600–605. IEEE. (cited on pages 6, 49, and 52)
- IRANI, M., 2002. Multi-frame correspondence estimation using subspace constraints. *International Journal of Computer Vision*, 48, 3 (2002), 173 – 194. (cited on page 111)
- JAIN, S. AND GOVINDU, V., 2013. Efficient higher-order clustering on the Grassmann manifold. In *Computer vision, 2013. The proceedings of the seventh IEEE international conference on*. (cited on page 7)
- JI, P.; SALZMANN, M.; AND LI, H., 2014. Efficient dense subspace clustering. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, 461–468. (cited on page 63)
- JIA, K.; CHAN, T.-H.; ZENG, Z.; GAO, S.; WANG, G.; ZHANG, T.; AND MA, Y., 2015. Roml: A robust feature correspondence approach for matching objects in a set of images. *International Journal of Computer Vision*, (2015), 1–25. (cited on pages 27, 73, 75, and 85)
- JOLLIFFE, I., 1986. *Principle Component Analysis*. Springer-Verlag, New York. (cited on page 33)
- JUNG, H.; JU, J.; AND KIM, J., 2014. Rigid motion segmentation using randomized voting. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1210–1217. IEEE. (cited on page 5)

-
- KANATANI, K., 2001. Motion segmentation by subspace separation and model selection. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, 586–591 vol.2. (cited on pages 6 and 52)
- KANATANI, K., 2002. Evaluation and selection of models for motion segmentation. In *Computer Vision – ECCV 2002*, vol. 2352 of *Lecture Notes in Computer Science*, 335–349. (cited on pages 6 and 52)
- KANATANI, K. AND SUGAYA, Y., 2003. Multi-stage optimization for multi-body motion segmentation. In *Australia-Japan Advanced Workshop on Computer Vision*, vol. 2, 7. (cited on page 52)
- KLAPPSTEIN, J.; VAUDREY, T.; RABE, C.; WEDEL, A.; AND KLETTE, R., 2009. Moving object segmentation using optical flow and depth information. *Advances in Image and Video Technology*, (2009), 611–623. (cited on page 3)
- LAUER, F. AND SCHNORR, C., 2009. Spectral clustering of linear subspaces for motion segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, 678–685. (cited on pages 52, 55, and 56)
- LEE, D. D. AND SEUNG, H. S., 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, 6755 (1999), 788–791. (cited on page 6)
- LEON, S. J., 2010. *Linear algebra with applications*. Prentice Hall. (cited on page 14)
- LEORDEANU, M. AND HEBERT, M., 2005. A spectral technique for correspondence problems using pairwise constraints. In *Computer vision, 2005. The proceedings of the seventh IEEE international conference on*, 1–8. (cited on page 75)
- LI, B.; ZHANG, Y.; LIN, Z.; AND LU, H., 2015. Subspace clustering by mixture of gaussian regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2094–2102. (cited on page 9)
- LI, C.-G. AND VIDAL, R., 2015. Structured sparse subspace clustering: a unified optimization framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 277–286. (cited on page 9)
- LI, G. AND PONG, T. K., 2015. Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization*, 25, 4 (2015), 2434–2460. (cited on page 27)
- LI, H., 2007. Two-view motion segmentation from linear programming relaxation. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. (cited on pages 4, 74, and 91)

