Imperial College
London

VISION-BASED KINEMATIC STRUCTURE LEARNING OF ARBITRARY ARTICULATED RIGID OBJECTS

URBANO MIGUEL GONÇALVES NUNES

Thesis submitted for the degree of Doctor of Philosophy

Supervised by PROFESSOR YIANNIS DEMIRIS

Personal Robotics Lab
Department of Electrical and Electronic Engineering
Imperial College London

October 2021

COPYRIGHT DECLARATION

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution Non-Commercial No Derivatives license (CC BY-NC-ND).

Under this licence, you may copy and redistribute the material in any medium or format on the condition that; you credit the author, do not use it for commercial purposes and do not distribute modified versions of the work.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

ORIGINALITY DECLARATION

I hereby declare that this thesis and the work herein detailed, was composed and originated by myself, except where appropriately referenced and credited.

London, October 2021

Urbano Miguel Gonçalves Nunes

ABSTRACT

The Kinematic Structure (KS) is a compact and structured representation that fully discloses the motion capabilities of an articulated rigid object. Estimating KSs is thus an active topic in the fields of computer vision and robotics, with applications in robot manipulation tasks, unsupervised motion retargeting, and robot-assisted dressing, to name a few possibilities. While previous approaches are typically offline or computationally demanding, in this thesis, novel KS estimation methods from vision-based data that are suitable for real-time applications will be developed.

The thesis starts by providing empirical evidence that initially representing the object by semi-dense three-dimensional (3D) points is a valid compromise between accuracy and computational processing costs. The problem of motion segmentation of articulated rigid bodies from semi-dense 3D points is then cast as a subspace clustering problem.

Online processing is explored to handle incomplete point trajectories and partial occlusions during KS estimation. A suitable incremental metric representation of the tracked semi-dense 3D points is proposed, based on the observation that the distance between points belonging to the same rigid part is constant. This representation allows mitigating noise and points' tracking errors while implicitly encoding motion information, which is combined with the object's topological distances to build more plausible KSs.

The use of event cameras to estimate KSs is considered for the first time in this thesis. A novel framework for event-based motion estimation is proposed, which can estimate the parameters of several motion models. The framework does not rely on any intermediate image-based representation and can thus handle augmented events from additional sensors. An incremental version of this framework is then used to perform joint shape-motion segmentation for event-based KS estimation without having to track feature points, which represents a paradigm shift on vision-based KS estimation. New challenging sequences for KS estimation are also made available. Experimental results corroborate that event cameras outperform frame-based cameras on motion-related tasks, and specifically on KS estimation.

This thesis advances the state-of-the-art on vision-based KS estimation by proposing new frame-based KS estimation methods and taking the first steps towards considering event cameras for KS estimation. Its contributions are likely to bolster real-time applications that rely on KSs.

ACKNOWLEDGEMENTS

With the development and conclusion of this thesis, a journey of great years has come to an end, with many memorable moments of joy but also lots of sacrifices. It is time to be grateful for the people and institutions that made this an easier and more pleasant journey.

First, I would like to thank Prof. Yiannis Demiris, who has been an incredibly supportive supervisor. I will forever remember his electrifying energy combined with genuine curiosity and passion for research when we first met. It was that same combination that helped me during moments of adversity, and I would be grateful to carry at least part of it with me throughout my career. I have learned so much from you during my PhD, and thank you for the opportunity to be part of the Personal Robotics Lab.

I also thank my examiners, Professor Michael Milford and Professor Andrew Davison, for their insightful and knowledgeable comments and constructive criticism regarding the thesis, which guided me to improve the quality of the manuscript.

To all the members of the Personal Robotics Lab, I am sincerely grateful for all the help, fruitful and insightful discussions and laughs from the social moments we have shared during the past years. I had the luck to share the lab with amazingly bright colleagues, from whom I will carry the fondest memories with me: Vinícius, Mark, Rodrigo, Tobias, Hyung Jin, Fan, Ahmed, Pierluigi, Ruohan, Antoine, Cat, Xingchen, Josh, Haining, Oya, and Vincent. Without you, this journey would not have been as enjoyable as it was.

I would also like to express my sincere gratitude to various other members of the Electrical and Electronic Engineering Department, with whom I have interacted regularly. Thank you, Max, Sahbi and Prof. Yiannis, for allowing me to contribute as a teaching assistant. It was definitely a very challenging and rewarding activity, and I am grateful for the opportunity. Thank you, Joan, for your diligence regarding all the administrative paperwork and for organising the Wednesdays coffee breaks.

This journey would also not be possible without the financial support from the Portuguese Fundação para a Ciência e Tecnologia (FCT), under Doctoral Grant with reference SFRH/BD/130732/2017. Given the global circumstances, I am also grateful for the scholarship extension conceded for almost up to a year.

To my family, whose unconditional support and motivation allowed me to fully embrace this journey. To my friends, from whom I could always count on a good laugh. To Gonçalo, with whom I could express whatever was on my mind. I express my deepest gratitude.

Lastly, to my dear partner, Joana, with whom I have shared beautiful moments, knowing that I would always have a place to call home at the end of each day. I know it was not always easy, but you made sure every step in this journey was and will be worthwhile. I can not wait for our next adventures!

CONTENTS

1 INTRODUCTION 19
- 1.1 Research Questions . . . 22
- 1.2 Contributions . . . 22
- 1.3 Publications . . . 23
- 1.4 Thesis Outline . . . 24

2 BACKGROUND 27
- 2.1 Standard Frame-based Cameras vs. Event Cameras . . . 27
- 2.2 Frame-based Motion Estimation and Segmentation . . . 29
  - 2.2.1 From 2D Data . . . 30
  - 2.2.2 From 3D Data . . . 31
- 2.3 Event-based Motion Estimation and Segmentation . . . 32
  - 2.3.1 Event-based Motion Estimation . . . 32
  - 2.3.2 Event-based Motion Segmentation . . . 34
  - 2.3.3 Limitations of Previous Works . . . 35
- 2.4 Frame-based Kinematic Structure Estimation . . . 35
  - 2.4.1 From 2D Data . . . 36
  - 2.4.2 From 3D Data . . . 37
- 2.5 Applications for Kinematic Structures . . . 37
- 2.6 Conclusion . . . 38

3 MOTION SEGMENTATION OF ARTICULATED RIGID OBJECTS FROM 3D POINTS 43
- 3.1 Problem Formulation . . . 44
- 3.2 Motion Segmentation based on Sparse Subspace Clustering . . . 45
- 3.3 Self-Tuning Spectral Clustering . . . 46
- 3.4 Influence of the Number of Frames . . . 47
- 3.5 Experimental Evaluation . . . 49
  - 3.5.1 Implementation Details . . . 49
  - 3.5.2 Quantitative Results . . . 50
  - 3.5.3 Qualitative Results . . . 52
  - 3.5.4 Influence of the Number of Frames . . . 53
- 3.6 Conclusion . . . 54

4 ONLINE KINEMATIC STRUCTURE ESTIMATION OF ARTICULATED RIGID OBJECTS FROM 3D POINTS 55

4.1 Problem Formulation . . . 57
4.2 Building the Similarity . . . 57
4.3 Clustering by Label Spreading . . . 59
4.3.1 Label Spreading . . . 59
4.3.2 Obtaining New Labels . . . 60
4.4 Building the Kinematic Structure . . . 60
4.5 Handling Lost Points . . . 61
4.5.1 Octree Representation of a 3D Point Cloud . . . 61
4.5.2 Maintaining an Even Distribution of Points Tracked . . . 61
4.5.3 Adding New Points . . . 62
4.6 Mitigating Noise . . . 62
4.7 Experimental Evaluation . . . 64
4.7.1 Implementation Details . . . 64
4.7.2 Quantitative Results . . . 64
4.7.3 Qualitative Results . . . 70
4.8 Conclusion . . . 72

5 EVENT-BASED VISION MOTION ESTIMATION BY DISPERSION MINIMISATION 73
5.1 Dispersion Minimisation Framework . . . 74
5.1.1 Pairwise Potential . . . 76
5.1.2 Entropy-based Measures . . . 77
5.1.3 Optimisation . . . 78
5.2 Efficient Approximation . . . 79
5.3 Incremental Extension . . . 81
5.4 Robust Estimation by Data Whitening . . . 82
5.5 Experimental Evaluation . . . 84
5.5.1 Datasets Overview . . . 84
5.5.2 Implementation Details . . . 85
5.5.3 2D Translation Estimation . . . 86
5.5.4 Rotational Motion Estimation . . . 87
5.5.5 Motion Estimation in Planar Scenes . . . 88
5.5.6 6-DOF Transformation Estimation . . . 90
5.5.7 Optical Flow Estimation . . . 92
5.5.8 Computational Processing Time . . . 95
5.6 Conclusion . . . 95

6 EVENT-BASED VISION INCREMENTAL KINEMATIC STRUCTURE ESTIMATION 97
6.1 Overview . . . 98

6.2 Incremental Joint Shape-Motion Segmentation . . . 99
6.2.1 Joint Optimisation Procedure . . . 99
6.2.2 Creating and Deleting Segments . . . 101
6.2.3 E-step . . . 101
6.2.4 M-step . . . 101
6.2.5 Incremental Segmentation . . . 102
6.2.6 Optimisation Prior Parameters . . . 102
6.3 Body Parts Affinity Estimation . . . 103
6.4 Kinematic Structure Estimation . . . 104
6.5 Experimental Evaluation . . . 106
6.5.1 Datasets Overview . . . 106
6.5.2 Implementation Details . . . 107
6.5.3 Quantitative Performance . . . 108
6.5.4 Qualitative Performance . . . 110
6.6 Conclusion . . . 111

7 CONCLUSIONS 113
7.1 Overview and Contributions of the Thesis . . . 113
7.2 Limitations . . . 114
7.2.1 Temporal Inconsistencies in Kinematic Structure Estimation . . . 115
7.2.2 Topological Inconsistencies in Kinematic Structure Estimation . . . 115
7.2.3 Distance Between Connected Links May Not Be Constant Over Time . . . 116
7.2.4 Event-based Kinematic Structure Dataset . . . 116
7.3 Future Directions . . . 116
7.3.1 Fusing Frames and Events for Motion Estimation . . . 117
7.3.2 Hybrid Estimation Approach for Kinematic Structure Generation . . . 117
7.3.3 SLAM and Simultaneous Kinematic Structure Estimation of Multiple Objects . . . 118
7.4 Final Remarks . . . 118

A COMPONENTS, SENSORS AND SOFTWARE PACKAGES 123
A.1 Robot Operating System . . . 123
A.2 Cameras . . . 123
A.3 Open-Source Contributions . . . 124

B MATHEMATICAL UTILITIES 127

B.1 Probability Distributions . . . 127
B.1.1 Gaussian Distribution . . . 127
B.1.2 Beta Distribution . . . 127
B.1.3 Dirichlet Distribution . . . 128
B.1.4 Inverse-Wishart Distribution . . . 128
B.1.5 Normal-Inverse-Wishart Prior . . . 129
B.2 Dirichlet Process . . . 129
B.2.1 Stick Breaking Construction . . . 129
B.2.2 The Chinese Restaurant Process . . . 130
B.2.3 Dirichlet Process Mixture Model . . . 130

C DISPERSION MINIMISATION FRAMEWORK: APPLICATIONS AND MODELS 133
C.1 2D Translational Motion Estimation . . . 133
C.2 Rotational Motion Estimation . . . 134
C.3 Isometry Transformation Estimation . . . 135
C.4 Similarity Transformation Estimation . . . 135
C.5 Affine Transformation Estimation . . . 136
C.6 Motion Estimation in Planar Scenes . . . 137
C.6.1 Why There is No Trivial Procedure to Linearise the Parameters of Motion Estimation in Planar Scenes . . . 137
C.7 6-DOF Estimation . . . 139

D DISPERSION MINIMISATION FRAMEWORK: ADDITIONAL RESULTS 141
D.1 Implementation Details . . . 141
D.2 Optimisation Procedure . . . 141
D.3 Additional Results on 2D Translational Motion Estimation . . . 143
D.4 Additional Results on Rotational Estimation . . . 143
D.5 Additional Results on 6-DOF Estimation . . . 145
D.6 Dispersion Minimisation Optimisation Profiles . . . 148
D.7 Truncated Kernel Size . . . 148
D.8 Real-Time Performance . . . 149
D.8.1 Implementation Description . . . 149
D.8.2 Experimental Evaluation . . . 151

E EVENT-BASED VISION INCREMENTAL KINEMATIC STRUCTURE ESTIMATION: EKS DATASET 153
E.1 Camera System . . . 153
E.2 Dataset Sequences . . . 153

F EVENT-BASED VISION INCREMENTAL KINEMATIC STRUCTURE ESTIMATION: ADDITIONAL RESULTS 155
F.1 Per-Event Evaluation 155
F.2 Considering Only Shape Information 155
F.3 Qualitative Results on EED 156

BIBLIOGRAPHY 159

LIST OF FIGURES

Figure 1.1 KS examples . . . 20
Figure 1.2 Vision-based KS estimation pipeline . . . 21
Figure 1.3 Thesis outline . . . 26
Figure 2.1 Frames vs. Events . . . 29
Figure 3.1 3D offline motion segmentation overview . . . 44
Figure 3.2 3D motion segmentation quantitative results . . . 51
Figure 3.3 3D motion segmentation qualitative results . . . 53
Figure 4.1 Online 3D KS learning example . . . 55
Figure 4.2 Online 3D KS learning overview . . . 56
Figure 4.3 Effect of incorporating the spectrum transformation . . . 63
Figure 4.4 Precision, recall and f-measure metrics comparison . . . 65
Figure 4.5 Estimation of the number of motion segments . . . 66
Figure 4.6 Comparison of the number of motion segments estimation . . . 66
Figure 4.7 Cumulative computational time per frame . . . 67
Figure 4.8 Influence of the number of points sub-sampled . . . 68
Figure 4.9 Influence of the μ parameter . . . 68
Figure 4.10 Influence of the number of leading eigenvalues . . . 69
Figure 4.11 Influence of incorporating the spectrum transformation 70
Figure 4.12 Consistency analysis in sequential estimation . . . 70
Figure 4.13 Qualitative results . . . 71
Figure 5.1 DMin framework overview . . . 74
Figure 5.2 DMin framework outcome . . . 75
Figure 5.3 DMin: exact vs approximate . . . 79
Figure 5.4 Data whitening benefit . . . 83
Figure 5.5 2D translational motion optimisation profiles . . . 86
Figure 5.6 Rotational motion estimation . . . 87
Figure 5.7 Motion estimation in planar scenes . . . 89
Figure 5.8 6-DOF estimation . . . 90
Figure 5.9 Trajectory estimation from 6-DOF estimates . . . 93
Figure 5.10 Optical flow estimation . . . 93

Figure 5.11 Runtime vs. accuracy comparison. A real-time ratio of 0 means real-time processing. The preceding W. indicates that the data whitening step described in Section 5.4 was applied. . . . 95
Figure 6.1 Event-based KS estimation pipeline . . . 97
Figure 6.2 Event-based KS estimation overview . . . 98
Figure 6.3 Object's topological shortest path . . . 103
Figure 6.4 Mode seeking based on mean-shift . . . 104
Figure 6.5 Object's biharmonic distance . . . 105
Figure 6.6 Qualitative results . . . 112
Figure A.1 RGB-D cameras . . . 124
Figure A.2 Event camera . . . 124
Figure D.1 2D translational motion estimation . . . 144
Figure D.2 6-DOF estimation . . . 146
Figure D.3 DMin optimisation profiles . . . 147
Figure D.4 Real-time implementation strategies . . . 150
Figure D.5 Real-time DMin incremental mode . . . 152
Figure E.1 EKS dataset overview . . . 153
Figure F.1 Qualitative results on EED . . . 157

LIST OF TABLES

Table 3.1 3D motion segmentation average quantitative results . 52
Table 3.2 Influence of number of frames . . . 53
Table 5.1 Rotational motion estimation accuracy . . . 88
Table 5.2 Motion estimation in planar scenes accuracy . . . 89
Table 5.3 6-DOF estimation accuracy . . . 91
Table 5.4 6-DOF estimation accuracy comparison . . . 92
Table 5.5 Optical flow estimation accuracy on DVSMOTION20 . 94
Table 6.1 $\mathbf{C}_0$ parameter . . . 108
Table 6.2 Quantitative results - precision, recall and f-measure . 109
Table 6.3 Quantitative results - number of segments and computation time . . . 110
Table 6.4 Quantitative results on EKS . . . 110
Table A.1 Specifications of the cameras . . . 125
Table D.1 Rotational motion estimation accuracy . . . 145
Table D.2 Truncated kernel size accuracy influence . . . 148
Table D.3 Real-time DMin incremental mode accuracy . . . 151

Table F.1 Per-event quantitative evaluation 155
Table F.2 Quantitative results considering only shape 156

LISTINGS

Listing 5.1 DMin framework 78
Listing 5.2 Online DMin framework 82

ACRONYMS

2D	Two-dimensional
3D	Three-dimensional
AAE	Average Angular Error
ADMin	Approximate Dispersion Minimisation
ADMM	Alternating Direction Method of Multipliers
AEE	Average Endpoint Error
ARPE	Average Relative Pose Error
ARRE	Average Relative Rotation Error
AVPED	Accumulated Variation of Pairwise Euclidean Distances
CMax	Contrast Maximisation
DMin	Dispersion Minimisation
DOF	Degrees of Freedom
EM	Expectation-Maximisation
fps	Frames Per Second
HRI	Human-Robot Interaction
IMU	Inertial Measurement Unit
IWE	Image of Warped Events
KS	Kinematic Structure
LRR	Low Rank Representation
MAP	Maximum A Posteriori
pdf	Probability Density Function
RAEE	Relative Average Endpoint Error
RGB-D	Red Green Blue-Depth

RMS	Root Mean Square
ROS	Robot Operating System
SSC	Sparse Subspace Clustering
SVD	Singular Value Decomposition
STSC	Self-Tuning Spectral Clustering

1

INTRODUCTION

Human-Robot Interaction (HRI) is a very active broad field of research that aims at endowing robots with the required capabilities to establish natural and realistic behaviours with humans and the environment. Although intuitive for humans, robots still struggle to interact in most simplistic scenarios where there is no plain rule-based guideline to follow.

Endowing robots with perception skills to appropriately understand the surrounding environment is an important step towards more fluid HRIs. On a high level, this problem is challenging mainly because of the huge amount of possibilities that may occur. Thus, it is crucial to develop methods that can extract meaningful and compact information from the robot's raw input space. This information can then serve as the input for other HRI stages downstream, e. g. planning and acting on the environment.

In this thesis, we focus on the problem of learning the Kinematic Structure (KS) of arbitrary articulated rigid objects from visual input data. The KS of an articulated rigid body provides a compact and meaningful representation that contains information regarding the entity's motion capabilities. By assuming that each part of the articulated object is rigid, it can thus be considered a mid-level representation of generic objects, encoding the relationship between the entity's rigid parts connected by kinematic joints. KS estimation can also be viewed as efficient and lossless compressors of information while only imposing the articulated rigidity assumption. Figure 1.1 illustrates some examples of the KS of articulated rigid objects.

KSs are thus useful representations of articulated rigid objects, which are used in robotics and computer vision applications. One example of such applications is the case where a robot is continuously learning by interacting with a human and the environment (Bohg et al., 2017; Moulin-Frier et al., 2018). In this example, the environment typically consists of multiple objects within the robot's field of view whose properties and categories are unknown. The robot must adaptively model these objects by observing them, by interacting with them, e. g. moving and poking, and by extracting useful representations, such as KSs. Other sensory modalities (e. g. touch) could also be merged to enrich the estimated KS representation. This scenario is challenging because not only does the robot need to estimate the KS of each

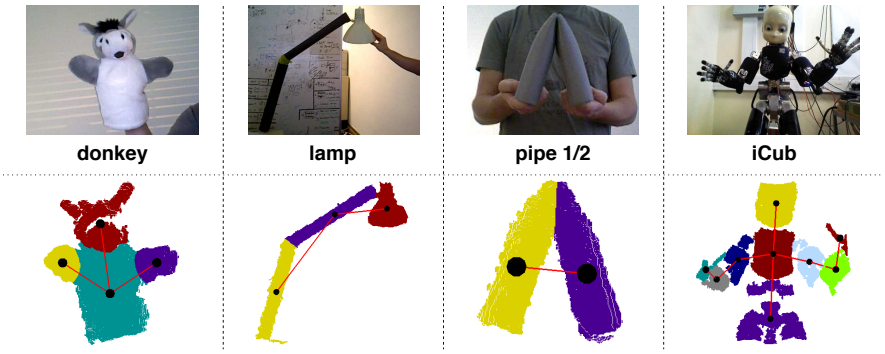


Figure 1.1: Examples of the KS of articulated rigid objects. The KS of an articulated rigid object can be represented as a set of distinct rigid body parts that are topologically linked. Each body part is represented by a distinct colour. Due to the rigidity assumption, each body part can be summarised by one point (e. g. centre of mass) since all the points belonging to the same body part contain the same kinematic information. This means that an articulated rigid object can be represented in an efficient and compact form without losing information about its kinematic capabilities.

unknown object, but it should also continuously do it in real-time. KS estimation may also be useful for other applications, such as robot manipulation tasks (Sturm et al., 2011), finding kinematic correspondences between objects or unsupervised motion retargeting, both in animation and robotics (Chang et al., 2018), and personalised robot-assisted tasks, such as personalised dressing (Gao et al., 2016; Zhang et al., 2019).

Supervised model-based KS learning methods have shown to be superior in domains where the model of the object is known, e. g. human bodies (Cao et al., 2017). However, when this is not the case, unsupervised methods are commonly used, and the problem of KS estimation is divided into two well-known problems, namely motion segmentation of rigid bodies (Elhamifar and Vidal, 2013; Elqursh and Elgammal, 2013; Judd et al., 2018) and KS generation based on motion and/or other cues (Chang and Demiris, 2018; Yan and Pollefeys, 2008). Throughout the thesis, we will adopt the unsupervised approach since we intend to estimate the KS of arbitrary articulated rigid objects.

Most unsupervised KS learning methods perform motion segmentation by tracking features over complete sequences of images and then applying a suitable clustering algorithm (Chang and Demiris, 2018; Yan and Pollefeys, 2008). These methods rely on complete point trajectories, which hampers its use in practice since as the sequence progresses, some points may be lost due to occlusion or measurement errors. Other methods try to overcome this issue by sequentially matching the object's surface (Lu et al., 2018; Zhang et al., 2013). Not only is this procedure computationally demanding, but it

also relies on an initial model of the object, which is sensitive to initialization and viewpoint.

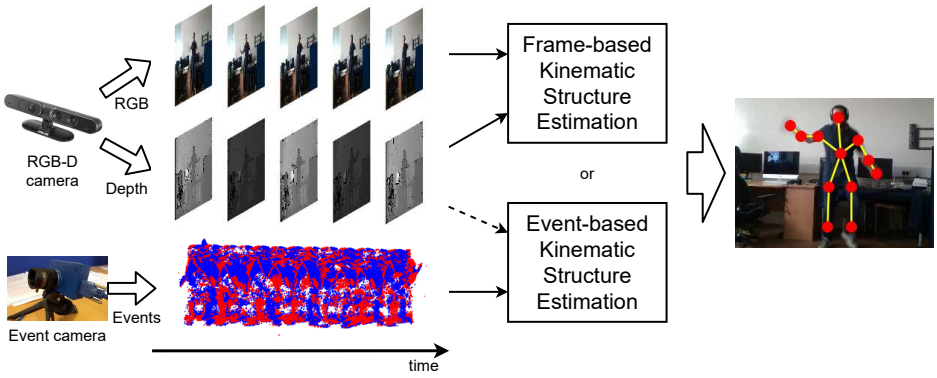


Figure 1.2: Vision-based KS estimation pipeline. By observing an object moving, we can estimate its KS. Three sources of data are considered, namely RGB and depth maps from frame-based Red Green Blue-Depth (RGB-D) cameras and events from event cameras. The frame-based methods developed in this thesis require three-dimensional (3D) data, which can be acquired by RGB-D cameras. The event-based methods process events, which can also be augmented with additional information, e. g. depth from frame-based depth maps.

This thesis contributes with new methods for KS estimation for online and real-time HRI applications by naturally handling lost points and not depending on prior models. Figure 1.2 illustrates the pipeline to estimate the KS of an articulated rigid object. We start by exploring frame-based RGB-D sensors to address the problem of KS estimation from semi-dense 3D points. Event cameras are also studied to address some of the encountered limitations of frame-based methods, which are largely originated from the working principle of standard frame-based cameras, such as difficulty in acquiring rapid or sudden movements due to low sampling rates. We thus argue that the use of event cameras for KS learning is a promising line of research, which, to the best of our knowledge, has not been previously considered. Event cameras react to brightness changes in the field-of-view of the camera by asynchronously reporting *events*. Such changes can occur due to motion, placing event cameras as natural candidates to perceive and estimate motion. Allied to their outstanding properties (e. g. very high dynamic ranges, high bandwidth in the order of millions of events per second, low latency), these cameras provide clear advantages over standard frame-based cameras to tackle several motion-related problems in computer vision (Gallego et al., 2018b; Stoffregen et al., 2019; Zhu et al., 2019). Section 2.1 provides further comparisons between the working principle of standard frame-based cameras and event cameras.

1.1 RESEARCH QUESTIONS

To summarise, the main thesis research questions are the following:

1. Can the KS estimation be improved by considering semi-dense 3D points instead of explicit two-dimensional (2D) feature points, and how?
2. How can a suitable representation that is robust to lost feature points be incrementally maintained for online KS estimation?
3. Are event cameras suitable sensors for KS estimation?
   a) How can event cameras estimate general motion, and can this be achieved incrementally (i. e. processing one event at a time, instead of a batch of events)?
   b) Can event cameras improve the KS estimation over standard frame-based cameras, and how?

1.2 CONTRIBUTIONS

The following is a summary of the contributions, new methods and approaches presented in this thesis:

- **Frame-based motion segmentation extension for KS estimation from (semi-dense) 3D points.** The 3D motion segmentation problem is cast into a Sparse Subspace Clustering (SSC) problem. The proposed method can automatically estimate the underlying number of segments, has few parameters to adjust, requires no post-processing and scales well with the number of frames of the observed sequence.
- **Frame-based online and unsupervised KS learning from (semi-dense) 3D points.** This is the first approach that directly tackles the problem of online and real-time 3D KS learning of arbitrary articulated rigid objects. Lost points and occurring noise are naturally handled, and thus the method does not depend on complete point trajectories, being suitable for incremental learning. A more plausible KS estimate is achieved, by considering the object's motion and topological properties.
- **Framework for arbitrary motion estimation for event cameras.** It can handle model outputs of arbitrary dimensions since no projection during optimisation is required. Thus, it can be seen as a generalisation

of previous event-based motion-compensation methods. A family of entropy-based measures that are suitable for batch optimisation are presented, as well as an efficient approximation technique. Based on the batch-based procedure, a first extension to incrementally (i. e. processing one event at a time, instead of a batch of events) estimate arbitrary motions and how to make it run in real-time are also described. Data whitening is proposed as a pre-processing step to handle current limitations of general event-based motion compensation methods in terms of estimation accuracy.

- **Event-based online and unsupervised KS learning.** The first KS estimation method for event cameras is presented. It relies on the previous event-based incremental motion estimation method for joint shape and motion segmentation. This representation is exploited to estimate the body parts' connectivity to generate the KS.

1.3 PUBLICATIONS

This thesis' contributions and results have been published:

Nunes, U. M. and Demiris, Y. (2018), 3D Motion Segmentation of Articulated Rigid Bodies based on RGB-D Data, *in* 'British Machine Vision Conference (BMVC)'. (Nunes and Demiris, 2018)

Nunes, U. M. and Demiris, Y. (2019), Online Unsupervised Learning of the 3D Kinematic Structure of Arbitrary Rigid Bodies, *in* 'IEEE/CVF International Conference on Computer Vision (ICCV)', pp. 3809–3817. doi: 10.1109/ICCV.2019.00391. (Nunes and Demiris, 2019)

Nunes, U. M. and Demiris, Y. (2020), Entropy Minimisation Framework for Event-Based Vision Model Estimation, *in* 'European Conference on Computer Vision (ECCV)', pp. 161-176. doi: 10.1007/978-3-030-58558-7_10. (**Spotlight**) (Nunes and Demiris, 2020)

Nunes, U. M. and Demiris, Y. (2021), Robust Event-based Vision Model Estimation by Dispersion Minimisation, *in* 'IEEE Transactions on Pattern Analysis and Machine Intelligence', doi: 10.1109/TPAMI.2021.3130049. (Nunes and Demiris, 2021b)

Nunes, U. M. and Demiris, Y. (2021), Live Demonstration: Incremental Motion Estimation for Event-based Cameras by Dispersion Minimisation,

in 'IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops', pp. 1322–1323. (Nunes and Demiris, 2021a)

Nunes, U. M. and Demiris, Y. (2021), Kinematic Structure Estimation of Arbitrary Articulated Rigid Objects for Event-based Cameras. (Under review)

1.4 THESIS OUTLINE

As presented in Figure 1.3, the rest of the thesis is comprised of six chapters and six appendices, and is organised as follows:

- Chapter 2 provides background material for the problems addressed in this thesis, whereby previous works on KS estimation, vision-based motion estimation and segmentation are reviewed. The working principle between standard frame-based cameras and event cameras are also compared.
- Chapter 3 presents a method for frame-based motion segmentation of (semi-dense) 3D points, with applications to KS estimation.
- Chapter 4 describes an online extension for KS estimation from (semi-dense) 3D points, which handles lost points and partial occlusions.
- Chapter 5 addresses the problem of event-based motion estimation, with applications to event-based KS estimation. It proposes a framework for general motion estimation from event-based data (that can also be augmented with additional sensors, e. g. RGB-D).
- Chapter 6 describes the first event-based KS estimation method by performing motion segmentation based on the incremental mode of the framework proposed in Chapter 5.
- Chapter 7 summarises the key points of the thesis and highlights possible future avenues of research by discussing the current limitations of the proposed work.
- Appendix A describes the components and sensors used throughout this work, as well as the open-source software packages developed.
- Appendix B contains mathematical utilities that grounded the explanations provided in the thesis.
- Appendix C presents a non-exhaustive list of models and applications that can be estimated using the Dispersion Minimisation (DMin) framework proposed in Chapter 5.

- Appendix D provides additional experimental results for the DMin framework proposed in Chapter 5.
- Appendix E describes the first KS dataset for event cameras, which has been used to experimentally evaluate the event-based KS estimation method proposed in Chapter 6.
- Appendix F provides additional experimental results for the event-based KS estimation method proposed in Chapter 6.

The topic of vision-based KSs estimation of articulated rigid objects was motivated and introduced in this chapter. The main research questions addressed were outlined, and the contributions made in this work were summarised. In the next chapter, we will compare the working principles of standard frame-based cameras and event cameras and will review in detail related works.

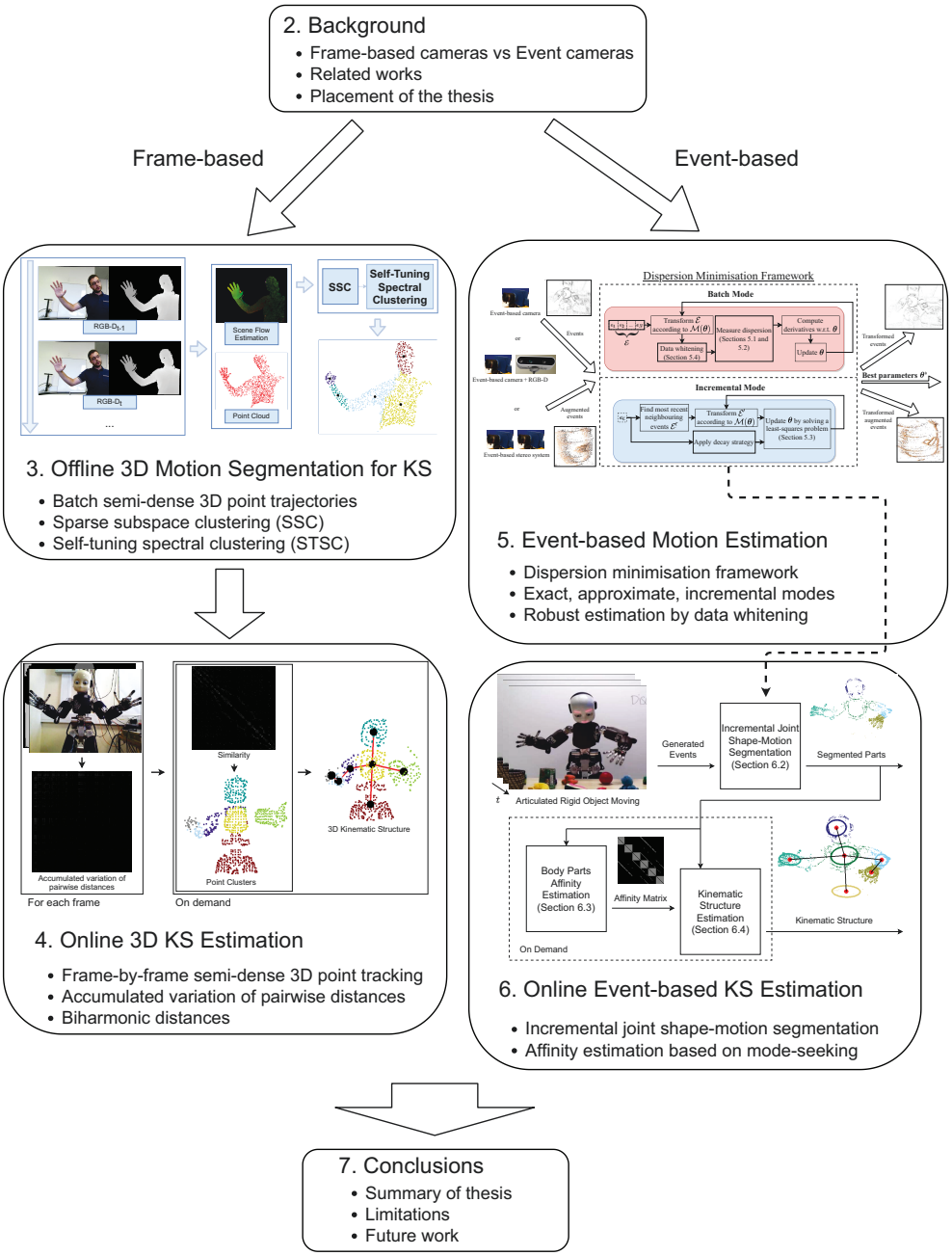


Figure 1.3: Thesis outline. Figures are shown in full size at the respective chapters.

2

BACKGROUND

Several previous works inspired this thesis and guided how to improve previous vision-based Kinematic Structure (KS) estimation approaches. In this chapter, we review such works, including vision-based motion estimation and segmentation, which is a core component of KS learning since it provides important cues to distinguish different body parts by assuming each part can exhibit distinct motion patterns.

We start by comparing the properties of standard frame-based cameras and event cameras in Section 2.1. We discuss their corresponding working principles and the advantages and limitations of each type of camera for vision-based motion perception tasks. We proceed by reviewing previous work on vision-based motion estimation and segmentation, based on standard frames in Section 2.2 and based on event data in Section 2.3. In Section 2.4, we review previous vision-based KS estimation methods from 2D and 3D visual data. The utility of KS representations is demonstrated in Section 2.5, where we present state-of-the-art applications that use KS representations to address other problems. To the best of our knowledge, at the time of writing the thesis, no event-based KS estimation methods have been previously proposed. Finally, Section 2.6 concludes this chapter by highlighting encountered limitations in the scientific literature and discussing the corresponding opportunities for further research that will guide the rest of the work proposed.

2.1 STANDARD FRAME-BASED CAMERAS VS. EVENT CAMERAS

Standard frame-based cameras output a sequence of image frames. For example, an RGB frame consists of a grid-like structure whose elements, usually referred by pixels, report the light intensity captured. The frames are captured at fixed time intervals (typically between 30 and 120 frames per second (fps) for commercially available cameras) via global or rolling shutter. Current frame-based cameras can reliably capture a scene with remarkable accuracy and detail, and thus providing rich sensory information for computer vision tasks and robot perception, to name a few possible broad applications. However, for vision-based motion perception tasks, these

cameras impose three major limitations regarding visual data acquisition, namely: 1) capture of redundant information, 2) blind time intervals between frames, and 3) motion blur effects due to high-speed motion[1]. All of these limitations result from the synchronous working principle of standard frame-based cameras and their limited frame-rate acquisition. As we will see in the following sections, these limitations can lead to sub-optimal motion-related estimations.

Event cameras operate on a distinct paradigm, whereby only pixel-wise (log) intensity changes are reported, commonly denoted by *events*. The output of event cameras is thus a stream of asynchronous events, which are defined by the pixel location, the microsecond-precise timestamp and the polarity of the occurring (log) intensity change. Each event is mathematically expressed by

$$e = (\mathbf{x}, t, p), \tag{2.1}$$

where $\mathbf{x} = (x, y)^\top$ represents the pixel coordinates, t is the timestamp it occurred and $p \in \{-1, +1\}$ is the polarity, i. e. negative or positive brightness change. This working principle has multiple advantages over standard cameras, namely: 1) lower bandwidth requirements to transmit, process and store data, 2) no redundant data is transmitted while preserving all the original content (at least up to scale), 3) no constraints related to the frame-rate acquisition, and 4) higher dynamic range. In particular, these characteristics are very appealing for vision-based motion perception since objects' movements typically cause intensity changes. Thus, conceptually there is an almost perfect match between triggered events and observed motion. As we will review in Section 2.3, event-based motion estimation and segmentation methods usually exploit this. In this work, we will also take advantage of this remark and further apply it for KS estimation of articulated rigid objects by assuming that the events generated correspond to body parts moving.

Figure 2.1 illustrates the distinct working principle of standard frame-based cameras and event cameras. Please refer to Appendix A.2 for detailed information regarding the cameras used throughout this work and respective specifications.

1 In reality, the motion blur effects depend on the frame-rate of the camera: the motion blur effects are mitigated for higher frame-rate acquisition. However, in the context of this work, we will consider that a standard frame-based camera has a frame-rate between 30 and 120 fps, as mentioned previously.

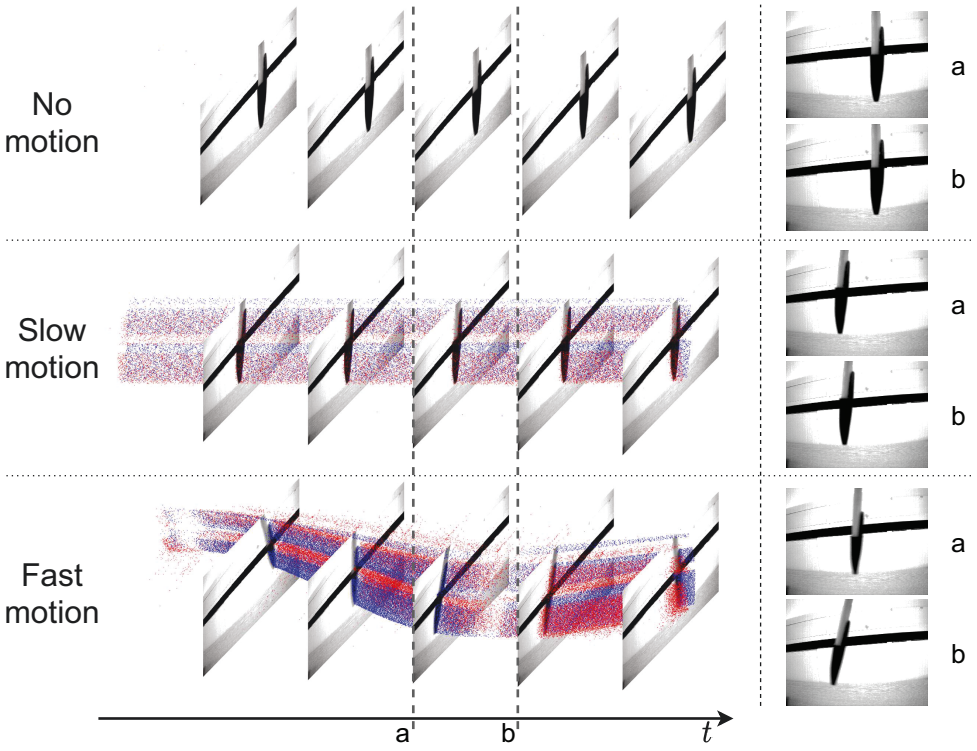


Figure 2.1: Standard frame-based camera vs. event camera. For comparison, the captured frames and the events triggered (which are represented as red and blue dots, depending on the respective polarity) are plotted on the same space-time axis (i. e. 2D space plus time) for each scenario. Three scenarios are illustrated, namely a pen still (top row), a pen slowly swinging (middle row), and a pen rapidly swinging (bottom row). The frames are captured at a fixed rate, regardless of the scene dynamics: when the pen is not moving, the same information is captured, leading to redundant data being transmitted, and when the pen is rapidly swinging, the quality of the frames deteriorates due to motion blur. In contrast, events are only generated when the camera detects intensity changes: when the pen is still, no events are triggered, and when the pen is swinging, the event rate adapts to the scene dynamics, whereby a "smooth" space-time event manifold is generated.

2.2 FRAME-BASED MOTION ESTIMATION AND SEGMENTATION

Motion estimation and segmentation have been very active topics of research in the computer vision community. Overall, since standard cameras capture the scene as snapshots at a constant rate, frame-based motion estimation is typically achieved by comparing consecutive frames. Sparse feature points may also be detected and tracked for later processing to reduce computational costs. We note that the literature on this field is extensive, and it is not our goal to exhaustively present all previous related works. In this section, we

will instead briefly review key common ideas previously explored which will then motivate our work.

2.2.1 *From 2D Data*

Several motion estimation and segmentation approaches have used factorisation methods (Boult and Brown, 1991; Costeira and Kanade, 1998; Tomasi and Kanade, 1992). These methods start by grouping feature points trajectories into a data matrix. Since points belonging to a single body lie in a three (or less) dimensional linear manifold, segmentation can then be performed by thresholding the entries of a similarity matrix built from a canonical form, by applying Singular Value Decomposition (SVD) to the data matrix. The rank of the data matrix is correlated with the number of distinct motions, but due to noise and errors introduced when tracking feature points, it is not trivial to estimate it. Thus, factorisation methods have difficulty in coping with noisy data. Additionally, they typically rely on complete points trajectories, which may be hard to enforce in real-world scenarios (e. g. due to occlusion, tracker limitations, bodies entering/exiting the field of view). Rao et al. (2008) proposed to address this issue by leveraging principles of data compression and sparse representation. However, their approach relies on solving an NP-hard problem, which in practice can be computationally intractable without sub-optimal approximations.

Spectral clustering algorithms have also been introduced to address motion segmentation problems (Keuper et al., 2015; Ochs and Brox, 2012). The aim of spectral clustering (Ng et al., 2002; Shi and Malik, 2000; von Luxburg, 2007) is to separate the connected components (i. e. clusters) in a similarity graph, by computing its graph Laplacian and then performing k-means clustering on the k leading eigenvectors, where k corresponds to the number of clusters. Label spreading methods (Zhou et al., 2004) can also segment a similarity graph by applying a message-passing procedure that iteratively clusters each data point according to initially known labelled and unlabelled points. These methods implicitly enforce smooth constraints on neighbouring points. Typically they are semi-supervised, although there are strategies to make them completely unsupervised. For example, Elhamifar and Vidal (2013) proposed an online motion segmentation based on label spreading that initially clustered all points into one cluster. For each incoming frame, clusters could be separated by performing normalised cuts (Shi and Malik, 2000), where the intra-cluster variability increased due to distinct motion patterns and thus increasing the number of clusters by one. However, in

general, it is not trivial to build the similarity graph or choose the number of clusters k (Zelnik-manor and Perona, 2004).

Subspace fitting methods, which build a similarity graph based on the feature points trajectories, have also been proposed to address motion segmentation problems. Vidal et al. (2005) presented an algebraic-geometric approach based on generalised principal component analysis. Although the method can handle subspaces of different dimensions, it is sensitive to noise and outliers. Additionally, its complexity increases exponentially in terms of the number and dimensions of the underlying subspaces. Liu et al. (2013) posed the problem of motion segmentation as finding a Low Rank Representation (LRR) of the data based on a dictionary of the data itself. Elhamifar and Vidal (2013) proposed a similar approach, whereby a sparse representation is sought instead. Both approaches rely on optimising a function whose global solution is used to build a similarity graph from which motion segmentation can then be performed. Although these approaches can handle noisy data and outliers, they require points trajectories as data points and thus are only suitable for offline processing.

Closely related to motion segmentation is also object segmentation and tracking (Ochs and Brox, 2011). Several methods have been proposed, which typically rely on previous motion segmentation methods to perform object segmentation (Keuper et al., 2015; Ochs et al., 2014; Rünz and Agapito, 2017). However, the main focus of these methods is geared towards segmenting completely disjoint motion patterns, which is not as general as previous motion segmentation methods.

2.2.2 *From 3D Data*

With the introduction of commercially available depth sensors, RGB-D data acquisition became feasible in real-time, and some motion estimation and segmentation approaches have been proposed, taking the provided additional depth information into account. Perera and Barnes (2013) proposed an approach to motion segmentation that does not explicitly estimate the underlying motion. The main observation is that the Euclidean distance between points that share the same motion does not vary. They build a similarity graph based on the standard deviation of the Euclidean distance between points where a thresholded binary decision is made accordingly (i. e. if the standard deviation is below a given threshold, then the corresponding points share the same motion). However, (dis)similarity measures based on the standard deviation are known to be sensitive to outliers and are thus not

suitable for long-term scenarios. Zografos et al. (2015) proposed a method that resorted to group theoretical invariants, where the canonical representation of points belonging to the same motion may be recovered by a unique alignment, corresponding to the associated rigid transformation. Although the method provides fast and accurate estimates, they require the number of motions to be given. Also, intending to estimate rigid transformations to perform motion segmentation, Judd et al. (2018) apply a multi-motion fitting technique, where the point trajectories and motion models are jointly estimated, based on observations and prior estimated motion models. The number of motions is estimated through an iterative procedure whereby motion candidates are proposed and merged.

2.3 EVENT-BASED MOTION ESTIMATION AND SEGMENTATION

As depicted in Figure 2.1, event cameras are suitable to tackle motion-related estimation problems by assuming that the events triggered are caused by motion, which is reasonable under a wide range of scenarios. This assumption has been the driving force for most of the state-of-the-art event-based motion estimation and segmentation methods. In this section, we will thus review how previous works have approached the problem of motion estimation and segmentation from event-based data.

2.3.1 *Event-based Motion Estimation*

Early event-based motion estimation research started by focusing on estimating the optical flow (Benosman et al., 2012). Benosman et al. (2014) proposed to estimate the optical flow by fitting a plane over a spatio-temporal neighbourhood of each event and computing the flow as the normal to the plane. Since the flow is estimated locally, the direction of the estimated flow is always orthogonal to the edge. This is known as the aperture problem in the computer vision field. Akolkar et al. (2020) addressed this problem by finding a spatial scale that maximises the average magnitude of the local flows and orienting the local flows according to their average direction. Almatrafi et al. (2020) proposed an event-based representation, based on spatial distances between events, that was suitable to estimate optical flow while overcoming the aperture problem.

Other methods estimate optical flow while also performing other tasks. Bardow et al. (2016) proposed to jointly estimate the optical flow and scene intensity using an event camera. The method performs dense optical flow

estimation by minimising a cost function that accounts for the asynchronous nature of the event data while imposing spatio-temporal constraints within a sliding window time interval. Stoffregen and Kleeman (2017) presented an approach to computing optical flow by estimating the underlying principle axis of the stream of events while simultaneously performing motion segmentation based on the velocities estimated. Zhu et al. (2017) proposed an event-based feature tracking method that also estimates the optical flow by applying an Expectation-Maximisation (EM) algorithm. In the E-step, the events' associations were updated given the estimated flow, and, in the M-step, the flow was updated according to the events' associations.

Event cameras have also been used to address ego-motion estimation. Kim et al. (2016a) proposed a method that jointly estimates the camera 6-Degrees of Freedom (DOF), depth relative to a keyframe and scene intensity gradient. Rebecq et al. (2017) proposed a geometric approach that jointly estimates the 6-DOF of the camera and a semi-dense depth map (Rebecq et al., 2018a). Gallego et al. (2018a) proposed a method for 6-DOF tracking from photometric depth maps (i. e. scene intensity plus depth). Conceptually, these methods rely on a global image-based registration strategy, whereby an image-like representation, which is obtained by accumulating the events spanning a small time window, is aligned to some keyframe image according to candidate 6-DOF parameters. A similar strategy has been used to train deep learning methods to perform optical flow, ego-motion and depth estimation (Ye et al., 2018; Zhu et al., 2018b, 2019).

Several event-based motion compensation methods have also been proposed to estimate other types of motions, such as rotational motion (Gallego and Scaramuzza, 2017) and similarity transformations (Mitrokhin et al., 2018), by solving an optimisation procedure, whereby the event alignment of point trajectories is maximised. Typically, these methods accumulate a batch of events spanning a small time window onto an image-based representation and maximise some focus score (Gallego et al., 2019) w. r. t. the motion parameters considered. These event-based motion compensation methods were unified by Gallego et al. (2018b) under the designated Contrast Maximisation (CMax) framework. Similarly to Gallego and Scaramuzza (2017), the CMax framework maximises an image-based contrast measure (i. e. variance) by warping a batch of events according to candidate model parameters and projecting them onto the Image of Warped Events (IWE). It is unifying in the sense that the optimisation procedure is independent of the model parameters to be estimated (e. g. optical flow, depth, rotational motion). Gallego et al. (2019) later added several new loss functions to this framework.

2.3.2 *Event-based Motion Segmentation*

Event-based motion segmentation has gathered much attention due to the success of event cameras to address vision-based motion-related estimation problems. Early work focused on segmenting and tracking multiple people (Piątkowska et al., 2012), whereby a spatio-temporal tracker is proposed based on Gaussian Mixture Models. This achieved by solely considering the events' spatial information and (softly) assigning each event to a cluster via Maximum a Posteriori (MAP) estimation. Other works (Mitrokhin et al., 2018, 2019) have also been proposed to simultaneously segment and track multiple objects without requiring prior knowledge about their shape.

Recent event-based motion segmentation methods (Mitrokhin et al., 2018; Parameshwara et al., 2020; Stoffregen and Kleeman, 2017; Stoffregen et al., 2019; Zhou et al., 2021) also do not require prior knowledge on the shape of the clusters and perform per-event segmentation. However, they process batches of events, typically in a sliding window fashion, which is a common approach since each event carries little or no information on its own. Additionally, they require special initialisation procedures or additional methods for initialisation to start working appropriately. Stoffregen et al. (2019) perform motion segmentation by optimising an iterative formulation similar to EM, whereby the event-cluster assignment and the motion parameters are iteratively estimated. However, the method requires the number of clusters to be given, even though it is not too sensitive to this parameter. Another similar joint optimisation method was proposed by Parameshwara et al. (2020), where feature tracking was included to add robustness for long term segmentation. Zhou et al. (2021) propose to perform event-based motion segmentation as a multi-model energy minimisation problem on a space-time graph, which is built based on the events' spatial-temporal structure.

Deep learning methods have also been proposed to address this problem. Kepple et al. (2020) proposed an architecture that jointly learns visual motion and respective confidence from local patches. The architecture consists of two networks: one to estimate motion and the other to provide confidence. They achieve state-of-the-art results, but the method can not perform per-event segmentation since it relies on segmenting local patches. Thus, its generality to perform event-based motion segmentation is not clear. Mitrokhin et al. (2020) presented a Graph Convolutional neural network, by converting the event stream into a 3D graph. The advantage of this type of architecture is that the time information is better preserved. However, the approach is computationally expensive since it takes large slices in time to improve its

overall performance, and thus the process to build the 3D graph does not scale well with the number of events considered. Additionally, both these methods disregard the asynchronous nature of the events since they require the event stream to be transformed onto synchronous representations to be processed by the neural networks.

2.3.3 *Limitations of Previous Works*

The vast majority of previous event-based motion estimation and segmentation methods processes a batch of events in a sliding time window fashion. The main reason for this approach is that each event carries little or no information on its own. However, batch-based approaches have several downsides, namely: 1) artificial discontinuities may be introduced between batches (a similar occurrence may happen when considering consecutive frames), 2) high latency since the estimation procedure requires waiting for a batch of events, 3) typically, higher computational processing costs, and 4) a batch size needs to be set, i. e. how many events to consider per batch, whose optimal value depends on the scene dynamics. This may be especially evident in the context of KS estimation, whereby the underlying structure needs to be maintained and its parts tracked. For example, each body part may exhibit distinct dynamics, and it is thus not clear how one could appropriately set the global batch size. On the other hand, thinking in terms of batches can be more intuitive. Thus, we will first consider event-based methods that rely on batches, from which incremental approaches can be readily constructed without significant performance degradation. Also, the majority of previous event-based motion segmentation methods do not consider fusing other sensory information to augment the events (e. g. depth from RGB-D cameras).

2.4 FRAME-BASED KINEMATIC STRUCTURE ESTIMATION

In this section, we will provide an overview of existing frame-based KS estimation methods. Typically, the problem of KS estimation of articulated rigid objects is divided into two core sub-problems: 1) understanding and segmenting the different body parts, and 2) constructing a representation that meaningfully links them according to the underlying kinematic constraints of the object. By assuming that distinct body parts can exhibit different motion patterns, the first sub-problem is usually addressed by performing motion segmentation. The second sub-problem is addressed by building a representative graph-based on motion and shape cues. In this thesis, we consider

that the graph nodes represent the body parts and the edges correspond to the respective nodes' connections. We note that the dual graph representation, whereby nodes represent joints and edges represent segments, can also be found in the literature (e. g. Chang et al. (2018) proposed a method for switching between the two graph representations).

2.4.1 *From 2D Data*

Similarly to frame-based motion segmentation from 2D data, early works relied on factorisation approaches to segment the articulated rigid objects' parts (Tresadern and Reid, 2005; Yan and Pollefeys, 2005, 2008). Probabilistic formulations were also proposed to segment the body parts (Ross et al., 2010; Sturm et al., 2011), whereby the number of segments and links is adaptively estimated. However, these approaches are relatively sensitive to feature tracking errors or recovering from poor initialisation. A tailored subspace analysis spanned by relative transformations is explored by Jacquet et al. (2013), which constrains the relative displacement between two linked rigid parts to detect the type of articulated motion. Chang and Demiris (2015, 2018) performed rigid motion segmentation based on randomised voting (Jung et al., 2014), by also adding an iterative fine-to-coarse procedure to adaptively estimate the number of parts.

The KS graph is then typically generated as the minimum spanning tree, whose methods to estimate the edge weights widely vary. Ross et al. (2010); Yan and Pollefeys (2008) proposed the minimum principal angle between the subspaces (i. e. clusters) as a measure of affinity between the graph nodes. In the context of the probabilistic framework proposed by Sturm et al. (2011), the edge weights were computed as the negative log posterior probability of the resulting KS given a sequence of pose observations of the articulated object. These methods, however, do not take into account the neighbouring region of the representative points and thus may have difficulty in meaningfully representing the objects' topology. This issue was addressed by Fayad et al. (2011), who proposed the edge weights to be given by the distance between neighbouring tracked points as a weighted average of the respective velocities and coordinates distances. Chang and Demiris (2018) also address this problem by fusing the object boundary and the density map estimated from feature points and the skeleton distance map into a density weighted skeleton surface. The edge weights are then estimated from a combination of geodesic distances on the density weighted skeleton surface and pairwise motion velocities.

2.4.2 *From 3D Data*

The depth is very informative since it allows to lift from a 2D image-based space (which is not Euclidean) to a 3D Euclidean space. Several KS estimation methods (Katz et al., 2013; Martín-Martín et al., 2016; Pillai et al., 2014) exploit this insight by performing motion segmentation based on pairwise Euclidean distances differences (Perera and Barnes, 2013). Other methods rely on explicit 6-DOF estimation (Martín Martín and Brock, 2014), on spectral clustering methods (Tzionas and Gall, 2016), subspace clustering approaches (Kim et al., 2016b), on point cloud registration methods (Lu et al., 2018) or on a mixture of pairwise Euclidean distances differences and non-rigid points matching (Zhang et al., 2013).

Similarly to methods from 2D data, the KS graph is also typically generated as the minimum spanning tree. Zhang et al. (2013) build a similarity matrix based on the standard deviation of the pairwise Euclidean distances between points, and the weights are also given based on this score. Similarly to Sturm et al. (2011), Pillai et al. (2014) compute the edge weights based on an approximate probabilistic score by maximising the posterior probability of the KS given the sequence of observed relative transformations between the body parts. Lu et al. (2018); Martín Martín and Brock (2014) estimate the weights by computing the relative displacements between pairwise body parts constrained to estimated rigid bodies transformations. These approaches, however, assume that the Euclidean displacements can properly approximate the object's surface topological distances, which typically is only valid if the body parts are relatively well separated. In general, the topological structure of the object needs to be considered, e. g. using biharmonic distance (Lipman et al., 2010). Tzionas and Gall (2016) propose to estimate the object's topology by extracting the mean curvature skeleton (Tagliasacchi et al., 2012) to skeletonise the object, based on which the corresponding KS was generated.

More recently, methods based on deep learning have also been proposed. Besides huge amounts of training data, the proposed methods still only cope with relatively simple articulated rigid bodies (Jain et al., 2021; Zeng et al., 2021) or need to be given the category of the object a-priori (Li et al., 2020).

2.5 APPLICATIONS FOR KINEMATIC STRUCTURES

The Kinematic Structure (KS) of an articulated rigid object is a compact and meaningful representation that contains skeleton information while also

revealing several motion-related properties about the object. Several works directly or indirectly use KSs as an important component of their main pipeline. This representation is thus useful to many scientific domains, and we will focus our attention on three broad fields of research where KSs are commonly used, namely computer vision, robotics and healthcare.

Chang et al. (2018) proposed a method that learns to match distinct body parts that essentially have identical kinematic properties. Interestingly, the method can match body parts that do not necessarily belong to entities of the same category (e. g. cats, dogs, humans). The key correspondence factor lies in the similarity of the objects' topologies and exhibited motions, which is cast as a hypergraph matching problem by considering multi-order similarities. This method can be useful for further applications, such as action imitation in humanoid robots (Wu and Demiris, 2010a,b), where the robot must match its own KS with the human's, or robot learning by demonstration (Wu et al., 2014), which is closely related.

Other exploratory works grounded on KS estimation methods try to learn robot representations through self-perception (Sturm et al., 2008, 2009), learning by instruction of proto-actions (Petit and Demiris, 2016) or interactive perception (Bohg et al., 2017). Along this line of work, Moulin-Frier et al. (2018) proposed a cognitive framework that endows a robot to proactively acquire and express knowledge about itself and its surroundings through Human-Robot Interaction (HRI).

More recently, personalised dressing assistance has gained much attention due to the potential practical impact that such techniques could have in a near future. Several methods have been recently proposed (Gao et al., 2016; Zhang et al., 2017, 2019), which assist the (potentially impaired) user to dress. This is achieved by estimating the user's KS and then modelling the movements of each body part. In this scenario, the KS carries all the necessary information regarding the user's motion capabilities.

2.6 CONCLUSION

In this chapter, previous works on vision-based KS estimation and related problems, such as motion estimation and segmentation, were discussed. Several applications for KSs were also presented, demonstrating the importance of having methods that can accurately estimate the KS of articulated rigid objects. We have also compared the working principle of standard frame-based cameras and event cameras, whose latter properties will be explored for the first time in this work in the context of KS estimation.

As discussed, previous methods share a few commonalities, with each new method trying to improve a particular aspect of the KS estimation pipeline, e. g. how to improve the motion segmentation accuracy, how to better capture the distance between distinct body parts. These aspects can be further decomposed into sub-problems, some of which have not garnered the deserved attention or have simply been overlooked. Next, we present a comprehensive summary of the main threads in estimating the KS of an articulated rigid object.

Initial object representation. There are two main approaches to initially represent an object, namely using *sparse feature points* or using *dense points*. The initial representation step is crucial since it sets the tone for the rest of the KS estimation pipeline. A representation based on sparse feature points relies on other methods to detect distinctive points, e. g. scale invariant feature transform (SIFT) (Lowe, 2004). Although this representation has several advantages, e. g. low redundant information, low computational costs, it assumes that the object contains balanced and rich textural information, which restricts its usage. Representing an object by dense points tries to address this issue by imposing spatial constraints on neighbouring points, such that they undergo coherent motion (Stückler and Behnke, 2015). However, this representation typically contains redundant information and is computationally demanding. This suggests that investigating a compromise between the two representations, i. e. considering semi-dense points from frame-based 3D data, may result in combining the best properties of both. Also, as discussed in Section 2.1, since event cameras naturally respond to intensity changes, and by assuming that such changes are due to object motion, the events generated already contain shape and motion information from the object. This indicates that we may process the events directly to extract the relevant information without having to have an explicit initial object representation, which corresponds to a paradigm shift in KS estimation.

Tracking of the representative points. Vision-based tracking is an active topic of research within the computer vision community since robust methods are needed for long-term perception. Having an initial point-based object representation in one frame, the task of tracking it throughout subsequent frames serves to differentiate which object regions exhibit distinct patterns, whereby in the context of KS estimation are typically related to different body parts exhibiting distinct motion patterns. Point tracking is especially important for frame-based KS estimation since distinct body parts are segmented by analysing the corresponding points' trajectories. In the KS estimation literature,

tracking is typically accomplished by points registration or by estimating the optical flow between consecutive frames to establish points' correspondences. Computing the optical flow is typically preferred due to lower computational costs and the availability of reliable off-the-shelf methods (Jaimez et al., 2015). In line with previous remarks concerning event cameras, we note that the event stream produced already adapts to the scene dynamics by forming a "smooth" spatio-temporal manifold (e. g. see Figure 2.1), suggesting that tracking individual points may also be bypassed.

Lost points and occlusion. Lost points may happen due to tracking errors or points leaving the field of view of the camera, while occlusion is a broad term for regions of interest not being visible for a certain time duration. Some methods try to handle lost points by explicitly modelling them in the formulation (Elhamifar and Vidal, 2013), but these techniques typically only work properly when a small fraction of the points are lost and can not recover from severe cases. Occlusion still remains a challenge in motion segmentation and KS estimation tasks, and, besides a few notable exceptions (Huang et al., 2014; Judd et al., 2018; Unger et al., 2012), is assumed to not occur or is simply overlooked (Chang and Demiris, 2018; Jung et al., 2014).