- LI, Z.; GUO, J.; CHEONG, L.-F.; AND ZHOU, S. Z., 2013. Perspective motion segmentation via collaborative clustering. In *Proceedings of the IEEE International Conference on Computer Vision*. (cited on pages 5, 91, 94, 101, and 106)
- LIN, Z.; CHEN, M.; WU, L.; AND MA, Y., 2009. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report*, UILU-ENG-09-2215 (2009). (cited on pages 27 and 42)
- LIU, G.; LIN, Z.; YAN, S.; SUN, J.; YU, Y.; AND MA, Y., 2013. Robust recovery of subspace structures by low-rank representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35, 1 (2013), 171–184. (cited on pages 8, 22, 33, 36, 38, 43, 44, 49, 53, 62, 63, 73, 74, 75, 76, 85, and 94)
- LIU, G.; LIN, Z.; AND YU, Y., 2010. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 663–670. (cited on pages 8, 22, 27, 28, 38, 39, 49, and 53)
- LIU, G. AND YAN, S., 2011. Latent low-rank representation for subspace segmentation and feature extraction. In *2011 International Conference on Computer Vision*, 1615–1622. IEEE. (cited on page 9)
- LOWE, D., 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2 (2004), 91–110. (cited on page 75)
- LU, C.; FENG, J.; LIN, Z.; AND YAN, S., 2013. Correlation adaptive subspace segmentation by trace lasso. In *Proceedings of the IEEE International Conference on Computer Vision*, 1345–1352. (cited on page 9)
- LU, C.-Y.; MIN, H.; ZHAO, Z.-Q.; ZHU, L.; HUANG, D.-S.; AND YAN, S., 2012. Robust and efficient subspace segmentation via least squares regression. In *Computer Vision – ECCV 2012*, 347–360. (cited on pages 9, 34, 73, 74, and 75)
- LUCAS, B. D.; KANADE, T.; ET AL., 1981. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, vol. 81, 674–679. (cited on pages 89 and 90)
- MA, Y. AND DERKSEN, H., 2007. Segmentation of multivariate mixed data via lossy data coding and compression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29, 9 (2007), 1–17. (cited on page 7)
- MA, Y.; DERKSEN, H.; HONG, W.; AND WRIGHT, J., 2007. Segmentation of multivariate mixed data via lossy data coding and compression. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29, 9 (2007), 1546–1562. (cited on page 56)

-
- MA, Y.; SOATTO, S.; KOSECKA, J.; AND SASTRY, S. S., 2012. *An invitation to 3-d vision: from images to geometric models*, vol. 26. Springer Science & Business Media. (cited on page 17)
- MACQUEEN, J. ET AL., 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, 281–297. Oakland, CA, USA. (cited on page 7)
- MÉMIN, E. AND PÉREZ, P., 2002. Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision*, 46, 2 (2002), 129–155. (cited on page 3)
- MIKOLAJCZYK, K. AND SCHMID, C., 2004. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60, 1 (2004), 63–86. (cited on page 75)
- MO, Q. AND DRAPER, B. A., 2012. Semi-nonnegative matrix factorization for motion segmentation with missing data. In *Computer Vision–ECCV 2012*, 402–415. Springer. (cited on page 6)
- MUNKRES, J., 1957. Algorithms for the assignment and transportation problems. *Journal of SIAM*, 5, 1 (1957), 32–38. (cited on page 80)
- MURPHY, K. P., 2012. *Machine learning: a probabilistic perspective*. MIT press. (cited on page 1)
- NAGEL, H.-H.; SOCHER, G.; KOLLNIG, H.; AND OTTE, M., 1994. Motion boundary detection in image sequences by local stochastic tests. In *Computer Vision–ECCV'94*, 305–315. Springer. (cited on page 3)
- NARAYANA, M.; HANSON, A.; AND LEARNED-MILLER, E., 2013. Coherent motion segmentation in moving camera videos using optical flow orientations. In *Proceedings of the IEEE International Conference on Computer Vision*, 1577–1584. (cited on page 3)
- NASIHATKON, B. AND HARTLEY, R., 2011. Graph connectivity in sparse subspace clustering. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2137–2144. IEEE. (cited on page 33)
- NG, A. Y.; JORDAN, M. I.; WEISS, Y.; ET AL., 2002. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2 (2002), 849–856. (cited on pages 4, 25, and 52)