Offline/batch vs online/incremental processing. Offline/batch and online/incremental processing are two distinct data processing design principles, which can be applied for motion estimation and segmentation and KS estimation. Methods that rely on offline/batch data processing store the information received for a certain time interval and then perform some estimation using all the stored data. Online/incremental processing methods perform estimation as soon as new data are available by maintaining an internal representation of the current state. Typically, offline/batch methods are more accurate than online/incremental methods since they can have a better overview of the data but are more computationally expensive and have higher latency.

Body parts segmentation. Motion patterns contain crucial information regarding how the object's body parts can be segmented, assuming distinct body parts exhibit different movements. Frame-based methods achieve motion segmentation by tracking representative points of each body part across frames and clustering points that exhibit similar motion patterns. Since each body part exhibits different movements by assumption, each cluster should correspond to an object's rigid part. Event-based methods can perform per-event motion segmentation without tracking points by instead explicitly estimating the motion parameters of each cluster directly from the events (Stoffregen et al., 2019). This is possible by assuming that motion is

the principal cause for the observed intensity changes in the field-of-view generating the events. Thus, the events already contain motion information, which can then be estimated and segmented.

Graph similarity estimation. As discussed in Section 2.4, there are several proposed approaches to estimate the edge weights connecting the nodes (i. e. body parts). However, most of them rely on distance measures that do not properly capture the underlying object's topology. Estimating the object's topology from a set of representative points is not a straightforward task and largely depends on the quality of the points, e. g. how many, how are they distributed. This, in turn, affects the estimation of distances on the object's manifold (Lipman et al., 2010), and consequently the edge weights.

The encountered key issues have largely motivated the course taken for the work developed by raising the main research questions investigated in this thesis (see Section 1.1 for the detailed list). Throughout the thesis, we will directly or indirectly address these problems by proposing novel approaches to handle them. The following chapter starts by addressing the issue of how to represent the object by considering semi-dense point trajectories from 3D frame-based data (e. g. from RGB-D cameras).

3

MOTION SEGMENTATION OF ARTICULATED RIGID OBJECTS FROM 3D POINTS

This chapter addresses the first research question:

> Can the Kinematic Structure (KS) estimation be improved by considering semi-dense 3D points instead of explicit 2D feature points, and how?

In this chapter, we address the problem of motion segmentation of articulated rigid bodies from a single-view RGB-D data sequence. As discussed in Chapter 2, previous methods either perform dense motion segmentation, and consequently are very computationally demanding, or rely on sparse 2D feature points, which may not be sufficient to represent the entire scene. As a compromise, we thus advocate the use of 3D semi-dense motion segmentation, which also bridges some limitations of standard 2D methods (e. g. background removal). We cast the 3D motion segmentation problem into a Sparse Subspace Clustering (SSC) problem (Elhamifar and Vidal, 2013), and perform Self-Tuning Spectral Clustering (STSC) (Zelnik-manor and Perona, 2004) to estimate the number of the object rigid parts. The resultant method has few parameters to adjust, requires no post-processing and scales well with the number of sequence frames. Figure 3.1 provides an overview of the proposed method.

Given a set of complete point trajectories from sequential frames, the goal of this chapter's work is to perform motion segmentation to then estimate the object's 3D articulated KS. In contrast to previous subspace clustering based motion segmentation methods, where 2D sparse feature points are tracked (Elhamifar and Vidal, 2013), we consider the case of semi-dense 3D points provided directly by an RGB-D sensor. To this end, a sub-sampled point cloud is generated from raw RGB-D data, and the respective point displacements between two consecutive frames are estimated based on the scene flow computed (Jaimez et al., 2015). Then, given complete point trajectories, the task of motion segmentation consists of clustering them according to their underlying motions, where we assume that distinct motions exist for each body part. In Section 3.1, the main problem addressed is mathematically formulated. We then cast the problem of motion segmentation into a SSC problem in Section 3.2 and show how we can use the STSC method to

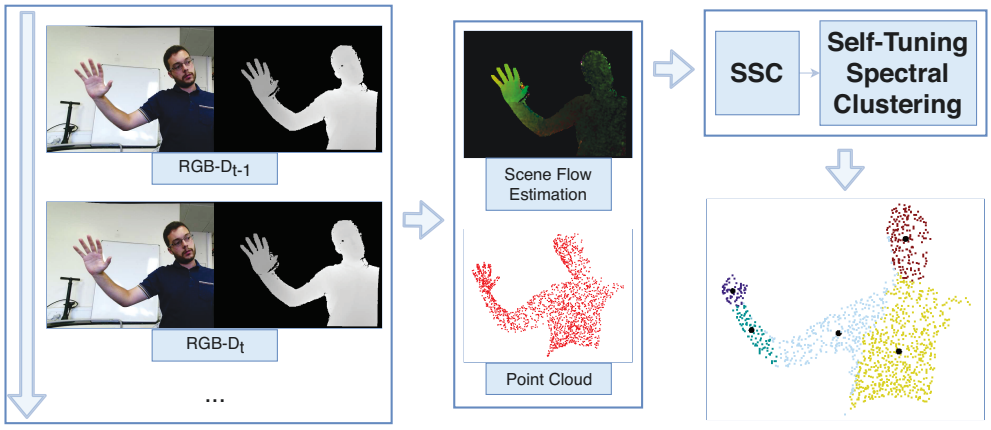


Figure 3.1: 3D offline motion segmentation overview. The proposed method receives an RGB-D image sequence corresponding to an object moving. For each frame, the respective semi-dense point cloud is generated, based on an adjustable parameter (i. e. the number of points to be sub-sampled). The scene flow between two consecutive frames is computed to keep track of the points throughout the sequence. Motion segmentation then is performed using SSC (Elhamifar and Vidal, 2013) with STSC (Zelnik-manor and Perona, 2004) for automatic number of object body parts selection.

estimate the number of segments of the articulated rigid object in Section 3.3. Section 3.4 theoretically analyses the influence of the number of frames on the overall method's computational complexity. In Section 3.5, we quantitatively and qualitatively evaluate the proposed method, and lastly Section 3.6 concludes this chapter.

3.1 PROBLEM FORMULATION

Given a set of point trajectories, corresponding to an RGB-D sequence of an articulated rigid object, arranged such that

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \dots & \mathbf{x}_N^1 \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \dots & \mathbf{x}_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1^F & \mathbf{x}_2^F & \dots & \mathbf{x}_N^F \end{bmatrix} \in \mathbb{R}^{3F \times N} \tag{3.1}$$

is the data matrix, the objective of motion segmentation is to cluster the point trajectories according to their underlying motion. The i-th point of frame f is defined as $\mathbf{x}_i^f \in \mathbb{R}^3$, where $i = \{1, \dots, N\}$, $f = \{1, \dots, F\}$, N is the number of points considered in each frame and F is the number of frames of the sequence. Similar to 2D motion segmentation (Yan and Pollefeys, 2008), where the point

trajectories of c rigid motions lie in a union of c low-dimensional subspaces of $\mathbb{R}^{2F}$ (Elhamifar and Vidal, 2013), 3D point trajectories also lie in a union of low-dimensional subspaces of $\mathbb{R}^{3F}$. This is because all point trajectories associated with a single rigid motion lie in an affine subspace of $\mathbb{R}^{3F}$. The 3D rigid body part motion segmentation may thus be cast as a subspace clustering problem.

3.2 MOTION SEGMENTATION BASED ON SPARSE SUBSPACE CLUSTERING

SSC (Elhamifar and Vidal, 2013) relies on the *self-expressiveness property* of the data to find a sparse representation, solving the convex optimization problem

$$\begin{aligned} &\min \|\mathbf{C}\|_1 + \lambda_e \|\mathbf{E}\|_1 + \frac{\lambda_z}{2}\|\mathbf{Z}\|_{\text{Fro}}^2 \\ &\text{s. t.} \quad \mathbf{Y} = \mathbf{Y}\mathbf{C} + \mathbf{E} + \mathbf{Z}, \quad \operatorname{diag}(\mathbf{C}) = \mathbf{0}. \end{aligned} \tag{3.2}$$

$\mathbf{C} \in \mathbb{R}^{N\times N}$ is the sparse representation matrix of the data matrix $\mathbf{Y} \in \mathbb{R}^{D\times N}$, $\mathbf{E} \in \mathbb{R}^{D\times N}$ represents sparse outlying entries, $\mathbf{Z} \in \mathbb{R}^{D\times N}$ represents occurring noise and $\operatorname{diag}(\mathbf{C}) \in \mathbb{R}^{N}$ corresponds to the vector of its diagonal elements. $\|\cdot\|_p$ denotes the p-norm and $\|\cdot\|_{\text{Fro}}$ denotes the Frobenius norm. Once the optimization program is solved, a sparse representation of the data $\mathbf{C}$ is obtained. A similarity graph with N nodes representing the data points can then be created, where the weights on the edges (i. e. affinity matrix) are computed as

$$\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^T. \tag{3.3}$$

This is done to ensure that if a given data point y_i is written as a linear combination of other data points including y_j (i. e. $c_{ij} > 0$), both are connected, even if the sparse representation y_j is not written as a linear combination that includes y_i (i. e. $c_{ji} = 0$). In other words, Equation (3.3) ensures that the similarity matrix is symmetric (i. e. $\mathbf{W} = \mathbf{W}^T$). Ideally, the similarity matrix has c connected components associated to each subspace, i. e.

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{W}_c \end{bmatrix}, \tag{3.4}$$

where $\mathbf{W}_l$ corresponds to the similarity matrix of data points belonging to subspace l.

In our context, $\mathbf{Y} = \mathbf{X} \in \mathbb{R}^{3F \times N}$ and the objective is to find a sparse representation $\mathbf{C}$, solving the optimization program given by Equation (3.2) with an additional constraint that captures the affine structure of the motion segmentation problem, i. e. $\mathbf{1}^T\mathbf{C} = \mathbf{1}^T$, where $\mathbf{1}$ represents a vector of ones. We assume that there are no sparse outlying entries (i. e. we do not consider the $\mathbf{E}$ term since only complete point trajectories are considered). However, noisy point trajectories are expected (e. g. due to noisy sensor measurements, or scene flow estimation innacuracies), which are handled by the $\mathbf{Z}$ term, and

$$\lambda_z = 800/\mu_z, \quad \mu_z = \min_i \max_{j \neq i} |\mathbf{y}_i^T \mathbf{y}_j|, \tag{3.5}$$

as suggested by Elhamifar and Vidal (2013).

3.3 SELF-TUNING SPECTRAL CLUSTERING

In order to cluster the data into subspaces, we apply spectral clustering (Ng et al., 2002) to the similarity matrix generated, according to Equation (3.3). Firstly, a symmetric normalized Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ is computed, i. e.

$$\mathbf{L} = \boldsymbol{\Delta}^{-\frac{1}{2}} \mathbf{W} \boldsymbol{\Delta}^{-\frac{1}{2}}, \tag{3.6}$$

where $\boldsymbol{\Delta} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with its elements given by

$$\delta_{ii} = \sum_{j=1}^{N} w_{ij}. \tag{3.7}$$

We then compute c eigenvectors of $\mathbf{L}$ corresponding to the associated c largest eigenvalues and form the matrix $\mathbf{V} \in \mathbb{R}^{N \times c}$ by stacking the eigenvectors column-wise. Lastly, K-means may be employed treating each row of $\mathbf{V}$ as a point, in order to cluster them into c clusters.[1]

However, the number of clusters c is not assumed to be given as input. A first attempt to estimate c could consist of performing eigenvalue decomposition of $\mathbf{L}$ and counting the number of eigenvalues above a given threshold, which would determine the number of clusters to segment. Although intuitive, this approach introduces an additional parameter to tune that may depend on the initial data matrix and thus on the observed object. Therefore,

1 Note that the Laplacian could also be computed as $\mathbf{L} = \mathbf{I} - \boldsymbol{\Delta}^{-\frac{1}{2}} \mathbf{W} \boldsymbol{\Delta}^{-\frac{1}{2}}$, as it appears more frequently in spectral graph theory. The only difference between these two formulations concerns the respective eigenvalues, i. e. from λ_i to $1 - \lambda_i$.

determining the number c based on the eigenvalues is infeasible for general applications.

Zelnik-manor and Perona (2004) introduced another approach, where a STSC algorithm was proposed based on eigenvector analysis instead of eigenvalues. The main idea is to find a rotation $\mathbf{R} \in \mathbb{R}^{c \times c}$ which best aligns the stacked eigenvector matrix $\mathbf{V}$ with the canonical coordinate system by computing its cost for every $c = 2, \ldots, N$. The number c^* that minimises the cost is selected as the number of clusters to segment. The algorithm follows by performing non-maximum suppression on the rows of $\mathbf{VR}^*$, where $\mathbf{R}^* \in \mathbb{R}^{c^* \times c^*}$ is the rotation matrix that minimises the cost of alignment.

In this work, we adopt the STSC algorithm (Zelnik-manor and Perona, 2004) to estimate the number of segments c of the articulated rigid object. For computational efficiency, the upper bound on range values of c was heuristically constrained to $c_{\text{upper}} = \lceil \log_2 N \rceil$. Also, the clusters obtained from non-maximum suppression on the rows of $\mathbf{VR}^*$ were used to initialise the K-means algorithm, as suggested by Zelnik-manor and Perona (2004) to accommodate for highly noisy data. In the end, a subset of n_g points $\mathbf{S}_g$ belonging to each segment g is obtained, where $N = n_1 + \cdots + n_g + \cdots + n_c$. The respective centre positions $\mathbf{m}_g^f$ are obtained by averaging all points belonging to the associated segment for each frame.

3.4 INFLUENCE OF THE NUMBER OF FRAMES

The data matrix $\mathbf{X}$, given by Equation (3.1), depends on the number of frames F of the sequence of RGB-D images, since $\mathbf{X} \in \mathbb{R}^{3F \times N}$. In the general case where the optimization program given by Equation (3.2) must be solved, each frame accounts for three constraints that must be satisfied. Thus, it is natural to assume that the complexity and processing time of the algorithm increase with the number of frames. However, that is not the case, and we mathematically demonstrate why next. Additionally, in Section 3.5, we experimentally support this claim.

The reason for this is related to which method is used to solve the sparse subspace optimization problem given by Equation (3.2). Since the problem is convex, one could use any generic convex solver. However, as suggested by (Elhamifar and Vidal, 2013), an Alternating Direction Method of Multipliers (ADMM) was adopted for an efficient implementation of the proposed problem. Since in this work we assume complete point trajectories (i. e. we do

not consider the $\mathbf{E}$ term since we are not considering outliers), Equation (3.2) becomes

$$\begin{aligned} &\min \|\mathbf{C}\|_1 + \frac{\lambda_z}{2}\|\mathbf{Z}\|_{\text{Fro}}^2 \\ &\text{s. t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C} + \mathbf{Z}, \quad \mathbf{C}^\top \mathbf{1} = \mathbf{1}, \quad \text{diag}(\mathbf{C}) = \mathbf{0}. \end{aligned} \tag{3.8}$$

By introducing auxiliary variables and Lagrange multipliers into the optimization program, we can obtain the following associated Lagrangian function of Equation (3.8) (Elhamifar and Vidal, 2013):

$$\begin{aligned} \mathcal{L}(\mathbf{C}, \mathbf{A}, \boldsymbol{\gamma}, \boldsymbol{\Gamma}) =& \|\mathbf{C}\|_1 + \frac{\lambda_z}{2}\|\mathbf{X} - \mathbf{X}\mathbf{A}\|_{\text{Fro}}^2 \\ &+ \frac{\rho}{2}\|\mathbf{A}^\top \mathbf{1} - \mathbf{1}\|_2^2 + \frac{\rho}{2}\|\mathbf{A} - \mathbf{C} + \text{diag}(\mathbf{C})\|_{\text{Fro}}^2 \\ &+ \boldsymbol{\gamma}^\top(\mathbf{A}^\top \mathbf{1} - \mathbf{1}) + \text{tr}(\boldsymbol{\Gamma}^\top(\mathbf{A} - \mathbf{C} + \text{diag}(\mathbf{C}))), \end{aligned} \tag{3.9}$$

where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is an auxiliary matrix, $\rho > 0$ is a penalty parameter, $\boldsymbol{\gamma} \in \mathbb{R}^N$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times N}$ are a vector and matrix of Lagrange multipliers, respectively, $\text{tr}(\cdot)$ denotes the trace operator of a matrix and we abuse the notation of $\text{diag}(\mathbf{C})$ throughout this section to denote a vector containing the diagonal elements of the matrix $\mathbf{C}$, as well as a diagonal matrix whose elements correspond to the entries of the diagonal of $\mathbf{C}$. This function can be iteratively solved via ADMM, until convergence or the number of iterations reaches a certain number:

- Update $\mathbf{A}^{(k+1)}$ according to

$$(\lambda_z \hat{\mathbf{X}} + \rho\mathbf{I} + \rho\mathbf{1}\mathbf{1}^\top)\mathbf{A}^{(k+1)} = \lambda_z \hat{\mathbf{X}} + \rho(\mathbf{1}\mathbf{1}^\top + \mathbf{C}^{(k)}) - \mathbf{1}\boldsymbol{\gamma}^{(k)\top} - \boldsymbol{\Gamma}^{(k)}, \tag{3.10}$$

  where $\hat{\mathbf{X}} = \mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{N \times N}$ and $\mathbf{I} \in \mathbb{R}^{N \times N}$ denotes the identity matrix.

- Update $\mathbf{C}^{(k+1)}$ according to

$$\mathbf{C}^{(k+1)} = \mathbf{J} - \text{diag}(\mathbf{J}), \quad \mathbf{J} = \mathcal{T}_{\frac{1}{\rho}}\left(\mathbf{A}^{(k+1)} + \frac{\boldsymbol{\Gamma}^{(k)}}{\rho}\right), \tag{3.11}$$

  where $\mathcal{T}_\eta(v) = (|v| - \eta)_+ \text{sgn}(v)$ and $(\cdot)_+$ returns its argument if it is non-negative or zero otherwise.

- Update $\boldsymbol{\gamma}^{(k+1)}$ according to

$$\boldsymbol{\gamma}^{(k+1)} = \boldsymbol{\gamma}^{(k)} + \rho(\mathbf{A}^{(k+1)\top}\mathbf{1} - \mathbf{1}). \tag{3.12}$$

- Update $\Gamma^{(k+1)}$ according to

$$\Gamma^{(k+1)} = \Gamma^{(k)} + \rho(\mathbf{A}^{(k+1)} - \mathbf{C}^{(k+1)}). \tag{3.13}$$

All the involved quantities, namely $\mathbf{A}$, $\mathbf{C}$, γ and Γ, only depend on the number of points considered in each frame, making the optimisation procedure not dependent on the number of frames. Only the auxiliary matrix $\hat{\mathbf{X}}$ depends on the number of frames since it is the product of two matrices that depend on the number of frames. However, it can be computed once before the actual iterative computation. In other words, the method's performance is only limited by one initial matrix multiplication that depends on the number of frames considered, which in practice is negligible since we may assume $N > F$. Therefore, the number of frames does not influence significantly the performance of the algorithm in terms of computation time.

3.5 EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the proposed method on the dataset provided by Tzionas and Gall (2016), designated by 3DKS Dataset. The 3DKS Dataset consists of RGB-D sequences of rigged articulated rigid objects moving: `donkey`, `lamp`, `pipe 1/2`, `pipe 3/4` and `spray`. Since the method requires complete point trajectories, only points that are tracked throughout the whole sequence are included in the set of point trajectories, given by Equation (3.1).[2]

3.5.1 *Implementation Details*

All experiments were performed using a PC with an Intel Core i7-8700k CPU @ 3.7Ghz (x6) and 32GB of RAM. The proposed method is deterministic and only a few parameters require adjustment, namely: parameter λ_z (i. e. parameter that balances the noise term in Equation (3.2)), the upper bound on range values of number of segments to be estimated c_{upper} and the number of initial sub-sampled points to be tracked N_{init}. Parameter λ_z is studied by Elhamifar and Vidal (2013) in the context of 2D motion segmentation. The resultant clustering error is not significantly affected by a wide range of values. We thus followed the suggested value, as presented in Equation (3.5).

2 Elhamifar and Vidal (2013) described two approaches to deal with missing entries in the data matrix (i. e. incomplete point trajectories). However, both only work properly when a small fraction of entries is missing, which may not occur in the context of our work, mainly due to sensor measurement errors.

The upper bound on range values for estimating the number of segments c_{upper} only affects the size of the search space for the optimal value c^*. In this sense, the proposed search constraint (i. e. $c_{upper} = \lceil \log_2 N \rceil$) is merely suggestive, as the only observable effect is the computational time. This means that depending on the knowledge of a specific problem one could be more conservative by setting the lowest possible upper bound (e. g. if a given body is not expected to have more than ten segments, one could set $c_{upper} = 10$). Nevertheless, the point to be made is that the method can be applied to other problems, even without changing the upper bound. Therefore, we will only study the effect of the number of initial sub-sampled points N_{init}. Since the method is deterministic, to test its robustness and reliability, we randomly sub-sample a subset of points, where its size is determined by N_{init}.

3.5.2 *Quantitative Results*

Five metrics were evaluated, namely: execution time, the number of segments estimated, precision, recall and f-measure. As described by (Ochs et al., 2014), the last three metrics for each pair of estimated segment $\mathbf{S}_i$ and ground-truth segment $\mathbf{S}_j^{GT}$ are defined as follows, respectively:

$$P_{ij} = \frac{|\mathbf{S}_i \cap \mathbf{S}_j^{GT}|}{|\mathbf{S}_i|} \quad R_{ij} = \frac{|\mathbf{S}_i \cap \mathbf{S}_j^{GT}|}{|\mathbf{S}_j^{GT}|} \quad F_{ij} = \frac{2P_{ij}R_{ij}}{P_{ij} + R_{ij}}. \tag{3.14}$$

These metrics try to capture a trade-off between measuring false positives and misses. The Hungarian method (Kuhn, 1955) is then applied to find the best allocation of segments to ground-truth ones. Empty segments are introduced if there are fewer segments estimated than ground-truth segments (i. e. recall is zero and precision is defined to be one).

Figure 3.2 shows the quantitative results obtained over one hundred trials for a range of values of N_{init}. The following conclusions can be drawn:

- *Computational time*: as the number of sub-sampled points increases, so does the execution time of the proposed method, as expected.

- *Number of segments estimated*: apart from the `donkey` object sequence, the number of segments estimated converges to the ground-truth for all other objects. Also, the associated standard deviation decreases as the number of initial sub-sampled points increases, suggesting that tracking more points improves the performance of the method.

- *Precision, recall and f-measure*: apart from the `spray` object sequence, increasing the number of sub-sampled points exhibit slight improvements, both

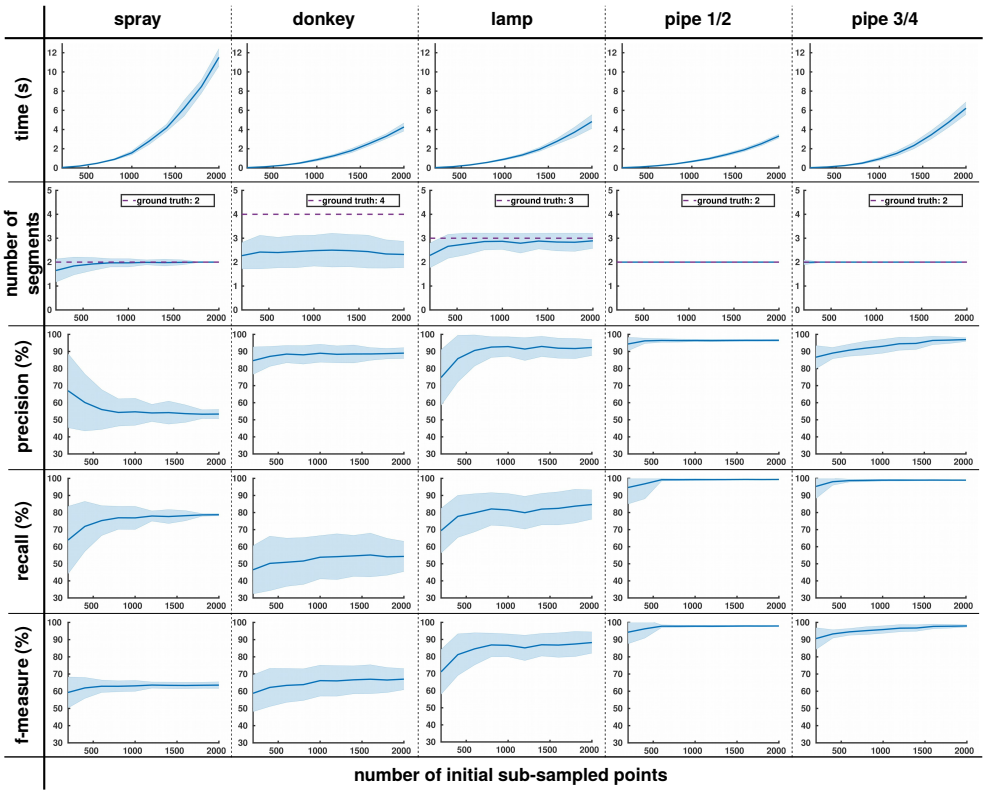


Figure 3.2: Quantitative results concerning five metrics: execution time (first row), number of segments estimated (second row), precision, recall and f-measure (third, fourth and fifth rows, respectively). These results were obtained by varying the number of initial randomly sub-sampled points across one hundred trials per number. The mean is represented by the solid line and the standard deviation is represented by the surrounding shaded area.

in terms of higher expected accuracy and lower variability, for the three metrics.

- There should be a compromise w. r. t. the number of point trajectories considered. The results obtained suggest that $N_{init} = 1000$ initial points considered is a good compromise since no significant improvements are observed for higher values, while the computational time still increases. Further discussion regarding the different results obtained for `donkey` and `spray` objects sequences, concerning the number of segments estimated and precision, respectively, will be done considering the qualitative results as well.

In Table 3.1, a summary of the quantitative obtained results is provided, using Low Rank Representation (LRR) (Liu et al., 2013) and SSC (Elhamifar and Vidal, 2013) methods to build the affinity matrix $\mathbf{W}$. Higher performance is obtained when the affinity matrix is built from SSC, which also corroborates

Table 3.1: Summary of the average quantitative results obtained.

		donkey	lamp	pipe 1/2	pipe 3/4	spray	Average
SSC	Time per point (msec)	1.03	1.14	0.80	1.32	2.39	1.34
	# of segments	2.41/4	2.77/3	2/2	2.00/2	1.93/2	-
	Precision (%)	88.03	89.68	96.21	93.07	56.04	84.61
	Recall (%)	52.53	80.34	98.58	98.46	75.59	81.10
	F-Measure (%)	65.80	84.75	97.38	95.69	64.37	82.82
LRR	# of segments	2.80/4	1.54/3	1.49/2	1.27/2	1.69/2	-
	Precision (%)	60.33	67.48	67.13	70.77	75.97	68.34
	Recall (%)	43.06	17.32	53.06	50.44	56.16	44.01
	F-Measure (%)	50.25	27.56	59.27	58.90	64.58	52.11

the choice of using it. The computation time for LRR is not provided since the respective author's code in MATLAB was used, whereas a version of SSC was implemented in C++. It is interesting to note that when 1000 points are tracked, the average computation time is approximately 1.34 seconds. Considering that every video takes more than 3 seconds, this means that the actual computation is faster than the video sequence, suggesting that a real-time online approach relying on this method would be feasible. This topic will be considered in the next chapter.

3.5.3 *Qualitative Results*

In Figure 3.3, we show qualitative results regarding the proposed 3D motion segmentation method (bottom row), as well as the respective objects and ground-truth segmentation. As shown by the quantitative results, for the `lamp`, `pipe 1/2` and `pipe 3/4` objects, the method achieves an average f-measure performance above 84%.[3] However the results obtained are lower in the case for `donkey` and `spray` objects.

Concerning the `donkey` object, we observe that a significant portion of the segmented body is not filled with points. This is because severe occlusions occur during the sequence, particularly between the tip of the arms and the body, and between the head and the body. Thus a significant number of point trajectories are discarded, which could provide better overall performance.

In the case of `spray` object, the associated motion might not exhibit sufficient evidence for a correct motion segmentation (as shown in the lower right corner image in Figure 3.3). Therefore the proposed method is not able to

3 The method could also segment body parts that did not belong to the object, i.e. `left hand` in the `pipe 3/4` object sequence. Nevertheless, only the motion segmentation of the actual object was evaluated.

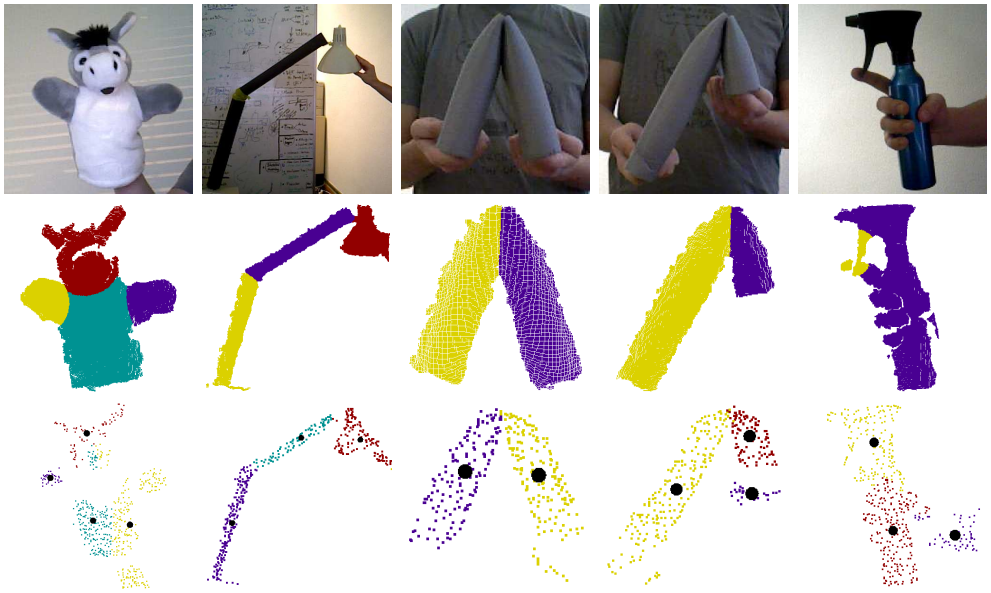


Figure 3.3: Qualitative results obtained with the same parameters (i. e. λ_z given by Equation (3.5)), $c_{upper} = \lceil \log_2 N \rceil$, $N_{init} = 1000$). The first row presents the articulated rigid objects. The second row depicts the motion segmentation ground truth, whereby each colour represents one segment. The third row shows the motion segments estimated (the black dots correspond to the centre coordinates of each body part). Figure best viewed in colour.

accurately segment subtle motions, which may occur in real-life scenarios, either due to noisy sensor measurements or errors during point tracking. This observation is corroborated by the fact that the precision results obtained for the `spray` object (third row of the first column in Figure 3.2) decrease for higher numbers of initial sub-sampled points since the ratio of correctly segmented points decreases.

3.5.4 *Influence of the Number of Frames*

Table 3.2: Summary of the average processing time per point obtained considering 2F frames.

	`donkey`	`lamp`	`pipe 1/2`	`pipe 3/4`	`spray`	Average
Time per point (msec)	1.14	1.03	1.00	1.32	2.36	1.37

The influence of the number of frames to be processed on the overall computational time was also evaluated and the results are reported in Table 3.2 (the length of each sequence was doubled, i. e. each sequence was considered forward and then backwards). The average processing time per point is identical, comparing the original sequence length (F frames) and the doubled one (2F

frames). This experimentally corroborates that the number of frames to be processed does not significantly affect the overall computational cost of the proposed method. Please refer to Section 3.4 for a mathematical justification.

3.6 CONCLUSION

A method for 3D motion segmentation of articulated rigid bodies from RGB-D data was proposed in this chapter. Given a set of complete point trajectories, the method estimates automatically 3D motion segments by solving a sparse optimization program and then applying adaptive spectral clustering to the resultant affinity matrix to estimate the number of body segments. The method is deterministic and can achieve low computational time depending on the number of points considered. The method was evaluated on a public dataset, achieving consistent results. It has few parameters to adjust, making it suitable for automatic motion segmentation of articulated rigid bodies in a wide range of scenarios. Such scenarios may include visual self-exploratory learning for lifelong autobiographic memories (Petit et al., 2016).

However, the method presented in this chapter is not capable of handling occlusions or segments entering/exiting the scene, which may occur in the context of Human-Robot Interaction (HRI) scenarios. In the next chapter, we will focus on this issue by performing frame-by-frame motion segmentation and estimating the object's KS on demand.

Contributions. The main contributions presented in this chapter have been published in:

Nunes, U. M. and Demiris, Y. (2018), 3D Motion Segmentation of Articulated Rigid Bodies based on RGB-D Data, *in* 'British Machine Vision Conference (BMVC)'. (Nunes and Demiris, 2018)

4

ONLINE KINEMATIC STRUCTURE ESTIMATION OF ARTICULATED RIGID OBJECTS FROM 3D POINTS

This chapter addresses the second research question:

> How can a suitable representation that is robust to lost feature points be incrementally maintained for online Kinematic Structure (KS) estimation?

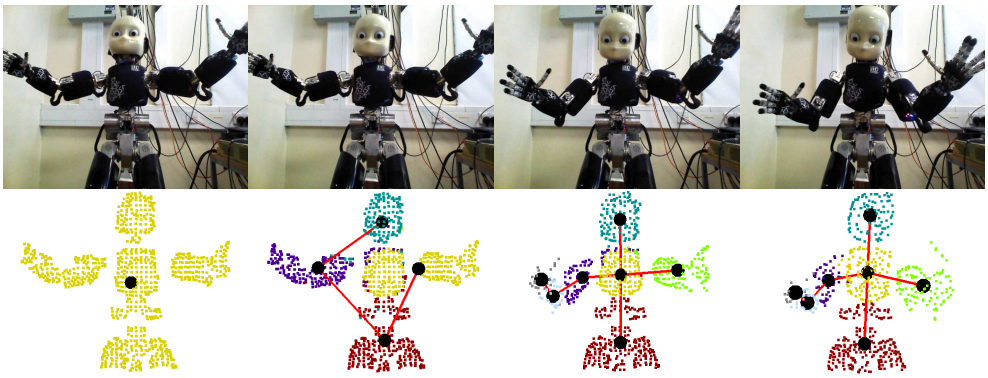


Figure 4.1: Frames 19, 24, 162 and 499 of the `iCub` sequence (first row) and the corresponding 3D KS estimation by the proposed method (second row). Until around frame 19, no significant motion is observed, and the whole object is thus segmented. By frame 24, some motions are perceived, and an initial sketch of the 3D KS is built. At frame 162, the right arm has been correctly segmented into three parts, with some points around the torso remaining misclassified, until they are correctly labelled, which is seen in frame 499. Figure best viewed in colour.

In the previous chapter, we presented an offline method that processes a batch of RGB-D frames and assumes that complete point trajectories are available. It provided empirical evidence that considering semi-dense 3D point trajectories is a valid compromise between accuracy and computational costs. However, as mentioned in Section 3.6, this approach is not feasible when considering scenarios that require continuity and fluidity, for instance, in Human-Robot Interaction (HRI) scenarios. Hence, in this chapter, we address the problem of estimating frame-by-frame the 3D KS estimation of arbitrary articulated rigid bodies from RGB-D sequences. We propose to tackle this problem in an online fashion by recursively maintaining a metric distance of the scene's 3D structure and aiming to achieve real-time performance. The noise influence

is mitigated by building a similarity measure based on a linear embedding representation and incorporating this representation into the original metric distance. The KS is then estimated based on a combination of implicit motion and spatial properties. An illustration of our method in action is presented in Figure 4.1.

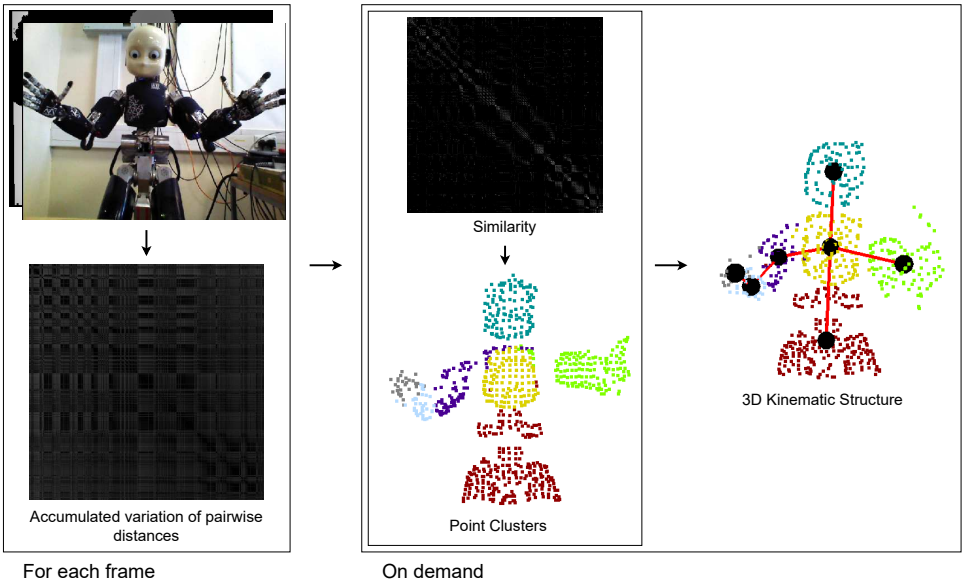


Figure 4.2: Online 3D KS estimation overview. For each RGB-D frame, we compute the respective Accumulated Variation of Pairwise Euclidean Distances (AVPED) between tracked 3D points. A similarity matrix can then be built based on the AVPED, from which we perform segmentation by *label spreading* to obtain point clusters. Finally, the 3D KS can be estimated by combining (implicit) motion and spatial properties of the tracked points. Figure best viewed in colour.

In Figure 4.2, we provide an overview of the proposed method. The method processes the information of incoming frames in an online fashion to estimate the KS of arbitrary articulated rigid bodies. We only make the following assumptions in the design of the proposed method: 1) a 3D point cloud of the object is assumed to be given (which can be provided by any suitable sensor or by a sequential 3D shape reconstruction method from monocular images (Agudo and Moreno-Noguer, 2017, 2018)), and 2) we assume the body is piecewise rigid, which implies that the pairwise Euclidean distance[1] between points belonging to the same body segment is constant throughout all frames; on the contrary, the pairwise Euclidean distance between points that belong to different body parts changes provided sufficient relative motion is observed.

1 We refer to the Euclidean distance as the l_2-norm.

Each point is tracked based on the scene flow (Jaimez et al., 2015), and the pairwise Euclidean distance between the point cloud points is computed for each incoming frame (Section 4.1). The AVPED is computed and used to build the correspondent similarity measure, which is presented in Section 4.2. In Section 4.3, we propose to use *Label spreading* (Zhou et al., 2004) to segment the points into distinct body parts. The KS of the object is then generated as the corresponding graph's minimum spanning tree in Section 4.4. Sections 4.5 and 4.6 present how lost points and noise are handled, respectively. In Section 4.7, we experimentally assess the proposed method's performance. The proposed approach achieves competitive performance both quantitatively and qualitatively in terms of estimation accuracy, even compared to offline methods. Finally, Section 4.8 concludes this chapter.

4.1 PROBLEM FORMULATION

At each frame f, we aim to estimate the KS of an articulated rigid body, given its point cloud and respective 3D point positions

$$\mathbf{X}^f = \begin{bmatrix} \mathbf{x}_1^f & \mathbf{x}_2^f & \dots & \mathbf{x}_N^f \end{bmatrix} \quad \in \mathbb{R}^{3\times N}. \tag{4.1}$$

We can compute the pairwise Euclidean distance $\mathbf{D}^f$ between all points $\mathbf{x}_i^f$ and then obtain the AVPED according to

$$\tilde{\mathbf{D}}^f = \sum_{j=1}^{f} \frac{|\mathbf{D}^j - \mathbf{D}^{j-1}|}{t_s} = \tilde{\mathbf{D}}^{f-1} + \frac{|\mathbf{D}^f - \mathbf{D}^{f-1}|}{t_s}, \tag{4.2}$$

where t_s is the sampling period. With this formulation, the AVPED can also be interpreted as an accumulation of the absolute temporal rate of change in pairwise Euclidean distances. Note that by construction, the AVPED defines a proper Euclidean distance matrix, which will be used to build a similarity matrix between every point considered and to cluster them. Since only the relative distance between points matters, this formulation does not require the static camera assumption, i. e. motion would be captured implicitly w. r. t. relative pairwise distances.

4.2 BUILDING THE SIMILARITY

The proposed method relies in the fact that the pairwise Euclidean distances between points belonging to the same body segment remain constant through-

out the frames, whereas the pairwise distances vary for points belonging to different body parts, which translates into

$$\begin{cases} \tilde{d}_{ij} \leqslant \epsilon & \text{if } p_i, p_j \in S_g \\ \tilde{d}_{ij} > \epsilon & \text{otherwise} \end{cases}. \tag{4.3}$$

The pairwise distance between points p_i and p_j is denoted by $\tilde{d}_{ij}$, which is computed according to Equation (4.2), S_g represents the set of points belonging to segment g, and ϵ is a value near 0. Thus, a natural construction of a similarity matrix is to associate points belonging to the same body part if the AVPED is near zero

$$\hat{w}_{ij} = \begin{cases} 1 & \text{if } \tilde{d}_{ij} \leqslant \epsilon \\ 0 & \text{otherwise} \end{cases}, \tag{4.4}$$

and thresholding accordingly. Perera and Barnes (2013) proposed a similar approach, whereby instead they constructed the similarity matrix considering the standard deviation of the pairwise distances between points. However, such an approach has a major limitation: a global threshold must be provided, which in practice may depend on the observed scene or the sensor measurement's errors. Typically, a different value must be manually found and set for each scenario, which can be cumbersome.

Instead, we embed the AVPED in a feature space and use the resultant representation to build the similarity matrix $\mathbf{W}$. We also denote this procedure by *spectrum transformation*. In particular, since the AVPED defines a proper Euclidean distance matrix, a configuration $\mathbf{Z} \in \mathbb{R}^{N \times k}$, $k \leqslant N$ can be found, such that the original Euclidean distances are preserved and we have that (Pekalska et al., 2002; Zhang and Zhou, 2009)

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\left(\tilde{\mathbf{D}} \odot \tilde{\mathbf{D}}\right)\mathbf{J} = \mathbf{Z}\mathbf{Z}^T \quad \in \mathbb{R}^{N \times N} \tag{4.5}$$

is the positive definite matrix of the inner products of the underlying configuration $\mathbf{Z}$, where $\mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T \in \mathbb{R}^{N \times N}$, $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity matrix and $\mathbf{1} \in \mathbb{R}^N$ is a vector of ones. $\mathbf{B}$ can be factorised by its eigendecomposition as

$$\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{4.6}$$

where $\mathbf{V}$ is the orthogonal matrix of eigenvectors corresponding to the eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_N)$ sorted in descending order in the diagonal of $\mathbf{\Lambda}$.

Since $\mathbf{B}$ is positive definite, we may obtain a k-dimensional ($k \leqslant N$ non-zero eigenvalues) representation of $\tilde{\mathbf{D}}$ as

$$\mathbf{Z} = \mathbf{V}_k \boldsymbol{\Lambda}_k^{\frac{1}{2}}, \tag{4.7}$$

where $\mathbf{V}_k \in \mathbb{R}^{N \times k}$ is a matrix of the k eigenvectors corresponding to the first k leading eigenvalues contained in the diagonal of $\boldsymbol{\Lambda}_k \in \mathbb{R}^{k \times k}$. Finally, we build a similarity matrix $\mathbf{W}$, whose entries w_{ij} are given by an exponential radial basis function kernel

$$w_{ij} = \exp\left[-\left(\frac{|\mathbf{z}_i - \mathbf{z}_j|}{\gamma}\right)^2\right], \tag{4.8}$$

where $\mathbf{z}_i \in \mathbb{R}^k$ corresponds to the i-th row of $\mathbf{Z}$ and $\gamma \in \mathbb{R}_+$ is related to the kernel parameter. The resulting matrix will closely match Equation (4.4), where points belonging to the same segment are associated since their AVPED is almost zero.

4.3 CLUSTERING BY LABEL SPREADING

Having built the similarity matrix $\mathbf{W}$, we can now employ any suitable clustering algorithm, e. g. spectral clustering (Ng et al., 2002; Zelnik-manor and Perona, 2004) since the continuity of the scene is maintained by the AVPED. The final results however may exhibit discontinuities between frames since we would not be taking full advantage of previous segmentation results (Elqursh and Elgammal, 2013). Thus, to take into consideration previous labelling information, we use *label spreading* (Zhou et al., 2004) on the graph defined by $\mathbf{W}$, which also allows re-labelling depending on a combination of past information and current graph structure. New labels are obtained by performing normalised cuts (Shi and Malik, 2000) on each existing sub-graph, each corresponding to one label. We briefly describe this approach next.

4.3.1 *Label Spreading*

Based on the similarity matrix $\mathbf{W}^f$, we compute its normalized graph Laplacian, according to Equation (3.6), from Section 3.3. Then, the new labels $\mathbf{Y}^f \in \mathbb{R}^{N \times c^f}$ are obtained given previous labels $\mathbf{Y}^{f-1} \in \mathbb{R}^{N \times c^{f-1}}$ with each row having the probability of point p_i belonging to label g as follows

$$\mathbf{Y}^f = (1 - \mu)(\mathbf{I} - \mu\mathbf{L})^{-1}\mathbf{Y}^{f-1}, \tag{4.9}$$

where $\mu \in [0, 1)$. The parameter μ controls how much the graph structure influences future labels in comparison to previous labels, e. g. when $\mu \to 1$ the graph structure is the dominant factor.

4.3.2 *Obtaining New Labels*

Since *label spreading* only propagates existing labels, we need to be able to find new ones, e. g. new segments may emerge. Points belonging to an emergent new label would be assigned some existing label, increasing its intra-cluster variation. This means we could perform a normalized cut to each label's sub-graph and evaluate if its cost is below some threshold τ. Given the respective similarity sub-matrix $\mathbf{W}_g$, we compute its normalized graph Laplacian $\mathbf{L}_g$, according to Equation (3.6), from Section 3.3, and obtain the eigenvector corresponding to the second largest eigenvalue[2]. The eigenvector is then thresholded according to the average of its values, and the normalized cut cost is obtained. If this cost is below τ, then we split the cluster, obtaining a new one; otherwise, we keep it intact.

4.4 BUILDING THE KINEMATIC STRUCTURE

The KS is represented as the minimum spanning tree of a non cyclic graph $\mathcal{G} = (V, E)$ (Chang and Demiris, 2018; Yan and Pollefeys, 2008), where V and E correspond to the set of vertices and edges respectively. Each segment's center $\mathbf{m}_g$ is treated as a vertex and the proximity between each pair of vertices (i. e. edges' weights) is given by

$$E_{ij} = \beta \tilde{d}_{ij} + (1 - \beta)\hat{d}_{ij}, \tag{4.10}$$

where $\tilde{d}_{ij}$ is the AVPED and $\hat{d}_{ij}$ corresponds to the biharmonic distance (Lipman et al., 2010) between the segments' centers i and j, respectively. This distance provides a balance between geodesic distances for local neighbourhoods and large distances across the global structure. Each edge's weight is thus a combination of the AVPED, which stores motion information, and the biharmonic distance, which contains information about the shape and spatial structure of the body, weighted by the scalar $\beta \in [0, 1]$.

2 The Laplacian can also be computed as $\mathbf{L} = \mathbf{I} - \mathbf{\Delta}^{-\frac{1}{2}}\mathbf{W}\mathbf{\Delta}^{-\frac{1}{2}}$, the only difference being the values of the eigenvalues, from λ_i to $1 - \lambda_i$; hence the selection of the eigenvector corresponding to the second largest eigenvalue, instead of the second smallest one.

4.5 HANDLING LOST POINTS

While observing the scene, some points may get lost, e. g. due to tracking/measurement errors or some body part naturally disappearing from the field of view. Thus, the proposed method must handle these occurrences, as in a real-world scenario, such that their influence is minimised. One can observe that the AVPED, according to Equation (4.2), is directly responsible for maintaining information about the scene. Therefore, we propose to handle lost points during the computation of $\tilde{\mathbf{D}}$. First, each point cloud is represented in an octree data structure (Huang et al., 2008), which allows us to have some knowledge about the 3D spatial structure of the scene and manipulate it accordingly. Whenever a point is lost, we then search for the region with fewer points and randomly select a point to start tracking. The approach is detailed next.

4.5.1 *Octree Representation of a 3D Point Cloud*

An octree representation is an efficient tree-based data structure, where each branch node represents a cube volume in a 3D space, known as a voxel. Each node is sub-divided into eight children-nodes until some stopping criterion is met, e. g. maximum depth or intra-resolution. Nodes that do not have children-nodes are designated by leaf nodes, and all the original points are mapped onto them. This representation allows different levels of branch depths, which leads to efficient data processing.

4.5.2 *Maintaining an Even Distribution of Points Tracked*

We are interested in maintaining an evenly balanced distribution of the 3D points being tracked, which is obtained from the density of points. The density can be approximated by estimating the number of neighbouring points, which is given by

$$\rho_i = \#\text{neighbourhood}(i) = \#\left\{p_j : |\mathbf{x}_i - \mathbf{x}_j| < r\right\}. \tag{4.11}$$

Essentially, we count the number of points inside a sphere with radius r and centre $\mathbf{x}_i$, i. e. the 3D position of the point of interest. This statistic allows us to have a simple yet effective policy to keep an even distribution of points tracked during the entire scene: if this number is greater than some threshold (i. e. $\rho_i > \rho_{\text{thresh}}$), it means too many points are concentrated in a given region,

and some should be discarded; afterwards, we search for the region with the lowest density and randomly select one point from it.

4.5.3 *Adding New Points*

In general, when a new point starts to be tracked, no previous information is known about it, since we have no prior knowledge about its respective AVPED from previous frames $\tilde{\mathbf{d}}_i^{f-1}$, as in Equation (4.2). When this is the case, $\tilde{\mathbf{d}}_i^{f-1}$ is initialised to zero, and the method runs as if this was the first frame for this particular point.

However, a new point can be selected in a region where previously tracked points are located. In line with our second assumption (please refer to the beginning of Chapter 4), we presume that this point will have close similarity to the new neighbouring points, especially if they belong to the same body part. Thus, we advocate that its AVPED should be initialized in function on the AVPED of its neighbouring points. In particular, we propose a weighted average according to

$$\tilde{\mathbf{d}}_i^{f-1} = \frac{\sum_j a_j \tilde{\mathbf{d}}_j^{f-1}}{\sum_j a_j} \quad \forall j \in \text{neighbourhood}(i), \tag{4.12}$$

where $a_j = \exp\left(-\alpha|\mathbf{x}_i^f - \mathbf{x}_j^f|\right)$ and $\alpha \in \mathbb{R}_+$. This means that closer points contribute more than farther ones.

Note that lost points may be seen as a special case that is naturally handled since whenever a point is lost, we randomly select another point from a lower density region and follow the procedure described above.

4.6 MITIGATING NOISE

In real-world scenarios, we must consider the occurrence of noise, e. g. due to measurement's errors or imprecise point tracking, since it may significantly influence the method's overall performance. We propose to mitigate the effects of noise by observing that the original distance matrix $\tilde{\mathbf{D}}$ must be a proper Euclidean distance matrix by construction, according to Equation (4.2). Consequently, $\mathbf{B}$ in Equation (4.5) must be positive definite, i. e. eigenvalues are in $[0, 1]$.

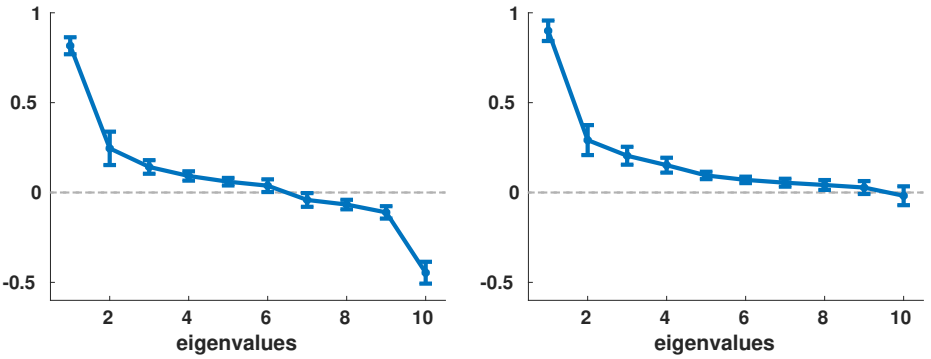
However, in practice, noise may introduce non-significant directions to the representation $\mathbf{Z}$ or even make $\tilde{\mathbf{D}}$ non-metric (Pekalska et al., 2002). Formally, this means that $\mathbf{B}$ will have small and/or negative eigenvalues.

The latter case is easily handled since we know that $\tilde{\mathbf{D}}$ is a proper Euclidean distance matrix, and consequently $\mathbf{B}$ must have non-negative eigenvalues. Therefore, we simply disregard any contribution from negative eigenvalues. The former case however has no immediate solution since the significance of each eigenvalue contribution, even if small, may depend on the particular scene.

We could solely employ the spectrum transformations as described above to obtain a representation for each incoming frame. However, the effects of the noise would still be present and accumulating in the original distance matrix $\tilde{\mathbf{D}}$ possibly reaching a point where the noise contribution would be dominant. The noise effects can be reduced by incorporating the spectrum transformations into the original distance matrix. We have that (Pekalska et al., 2002)

$$\tilde{\mathbf{D}} \odot \tilde{\mathbf{D}} = \mathbf{b}\mathbf{1}^T + \mathbf{1}\mathbf{b}^T - 2\mathbf{B}, \tag{4.13}$$

where $\mathbf{b} \in \mathbb{R}^N$ is a vector containing the diagonal elements of $\mathbf{B}$, as defined in Equation (4.5). Thus, a noiseless distance matrix, denoted as $\tilde{\mathbf{D}}_l$, can be obtained by only considering the contribution of the l leading eigenvalues: first, based on Equation (4.6), $\mathbf{B}_l$ is obtained, where only the l leading eigenvalues and corresponding eigenvectors are considered; then, we compute $\tilde{\mathbf{D}}_l$ according to Equation (4.13), where $\mathbf{B}_l$ is used instead; finally, in the next frame, we consider $\tilde{\mathbf{D}}_l$ in Equation (4.2) instead.



(a) Without incorporation (b) With incorporation

Figure 4.3: Effect of incorporating the spectrum transformation into the original distance matrix. (a) the large negative eigenvalue reflects the significant noise contribution. (b) incorporating the spectrum transformation, the noise contribution is negligible.

Figure 4.3 illustrates the effect of considering the spectrum transformation into the original distance matrix. Without incorporation, the metric nature of the distance matrix $\tilde{\mathbf{D}}$ is degraded, as depicted by the large negative

eigenvalue(s) in Figure 4.3a. This is mitigated if we incorporate the spectrum transformation into the original distance matrix, which can be observed in Figure 4.3b, where only one very small negative eigenvalue is present. In Section 4.7, we experimentally validate this approach to handle noise.

4.7 EXPERIMENTAL EVALUATION

Here, we experimentally evaluate the proposed method's performance on the dataset provided by Tzionas and Gall (2016), to compare with the offline KS estimation method described in Chapter 3, and on two custom recorded sequences, which exhibit partial occlusions.

4.7.1 *Implementation Details*

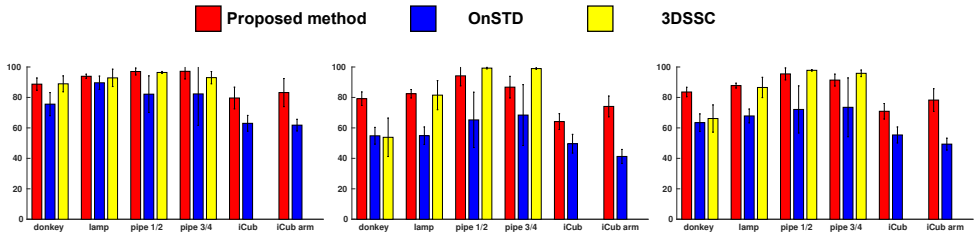
All experiments were performed using a PC with an Intel Core i7-8700k CPU @ 3.7Ghz (x6) and 32GB of RAM, using a C++ implementation. The same parameters were used across all the experiments, except where mentioned otherwise, and were set as follows: $N = 1000$, $\mu = 0.75$, $\gamma = \frac{e^{-2}}{N^2}\sum_{i,j}\tilde{d}_{ij}$, $\alpha = 50$, $\rho_{thresh} = \left\lceil \frac{2}{N}\sum_i \rho_i \right\rceil$, r as the octree resolution, $l = 3$, $\tau = 0.025$ and $\beta = 0.275$. Also, we compare our approach with the method proposed in Chapter 3 and a variant where the similarity is built based on the standard deviation (Perera and Barnes, 2013) in an online fashion, denoted as 3DSSC and OnSTD, respectively. We note that since 3DSSC can not handle occlusion, we did not compare it for the two custom recorded sequences, as they exhibit partial occlusion.

4.7.2 *Quantitative Results*

Five metrics were evaluated: precision P, recall R, f-measure F, number of segments estimated and execution time per frame. The precision, recall and f-measure are given by Equation (3.14), from Section 3.5.2, which try to capture the trade-off between false positives and misses (Ochs et al., 2014). The overall evaluation of these metrics follows the description in Section 3.5.2. Each metric is evaluated for one hundred trials, and for each one the points are randomly sub-sampled.

4.7.2.1 *Precision, Recall and F-Measure*

In Figure 4.4, we show the comparison of precision, recall and f-measure metrics with 3DSSC and OnSTD, in percentage. These results are obtained from the last frame of each sequence. We highlight that, even though our method is online, since the estimation is always only based on current and past information, we achieve comparable results with the offline method 3DSSC, and even outperform for the `donkey` sequence, which by design takes into consideration the entire sequence. The variant OnSTD based on (Perera and Barnes, 2013), where the similarity matrix is built based on the standard deviation of the pairwise distances, achieves the lowest performance. One reason for this might be because this distance gradually stagnates as the number of frames increases. On the contrary, our method does not have such a limitation since the AVPED does not stagnate if motions keep occurring. This means that motions will be segmented, regardless of when they happen, which leads to higher overall performance and not impairing the idea of online motion segmentation.

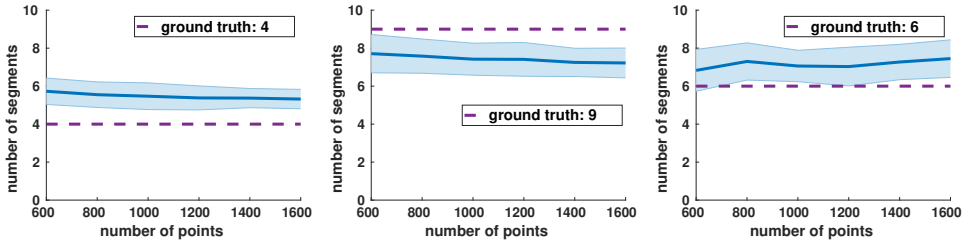


(a) Precision (%) (b) Recall (%) (c) F-Measure (%)

Figure 4.4: Precision, recall and f-measure metrics comparison, in percentage (%). All results are obtained from one hundred trials, where 1000 points are initially sub-sampled and selected randomly. We do not compare the `iCub` and `iCub arm` sequences with 3DSSC, because it does not handle lost points and occlusions.

4.7.2.2 *Number of Segments*

Figure 4.5 shows results obtained for the number of segments estimated in function of the number of points sub-sampled. For some objects, there is a constant bias, which may indicate that either some motions are too subtle and not detected, as seen in the `iCub` sequence in Figure 4.5b, or that severe point tracking errors introduce additional motions, as seen in the `donkey` sequence in Figure 4.5a. We will expand on this when discussing the qualitative results in Section 4.7.3. The number of points considered does not significantly influence the overall number of segments estimation. This



(a) donkey (b) iCub (c) iCub arm

(d) lamp (e) pipe 1/2 (f) pipe 3/4

Figure 4.5: The number of motion segments estimated by the proposed method in function of the number of points sub-sampled.

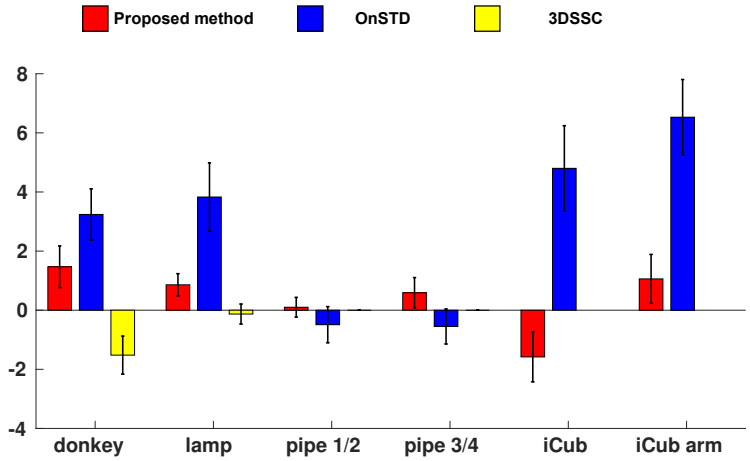


Figure 4.6: Comparison of the number of segments error for 1000 points initially sub-sampled and selected randomly.

means that it is possible to sub-sample fewer points without compromising the overall performance.

In Figure 4.6, we compare the proposed method with two other KS estimation methods regarding number of segments estimation. Overall, we see that the proposed method achieves better performance than OnSTD and comparable performance with the offline method 3DSSC. We note that the errors of 3DSSC for the iCub and iCub arm sequences are not plotted since these sequences exhibit some occlusion, which 3DSSC does not handle.