- OCHS, P. AND BROX, T., 2012. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition, 2012. CVPR 2012. IEEE Conference on*. (cited on page 5)
- OCHS, P.; MALIK, J.; AND BROX, T., 2014. Segmentation of moving objects by long term video analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 6 (2014), 1187–1200. (cited on pages 1, 5, 74, and 75)
- OLIVEIRA, R.; COSTEIRA, J.; AND XAVIER, J., 2005. Optimal point correspondence through the use of rank constraints. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Conference on*, 1–6. (cited on pages 73, 75, and 76)
- PARK, J.; ZHA, H.; AND KASTURI, R., 2004. Spectral clustering for robust motion segmentation. In *Computer Vision - ECCV 2004*, vol. 3024 of *Lecture Notes in Computer Science*, 390–401. (cited on pages 52 and 61)
- PATEL, V. M.; VAN NGUYEN, H.; AND VIDAL, R., 2013. Latent space sparse subspace clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, 225–232. (cited on page 9)
- PATEL, V. M. AND VIDAL, R., 2014. Kernel sparse subspace clustering. In *2014 IEEE International Conference on Image Processing (ICIP)*, 2849–2853. IEEE. (cited on page 9)
- PENG, X.; LU, C.; YI, Z.; AND TANG, H., 2015. Connections between nuclear norm and frobenius norm based representation. *arXiv preprint arXiv:1502.07423*, (2015). (cited on page 53)
- PENG, X.; ZHANG, L.; AND YI, Z., 2013. Scalable sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 430–437. (cited on page 8)
- PHAM, D.-S.; BUDHADITYA, S.; PHUNG, D.; AND VENKATESH, S., 2012. Improved subspace clustering via exploitation of spatial constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 550–557. IEEE. (cited on page 8)
- PICCARDI, M., 2004. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4, 3099–3104. IEEE. (cited on page 2)
- PICCINI, T.; PERSSON, M.; NORDBERG, K.; FELSBURG, M.; AND MESTER, R., 2014. Good edgels to track: Beating the aperture problem with epipolar geometry. In *Computer Vision and ECCV Workshops*. (cited on pages 91 and 92)

-
- POLING, B.; LERMAN, G.; AND SZLAM, A., 2014. Better feature tracking through subspace constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3454–3461. (cited on pages 89, 90, 92, 93, and 100)
- PURKAIT, P.; CHIN, T.-J.; ACKERMANN, H.; AND SUTER, D., 2014. Clustering with hypergraphs: the case for large hyperedges. In *Computer Vision–ECCV 2014*, 672–687. Springer. (cited on page 8)
- RAO, S.; TRON, R.; VIDAL, R.; AND MA, Y., 2008. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8. (cited on page 64)
- RAO, S.; TRON, R.; VIDAL, R.; AND MA, Y., 2010a. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32, 10 (2010), 1832–1845. (cited on page 49)
- RAO, S.; YANG, A.; SASTRY, S.; AND MA, Y., 2010b. Robust algebraic segmentation of mixed rigid-body and planar motions in two views. *International Journal of Computer Vision*, 88 (2010), 425–446. (cited on page 6)
- SCOTT, G. L. AND LONGUET-HIGGINS, H. C., 1990. feature grouping by "relocalisation" of eigenvectors of the proximity matrix. In *Proceedings of British Machine Vision Conference*, 103–108. (cited on page 54)
- SHEIKH, Y.; JAVED, O.; AND KANADE, T., 2009. Background subtraction for freely moving cameras. In *Computer Vision, 2009 IEEE 12th International Conference on*, 1219–1225. IEEE. (cited on page 2)
- SHEN, Y.; WEN, Z.; AND ZHANG, Y., 2014. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29, 2 (2014), 239–263. (cited on pages 27 and 78)
- SHI, J. AND MALIK, J., 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22, 8 (2000), 888–905. (cited on pages 4, 24, 44, 52, 57, and 74)
- SHI, J. AND TOMASI, C., 1994. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. (cited on pages 75, 89, and 90)
- SIVIC, J.; SCHAFFALITZKY, F.; AND ZISSERMAN, A., 2006. Object level grouping for video shots. *International Journal of Computer Vision*, 67, 2 (2006), 189–210. (cited on page 75)