4.7.2.3 *Computational Time*

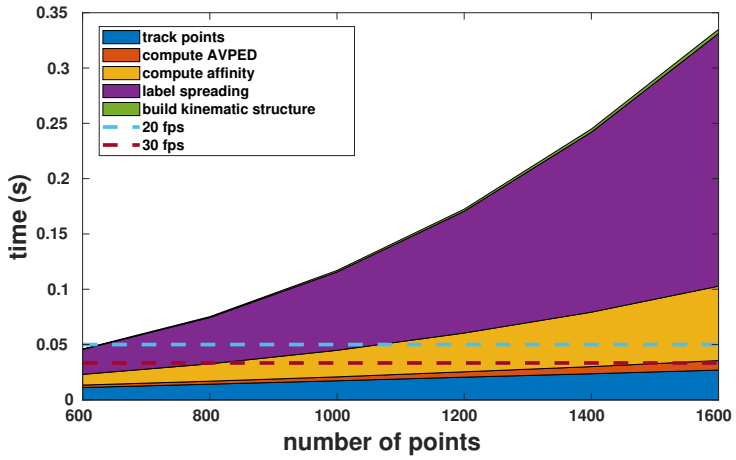
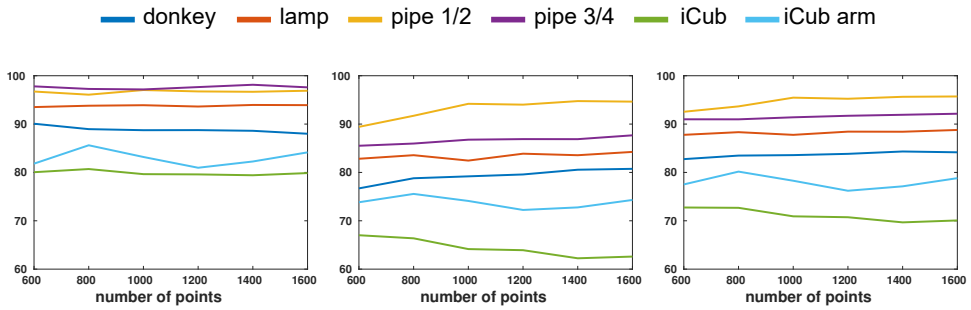


Figure 4.7: Cumulative computational time per frame in function of the number of points sub-sampled. Tracking points and computing the AVPED can be done in real-time, considering up to around 1400 points. Note that the time spent handling lost points and noise is also included, respectively.

Figure 4.7 presents the average cumulative computational time per frame that is required by the proposed method. We can observe that the AVPED can be maintained in real-time, considering up to around 1400 points (we also have to include the time spent tracking the points). This is the only computation required for each incoming frame since the AVPED is responsible for maintaining information across the entire scene. The rest of the computation, i. e. affinity, *label spreading* and KS estimation, can be done on demand and/or in parallel, not necessarily for each frame, depending on the computational resources, since they do not influence the information that is preserved by AVPED. The proposed method is thus suitable for real-time applications.

4.7.2.4 *Influence of the Number of Points Sub-Sampled*

The input point cloud tends to be better represented by a larger number of points, but the corresponding computational cost increases, as illustrated in Figure 4.7. In practice, we must reach a compromise and based on Figure 4.8 we can consider a lower number of points without significant performance degradation depending on the application. Also, based on Figure 4.5, the number of segments estimated is not significantly influenced by the number of points. Thus, we recommend selecting a number of points between 1000 and 1500 for most applications.

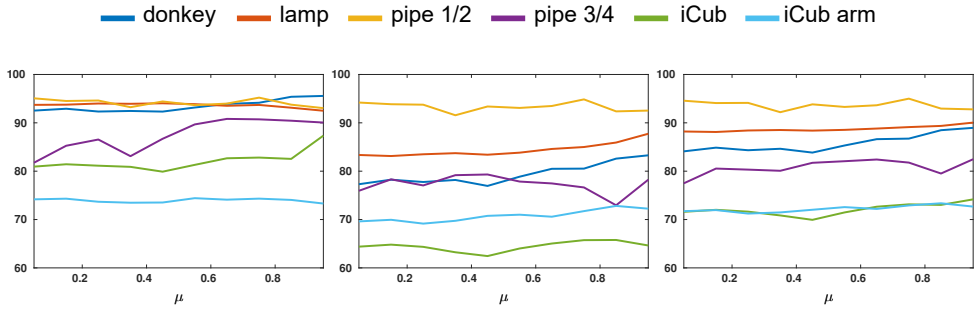


(a) Average precision (%) (b) Average recall (%) (c) Average f-measure (%)

Figure 4.8: (From left to right) Average precision, recall and f-measure in function of the number of points sub-sampled, respectively.

4.7.2.5 *Influence of the μ Parameter*

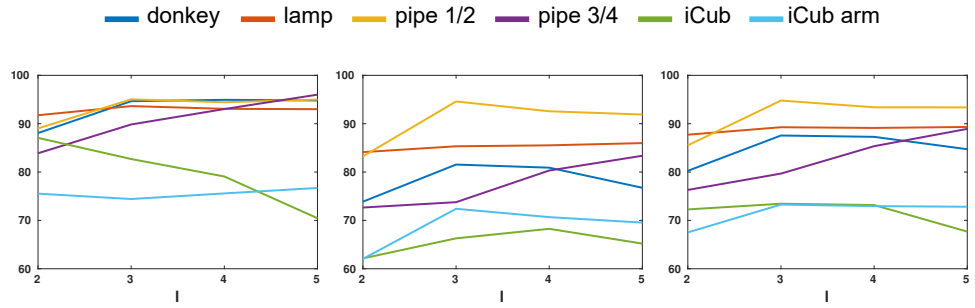
The $\mu \in [0, 1)$ parameter controls the contribution of the graph structure for estimating future labels, according to Equation (4.9). If $\mu \to 1$, the graph structure is the dominant factor, whereas if $\mu \to 0$, future labels are more influenced by previous ones. We want to find a suitable value for μ that accounts for new information while also considering information from previous labels. Figure 4.9 shows the average results obtained, in terms of precision, recall and f-measure. We observe that higher μ values yield slightly better average results. This corroborates the fact that the AVPED can maintain current and past information of the scene. Moreover, the results of lower μ values depend significantly on the initial point cloud distribution and corresponding point tracking errors, which is not desirable for online and real-time applications. Based on the results from Figure 4.9, we selected $\mu = 0.75$ as a balanced compromise between the contribution of the graph structure and previous labelled points.



(a) Average precision (%) (b) Average recall (%) (c) Average f-measure (%)

Figure 4.9: (From left to right) Average precision, recall and f-measure in function of the μ parameter, respectively.

4.7.2.6 *Influence of the Number of Leading Eigenvalues*



(a) Average precision (%) (b) Average recall (%) (c) Average f-measure (%)

Figure 4.10: (From left to right) Average precision, recall and f-measure in function of considering l leading eigenvalues.

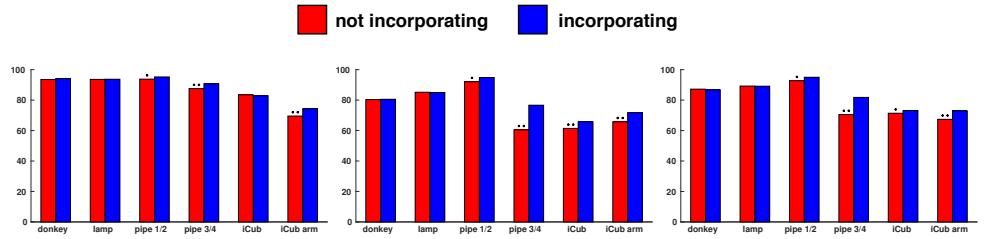
Figure 4.10 shows the influence of the number of leading eigenvalues considered to build the similarity matrix, according to Equation (4.8). We observe that, aside from `pipe 3/4` and `iCub` sequences, the performance is not significantly improved considering more than three eigenvalues. For a higher number of eigenvalues considered, the average performance is only improved for the `pipe 3/4` sequence, and it decreases for the `iCub` sequence. We thus selected the number of leading eigenvalues to be 3.

4.7.2.7 *Influence of Incorporating the Spectrum Transformation*

Due to measurement errors or faulty point tracking, noise may be introduced into the AVPED. Since the AVPED is a proper distance matrix, we cast the problem of handling noise into embedding the AVPED in a feature space and use the resultant representation to build the similarities (as described in Section 4.2, while also incorporating the spectrum transformation into the original distance matrix $\tilde{\mathbf{D}}$ (as described in Section 4.6). Figure 4.11 illustrates the benefits of incorporating this spectrum transformation process, for which there are statistically significant gains across most sequences. Also, the results are most significant when the sequences are longer and/or exhibit more noise and lost points, e. g. `iCub` and `iCubArm` sequences. Since these sequences reflect real-world scenarios more accurately, we advocate for the incorporation of spectrum transformation into AVPED.

4.7.2.8 *Consistency Analysis in Sequential Estimation*

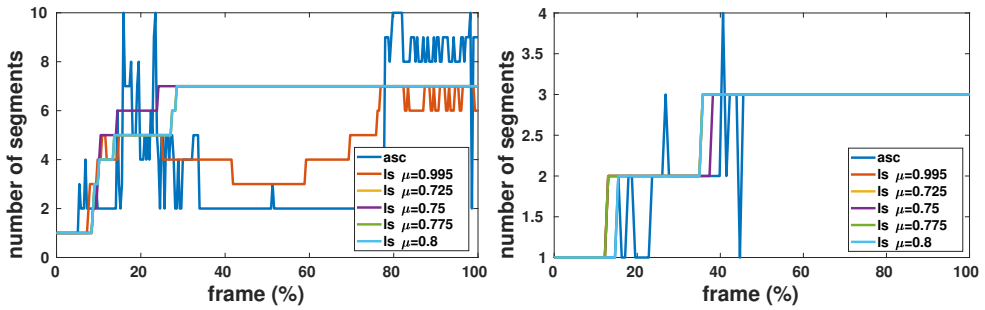
The proposed method provides sequential consistency in two ways: by construction from Equation (4.2), and by clustering via *label spreading*. An analysis



(a) Average precision (%) (b) Average recall (%) (c) Average f-measure (%)

Figure 4.11: (From left to right) Average precision, recall and f-measure in function of incorporating the spectrum transformation, respectively. * and ** represent statistical significance with p-value<0.05 and p-value<0.005, respectively.

of the number of segments estimated throughout the `iCub` and `lamp` sequences compared to clustering via Self-Tuning Spectral Clustering (STSC) is provided in Figure 4.12. It can be seen that the *label spreading* provides additional consistency to the overall method's performance. In particular, the method's consistency is empirically robust to neighbouring values of $\mu = 0.75$, further corroborating its choice.



(a) `iCub` (b) `lamp`

Figure 4.12: Number of segments estimated using *label spreading* (ls) and adaptive spectral clustering (asc) for the `iCub` and `lamp` sequences. We see that the number of segments estimated by applying STSC at each frame is not very consistent. Applying *label spreading* with $\mu = 0.995$ is also not very consistent compared to neighbouring values of $\mu = 0.75$.

4.7.3 *Qualitative Results*

Figure 4.13 presents some qualitative results for the last frame of each sequence. We can see that the proposed method produces plausible KSs compared to the ground truth while being online and running in real-time. The results obtained for the variant OnSTD based on (Perera and Barnes, 2013) seem to corroborate the fact that the similarity built from the standard devi-

ation of the pairwise distances can not accommodate for motions that appear later in the sequence, e. g. the head of the `donkey` is the last part to move and it is not segmented. Furthermore, the results seem to indicate that the OnSTD variant yields a noisier segmentation.

It is also worth mentioning that the bias exhibited for some sequences regarding the number of motions segmented presented in Figure 4.5 can now be better understood. For instance, the number of segments of `iCub` is consistently underestimated by around two segments; however, we can see from Figure 4.13 that the three segments of the left arm are estimated as being one segment, which explains the bias. One possible reason might be that the motions exposed are not significant and/or too subtle, and the method can not distinguish them. Even in these cases, the KSs estimated are plausible.

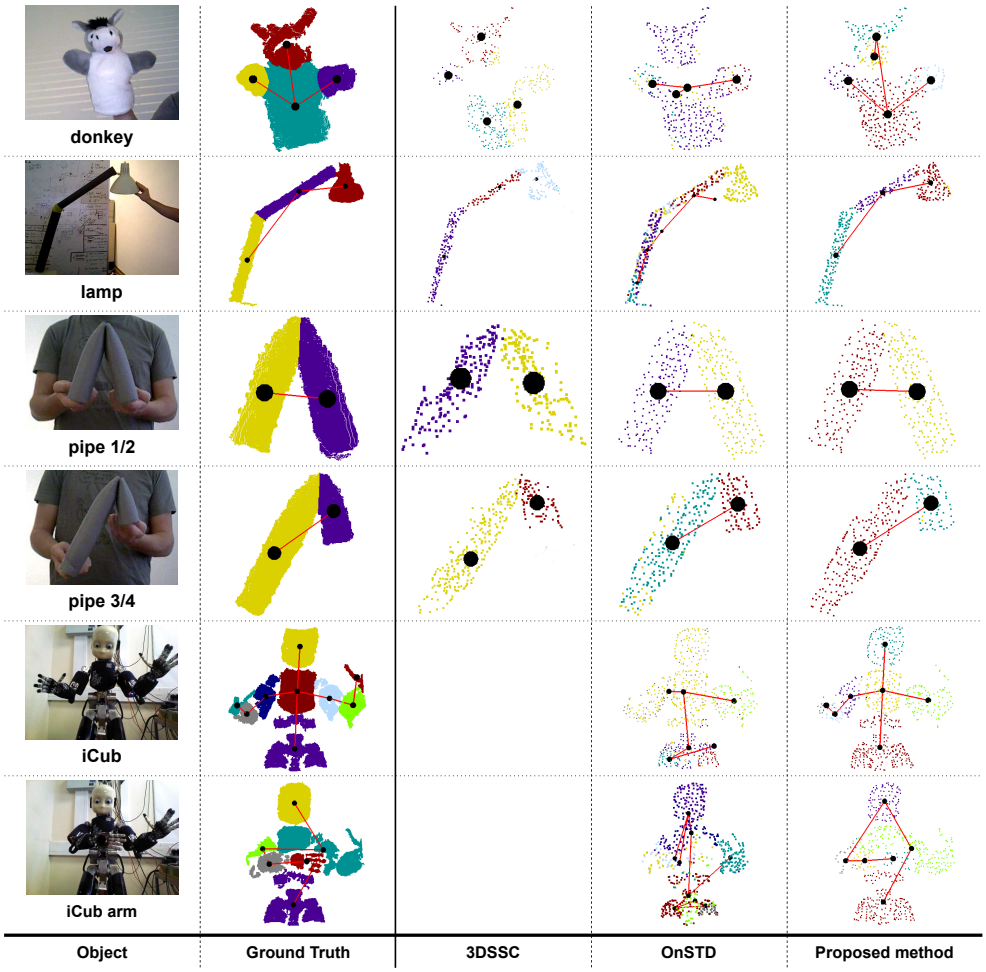


Figure 4.13: Qualitative results. These results shown are obtained for the last frame of each sequence. The proposed method learns plausible 3D KSs of the objects. Figure best viewed in colour.

4.8 CONCLUSION

An unsupervised 3D online KS learning method was proposed in this chapter, which can naturally handle noise and lost points. To the best of our knowledge, this is the first method that estimates arbitrary 3D KSs in an online fashion and in real-time. The experimental results show that its performance is comparable to offline methods, and the KSs produced are plausible and compact representations of the structure of each object.

We envision our method to be applied in real-world scenarios, such as HRI scenarios. Contrary to offline methods that can only process a batch at a time, e. g. the one proposed in Chapter 3, the proposed method can run continuously, which is crucial in enabling fluid HRI. For instance, enabling a humanoid robot to adaptively learn kinematic properties of surrounding objects, either by demonstration or exploration, for better object manipulation and grasping.

However, point tracking is still an open problem in the computer vision field, e. g. due to large displacements and occlusions. Even though we devised an approach that mitigates the impact of lost points and noise, this is a component that can still be improved. Thus, in the next chapters, we will be bridging this aspect by not explicitly associating points in time. To achieve this, we will consider event cameras, which, as discussed in Section 2.1, output asynchronous signals (i. e. events) that are generated based on observed intensity changes. If we assume that the intensity changes are mainly due to motion, in principle, we could retrieve the underlying motion parameters without explicitly having to establish correspondences between points. The next chapter will explore this insight.

Contributions. The main contributions presented in this chapter have been published in:

Nunes, U. M. and Demiris, Y. (2019), Online Unsupervised Learning of the 3D Kinematic Structure of Arbitrary Rigid Bodies, *in* 'IEEE/CVF International Conference on Computer Vision (ICCV)', pp. 3809–3817. doi: 10.1109/ICCV.2019.00391. (Nunes and Demiris, 2019)

5

EVENT-BASED VISION MOTION ESTIMATION BY DISPERSION MINIMISATION

This chapter begins to address the third research question, focusing on the first part:

> How can event cameras estimate general motion, and can this be achieved incrementally (i. e. processing one event at a time, instead of a batch of events)?

In this chapter, we focus our attention on motion estimation for event cameras, whose events can be augmented with additional information from other sensors, e. g. depth from RGB-D. We describe a general event-based framework, denoted Dispersion Minimisation (DMin), which will be a crucial component in performing motion segmentation for the next chapter. The framework extends previous event-based motion compensation algorithms (Gallego and Scaramuzza, 2017; Gallego et al., 2018b; Mitrokhin et al., 2018) by avoiding computing an optimisation score based on an explicit image-based representation, which provides three main benefits: 1) The framework can be extended to perform incremental estimation, i. e. on an event-by-event basis. 2) Besides purely visual transformations in 2D, the framework can readily use additional information, e. g. by augmenting the events with depth, to estimate the parameters of motion models in higher dimensional spaces. 3) The optimisation complexity only depends on the number of events. We assume that events are caused by moving edges on the scene and that by aligning events of the same scene edge, the underlying motion can be recovered. We achieve this by modelling the event alignment according to candidate motion parameters and minimising the resultant dispersion, which is computed by a family of suitable entropy-based measures. Data whitening is also proposed as a simple and effective pre-processing step to make the framework's accuracy performance more robust, as well as other event-based motion-compensation methods. Figure 5.1 provides an overview of the proposed framework.

In Section 5.1, we describe the DMin framework and provide a family of suitable entropy-based measures. In Section 5.2, we describe an Approximate Dispersion Minimisation (ADMin) approach, which is an efficient approximation whose complexity is only linear with the number of events (e. g. the complexity does not depend on the number of image pixels). Section 5.3

shows how the DMin framework can be extended to incrementally estimate motion on an event-by-event basis and achieve real-time processing. In Section 5.4, we describe a simple pre-processing step based on data whitening that increases the robustness of event-based motion compensation methods. The DMin framework's performance is evaluated on several challenging sequences using distinct motion models in Section 5.5. Finally, Section 5.6 summarises and concludes the chapter.

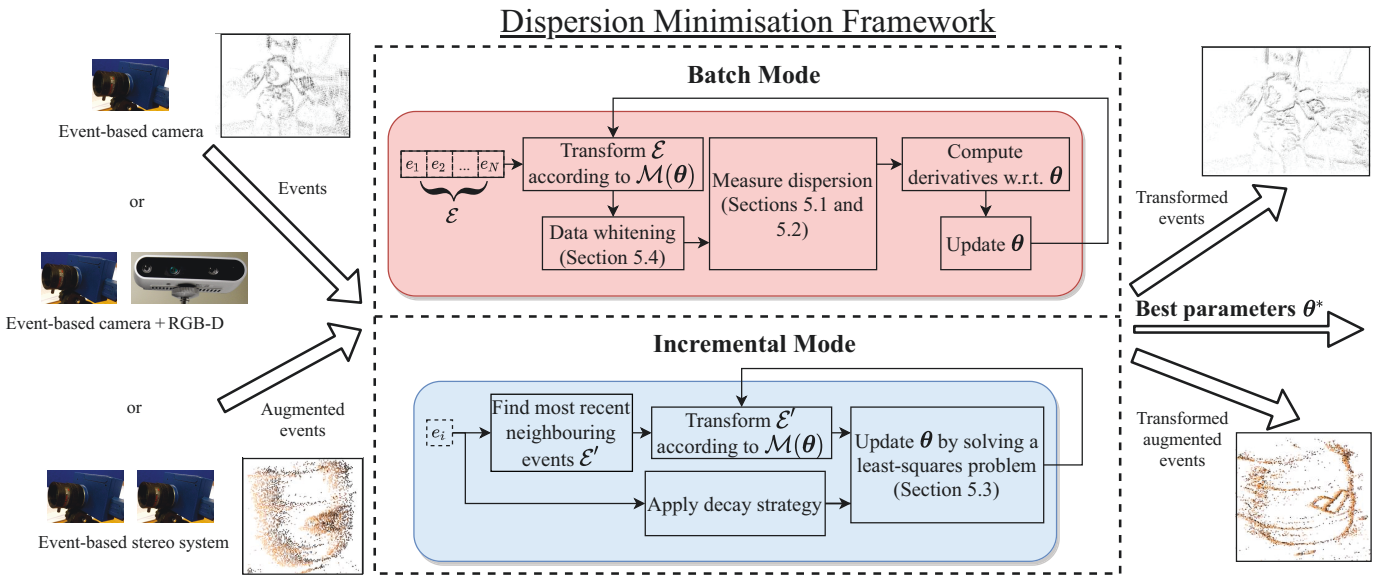


Figure 5.1: DMin framework overview. The DMin framework estimates the best model parameters that minimise the events' dispersion. The input may consist of only events or augmented events, e. g. with depth information. We propose a *batch mode*, whereby dispersion minimisation is achieved by transforming a batch of events according to candidate motion parameters and evaluating a dispersion measure. We also propose an *incremental mode*, whereby the motion parameters are estimated on an event-by-event basis. The DMin framework does not rely on any intermediate representation, e. g. Image of Warped Events (IWE). The images shown are formed by accumulating the events projected onto the image plane and are only included to aid visualisation.

5.1 DISPERSION MINIMISATION FRAMEWORK

The core idea sustaining the proposed framework relies on the assumption that events are generated according to some underlying *generation model*. For example, an event-based camera moving triggers events according to the observed scene texture and the motion $\mathcal{M}(\theta)$ it undergoes, which is illustrated in Figure 5.2. The generated events are thus dispersed according to the scene texture and motion. This observation is in line with previous works on

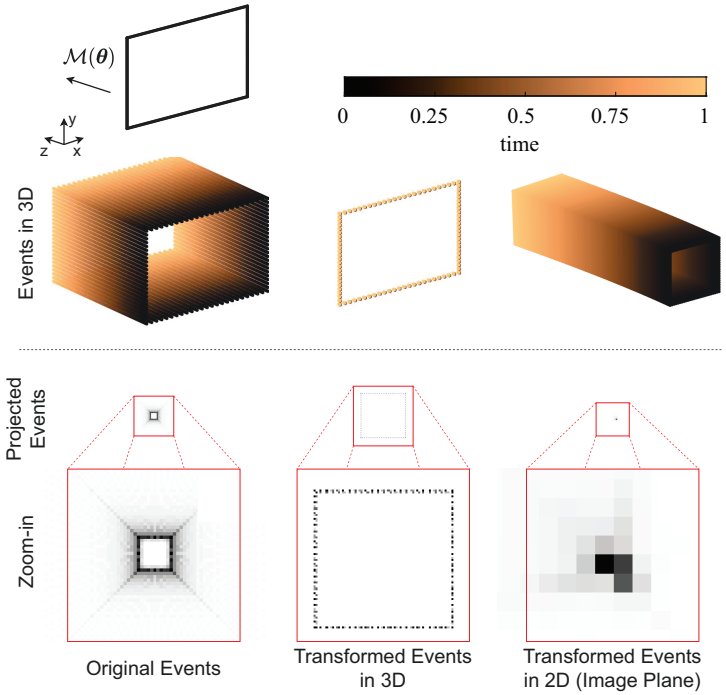


Figure 5.2: Consider a square moving with a constant linear velocity in the z-direction, i. e. moving perpendicularly away from the camera, and the corresponding events generated, as depicted on the left column. From top to bottom, we show the events generated in 3D space, e. g. which can be obtained by augmenting the events with depth information, the corresponding events projected onto the image plane, and zoom-in for better visualisation. In the middle column, we show the transformed depth-augmented events according to the motion parameters estimated using the proposed framework directly in 3D. In the right column, we show the transformed depth-augmented events according to the motion parameters estimated using the Contrast Maximisation (CMax) framework (Gallego et al., 2018b), which projects the depth-augmented events onto the IWE to then compute its variance for the optimisation procedure. The proposed framework can accurately estimate the 6-Degrees of Freedom (DOF) transformation model parameters, whereas the CMax framework can not.

event-based motion estimation (Gallego and Scaramuzza, 2017; Gallego et al., 2018b; Kim et al., 2016a), which also assume that events are caused by moving edges on the scene. By locally aligning neighbouring events associated to the same edge, we can recover the apparent motion that triggered them. We exploit this insight by considering measures of events' dispersion and then minimising them according to the motion model parameters. Previous event-based motion compensation methods (Gallego and Scaramuzza, 2017; Gallego et al., 2018b; Mitrokhin et al., 2018) also exploit this insight: they maximise a contrast measure of the resultant IWE, while also revealing a

strong edge structure. Distinctively, however, we can measure the events' dispersion without relying on any intermediate image-based representation, which allows us to consider models whose outputs have more than two dimensions, e. g. events augmented with depth information. As illustrated in Figure 5.2, performing the optimisation directly in 3D space has the main benefit of avoiding possible degenerate solutions since, for example, the CMax framework (Gallego et al., 2018b) may estimate motion parameters that push the projected events onto a point/line (Zhu et al., 2019).

We consider two types of dispersion measures: pairwise potential (Section 5.1.1), which measures the similarity between events, and entropy-based (Section 5.1.2), which measures the dispersion of the events' distribution.

5.1.1 *Pairwise Potential*

Events are optimally modelled if they become more concentrated (on the edges), which minimises their average pairwise distance (e. g. the average distance is zero if all events are transformed onto the same point). The similarity between two events can be measured according to the d-dimensional multivariate Gaussian kernel parameterised by the covariance matrix $\mathbf{\Sigma}$:

$$\mathcal{K}_{\mathbf{\Sigma}}(\mathbf{f}_i, \mathbf{f}_j; \theta) = \frac{\exp\left(-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^T \mathbf{\Sigma}^{-1}(\mathbf{f}_i - \mathbf{f}_j)\right)}{(2\pi)^{\frac{d}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}}}, \tag{5.1}$$

where $\mathbf{f}_i$ represents a d-dimensional feature associated with event i (e. g. transformed event's spatial coordinates), which is transformed according to a model $\mathcal{M}$ and candidate parameters θ (please refer to Appendix C for possible motion models), according to

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta). \tag{5.2}$$

The pairwise potential measure, hereinafter designated by *Potential* energy, can be defined by:

$$P(\mathbf{f}; \theta) \coloneqq -\sum_{i,j}^{N_e} \mathcal{K}_{\mathbf{\Sigma}}(\mathbf{f}_i, \mathbf{f}_j; \theta). \tag{5.3}$$

The *Potential* energy measures the average similarity between all pairs of events. Another interpretation for this equation is related to a simplified version of pairwise potentials from fully connected Conditional Random Field models (Lafferty et al., 2001), whereby events are considered as nodes in a graph.

5.1.2 *Entropy-based Measures*

Entropy measures the dispersion of a distribution. For ease exposition, we start by considering the expression of the well-known *Shannon* entropy of a random variable x with probability density function (pdf) given by $p(x)$:

$$S(p) = -\int p(x) \log p(x) dx. \tag{5.4}$$

For discrete random variables, the integral is substituted by a summation. The Shannon entropy is just a particular case of a larger parametric family of entropies, namely *Sharma-Mittal*, *Rényi* and *Tsallis* (Sharma and Mittal, 1975).

Since we do not assume any prior event-based distribution model, we consider that the distribution model can be approximated by the aggregation of each event inducing a certain distribution in the feature space. The distribution model is thus only defined over the events in the batch $\mathcal{E}$ (i. e. the random variable domain). For simplicity, we assume that each event e_j induces a Gaussian distribution centred at the event and is parameterised by the covariance matrix $\mathbf{\Sigma}$. The respective pdf is thus given by

$$p(\mathbf{f}_i, \mathbf{f}_j; \theta) = \frac{1}{Z} \mathcal{K}_{\mathbf{\Sigma}}(\mathbf{f}_i, \mathbf{f}_j; \theta), \; Z = \sum_{i,j}^{N_e} \mathcal{K}_{\mathbf{\Sigma}}(\mathbf{f}_i, \mathbf{f}_j; \theta) \tag{5.5}$$

where Z is a normalisation factor, such that the probabilities sum up to 1 (i. e. $\sum_{i,j}^{N_e} p(\mathbf{f}_i, \mathbf{f}_j; \theta) = 1$).

Substituting the pdf given by Equation (5.5) into the *Shannon* entropy given by Equation (5.4), we have

$$S(p) = -\sum_{i,j}^{N_e} p(\mathbf{f}_i, \mathbf{f}_j; \theta) \log p(\mathbf{f}_i, \mathbf{f}_j; \theta), \tag{5.6}$$

which is given considering all the events in the batch.

We can already use Equation (5.6) to minimise the events' dispersion. However, in practice and in the context of the present work, we empirically

advocate using a modified expression by removing the normalisation factor Z:

$$\begin{aligned} S(p) &= -\sum_{i,j}^{N_e} p(\mathbf{f}_i, \mathbf{f}_j; \theta) \log p(\mathbf{f}_i, \mathbf{f}_j; \theta) \\ &= -\sum_{i,j}^{N_e} \frac{1}{Z} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta) \log \left(\frac{1}{Z} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta) \right) \\ &= -\frac{1}{Z} \sum_{i,j}^{N_e} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta) \left(\log \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta) - \log Z \right) \\ &\propto \sum_{i,j}^{N_e} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta) \log \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta) =: S(\mathbf{f}; \theta). \end{aligned} \tag{5.7}$$

Following the steps above, we can reach the following expressions for the *Sharma-Mittal*, *Rényi* and *Tsallis* (Sharma and Mittal, 1975) entropy-based measures, respectively:

$$H_{\alpha,\beta}(p) := \frac{1}{1-\beta} \left[\left(\sum_{i,j}^{N_e} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta)^{\alpha} \right)^{\gamma} - 1 \right], \tag{5.8}$$

$$R_{\alpha}(\mathbf{f}; \theta) := \frac{1}{1-\alpha} \log \left(\sum_{i,j}^{N_e} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta)^{\alpha} \right), \tag{5.9}$$

$$T_{\alpha}(\mathbf{f}; \theta) := \frac{1}{1-\alpha} \left(\sum_{i,j}^{N_e} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j; \theta)^{\alpha} - 1 \right). \tag{5.10}$$

The construction presented is not unique. For example, we could also consider the case by which the pdf is given as a sum of all the Gaussians induced by the events, i. e. $p(\mathbf{f}_i, \mathbf{f}_j; \theta) = \frac{1}{Z} \sum_j^{N_e} \mathcal{K}_{\Sigma}(\mathbf{f}_i, \mathbf{f}_j, \theta)$. Another case to consider could be that the pdf is instead given by some known model of the shape that is generating the events (Lagorce et al., 2015). We leave these lines for future work.

5.1.3 *Optimisation*

The flow of the proposed framework is summarised in Algorithm 5.1. A set of events $\mathcal{E}$ is transformed according to candidate motion parameters

Algorithm 5.1 DMin framework - Batch mode

Input: Set of events $\mathcal{E} = \{e_k\}_k^{N_e}$.
Output: Estimated model parameters θ^*.
Procedure:

1: Initialise the candidate model parameters θ.
2: Transform events according to parameters θ, Equation (5.2).
3: Compute the dispersion $D(\mathbf{f}; \theta)$ (Sections 5.1 and 5.2).
4: Find the best parameters θ^* by minimising the dispersion $f(\theta)$, Equation (5.11).

θ, given by Equation (5.2). We then find the best motion parameters θ^*, by minimising the dispersion in the feature space

$$\theta^* = \arg\min_{\theta} f(\theta) \equiv \arg\min_{\theta} D(\mathbf{f}; \theta), \tag{5.11}$$

where $D(\mathbf{f}; \theta)$ is one of the proposed dispersion measures.