- SUN, D.; SUDDERTH, E. B.; AND BLACK, M. J., 2012. Layered segmentation and optical flow estimation over time. In *Computer Vision and Pattern Recognition, 2012. CVPR 2012. IEEE Conference on*, 1768–1775. (cited on page 74)
- SUNDARAM, N.; BROX, T.; AND KEUTZER, K., 2010. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Computer Vision–ECCV 2010*, 438–451. Springer. (cited on page 4)
- SZELISKI, R., 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media. (cited on page 92)
- THOMPSON, W. B.; MUTCH, K. M.; AND BERZINS, V. A., 1985. Dynamic occlusion analysis in optical flow fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , 4 (1985), 374–383. (cited on page 3)
- TIERNEY, S.; GAO, J.; AND GUO, Y., 2014. Subspace clustering for sequential data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1019–1026. (cited on page 8)
- TIPPING, M. E. AND BISHOP, C. M., 1999. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11, 2 (1999), 443–482. (cited on page 7)
- TOLDO, R. AND FUSIELLO, A., 2008. Robust multiple structures estimation with j-linkage. In *Computer Vision–ECCV 2008*, 537–547. Springer. (cited on page 4)
- TOMASI, C. AND KANADE, T., 1991. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh. (cited on pages 89 and 90)
- TOMASI, C. AND KANADE, T., 1992. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9, 2 (1992), 137–154. (cited on page 50)
- TORKI, M. AND ELGAMMAL, A., 2010. One-shot multi-set non-rigid feature-spatial matching. In *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*, 3058–3065. (cited on pages 75 and 85)
- TORRESANI, L. AND BREGLER, C., 2002. Space-time tracking. In *Computer Vision–ECCV 2002*. (cited on pages 89, 90, and 92)
- TORRESANI, L.; KOLMOGOROV, V.; AND ROTHER, C., 2008. Feature correspondence via graph matching: Models and global optimization. In *Computer Vision – ECCV 2008*, 596–609. (cited on page 75)

-
- TRON, R. AND VIDAL, R., 2007. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 1–8. IEEE. (cited on pages xv, 2, 10, 63, 73, 91, and 101)
- TSENG, P., 2000. Nearest q-flat to m points. *J. of Optimization Theory and Applications*, 105, 1 (2000), 249–252. (cited on page 7)
- TUZEL, O.; PORIKLI, F.; AND MEER, P., 2009. Kernel methods for weakly supervised mean shift clustering. In *Proceedings of the IEEE International Conference on Computer Vision*. (cited on page 111)
- VALGAERTS, L.; BRUHN, A.; AND WEICKERT, J., 2008. A variational model for the joint recovery of the fundamental matrix and the optical flow. In *Pattern Recognition*, 314–324. (cited on pages 91 and 92)
- VIDAL, R., 2011. Subspace clustering. *IEEE Signal Processing Magazine*, 28, 2 (2011), 52–68. (cited on pages 5 and 33)
- VIDAL, R. AND FAVARO, P., 2014. Low rank subspace clustering (LRSC). *Pattern Recognition Letters*, 43 (2014), 47–61. (cited on pages 44, 62, 73, and 74)
- VIDAL, R.; MA, Y.; AND SASTRY, S., 2005a. Generalized principle component analysis(gpca). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27, 12 (2005), 1945–1959. (cited on page 6)
- VIDAL, R.; MA, Y.; AND SASTRY, S., 2005b. Generalized principle component analysis(gpca). *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27, 12 (2005), 1945–1959. (cited on page 91)
- VIDAL, R.; MA, Y.; SOATTO, S.; AND SASTRY, S., 2006. Two-view multibody structure from motion. *International Journal of Computer Vision*, 68, 1 (2006), 7–25. (cited on page 4)
- VIDAL, R.; SOATTO, S.; MA, Y.; AND SASTRY, S., 2002. Segmentation of dynamic scenes from the multibody fundamental matrix. In *ECCV Workshop on Vision and Modeling of Dynamic Scenes*. (cited on page 91)
- VIDAL, R.; TRON, R.; AND HARTLEY, R., 2008. Multiframe motion segmentation with missing data using powerfactorization and gpca. *International Journal of Computer Vision*, 79, 1 (2008), 85–105. (cited on page 49)
- VOLZ, S.; BRUHN, A.; VALGAERTS, L.; AND ZIMMER, H., 2011. Modeling temporal coherence for optical flow. In *Computer vision, 2011. The proceedings of the seventh IEEE international conference on*, 1116–1123. (cited on page 74)