5.2 EFFICIENT APPROXIMATION

The complexity of the proposed approach is quadratic with the number of events. It is thus more computationally demanding than previous frameworks (Gallego and Scaramuzza, 2017; Gallego et al., 2018b), which are linear with the number of events and the number of pixels of the IWE. To overcome the increased complexity, we propose an efficient approximation to compute the dispersion functions, as explained next.

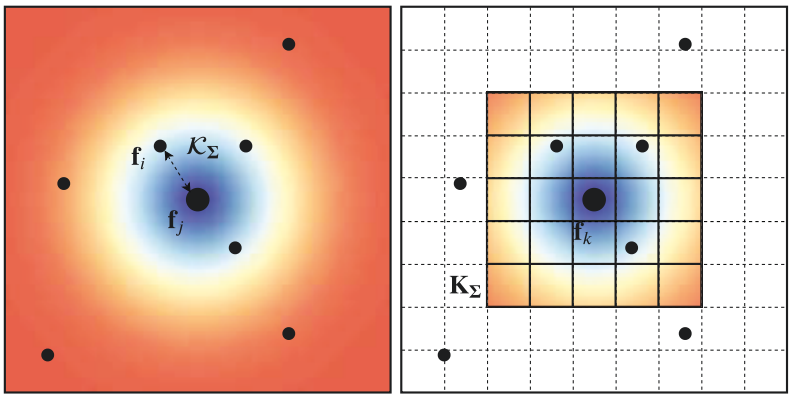


Figure 5.3: (Left) In the DMin approach, all features pairwise distances are computed. (Right) In the ADMin approach, the feature space is discretised and each feature $\mathbf{f}_k$ is convolved with a truncated kernel $\mathbf{K}_\Sigma$.

This is achieved by approximating the kernel $\mathcal{K}_\Sigma$ given by Equation (5.1) with a truncated version $\mathbf{K}_\Sigma$, where values beyond certain standard deviations

are set to zero, and then each feature vector is convolved with $\mathbf{K}_{\boldsymbol{\Sigma}}$. This idea is illustrated in Figure 5.3. To achieve linear complexity with the number of events, each feature vector is asynchronously convolved using the event-based convolution method proposed by Scheerlinck et al. (2019).

The *Approximate Potential* energy can be expressed as

$$\tilde{P}(\mathbf{f};\theta) \coloneqq -\sum_{i,j}^{N_e} \mathbf{K}_{\boldsymbol{\Sigma}} * \delta_{\mathbf{f}_i}(\mathbf{f}_j), \tag{5.12}$$

where each feature $\mathbf{f}_i$ is convolved with the truncated kernel $\mathbf{K}_{\boldsymbol{\Sigma}}$ in the feature space. $\delta_{\mathbf{x}_i}(\mathbf{x})$ represents the Kronecker delta function, where $\delta_{\mathbf{x}_k}(\mathbf{x}) = 1$ when $\mathbf{x} = \mathbf{x}_k$ and 0 otherwise.

Similarly, the *Sharma-Mittal*, *Rényi* and *Tsallis* entropy-based measures can be approximately expressed based on the kernel $\mathbf{K}_{\boldsymbol{\Sigma}}^{\alpha}$, and the *Shannon* entropy-based measure can be approximately expressed based on the kernel $\mathbf{K}_{\boldsymbol{\Sigma}} \odot \log \mathbf{K}_{\boldsymbol{\Sigma}}$:

$$\tilde{H}_{\alpha,\beta}(\mathbf{f};\theta) \coloneqq \frac{1}{1-\beta}\left[\left(\sum_{i,j}^{N_e} \mathbf{K}_{\boldsymbol{\Sigma}}^{\alpha} * \delta_{\mathbf{f}_i}(\mathbf{f}_j)\right)^{\gamma} - 1\right], \tag{5.13}$$

$$\tilde{R}_{\alpha}(\mathbf{f};\theta) \coloneqq \frac{1}{1-\alpha}\log\left(\sum_{i,j}^{N_e} \mathbf{K}_{\boldsymbol{\Sigma}}^{\alpha} * \delta_{\mathbf{f}_i}(\mathbf{f}_j)\right), \tag{5.14}$$

$$\tilde{T}_{\alpha}(\mathbf{f};\theta) \coloneqq \frac{1}{1-\alpha}\left(\sum_{i,j}^{N_e} \mathbf{K}_{\boldsymbol{\Sigma}}^{\alpha} * \delta_{\mathbf{f}_i}(\mathbf{f}_j) - 1\right), \tag{5.15}$$

$$\tilde{S}(\mathbf{f};\theta) \coloneqq \sum_{i,j}^{N_e} \left(\mathbf{K}_{\boldsymbol{\Sigma}} \odot \log \mathbf{K}_{\boldsymbol{\Sigma}}\right) * \delta_{\mathbf{f}_i}(\mathbf{f}_j), \tag{5.16}$$

where $\odot$ represents the Hadamard product and log represents the natural logarithm applied element-wise.

For simplicity, assuming the kernel $\mathbf{K}_{\boldsymbol{\Sigma}}$ has size κ^d, evaluating the approximate functions has a complexity of $O(N_e\kappa^d)$, which is linear with the number of events. The complexity may still be exponential with the number of dimensions if we do not consider efficient high-dimensional convolution operations that reduce the computational complexity to become linear with the number of dimensions (Adams et al., 2010). Although we need to discretise the feature space, the computational complexity of the ADMin approach is independent of the actual number of discretised bins. In contrast, the complexity of previous event-based motion compensation methods (Gallego and Scaramuzza, 2017; Gallego et al., 2018b; Mitrokhin et al., 2018) is also linear

with the number of discretised bins (e. g. number of image pixels N_p). This means that although directly extending their framework to handle higher dimensions is possible, in practice it would be inefficient and not scalable without considering sophisticated data structures.

5.3 INCREMENTAL EXTENSION

The DMin framework can also be extended to work online, i. e. on an event-by-event basis. This means that we can incrementally estimate the model parameters for each incoming event instead of considering batches of events. This approach has three main benefits: 1) the estimation is available for each event, 2) the constancy assumption can be relaxed, and 3) the method's latency is highly reduced since it does not require waiting for a batch of events. For ease exposition, we will consider the extension for the *Potential* energy given by Equation (5.3), since a similar derivation can be considered for the rest of the entropy-based measures.

We begin by observing that the *Potential* energy is (locally) minimised when its derivative w. r. t. the model parameters θ is zero:

$$\frac{\partial P(\mathbf{f};\theta)}{\partial \theta} = -\sum_{i,j}^{N_e} \underbrace{w_{i,j} \frac{\partial \Delta \mathbf{f}_{i,j}^T}{\partial \theta} \boldsymbol{\Sigma}^{-1}}_{\mathbf{W}_{i,j}} \Delta \mathbf{f}_{i,j} = \mathbf{0}, \tag{5.17}$$

where $w_{i,j} = \mathcal{K}_{\boldsymbol{\Sigma}}(\mathbf{f}_i, \mathbf{f}_j; \theta)$ and $\Delta \mathbf{f}_{i,j} = \mathbf{f}_i - \mathbf{f}_j$. By expanding $\mathbf{f}_i$, according to Equation (5.2), such that it linearly depends on the model parameters θ, we have

$$\begin{aligned}
\mathbf{f}_i = \mathcal{M}(e_i;\theta) &= \mathcal{M}(e_i;\theta) - \left[\mathbf{G}(t_i;\theta) - \mathbf{G}(t_i;\theta)\right] \mathbf{y}_i && (5.18)\\
&= \mathcal{M}(e_i;\theta) - \mathbf{G}(t_i;\theta)\mathbf{y}_i + \mathbf{G}(t_i;\theta)\mathbf{y}_i && (5.19)\\
&= \tilde{\mathcal{M}}(e_i;\theta) + \mathbf{G}(t_i;\theta)\mathbf{y}_i && (5.20)\\
&= \tilde{\mathbf{f}}_i + \mathbf{Y}_i \theta, && (5.21)
\end{aligned}$$

where $\mathbf{y}_i$ represents a feature associated with event i before any transformation (e. g. events' spatial coordinates). Plugging Equation (5.21) into Equation (5.17) and rearranging the terms, we obtain

$$\underbrace{-\sum_{i,j}^{N_e} \mathbf{W}_{i,j} \Delta\tilde{\mathbf{f}}_{i,j}}_{\psi} = \underbrace{\sum_{i,j}^{N_e} \mathbf{W}_{i,j} \left(\mathbf{Y}_i - \mathbf{Y}_j\right) \theta}_{\Psi}$$
$$\implies \theta^* = \Psi^{-1}\psi \tag{5.22}$$

which can be iteratively solved by linear least-squares. Equation (5.22) can be solved exactly, i. e. for all pairs of events, or approximately, applying the efficient approximation described in Section 5.2, considering batches of events.

Algorithm 5.2 DMin framework - Incremental mode

Input: Stream of events $\mathcal{E} = \{e_1, e_2, ..., e_k, ...\}$.
Output: Estimated model parameters θ^*.
Procedure:

1: Initialise the candidate model parameters θ.
2: **for** each event e_k **do**
3: Transform events according to parameters θ, Equation (5.2).
4: Apply decay strategy to Ψ and ψ.
5: Find the best parameters θ^* by iteratively solving Equation (5.22).
6: Update Ψ and ψ.
7: **end for**

Equation (5.22) can also be solved incrementally as new events arrive, by maintaining and updating the components ψ and Ψ separately and applying some strategy to give more importance to recent events (e. g. temporal window, decay factor). The online extension is summarised in Algorithm 5.2.

Please refer to Appendix D.8 for details on how to use the proposed online extension and to Appendix C on how to adapt Equation (5.21) to specific motion models (e. g. similarity transformation, 3D rotational motion, 6-DOF motion).

5.4 ROBUST ESTIMATION BY DATA WHITENING

The DMin framework estimates the motion parameters by minimising the dispersion of events. In the limit, the dispersion is minimised if all events are mapped onto the same point, which in practice can occur, e. g. by estimating up-to scale 3D linear velocities. Figure 5.4 illustrates this situation, where we

show a projection of the accumulated original events and transformed events according to the estimated parameters. Previous frameworks (Gallego and Scaramuzza, 2017; Gallego et al., 2018b) also exhibit a similar limitation since the contrast measure of the IWE is maximised if all events are warped onto a line.



(a) (b) (c)

Figure 5.4: (a) Original events accumulated onto the image plane. Respective transformed events according to the best estimated parameters (b) without data whitening and (c) with data whitening. The spatial covariance of the original events is similar to the covariance of the respective transformed events by applying data whitening.

To cope with this limitation, we propose to pre-process the events by whitening them before computing any of the measures above. Data whitening is a standard technique to decorrelate the data dimensions from one another (Koivunen and Kostinski, 1999). For our purpose, we are interested in enforcing that the *layout* of the events' distribution remains *similar* before and after the transformations (which does not occur in failure cases, as shown in Figure 5.4). Since we do not assume any prior distribution of the events, we consider the covariance of the associated features as a measure of the *layout*, which should remain more or less constant throughout the optimisation procedure. We can enforce this by whitening the transformed features, which restricts the optimisation procedure to search for model parameters that do not significantly alter the underlying events' *layout*. This step can thus be seen as a soft constraint on the dispersion measures, and it can also be employed by other event-based motion compensation methods (Gallego et al., 2018b, 2019).

We adopted ZCA whitening, which we briefly review next. Grouping the d-dimensional feature vectors $\mathbf{f}_i$ into a $d \times N_e$ matrix $\mathbf{F} = \begin{pmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_{N_e} \end{pmatrix}$, we first remove its mean, obtaining the zero mean matrix $\bar{\mathbf{F}}$. The covariance matrix is thus given by

$$\mathbf{\Sigma}_{\mathbf{F}} = \frac{1}{N_e - 1} \bar{\mathbf{F}} \bar{\mathbf{F}}^\top. \tag{5.23}$$

The whitened matrix $\mathbf{F_W}$ is obtained from $\bar{\mathbf{F}}$ by a linear transformation $\mathbf{W}$, such that

$$\begin{aligned} \mathbf{I} &= \frac{1}{N_e - 1}\mathbf{F_W}\mathbf{F_W}^\top = \frac{1}{N_e - 1}\mathbf{W}\bar{\mathbf{F}}\bar{\mathbf{F}}^\top\mathbf{W}^\top = \mathbf{W}\boldsymbol{\Sigma}_\mathbf{F}\mathbf{W}^\top \\ &\implies \mathbf{W} = \boldsymbol{\Sigma}_\mathbf{F}^{-\frac{1}{2}} \end{aligned} \tag{5.24}$$

The whitening matrix $\mathbf{W}$ can be computed by performing singular value decomposition $\mathbf{W} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}\mathbf{U}^\top$.

5.5 EXPERIMENTAL EVALUATION

In this section, we quantitatively and qualitatively evaluate the DMin framework in several challenging event-based estimation problems. In every experiment, we assume that the camera is calibrated and lens distortion has been removed.

5.5.1 *Datasets Overview*

We tested our framework on challenging real sequences from three datasets: Event-Camera Dataset (Mueggler et al., 2017), MVSEC Dataset (Zhu et al., 2018a) and DVSMOTION20 Dataset (Almatrafi et al., 2020).

The Event-Camera Dataset (Mueggler et al., 2017) consists of real sequences acquired by a DAVIS240 camera (Brandli et al., 2014), each with approximately one-minute duration and increasing motion magnitude, reaching angular velocities of 1000 deg/s. A motion-capture system provides the camera's pose at 200Hz, and a built-in Inertial Measurement Unit (IMU) provides acceleration and angular velocity measurements at 1000Hz. Using this dataset, we evaluate our framework on 2D translational motion, rotational motion estimation and motion estimation in planar scenes.

The MVSEC Dataset (Zhu et al., 2018a) consists of real sequences acquired by a set of different sensors for event-based 3D perception. The set of sensors includes an event-based stereo system of two DAVIS346B cameras and a frame-based stereo system of two Aptina MT9V034 cameras at 20 frames per second (fps), which provides the depth of the events generated. Due to the low sampling rate of the frame-based stereo system, the depth information provided may be highly corrupted, which our framework needs to handle. We use this dataset to assess our framework on 6-DOF transformation estimation and optical flow estimation from global and local estimates.

The DVSMOTION20 Dataset (Almatrafi et al., 2020) consists of real sequences acquired by a DAVIS346 camera, whose movements are restricted to rotations only (i. e. no translations). This is to ensure that the motion can be recovered entirely from the angular velocity provided by IMU measurements, allowing reliable optical flow ground-truth inference. Using this dataset, we evaluate our framework on optical flow estimation from global and local estimates.

5.5.2 *Implementation Details*

We assume that each batch of events spans a short time interval, such that the model parameters can be considered constants (which is reasonable since events can be triggered with a microsecond temporal resolution). We will consider the *Tsallis* entropy and respective approximation, given by Equations (5.10) and (5.15), respectively, where $\alpha = 2$. Additional results using the rest of the measures are provided in Appendix D. Note that α and β are not tunable parameters since each one specifies an entropy. For the online approach, we will consider the *Potential* energy (designated as *On. Potential*), as presented in Section 5.3. In case the data whitening step described in Section 5.4 is used, we precede the name of the measure with *W.*, e. g. *W. Tsallis*.

The proposed ADMin approach requires that we discretise the feature space and perform convolution operations. We use bilinear/trilinear interpolation/voting for each feature vector $\mathbf{f}_i$ to update the nearest bins. We asynchronously convolve each feature vector, using the event-based asynchronous convolution method proposed by Scheerlinck et al. (2019). This convolution method was also used in our implementation of the CMax framework (Gallego et al., 2018b) and the decay factor was set to zero to emulate a synchronous convolution over the entire space. We use a 3×3 Gaussian truncated kernel with one bin as standard deviation.

For the incremental mode, we maintain a fixed number of the most recent events similar to a first-in-first-out queue policy. The components $\boldsymbol{\Psi}$ and $\boldsymbol{\psi}$ of Equation (5.22) are incrementally computed by considering a spatial neighbourhood centred at the incoming event (the exact size of the neighbourhood is mentioned for each estimation case). These components are also decayed according to the factor $\exp\left(-(t_i - t_{i-1})n_{\text{decay}}\right)$, where t_i is the timestamp of the current event, t_{i-1} is the timestamp of the previous event and n_{decay} is incremented by one for each new event and is decayed according to the same factor.

In batch mode, only one estimate is computed for each batch, which is time-stamped according to the time-stamp of the last event in the batch. In online mode, a new estimate is computed for each event, which is time-stamped according to the corresponding event's time-stamp; however, we only compare the estimates that are spaced according to the fixed number of the most recent events considered. For example, if we maintain 10 of the most recent events, for evaluation purposes, we only consider the estimates computed for the 10th, 20th events, and so on. The quantitative errors reported below are computed by temporally aligning the ground-truth and the framework's estimates.

We refer the reader to Appendix D for additional experimental results, practical implementation details and a possible strategy to run the online mode in real-time.

5.5.3 *2D Translation Estimation*

The 2D translation motion can be parameterised by the 2D translational motion model (please refer to Appendix C.1 for the full mathematical description).

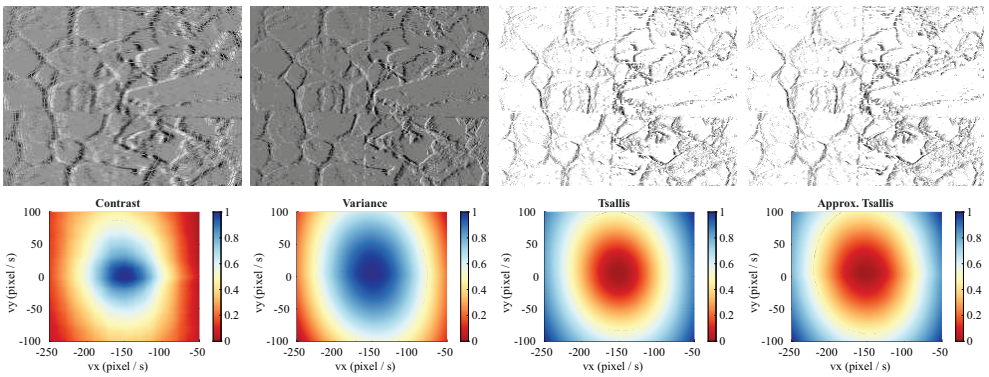


Figure 5.5: 2D Translation estimation between frames 17 and 18 of the `poster_translation` sequence (Mueggler et al., 2017). Top row, from left to right: Original events projected onto the IWE, where the flow is dominated by the horizontal component $\left(\theta^* \approx (-150, 0)^T\right)$; Motion compensated events projected onto the IWE, according to the CMax framework (Gallego et al., 2018b) $\left(\theta^* = (-150.3, 3.7)^T\right)$, *Tsallis* entropy-based measure $\left(\theta^* = (-150.8, 5.6)^T\right)$ and *Approx. Tsallis* measure $\left(\theta^* = (-150.7, 3.8)^T\right)$, respectively. Bottom row, from left to right: IWE contrast, *Variance* (Gallego et al., 2018b), *Tsallis* entropy-based measure and respective approximation profiles in function of the 2D translational motion parameters, respectively.

Figure 5.5 shows the results of the 2D translational motion estimation between frames 17 and 18 of the `poster_translation` sequence (Mueggler et al., 2017). If the original events are projected onto the image plane, the resulting image is blurry, whereas using our framework, the transformed events produce a sharp image with high contrast. Similar results are obtained if we use the CMax framework (Gallego et al., 2018b). Distinctively, however, by using an entropy-based measure, our framework minimises the transformed events' dispersion in a 2D feature space. The profiles of the *Tsallis* entropy-based measure and corresponding approximation in function of the 2D translational motion parameters are also presented (two most right plots in the bottom row). We can see that both profiles are similar, and the 2D translational motion parameters are correctly estimated.

5.5.4 *Rotational Motion Estimation*

The rotational motion can be parameterised by the rotational model (please refer to Appendix C.2 for the full mathematical description).

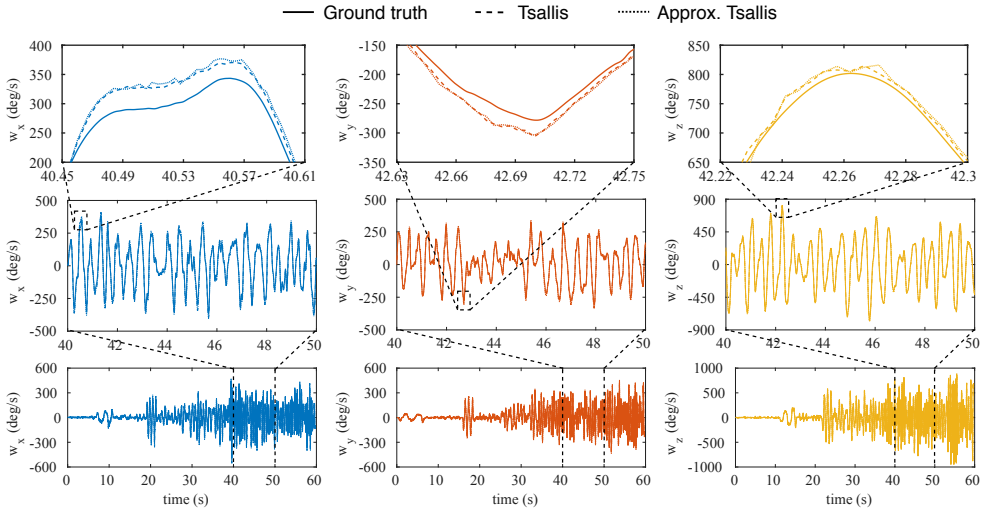


Figure 5.6: Comparison of the angular velocity estimated against the ground truth from IMU measurements on the `poster_rotation` sequence (Mueggler et al., 2017), considering batches of 20000 events. Bottom row: Whole sequence. Middle and top rows: Zoomed-in plots of the corresponding bounded regions. Both angular velocity profiles are almost identical to the ground truth.

Figure 5.6 presents a comparison of the angular velocity parameters estimated by the proposed framework using the *Tsallis* entropy-based measure and the corresponding approximate measure against the ground truth provided by IMU measurements on the `poster_rotation` sequence (Mueggler et al.,

2017). Qualitatively, both angular velocity profiles are identical to the ground truth, even during peak excursions of approximately ± 1000 deg/s, corresponding to rotations of more than 2.5 turns per second.

A detailed quantitative performance comparison is presented in Table 5.1, in terms of errors for each angular velocity component ($e_{w_x}, e_{w_y}, e_{w_z}$) and the respective standard deviation (σ_{e_w}) and Root Mean Square (RMS), on the `boxes_rotation` and `poster_rotation` sequences (Mueggler et al., 2017). We also report the average processing time per event. We compare the CMax (Gallego et al., 2018b), the *Potential* and the *Tsallis*, as well as the respective approximate and incremental versions, which are preceded by *Approx.* and *On.*, respectively. For the batch mode, we consider batches of 20000 events, and for the incremental mode, we maintain the most recent 10000 events, considering a 9×9 spatial neighbourhood centred at the incoming event.

Table 5.1: Rotational motion estimation accuracy comparison on the `boxes_rotation` and `poster_rotation` sequences (Mueggler et al., 2017). The angular velocity errors for each component ($e_{w_x}, e_{w_y}, e_{w_z}$), their standard deviation (σ_{e_w}) and RMS are presented in deg/s, w. r. t. IMU measurements, as well as the RMS error compared to the maximum excursions in percentage (RMS %). The processing time per event T_e is also reported in microseconds. The best value per column is highlighted in bold.

	`boxes_rotation`							`poster_rotation`						
Function	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %	T_e	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %	T_e
Variance	**7.51**	**6.43**	7.07	9.96	10.05	1.40	61.15	13.53	**8.64**	6.90	13.89	13.97	1.49	51.67
Potential	7.69	7.06	**6.49**	**9.34**	**9.43**	**1.29**	1202.22	14.23	8.81	**6.48**	**13.05**	**13.14**	**1.40**	475.48
Approx. Potential	7.91	7.07	7.78	9.94	10.04	1.37	57.38	13.94	8.91	7.69	13.51	13.62	1.45	59.40
On. Potential	7.99	8.04	10.84	12.13	12.25	1.67	**2.60**	**11.42**	9.86	14.27	16.98	17.16	1.83	**2.90**
Tsallis	7.91	7.53	6.51	9.48	9.56	1.30	1503.87	14.55	9.30	6.54	13.37	13.44	1.43	657.24
Approx. Tsallis	8.13	7.36	7.87	10.07	10.16	1.38	59.92	14.18	9.24	7.82	13.70	13.78	1.47	61.75
On. Tsallis	8.79	9.32	11.45	14.02	14.09	1.92	2.64	12.93	10.86	14.21	19.82	19.97	2.13	2.98

Overall, the exact measures, i. e. *Potential* and *Tsallis*, achieve the best results, while the corresponding efficient approximations achieve competitive performance. The incremental versions also achieve competitive results while being the most computationally efficient.

5.5.5 *Motion Estimation in Planar Scenes*

The motion in planar scenes can be parameterised by the homography model (please refer to Appendix C.6 for the full mathematical description).

Figure 5.7 illustrates the outcome of projecting the transformed events by performing motion estimation in a planar scene, according to the homography

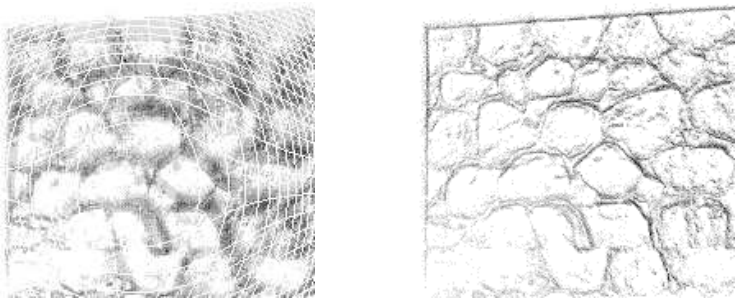


Figure 5.7: Motion estimation in planar scenes. (Left) Original events projected. (Right) Transformed events projected onto the image plane.

model, between frames 673 and 674 of the `poster_6dof` sequence (Mueggler et al., 2017).

Table 5.2: Motion estimation on planar scenes accuracy comparison on the `poster_translation` and `poster_6dof` sequences (Mueggler et al., 2017). The average angular and linear velocity errors (e_w, e_v), their standard deviation (σ_{e_w}, σ_{e_v}) and RMS (RMS_{e_w}, RMS_{e_v}) are presented in deg/s and m/s, respectively, as well as the RMS errors compared to the maximum excursions in percentage (RMS_{e_w} %, RMS_{e_v} %).

Sequence	Whitened?	**Function**	Failed?	e_w	σ_{e_w}	RMS_{e_w}	RMS_{e_w} %	e_v	σ_{e_v}	RMS_{e_v}	RMS_{e_v} %
`poster_translation`	No	Variance	No	15.09	20.14	20.61	8.91	0.34	0.38	0.45	16.86
		Potential	Yes	-	-	-	-	-	-	-	-
		Approx. Potential	Yes	-	-	-	-	-	-	-	-
	Yes	Variance	No	12.99	16.82	18.44	7.97	0.23	0.30	0.32	12.04
		Potential	No	20.36	20.76	26.39	11.40	0.31	0.34	0.40	15.06
		Approx. Potential	No	21.00	18.82	26.40	11.41	0.34	0.34	0.43	16.09
`poster_6dof`	No	Variance	No	25.76	34.24	35.07	3.26	0.48	0.59	0.64	26.12
		Potential	Yes	-	-	-	-	-	-	-	-
		Approx. Potential	Yes	-	-	-	-	-	-	-	-
	Yes	Variance	No	22.67	28.34	31.28	2.91	0.40	0.50	0.53	21.85
		Potential	No	34.62	46.82	55.23	5.13	0.47	0.56	0.62	25.67
		Approx. Potential	No	32.28	38.04	43.70	4.06	0.47	0.55	0.61	25.06

In Table 5.2, we provide a quantitative performance comparison on the `poster_translation` and `poster_6dof` sequences (Mueggler et al., 2017). We compare the *Variance* (Gallego et al., 2018b), the *Potential* and the respective approximate version *Approx. Potential* in terms of errors for the angular and linear velocities (e_w, e_v), their standard deviation (σ_{e_w}, σ_{e_v}) and RMS (RMS_{e_w}, RMS_{e_v}) are presented in deg/s and m/s, respectively, as well as the RMS errors compared to the maximum excursions in percentage (RMS_{e_w} %, RMS_{e_v} %), considering batches of 20000 events. The homography model

estimates the linear velocity up to a scale factor, which corresponds to the distance between the camera and the plane in the scene. To compute the linear velocity errors, we scale the estimated linear velocities according to the estimated distance. We also present the results by applying the data whitening step described in Section 5.4. The results for both measures are competitive, even though the Variance achieves the best performance overall. We highlight that all measures benefit from the data whitening step, which effectively mitigates the convergence to degenerate solutions.

5.5.6 *6-DOF Transformation Estimation*

So far, the models tested only require the original events' coordinates. In this section, we will evaluate our framework using the 6-DOF transformation model, which works for points with 3D coordinates, by augmenting the events with depth information. The 6-DOF transformation is a rigid body transformation that can be parameterised by a 3D rotation and a 3D linear translation (please refer to Appendix C.7 for the full mathematical description).

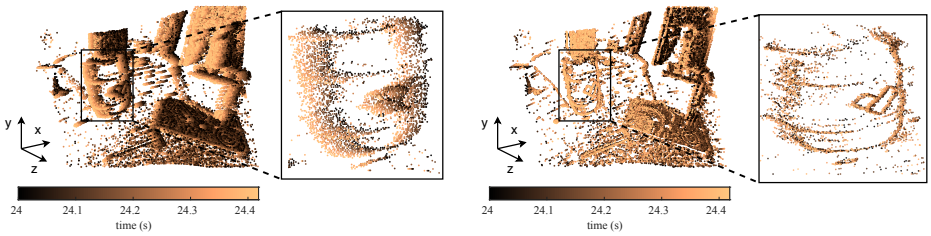


Figure 5.8: 6-DOF estimation on the `indoor_flying1` sequence (Zhu et al., 2018a). The trajectory of the original augmented events (left) can be modelled in 3D to retrieve the 6-DOF parameters of the moving camera while also aligning the events (right).

In Figure 5.8, we compare the original and transformed events from a moving 3D scene of the `indoor_flying1` sequence (Zhu et al., 2018a). Due to the motion, the original events generated are spread according to the 6-DOF motion (Figure 5.8 left). We can retrieve the 6-DOF parameters by minimising the transformed augmented events' dispersion directly in 3D space (avoiding possible projection steps) while conceptually also aligning the events to a reference frame.

We present a quantitative evaluation in Table 5.3, in terms of errors for the angular and linear velocities and the respective standard deviation and RMS, on the `indoor_flying1` and `outdoor_driving_day1` sequences (Zhu

Table 5.3: 6-DOF estimation accuracy comparison on the `indoor_flying1` and `outdoor_driving_day1` sequences (Zhu et al., 2018a). The average angular and linear velocity errors (e_w, e_v), their standard deviation (σ_{e_w}, σ_{e_v}) and RMS (RMS_{e_w}, RMS_{e_v}) are presented in deg/s and m/s, respectively, as well as the RMS errors compared to the maximum excursions in percentage (RMS_{e_w} %, RMS_{e_v} %). The processing time per event T_e is also reported in micro-seconds. The best value per column is highlighted in bold.

	`indoor_flying1`									`outdoor_driving_day1`								
Function	e_w	σ_{e_w}	RMS_{e_w}	RMS_{e_w} %	e_v	σ_{e_v}	RMS_{e_v}	RMS_{e_v} %	T_e	e_w	σ_{e_w}	RMS_{e_w}	RMS_{e_w} %	e_v	σ_{e_v}	RMS_{e_v}	RMS_{e_v} %	T_e
Potential	1.44	1.93	1.93	13.70	0.05	**0.06**	0.06	9.55	6575.61	1.69	3.07	3.08	10.69	0.44	0.66	0.67	8.61	19014.47
Approx. Potential	1.48	1.95	1.96	13.92	0.05	**0.06**	0.06	9.82	1333.34	2.04	3.39	3.40	11.79	0.41	0.61	0.62	7.90	823.68
On. Potential	1.47	1.97	1.97	14.02	0.06	0.08	0.08	11.71	**11.80**	1.90	3.10	3.10	10.77	0.63	0.87	0.94	12.09	**65.56**
Tsallis	**1.41**	**1.90**	**1.89**	**13.48**	**0.04**	**0.06**	**0.06**	**8.99**	7646.93	**1.63**	**3.05**	**3.06**	**10.64**	0.42	0.66	0.67	8.59	14767.06
Approx. Tsallis	1.43	1.92	1.92	13.68	0.05	**0.06**	0.06	9.10	1311.17	2.01	3.30	3.31	11.50	**0.39**	**0.58**	**0.59**	**7.57**	828.28
On. Tsallis	1.46	1.94	1.95	13.84	0.05	0.07	0.07	10.54	14.18	1.93	3.24	3.25	11.29	0.59	0.86	0.94	12.08	68.81

et al., 2018a). We also report the average processing time per event. We compare the *Potential* and the *Tsallis*, as well as the respective approximate and incremental versions, which are preceded by *Approx.* and *On.*, respectively. For the batch mode, we consider batches of 20000 and 30000 events for the `indoor_flying1` and `outdoor_driving_day1` sequences, respectively. For the incremental mode, we maintain the most recent 15000 and 30000 events for the `indoor_flying1` and `outdoor_driving_day1` sequences, respectively, and consider a 9×9 spatial neighbourhood centred at the incoming event.

The 6-DOF parameters were estimated from highly corrupted data since the events' spatial coordinates were augmented from synchronous depth measurements at 20fps. This frame rate is several orders of magnitude lower than the temporal resolution of the event-based data.

Moreover, in the `outdoor_driving_day1` sequence, the events generated are significantly influenced by several other relative motions, e. g. due to other cars moving in the field-of-view. The results suggest that our framework is capable of coping with noise, provided it is not the predominant factor.

In Table 5.4, we also compare our framework with two deep learning methods (Ye et al., 2018; Zhu et al., 2019) that predict the ego-motion of a moving camera from events. The methods are evaluated in terms of Average Relative Pose Error (ARPE) (RPE $= \arccos \frac{\mathbf{t}_{\text{pred}} \cdot \mathbf{t}_{\text{gt}}}{\|\mathbf{t}_{\text{pred}}\| \cdot \|\mathbf{t}_{\text{gt}}\|}$), Average Relative Rotation Error (ARRE) (RRE $= \left\| \operatorname{logm}\left(\mathbf{R}_{\text{pred}}^{\mathsf{T}} \mathbf{R}_{\text{gt}}\right) \right\|$, where logm is the matrix log), and Average Endpoint Error (AEE) (EE$= \left\| \mathbf{t}_{\text{pred}} - \mathbf{t}_{\text{gt}} \right\|$).

All the methods achieve competitive performance in the `outdoor_driving_day1` sequence, although the method proposed by Ye et al. (2018) achieves the best results. However, deep learning methods are known to require huge

Table 5.4: Quantitative 6-DOF estimation comparison on the `indoor_flying1` and `outdoor_driving_day1` sequences (Zhu et al., 2018a). The ARPE in degrees, the ARRE in radians and the AEE in m/s are presented. The best value per column is highlighted in bold.

	`indoor_flying1`			`outdoor_driving_day1`		
Method	ARPE	ARRE	AEE	ARPE	ARRE	AEE
Zhu et al. (2019)	-	-	-	7.74	0.00867	-
Ye et al. (2018)	-	-	-	**3.98**	**0.00267**	**0.70**
Potential	11.01	0.00908	0.10	9.49	0.01331	0.94
Approx. Potential	11.41	0.00917	0.10	8.76	0.01540	0.88
On. Potential	13.69	0.00681	0.12	14.16	0.01136	1.35
Tsallis	**10.42**	0.00896	**0.09**	9.41	0.01298	0.92
Approx. Tsallis	10.45	0.00896	**0.09**	8.04	0.01524	0.83
On. Tsallis	12.35	**0.00679**	0.11	13.98	0.01129	1.29

amounts of data. As such, they can not handle the `indoor_flying1` sequence, which has only one-minute duration. In contrast, our framework can provide robust estimations regardless of the sequence duration. Additionally, in the `outdoor_driving_day1` sequence only 2-DOF play a major role (i. e. accounting for driving forward and turning in curves) and, as mentioned by Ye et al. (2018), this may lead to network overfitting. However, the full 6-DOF are utilised in the `indoor_flying1` sequence, accounting for more complex motions, which the network may not handle if overfitted to simpler motions. The DMin framework does not have this limitation, as corroborated by the results.

In Figure 5.9, the estimated trajectories are compared with the ground-truth on the `indoor_flying1` and `outdoor_driving_day1` sequences (Zhu et al., 2018a). The trajectories were obtained by integrating the estimated 6-DOF parameters. The estimated trajectories roughly follow the ground truth, exhibiting small drifts. Notably, even though the complex motion is exhibited on the `indoor_flying1` sequence, the estimated trajectories are very close to the ground truth.

5.5.7 *Optical Flow Estimation*

The optical flow model can be parameterised by local 2D translations (please refer to Appendix C.1 for the full mathematical description). The optical flow can also be obtained by estimating some global motion, according to some other motion model, and computing the difference between the original and transformed image coordinates, e. g. for a similarity transformation:

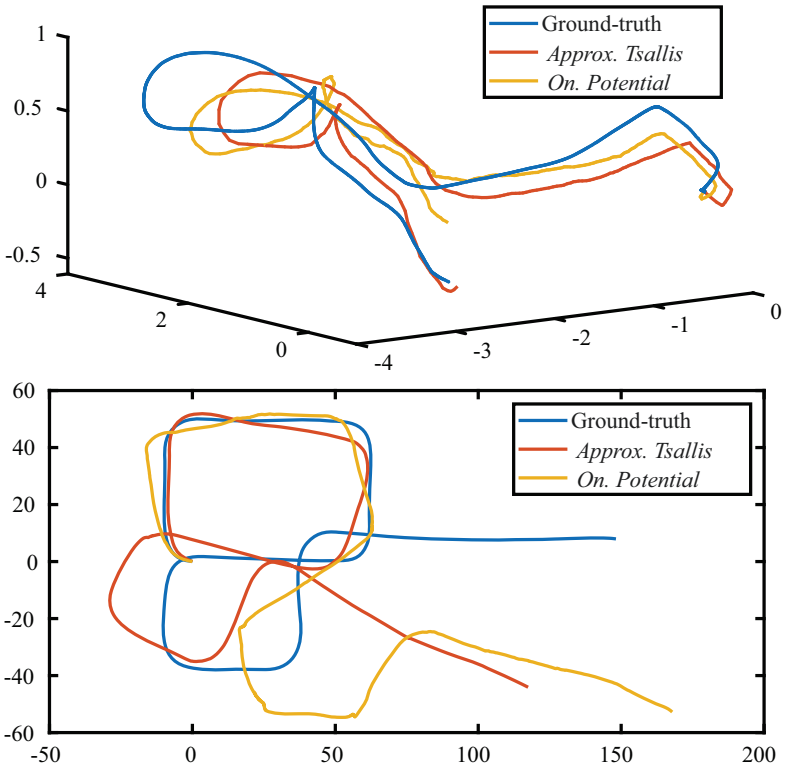


Figure 5.9: Estimated trajectories on the (top) `indoor_flying1` and (bottom) `outdoor_driving_day1` sequences (Zhu et al., 2018a).

$\mathbf{x} - \mathbf{P}\mathcal{S}^{-1}(t;\theta)(\mathbf{x}^\mathsf{T}, 1)^\mathsf{T}$. However, this assumes that there is only one motion occurring and is thus less general.

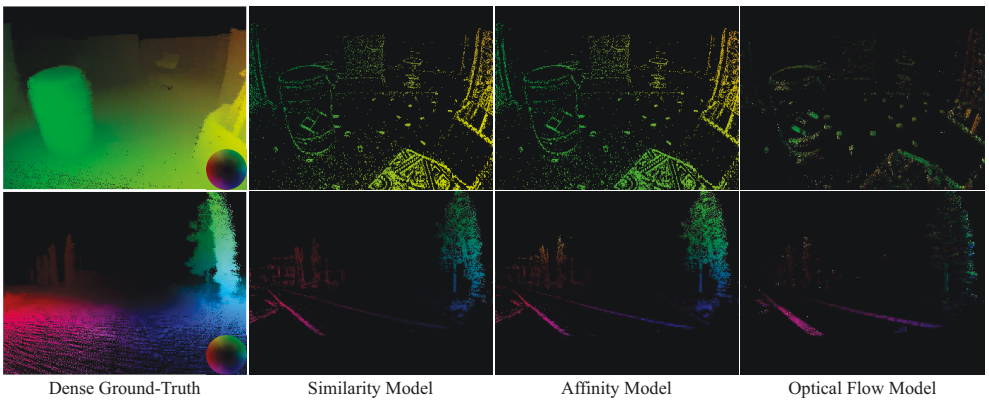


Figure 5.10: Qualitative optical flow estimation on the (top row) `indoor_flying1` and (bottom row) `outdoor_driving_day1` sequences (Zhu et al., 2018a).

Figure 5.10 shows some qualitative results of optical flow estimation using the *On. Potential* measure, for the global similarity and affinity transformation models, given by Equations (C.11) and (C.14), respectively, and the local

optical flow model. The flow is only estimated where events are generated. The estimated and ground-truth flows are visually very similar.

Table 5.5: Quantitative optical flow estimation comparison on the `classroom` and `conf_translation` sequences (Almatrafi et al., 2020). The Relative Average Endpoint Error (RAEE) in percentage and the Average Angular Error (AAE) in degrees are presented. The best value per column is highlighted in bold.

			`conf_trans`		`classroom`	
Method			RAEE	AAE	RAEE	AAE
EV-Flow Net (Zhu et al., 2018b)			23.62	8.04	44.62	15.14
Almatrafi et al. (2020)			18.71	5.42	25.62	6.84
Model	Whitened?	**Function**	RAEE	AAE	RAEE	AAE
Affinity	No	*Potential*	20.76	3.61	30.48	7.91
		Approx. Potential	21.37	4.18	30.36	7.97
		On. Potential	39.50	13.84	30.98	9.98
	Yes	*Potential*	20.12	3.26	23.34	3.63
		Approx. Potential	20.30	3.75	23.18	3.90
		On. Potential	39.84	14.01	30.86	9.92
Similarity	No	*Potential*	20.76	3.64	28.46	6.62
		Approx. Potential	21.45	4.18	28.86	7.04
		On. Potential	24.44	5.24	26.70	7.29
	Yes	*Potential*	20.07	**3.19**	23.28	**3.57**
		Approx. Potential	20.34	3.60	**23.09**	3.80
		On. Potential	24.34	5.24	26.82	7.37
Rotation	No	*Potential*	20.44	4.78	24.67	4.58
		Approx. Potential	21.05	5.59	24.87	5.28
		On. Potential	**18.15**	5.40	23.85	5.82

In Table 5.5, we present a quantitative evaluation on the `conf_translation` and `classroom` sequences (Almatrafi et al., 2020), in terms of RAEE and AAE. We compare the *Potential*, as well as the respective approximate and incremental versions, which are preceded by *Approx.* and *On.*, respectively, against two state-of-the-art methods (Almatrafi et al., 2020; Zhu et al., 2018b). We report results using three global models, namely affinity transformation, similarity transformation and rotational motion transformation (please refer to Appendix C for the complete description of the models). For the affinity and similarity transformations, Table 5.5 also presents the results obtained by applying the data whitening pre-process step described in Section 5.4 since these models have at least one scaling parameter. The proposed measures achieve the best results, while the method proposed by Almatrafi et al. (2020) achieves competitive performance. We highlight that applying data whitening

using the affinity and similarity transformation models improves the overall quantitative performance.

5.5.8 *Computational Processing Time*

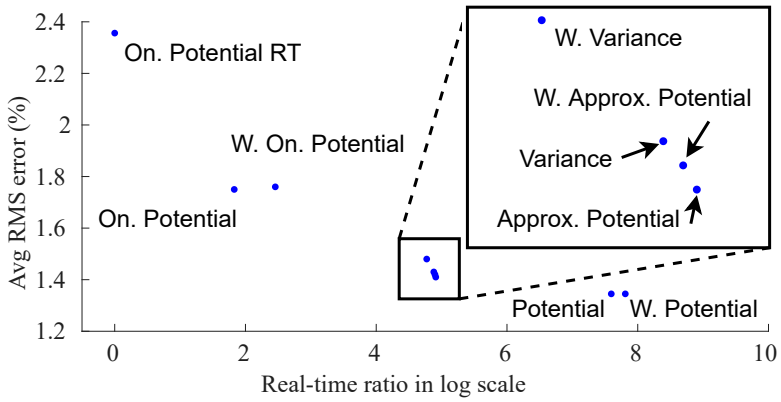


Figure 5.11: Runtime vs. accuracy comparison. A real-time ratio of 0 means real-time processing. The preceding W. indicates that the data whitening step described in Section 5.4 was applied.

In general, the performance of the proposed measures follows the trend depicted in Figure 5.11. The batch mode measures are computationally slower, exhibiting the best accuracy. On the other extreme, the *Potential* measure in online mode is considerably faster but on average exhibits larger RMS errors. In a nutshell, the proposed measures and modes provide a trade-off between accuracy and computational runtime.

5.6 CONCLUSION

This chapter described a framework for event-based model estimation that does not rely on any intermediate image-based representation to estimate the underlying motion causing events to be triggered. The framework can thus perform incremental motion estimation for each incoming event while handling arbitrary motion models. Besides estimating visual transformations in 2D, the framework also allows incorporating additional sensory information by augmenting the events to estimate the motion parameters of models in higher dimensional spaces. We also proposed a simple pre-processing step based on data whitening that mitigates degenerate solutions while benefiting the framework's overall performance. The proposed framework can thus be seen as an extension of previous motion compensation approaches. The framework was evaluated on several challenging datasets and motion

models, achieving state-of-the-art performance in complex motion estimation from event-based data. The incremental mode is also suitable for real-time applications, and we refer the reader to Appendix D.8 for additional details on how that can be achieved.

In the next chapter, we will use the incremental mode of the DMin framework as a core component to perform joint shape-motion segmentation in the context of event-based Kinematic Structure (KS) estimation.

Contributions. The main contributions presented in this chapter have been published:

Nunes, U. M. and Demiris, Y. (2020), Entropy Minimisation Framework for Event-Based Vision Model Estimation, *in* 'European Conference on Computer Vision (ECCV)', pp. 161-176. doi: 10.1007/978-3-030-58558-7_10. (**Spotlight**) (Nunes and Demiris, 2020)

Nunes, U. M. and Demiris, Y. (2021), Robust Event-based Vision Model Estimation by Dispersion Minimisation, *in* 'IEEE Transactions on Pattern Analysis and Machine Intelligence', doi: 10.1109/TPAMI.2021.3130049. (Nunes and Demiris, 2021b)

Nunes, U. M. and Demiris, Y. (2021), Live Demonstration: Incremental Motion Estimation for Event-based Cameras by Dispersion Minimisation, *in* 'IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops', pp. 1322–1323. (Nunes and Demiris, 2021a)

6

EVENT-BASED VISION INCREMENTAL KINEMATIC STRUCTURE ESTIMATION

This chapter addresses the third research question, focusing on the second part:

> Can event cameras improve the Kinematic Structure (KS) estimation over standard frame-based cameras, and how?

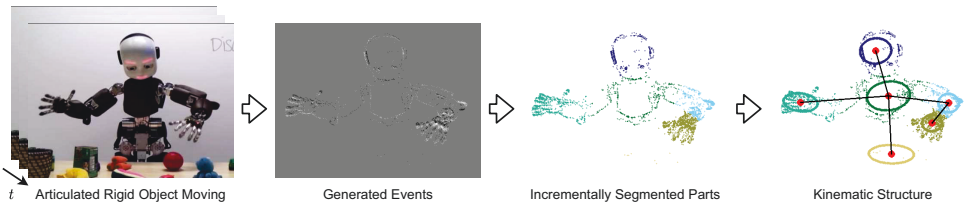


Figure 6.1: Incremental event-based KS estimation of articulated rigid objects pipeline. An event camera observes an articulated rigid object moving, causing events to be triggered due to the perceived motions. The proposed method incrementally segments the different body parts by jointly estimating the corresponding motion and shape parameters. The KS is then generated as a graph based on the segmented parts and their respective biharmonic distance. RGB and accumulated generated events images only included for visualisation.

In this chapter, we present the first event-based KS estimation method for arbitrary articulated rigid objects. Figure 6.1 presents the pipeline of the proposed method. Each generated event is assigned to an existing segment, or a new one is created, by combining an adapted Bayesian inference approach and the result from event-based motion estimation (please refer to Chapter 5 for a detailed description of the event-based motion estimation method, and in particular 5.3, which describes the incremental approach adopted in this chapter). We then estimate the object's KS and represent it as a graph. The graph's connections are computed by sampling points from each segment's distribution and estimating the underlying biharmonic distances (Lipman et al., 2010). This approach overcomes potential issues related to discontinuities in the original events' topological surface.

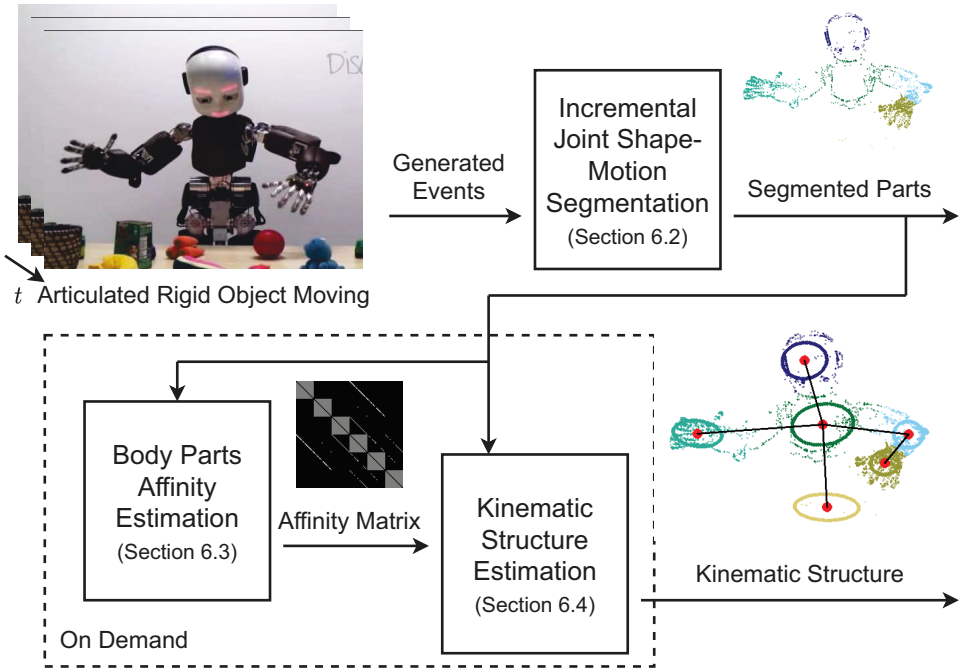


Figure 6.2: Overview of the incremental event-based KS estimation method. For each incoming event, we find the most plausible body part responsible for generating it and update its parameters accordingly. An affinity matrix is built by estimating the affinity between each body part based on points sampled from the corresponding estimated spatial distributions. We then generate the KS as a graph by combining the segmented parts and the estimated affinity between them. The KS can be estimated on demand to save computation. RGB images only included for visualisation.

6.1 OVERVIEW

Figure 6.2 presents an overview of the proposed approach. The method only assumes that a moving articulated rigid object generates events and that the shape of a multivariate Gaussian distribution approximately describes each body part. In Section 6.2, we jointly segment the different parts by combining an adapted Bayesian inference approach and the incremental event-based motion estimation method described in Chapter 5. We exploit this representation 1) to estimate the number of the object's segments and 2) to find the connectivity of the graph that represents the object's KS. The number of segments and each body part parameters are estimated using an adapted Bayesian inference approach inspired by Dirichlet Processes. Section 6.3 describes how the graph's connectivity can be estimated based on the biharmonic distance (Lipman et al., 2010) between each pair of segments, which in turn is estimated by computing the modes of sampled points from each segment's distribution. The respective KS generation process is then

described in Section 6.4 based on the segmented parts and their respective biharmonic distance. In Section 6.5, we show that this simple approach outperforms state-of-the-art frame-based methods in sequences obtained by simulating events from video sequences (Rebecq et al., 2018b). We also show that the proposed method achieves state-of-the-art performance on new high-speed motions sequences, which typical frame-based KS estimation methods can not handle (please refer to Appendix E for additional details about the dataset). Finally, Section 6.6 concludes this chapter.

6.2 INCREMENTAL JOINT SHAPE-MOTION SEGMENTATION

The proposed KS estimation framework is designed to work with a stream of visual events $\mathcal{E} = \{e_k\}_{k=1,2,\ldots}$. For ease exposition, we assume that the camera does not move, and thus all the events are generated from the moving parts of an articulated rigid object. We also begin by considering that we want to segment a batch of events into distinct body parts and then explain how we can extend the method to perform the incremental estimation. We refer the reader to Appendix B for additional mathematical details regarding the probability distributions and processes adopted.

Each segment $S_g = (\mathbf{m}_g, \mathbf{C}_g, \theta_g)$ is assumed to be described by a spatial multivariate Gaussian distribution $\mathcal{N}(\mathbf{m}_g, \mathbf{C}_g)$, defining its position $\mathbf{m}_g$ and shape $\mathbf{C}_g$, and corresponding motion parameters θ_g. Additionally, at time t there are n^t estimated segments (thus $g = \{1, \ldots, n^t\}$). To ease notation, the relevant subscripts and/or superscripts will only be included whenever necessary.

6.2.1 *Joint Optimisation Procedure*

We can formalise the problem of estimating the segments' parameters from generated events and how many exist, as the problem of maximising the log-likelihood of the observed events given the segments' parameters. This entails solving the following optimisation problem:

$$\mathbf{S}^* = \arg\max_{\mathbf{S}} \sum_k \log p(e_k|\mathbf{S}) = \sum_k \log l(\mathbf{S}), \tag{6.1}$$

where $l(\mathbf{S})$ is the complete data log-likelihood of the set of all estimated segments $\mathbf{S} = \{S_g\}_{g=1}^n$

$$l(\mathbf{S}) = \sum_k p(e_k, z_k|\mathbf{S}), \tag{6.2}$$

and $z_k \in \{1, \dots, n\}$ is a hidden variable that represents the cluster assignment of event e_k.

We solve this optimisation problem via Expectation-Maximisation (EM) and Maximum a Posteriori (MAP) estimation:

$$\begin{aligned} Q(\mathbf{S}, \mathbf{S}^{\text{prev}}) = \underbrace{\sum_{k,g} r_{kg} \log \pi_g + \sum_{k,g} r_{kg} \log p(e_k|S_g)}_{\text{expected complete data log-likelihood}} + \\ \underbrace{\log p(\boldsymbol{\pi}) + \sum_g \log p(S_g)}_{\text{log prior}}, \end{aligned} \tag{6.3}$$

where r_{kg} is the responsibility that cluster g takes for event e_k, $\boldsymbol{\pi} = \{\pi_g\}_{g=1}^n$ are the mixing weights and $\mathbf{S}^{\text{prev}}$ refers to the previous estimate in the iterative optimisation procedure.

To compute $p(e_k|S_g)$ and $p(S_g)$, we assume that the shape parameters $(\mathbf{m}_g, \mathbf{C}_g)$ and the motion parameters θ_g can be factorised since a given body part can exhibit any kind of motion within its possible range of motion. The likelihood of event e_k given segment S_g can thus be computed as

$$p(e_k|S_g) \propto p(e_k|\mathbf{m}_g, \mathbf{C}_g) p(e_k|\theta_g), \tag{6.4}$$

where $p(e_k|\mathbf{m}_g, \mathbf{C}_g) = \mathcal{N}(\mathbf{x}_k|\mathbf{m}_g, \mathbf{C}_g)$ (i. e. Gaussian distribution) and

$$p(e_k|\theta_g) \propto \exp\left[-D(e_k; \theta_g)\right]. \tag{6.5}$$

$D(e_k; \theta_g)$ is a dispersion-based score, which measures how the motion parameters θ_g can compensate the underlying motion of the observed events assigned to segment g (please refer to Chapter 5 for dispersion measures). For event-based KS estimation, we adapted the expression given by Equation (5.3) to take into account the responsibilities, as follows:

$$D(e_k; \theta_g) \propto -\sum_j r_{jg} \mathcal{K}_{\Sigma}(\mathbf{x}'_{kg}, \mathbf{x}'_{jg}; \theta_g), \tag{6.6}$$

where $\mathbf{x}'_{kg}$ corresponds to the transformed event spatial coordinates according to motion parameters θ_g, given by Equation (5.2). The segment's prior is given by

$$p(S_g) = p(\mathbf{m}_g, \mathbf{C}_g) p(\theta_g) \propto p(\mathbf{m}_g, \mathbf{C}_g), \tag{6.7}$$

as it is intractible to introduce a proper prior on the motion parameters θ_g such that all motions have equal probability. Each segment's position

$p(\mathbf{m}_g, \mathbf{C}_g)$ has the conjugate-prior form of a Normal-inverse-Wishart distribution, i. e. $\mathrm{NIW}(\mathbf{m}_g, \mathbf{C}_g | \mathbf{m}_0, \kappa_0, \nu_0, \mathbf{C}_0)$. The mixing weights' prior is given by the Dirichlet distribution with parameter α.

6.2.2 *Creating and Deleting Segments*

Inspired by Dirichlet Process Mixture Models, which estimate the number of clusters (Escobar and West, 1995; Maceachern and Müller, 1998), we maintain a virtual segment with no events assigned to it (i. e. it only contains prior information). Whenever an event is to be assigned to the virtual segment, a new segment is created and updated accordingly, and the number of segments n is incremented by 1. Conversely, a segment g is erased whenever the corresponding mixing weight π_g is below a predefined threshold.

6.2.3 *E-step*

The responsibilities are computed as

$$r_{kg} \propto \pi_g p(e_k | S_g^{\text{prev}}), \quad \text{s.t.} \sum_{g'} r_{kg'} = 1. \tag{6.8}$$

6.2.4 *M-step*

The mixing weights are computed as

$$\pi_g \propto \begin{cases} \sum_k r_{kg} = r_g & \text{if } g \text{ has been seen before} \\ \alpha & \text{otherwise (virtual segment)} \end{cases}, \tag{6.9}$$

such that $\sum_g \pi_g = 1$. Each mean is computed as

$$\mathbf{m}_g = \frac{r_g \bar{\mathbf{x}}_g + \kappa_0 \mathbf{m}_0}{r_g + \kappa_0}, \quad \bar{\mathbf{x}}_g = \frac{\sum_k r_{kg} \mathbf{x}_k}{r_g} \tag{6.10}$$

and each covariance is computed as

$$\mathbf{C}_g = \frac{\mathbf{C}_0 + \mathbf{Z}_g + \frac{\kappa_0 r_g}{\kappa_0 + r_g} (\bar{\mathbf{x}}_g - \mathbf{m}_0)(\bar{\mathbf{x}}_g - \mathbf{m}_0)^\mathsf{T}}{\nu_0 + r_g + d + 2}, \tag{6.11}$$

where $\mathbf{Z}_g = \sum_k r_{kg}(\mathbf{x}_k - \bar{\mathbf{x}}_g)(\mathbf{x}_k - \bar{\mathbf{x}}_g)^\top$. The motion parameters are estimated based on Chapter 5, according to

$$\theta_g = \boldsymbol{\Psi}_g^{-1}\boldsymbol{\psi}_g, \tag{6.12}$$

$$\boldsymbol{\psi}_g = -\sum_k r_{kg} \sum_j r_{jg}\mathbf{W}_{kjg}\left(\tilde{\mathbf{x}}_{kg} - \tilde{\mathbf{x}}_{jg}\right), \tag{6.13}$$

$$\boldsymbol{\Psi}_g = \sum_k r_{kg} \sum_j r_{jg}\mathbf{W}_{kjg}\left(\mathbf{X}_k - \mathbf{X}_j\right), \tag{6.14}$$

$$\mathbf{W}_{kjg} = \mathcal{K}_{\boldsymbol{\Sigma}}(\mathbf{x}'_{kg}, \mathbf{x}'_{jg}; \theta_g)\frac{\partial\left(\mathbf{x}'_{kg} - \mathbf{x}'_{jg}\right)^\top}{\partial\theta_g}\boldsymbol{\Sigma}^{-1}, \tag{6.15}$$

where $\mathbf{x}'_{kg} = \tilde{\mathbf{x}}_{kg} + \mathbf{X}_k\theta_g$ implicitly defines $\tilde{\mathbf{x}}_{kg}$ and $\mathbf{X}_k$ (please refer to Appendix C for additional details).

Equations (6.8) to (6.12) are iteratively solved until the estimate converges or a certain number of iterations is reached. After the optimisation, events are assigned to segments by performing hard assignment based on the corresponding responsibilities.

6.2.5 *Incremental Segmentation*

For each incoming event e_k, we want to find from which body part it was generated, while the corresponding parameters S_g are incrementally updated. To achieve this, we adopt a decaying strategy applied to cumulative variables of interest, such that more recent events are given more importance. Assuming the cumulative responsibility at time t_{k-1} is given by $r_g^{t_{k-1}}$, we update it according to

$$r_g^{t_{k-1}} \leftarrow d(\gamma, t_k)r_g^{t_{k-1}}, \tag{6.16}$$

where $d(\gamma, t_k)$ is a user defined decay function. This decay function is also applied to $\boldsymbol{\psi}_g^{t_{k-1}}$ and $\boldsymbol{\Psi}_g^{t_{k-1}}$, as mentioned in Section 5.5.2.

6.2.6 *Optimisation Prior Parameters*

The joint optimisation procedure described introduces a few prior parameters that need to be set. It is thus worth briefly mentioning their potential influence in the context of the proposed optimisation procedure. We refer the reader to Appendix B and to (Murphy, 2012) Chapters 2 and 25 for additional details.

The Dirichlet concentration parameter α sets the prior value for how probable is the existence of additional segments, according to Equation (6.9).

Looking at the lower-end extreme case, whereby the parameter is set close to zero, then only one segment would emerge since the contribution of the virtual segment would be close to zero. Conversely, if the parameter is set to a value that tends to infinite, then a new segment would emerge for each incoming event.

The Normal-inverse-Wishart distribution encapsulates our priors regarding the position, size and shape of the segments to estimate. The prior position is given by $\mathbf{m}_0$ and κ_0 defines how strongly we believe the position prior. The prior size and shape is given by $\mathbf{C}_0$ and ν_0 defines how strongly we believe the shape prior.