- VON LUXBURG, U., 2007. A tutorial on spectral clustering. *Statistics and Computing*, 17 (2007), 1–32. (cited on pages 22, 23, and 25)
- WANG, H.; KLASER, A.; SCHMID, C.; AND LIU, C.-L., 2011a. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 3169–3176. IEEE. (cited on page 65)
- WANG, S.; YUAN, X.; YAO, T.; YAN, S.; AND SHEN, J., 2011b. Efficient subspace segmentation via quadratic programming. In *AAAI*, vol. 1, 519–524. (cited on page 9)
- WANG, Y.; YIN, W.; AND ZENG, J., 2015. Global convergence of admm in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*, (2015). (cited on page 27)
- WANG, Y.-X.; XU, H.; AND LENG, C., 2013. Provable subspace clustering: When lrr meets ssc. In *Advances in Neural Information Processing Systems*, 64–72. (cited on page 8)
- WEBER, J. AND MALIK, J., 1997. Rigid body segmentation and shape description from dense optical flow under weak perspective. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19, 2 (1997), 139–143. (cited on page 3)
- WEDEL, A.; CREMERS, D.; POCK, T.; AND BISCHOF, H., 2009. Structure- and motion-adaptive regularization for high accuracy optic flow. In *IEEE 12th International Conference on Computer Vision*. (cited on page 91)
- WEDEL, A.; POCK, T.; BRAUN, J.; FRANKE, U.; AND CREMERS, D., 2008. Duality TV-L1 flow with fundamental matrix prior. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*. (cited on pages 91 and 92)
- WEISS, Y., 1999. Segmentation using eigenvectors: a unifying view. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, 975–982. IEEE. (cited on page 53)
- WU, Y.; ZHANG, Z.; HUANG, T.; AND LIN, J., 2001. Multibody grouping via orthogonal subspace decomposition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2, II–252–II–257 vol.2. (cited on pages 6, 49, and 52)
- YAN, J. AND POLLEFEYS, M., 2006. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision – ECCV 2006*, vol. 3954, 94–106. (cited on pages 5, 7, 22, 44, 52, 75, and 91)

-
- YANG, A.; WRIGHT, J.; MA, Y.; AND SASTRY, S., 2008. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, 110, 2 (2008), 212–225. (cited on page 33)
- YILMAZ, A.; JAVED, O.; AND SHAH, M., 2006. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38, 4 (2006), 13. (cited on page 3)
- YIN, M.; GUO, Y.; GAO, J.; HE, Z.; AND XIE, S., 2016. Kernel sparse subspace clustering on symmetric positive definite manifolds. *arXiv preprint arXiv:1601.00414*, (2016). (cited on page 9)
- YOU, C.; LI, C.-G.; ROBINSON, D. P.; AND VIDAL, R., 2016. Oracle based active set algorithm for scalable elastic net subspace clustering. *arXiv preprint arXiv:1605.02633*, (2016). (cited on page 9)
- ZELNIK-MANOR, L. AND IRANI, M., 2003. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, II–287. IEEE. (cited on pages 6, 52, and 75)
- ZENG, Z.; CHAN, T.-H.; JIA, K.; AND XU, D., 2012. Finding correspondence from multiple images via sparse and low-rank decomposition. In *Computer Vision – ECCV 2012*, 325–339. (cited on pages 27, 73, 75, 76, 78, and 85)
- ZHANG, S.; QIAN, H.; AND GONG, X., 2016. An alternating proximal splitting method with global convergence for nonconvex structured sparsity optimization. In *Thirtieth AAAI Conference on Artificial Intelligence*. (cited on page 27)
- ZHANG, T.; SZLAM, A.; WANG, Y.; AND LERMAN, G., 2010. Hybrid linear modeling via local best-fit flats. In *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*. (cited on page 16)
- ZHANG, Y., 2010. An alternating direction algorithm for nonnegative matrix factorization. Technical report, Rice University. (cited on pages 27 and 78)
- ZHANG, Y.; SUN, Z.; HE, R.; AND TAN, T., 2013. Robust subspace clustering via half-quadratic minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, 3096–3103. (cited on page 9)
- ZHU, S. C. AND YUILLE, A., 1996. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18, 9 (1996), 884–900. (cited on page 3)