6.3 BODY PARTS AFFINITY ESTIMATION

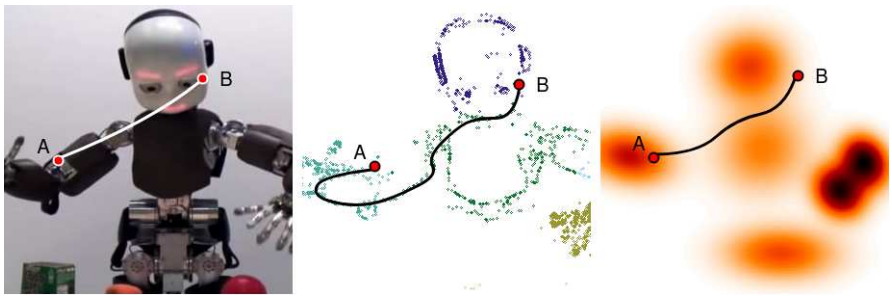


Figure 6.3: Topological shortest paths between points A and B, on the objects' surface (left), on the corresponding generated events surface (middle) and on the estimated spatial segments' surface (right). The spatial segments' topology provides a good approximation for the object's topology.

At this stage, we have incrementally segmented past events into n parts $S_g = (\mathbf{m}_g, \mathbf{C}_g, \theta_g)$. To estimate the underlying KS, these segments need to be connected according to the object's topological surface. Events are typically generated from contrasting textures and objects' boundaries. Thus, the resultant event-based topological structure does not adequately represent the object's actual topology, as illustrated in Figure 6.3. We propose instead to leverage the estimated spatial segments' components as an approximation of the object's topological structure.

An affinity matrix $\mathbf{A}$ can be built by sampling points from each estimated spatial Gaussian distribution and performing mean-shift iterations (Comaniciu and Meer, 2002) to compute the mode of each sample. This is achieved, while only considering the remaining Gaussian distributions, as depicted in Figure 6.4. The mode found for each sample indicates the closest body part to the part that generated the samples. For example, in Figure 6.4, the segment S_1 corresponds to the mode of four samples, while segment S_2 corresponds

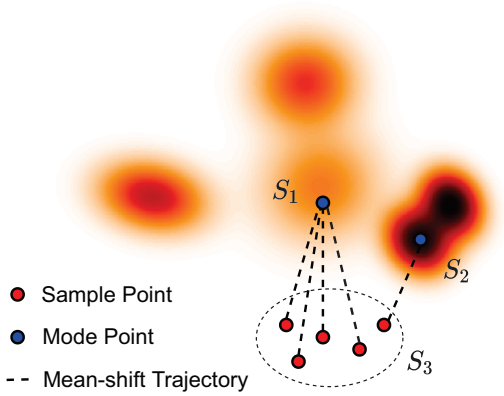


Figure 6.4: Mode seeking based on mean-shift iterations.

to the mode of one sample. This suggests that S_1 is the topologically closest segment to S_3, and they should be connected.

The affinity matrix $\mathbf{A}$ is a symmetric matrix with $n \cdot n_s$ rows and columns, where n_s is the number of samples drawn from each segment. The number of samples n_s controls the trade-off between accuracy and computational costs on the underlying object's topology approximation. In the lower-end limit, if only one sample is drawn, the object's segment topology would be approximated by just one point, which does not reliably represent the underlying segment's topology. On the other extreme limit, whereby an infinite number of points are drawn, the segment's topology would be fully captured, although, in practice, it would be intractable to compute.

The entries $a_{i_g j_h}$, $\forall j_h$ are incremented by 1, whenever the mode of sample i_g (i. e. the i-th sample from segment g) is the mean position of segment h (also, $a_{i_g j_h} = a_{i_h j_g}$ since the matrix is symmetric). The entries corresponding to samples drawn from the same segment are set to one, except for the diagonal elements, which are set to zero.

6.4 KINEMATIC STRUCTURE ESTIMATION

As in the previous Chapter 4, we assume that the object's KS is not cyclic (Chang and Demiris, 2018; Yan and Pollefeys, 2008), which covers most articulated rigid objects. We model the topological connections between segments as a complete graph $\mathcal{G} = (V, E)$ whose vertices V represent the segments' position, given by the respective mean of the spatial Gaussian

distribution. The proximity E between two segments g and h is given by the respective biharmonic distance (Lipman et al., 2010):

$$E(S_g, S_h) = d_H(\mathbf{m}_g, \mathbf{m}_h). \tag{6.17}$$

The graph's minimum spanning tree is then generated as the articulated rigid object's KS.

The biharmonic distance provides several important properties, including a balance between approximate geodesic distance for local neighbourhoods and shape awareness across the global structure. To estimate the biharmonic distance between pairs of points on a surface (e. g. surface of an articulated rigid object), we first need to compute the corresponding Laplacian matrix $\mathbf{L}$, which encodes the affinity of the surface's points, and the mass matrix $\mathbf{M}$, which is a diagonal matrix whose entries correspond to the Voronoi area of each mesh vertex.

The Laplacian matrix $\mathbf{L}$ is obtained from the previously built affinity matrix $\mathbf{A}$, according to Equation (3.6), from Section 3.3. The spectrum of the Laplacian is obtained by solving the generalised eigenvalue problem $\mathbf{L}\mathbf{v}_j = \mathbf{M}\lambda_j\mathbf{v}_j$, where $\mathbf{v}_j$ and λ_j correspond to the j-th eigenvector and eigenvalue, respectively. The $\mathbf{M}$ matrix is denoted by the area/mass matrix since it contains information regarding the area of each mesh formed by three points that represent the object's topology. By taking the first J eigenvectors to reduce computational costs, the biharmonic distance between two points a and b on the surface is given by (Lipman et al., 2010)

$$d_H(\mathbf{p}_a, \mathbf{p}_b) = \sum_j^J \left(\frac{v_{j,a} - v_{j,b}}{\lambda_j} \right)^2, \tag{6.18}$$

where $v_{j,a}$ represents the a-th entry of the eigenvector $\mathbf{v}_j$.

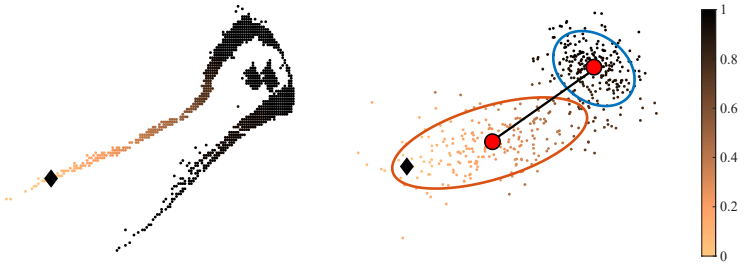


Figure 6.5: Biharmonic distance between the marked point (⧫) and all other points, from the original events (left) and the samples drawn from each estimated part (right).

As discussed in Section 6.3, the surface of a moving object is not well represented by the corresponding generated events, as several event points that belong to the same body part may exhibit a large biharmonic distance, which is illustrated on the left plot of Figure 6.5. The simple strategy of sampling points from the estimated segments' distribution provides a reasonable approximation of the object's topology, which in turn, provides a better estimate of the actual distance between points on the object's surface, as depicted on the right plot of Figure 6.5.

6.5 EXPERIMENTAL EVALUATION

In this section, we quantitatively and qualitatively assess the performance of the proposed method in several highly complex articulated KS sequences.

6.5.1 *Datasets Overview*

We evaluate our method on challenging real sequences from three datasets: HKS Dataset (Chang and Demiris, 2018), 3DKS Dataset (Tzionas and Gall, 2016), already used in Chapters 3 and 4, and the new EKS Dataset (please refer to Appendix E for additional details regarding the dataset). To evaluate the proposed method, we simulated event-based data from the HKS and 3DKS datasets sequences using the Event Camera Simulator (Rebecq et al., 2018b). Additionally, we augment the simulated events from the 3DKS dataset' sequences with frame-based depth information. Since the original frame rate is low (i. e. between 20 and 30 frames per second (fps)), the simulated events are of poor quality in some parts of the sequences, increasing the estimation difficulty.

The HKS Dataset (Chang and Demiris, 2018) consists of RGB sequences of highly articulated rigid objects moving. The sequences exhibit relative slow but diverse motions, including concurrency, rotations, and scaling. This dataset is useful to assess our method's performance on regular articulated rigid objects' sequences.

To the best of our knowledge, the EKS Dataset is the first event-based KS dataset, containing three challenging sequences which exhibit fast and diverse motions. Each sequence was recorded by a system composed of an event-based DAVIS346B camera and a frame-based RGB-D Intel RealSense D415 camera. The frame-based data exhibit massive motion blur effects, making frame-based estimation using commercially available cameras unfeasible.

Please refer to Appendix E for additional details about the proposed EKS Dataset.

6.5.2 *Implementation Details*

All experiments were performed on a single-threaded Intel Core i7-9700k CPU @ 3.6Ghz, using a C++ implementation. Each object graph's minimum spanning tree was generated using the igraph library[1].

The same parameters were used across all experiments, except for the Normal-inverse-Wishart $\mathbf{C}_0$ parameter. As mentioned in Section 6.2.6, the Normal-inverse-Wishart $\mathbf{C}_0$ parameter sets the prior information for the size and shape of each segment. Since the apparent size and shape of each objects' segments varies, this parameter had to be empirically set for each sequence. For most of the sequences, we set it to $0.75\mathbf{I}$ and provide the exceptions in Table 6.1. For the remaining parameters, we set them with the weakest prior possible, while still being proper, which amounts to having $\kappa_0 = 0$ and $\nu_0 = d + 2$. In this setting, $\mathbf{m}_0$ does not need to be set. The Dirichlet parameter was set to $\alpha = 1$ which emulates a virtual segment to which has been assign one event. Empirically, we found that this value provides a balance between being able to generate possible new segments, while converging to a reasonable estimated number of segments. The number of samples drawn from each segment's spatial distribution to build the affinity matrix $\mathbf{A}$ was set to $n_s = 16$ since we have empirically found that this value provided a valid trade-off between computational runtime performance and suitable topological representation for most of the sequences considered. The area matrix $\mathbf{M}$ was set to the identity matrix and $J = n + 1$. While estimating the area of each mesh formed by the sampled points is possible, the actual computation is quite time consuming since it entails finding all the meshes formed (e. g. by Delaunay triangulation), and then computing the respective areas. Reducing this computation load to fit the real-time runtime constraint could improve the proposed event-based KS estimation, and we leave this for future work.

The decay strategy was experimentally determined by adapting the one proposed in Section 5.5.2, where the decay function is given by $d(\gamma, t_k) = \exp\left(-(t_k - t_{k-1})\gamma\right)$, where $\gamma = r_g^{t_{k-1}}$ and t_k is the timestamp of the current event. This decay strategy has the benefit of applying different decay weights to each segment depending on the responsibilities $r_g^{t_{k-1}}$.

1 https://igraph.org/c/

Table 6.1: The $\mathbf{C}_0$ parameter set for each sequence. We set $\mathbf{C}_0$ according to $\alpha\mathbf{I}$ and provide the respective α value for each sequence. For the remaining sequences $\alpha = 0.75$.

Dataset	Sequence	α
HKS	`iCub_hand`	0.10
	`yellow_crane`	0.20
3DKS	`donkey`	0.05
	`lamp`	0.50
	`pipe 1/2`	0.05
	`pipe 3/4`	0.05
EKS	`hand_fast`	0.25
	`human_fast`	0.01

A segment g is erased in case the corresponding mixing weight π_g is below a certain threshold. Equivalently, in practice, we define this threshold in terms of r_g (sum of responsibilities for segment g, according to Equation (6.9)) to be 0.99. For this case, this means that very few events are assigned to the segment (in fact, less than 1 after decay), and most probably it corresponds to an outlier. We also introduce another threshold in terms of r_g (ranging from 20 to 30) for a given segment to be considered a *proper* segment, from which the KS is built, and evaluations are performed.

6.5.3 *Quantitative Performance*

We quantitatively evaluate our method on five metrics: precision P, recall R, f-measure F, number of segments estimated n and execution time per sequence T^{comp}. The precision, recall and f-measure are given by Equation (3.14), which try to capture the trade-off between false positives and misses (Ochs et al., 2014). The overall evaluation of these metrics follows the description in Section 3.5.2.

Sparse feature points are provided as ground truth in the HKS dataset. As the proposed method does not rely on feature points, we compute the precision, recall and f-measure metrics by determining the spatially closest segment to each feature point. However, 3DKS and EKS datasets do not provide sparse feature points as ground-truth. Instead, we manually labelled each event according to the corresponding underlying segment. To be consistent, we computed the precision, recall and f-measure metrics as explained previously for each labelled event. We compare the proposed event-based KS estimation method (OnEKSM) against two state-of-the-art frame-based KS

estimation methods: Chang and Demiris (2018) (HKSM) and the frame-based method proposed in Chapter 4 (OnFKSM).

Table 6.2: Quantitative results on the HKS and 3DKS Datasets, in terms of average precision (P %), recall (R %) and f-measure (F %). Higher values are better.

		HKSM			OnEKSM		
Dataset	Sequence	P	R	F	P	R	F
HKS	Baxter	74.1	46.6	58.0	79.4	74.0	76.6
	dancing	92.9	53.9	68.0	78.3	80.9	79.6
	head	92.5	57.0	69.7	81.3	94.6	87.4
	iCub_body	83.7	49.5	62.0	76.9	73.9	75.4
	iCub_hand	72.5	59.3	65.0	70.1	58.5	63.8
	robot_arm	74.1	46.2	56.2	63.8	64.1	63.9
	yellow_crane	82.5	71.9	76.5	74.7	75.6	75.1

		OnFKSM			OnEKSM		
Dataset	Sequence	P	R	F	P	R	F
3DKS	donkey	88.7	79.2	83.6	90.0	83.5	86.6
	lamp	93.9	82.5	87.8	93.9	92.0	92.9
	pipe 1/2	97.0	94.2	95.5	96.3	87.6	91.7
	pipe 3/4	97.2	86.8	91.4	93.9	98.6	96.2

Overall, our method quantitatively outperforms the state-of-the-art frame-based methods on the HKS and 3DKS datasets, as presented in Table 6.2. This demonstrates that the proposed method is capable of capturing the KS from slow but complex and diverse motions. Additionally, not only does it works with 2D event-based data but can also readily take advantage of additional depth information if available without changing the method's architecture.

In Table 6.3, we present the quantitative results obtained for the number of segments and computation time for each sequence on the HKS and 3DKS Datasets. We provide the estimated number of segments normalised by the true number of segments (i. e. $N = n/n^{GT}$). Thus, values closer to 1 correspond to a better number of segments estimation. We also provide the computation time normalised by the actual length of each sequence (i. e. $T = T^{comp}/T^{seq}$). Values below 1 correspond to real-time processing. Overall, the proposed method compares favourably in terms of the number of segments estimated and achieves real-time processing in several sequences.

In Table 6.4, we present the proposed event-based method's quantitative performance on the new EKS Dataset. Overall, it achieves solid performance on these challenging sequences. Both frame-based methods (HKSM and OnFKSM) cannot cope with these sequences, exhibiting very rapid and diverse motions.

Table 6.3: Quantitative results on the HKS and 3DKS Datasets, in terms of the normalised number of segments estimated (N) and normalised computation time (T). Values of N closer to 1 are better, and values of T below 1 correspond to real-time processing.

		HKSM		OnEKSM	
Dataset	Sequence	N	T	N	T
HKS	`Baxter`	0.65	119.6	1.73	11.5
	`dancing`	0.89	77.6	1.00	2.47
	`head`	5.24	11.2	1.00	0.30
	`iCub_body`	0.77	145.4	1.00	0.80
	`iCub_hand`	0.725	9.1	1.13	0.66
	`robot_arm`	0.69	30.5	1.13	1.07
	`yellow_crane`	0.72	19.74	1.50	0.33

		OnFKSM		OnEKSM	
Dataset	Sequence	N	T	N	T
3DKS	`donkey`	1.37	2.41	1.00	6.60
	`lamp`	1.29	2.42	1.00	50.71
	`pipe 1/2`	1.05	2.85	1.00	4.78
	`pipe 3/4`	1.30	2.86	1.00	5.90

Table 6.4: Quantitative results on the EKS Dataset, in terms of average precision (P %), recall (R %) and f-measure (F %), normalised number of segments estimated (N) and normalised computation time (T). Higher values of P, R and F are better. Values of N closer to 1 are better, and values of T below 1 correspond to real-time processing.

		OnEKSM				
Dataset	Sequence	P	R	F	N	T
EKS	`arm_fast`	92.5	94.3	92.8	1.00	4.01
	`hand_fast`	85.1	80.5	82.7	1.29	15.02
	`human_fast`	80.2	62.8	70.4	0.83	2.92

6.5.4 *Qualitative Performance*

Figure 6.6 presents some qualitative results obtained by the proposed event-based KS estimation method. Overall, the proposed event-based method produces qualitative estimations of plausible articulated structures compared to the ground truth. The method can cope with a wide variety of motion ranges, as well as diverse kinds of spatial topological structures.

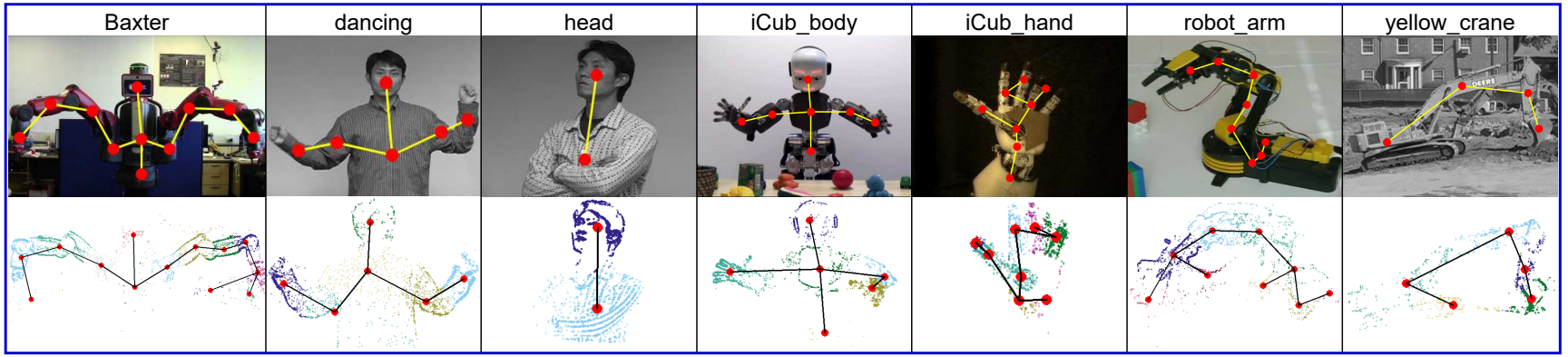
From Figure 6.6, we can see that the proposed method struggles to correctly identify body parts that exhibit a low density of events. This is clearly noticeable in the `Baxter`, `donkey`, `lamp` and `hand_fast` sequences. Particularly, the method fails to detect the lower part of `Baxter` and the palm in the

hand_fast sequence. Since the lower part of Baxter's topological structure is an extremity, the overall estimated KS still looks plausible. However, in the case of the hand_fast sequence, the palm can be considered as a root from which all other segments are connected. Since the method fails to segment it, the overall estimated KS is quite different from the ground-truth.

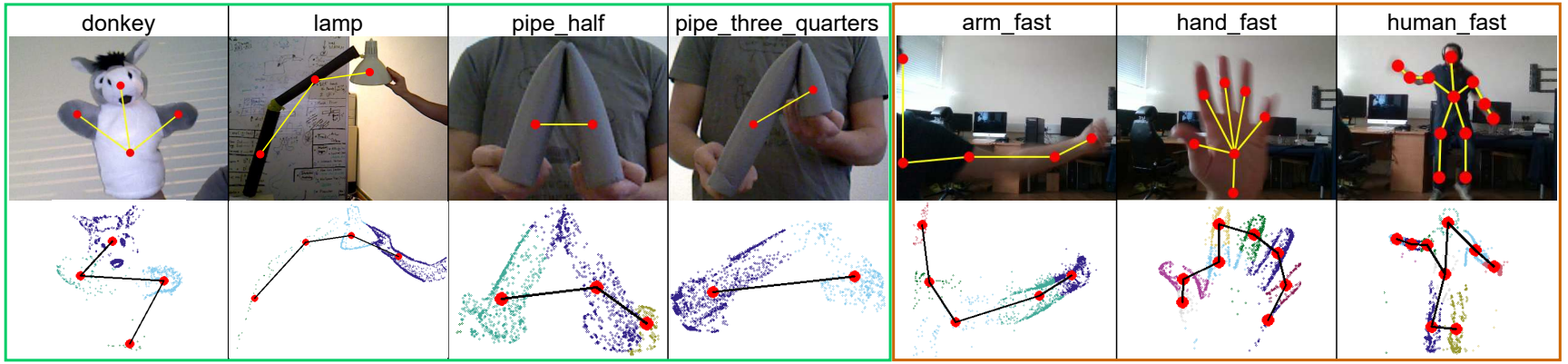
6.6 CONCLUSION

This chapter presented a novel approach for KS estimation of arbitrary articulated rigid objects for event cameras. We achieve this by proposing an incremental approach to jointly perform per-event shape-motion segmentation and estimating the underlying KS based on the inferred spatial body parts. The results obtained indicate that the proposed event-based method outperforms state-of-the-art frame-based methods on this task, further corroborating the suitability of event cameras to tackle motion-related problems in computer vision.

However, the proposed method has a few shortcomings, providing avenues for future work. Considering more sophisticated shapes may improve the method's performance, particularly if the underlying shape cannot be well represented by a Gaussian-like shape. Merging the spatially rich information from frame-based data may also help the KS estimation, especially in regions that exhibit low density of events, e. g. due to the lack of motion (lower part of Baxter sequence (Chang and Demiris, 2018)).



(a) HKS sequences



(b) 3DKS sequences

(c) EKS sequences

Figure 6.6: Qualitative results. The top rows of each block illustrate the object and respective KS. The bottom rows of each block show the corresponding segmented events and estimated KS. Figure best viewed in colour.

7

CONCLUSIONS

This chapter serves to conclude the thesis. First, this chapter presents an overview of the thesis by summarising the contributions made. Second, it discusses the limitations of the work developed. Lastly, this chapter highlights possible future research avenues that are mainly motivated by the limitations discussed previously.

7.1 OVERVIEW AND CONTRIBUTIONS OF THE THESIS

The primary focus of this thesis has been to develop unsupervised vision-based methods for Kinematic Structure (KS) estimation of articulated rigid bodies that are suitable for real-time applications. KS estimation methods rely on several other components, which also need to be considered for accurate estimations. By considering the whole KS estimation pipeline process, this thesis has advanced the state-of-the-art on various computer vision domains.

First, the problem of motion segmentation of articulated rigid bodies from semi-dense 3D points was cast into a Sparse Subspace Clustering (SSC) problem, whereby the number of rigid parts was estimated by performing Self-Tuning Spectral Clustering (STSC). This provided empirical evidence that initially representing an articulated rigid object from semi-dense 3D points was a valid compromise between accuracy and computational processing costs. However, similarly to previous methods that performed batch processing, the method could not handle lost points, e. g. due to tracking errors or partial occlusion.

This limitation motivated the development of an online KS estimation approach that maintained a suitable metric representation between all the semi-dense 3D points in a frame-by-frame fashion. The metric representation is exploited to naturally handle noise, e. g. due to points' tracking errors, and lost points, e. g. due to partial occlusion while maintaining an even distribution of representative points tracked. Experimental results showed that the metric representation outperforms previous representations, and the online KS estimation method achieves a performance on par with the offline method previously presented. Although the method can mitigate the effects of noise introduced by point tracking, it can not handle rapid or

sudden movements, which can naturally occur in real-world scenarios. This stems from the limited sampling rate of standard commercially available frame-based cameras.

To overcome the limited sampling rate problem, event cameras were thus considered for vision-based KS estimation. In several scenarios, it is reasonable to assume that events are mainly generated by occurring movements. An event-based motion estimation framework was developed by exploiting this observation, denoted Dispersion Minimisation (DMin) framework. Besides being able to estimate the parameters of several motion models without having to perform point tracking, the DMin framework was also designed to process events augmented with additional information, e. g. depth information. Previous event-based motion compensation approaches have not considered this. Additionally, a data whitening pre-processing step was proposed to address previously encountered limitations, and an incremental approach was derived from the general framework's formulation, allowing event-by-event processing. The incremental mode can achieve real-time performance without significantly decreasing the method's overall accuracy.

The first event-based KS estimation approach was then proposed. The method relied on performing incremental and joint shape-motion segmentation by building on the incremental mode of the DMin framework previously proposed. The estimated shape was exploited as an intermediate representation of the spatial distribution of the object to build the underlying KS. Experimental results further corroborated the trend that event cameras are outperforming standard frame-based cameras in vision-based motion-related problems.

These contributions sought to provide small steps in a long journey towards robust, fully automated and real-time KS estimation of articulated rigid objects for natural and fluid Human-Robot Interactions (HRIs). I hope that this will promote further research on vision-based KS estimation methods and real-world HRI scenarios where understanding an object's KS is informative. To that end, all the developed software packages have also been released under open-source code, and Appendix A.3 provides a list of the publicly available repositories.

7.2 LIMITATIONS

Although this thesis has advanced the state-of-the-art on vision-based KS estimation, the work developed still has limitations. This section discusses such limitations, which will largely motivate future research directions.

7.2.1 *Temporal Inconsistencies in Kinematic Structure Estimation*

The online/incremental KS methods may exhibit *temporal discontinuity* issues throughout the sequence. This typically manifests by previously correct connected rigid body parts being disconnected afterwards. Recall that for online/incremental methods, only the current state of the system is available, whereby a representation is recursively maintained and updated by encoding past states/information. Thus by construction, online/incremental methods only impose temporally local constraints (e. g. between consecutive frames). Additionally, past information is usually weighted according to a decay factor to give more relevance to new data. This procedure can be seen as a low-pass filter that smooths out noise and handles outliers. These two factors combined can lead to unexpected outcomes over time, and, in the context of KS estimation, they can manifest as mentioned above. Offline/batch methods are typically more robust to temporal inconsistencies since they can have a global view of the data. However, they are more computationally demanding and can only process relatively short sequences.

As pointed out in Section 4.8, robust point tracking is still an open problem in the computer vision field. Method's that rely on point tracking should hence mitigate the effects of point tracking errors, e. g. as described in Sections 4.5 and 4.6. However, there are still cases where point tracking completely fails (e. g. rapid and sudden movements), compromising the rest of the method's pipeline.

7.2.2 *Topological Inconsistencies in Kinematic Structure Estimation*

Topological inconsistent KSs may also be generated, e. g. as shown in Figure 6.6 for the `human_fast` sequence. The biharmonic distance (Lipman et al., 2010) can adequately approximate the distance between two points that live on an topological surface. However, the quality of the biharmonic distance depends on how the topological surface is represented. For example, in the case of event-based KS estimation, initially representing an object, e. g. by a set of points as in the frame-based case, is not a straightforward task to achieve. The main reason for this concerns the fact that events are typically generated at the object's boundary (see Figure 6.1 for an example). This leads to a certain ambiguity regarding what can be considered as the object's interior or exterior.

Section 6.3 tries to address this issue by assuming that the shape of each body part can be represented by a Gaussian-like shape. However, not all body

parts' shapes can be appropriately represented by a Gaussian-like shape. Another limitation with this approach is that the object's boundary is not explicitly estimated. Therefore, topological inconsistencies can still occur in cases where two topologically distant body parts are spatially very close.

7.2.3 *Distance Between Connected Links May Not Be Constant Over Time*

The distance between connected segments of an articulated rigid body remains constant over time. A version of the KS estimation method described in Chapter 6 was implemented imposing such constraint. However, in terms of performance, no improvements were obtained, and the presented version does not impose the constraint. A reason for this is related to the incremental estimation nature of the method, whereby it is difficult to have a global view based on which constraints could be more effectively imposed, as discussed previously in Section 7.2.1.

7.2.4 *Event-based Kinematic Structure Dataset*

Only three real sequences were recorded for the EKS Dataset, as presented in Section 6.5.1 and in Appendix E in more detail, while the remaining event-based sequences where simulated from previously frame-based recorded sequences (Mueggler et al., 2017). For evaluation purposes, additional real sequences should be directly recorded from event cameras. However, instead of performing per-event manual labelling, which is extremely time-consuming, other forms of labelling should be considered, e. g. tracking the Degrees of Freedoms (DOFs) of the object's body parts with a Motion Capture System. The downside of this approach, however, would be that per-event labelling would not be available. Thus, the experimental evaluation performed would have to be different from the one typically adopted in the literature.

7.3 FUTURE DIRECTIONS

The shortcomings of the presented research and immediate emergent future work have been addressed above. This section is devoted to expanding on possible future research directions related to KS estimation.

7.3.1 *Fusing Frames and Events for Motion Estimation*

Chapter 5 presents a framework for general event-based motion estimation, which can naturally handle augmented events, e. g. with depth information. The framework was thus evaluated on sequences where the events were augmented with depth values from depth maps provided by an RGB-D sensor. A simple approach was taken in which each event pixel coordinates was mapped to the available depth map to get the corresponding depth value. However, gross measurement errors are introduced since the pixel coordinates of the events generated may no longer correctly map to the coordinates of the previously captured frame. This issue is related to the frame-to-frame interpolation problem.

Approaches on how to fuse frame-based RGB data and events have been proposed (Scheerlinck et al., 2018; Wang et al., 2021). These methods focus on video reconstruction for high dynamic range scenarios by temporally interpolating RGB frames using events. It would be interesting to consider similar fusion procedures which focused on motion-related estimation scenarios. In particular, temporally interpolating depth maps using events is a promising research direction that has not been considered. This could mitigate some of the gross measurement errors mentioned above and, in turn, allow more accurate estimations from augmented events.

Open questions remain on whether augmenting the events with absolute intensity values can improve the estimates of motion-related tasks and how should such an approach be integrated. As argued throughout the thesis and supported by recent advances on event-based motion-related applications, events already carry much of the motion content. In principle, the intensity values should therefore be used to complement the events in situations where appearance and texture information could provide additional cues. For example, as briefly mentioned in Section 6.6, RGB frames could help the KS estimation in regions that do not generate many events due to the lack of motion, as it occurs for the `Baxter` sequence (Chang and Demiris, 2018).

7.3.2 *Hybrid Estimation Approach for Kinematic Structure Generation*

Closely related to the topic previously discussed is the possibility of merging high and low-level information in a hybrid approach (i. e. merging top-down and bottom-up approaches) to improve the KS generation process. Semantic and category level approaches have recently been proposed: Muthu et al. (2020) describe a motion segmentation method for frame-based RGB-D

sequences by combining semantic and motion information, whereby only coarse objects are segmented; category level information was considered by Li et al. (2020) to estimate the KS of simple objects of known categories.

Integrating these top-down approaches into the KS estimation process could be an interesting research avenue. However, instead of just staying at the object semantic/category level, another promising approach could be to consider one level lower, namely at the body part's level. This approach could immediately improve three encountered limitations discussed previously. First, temporal inconsistencies could be mitigated since the body parts could be labelled and then tracked based on a known model, which typically is more accurate. Topological consistency could also be improved by imposing constraints at the body parts' level on which models could be linked. Lastly, imposing a constant distance between connected segments would also be more straightforward.

7.3.3 *SLAM and Simultaneous Kinematic Structure Estimation of Multiple Objects*

Throughout this thesis, only single object sequences were considered. Although the KS estimation methods developed can, in principle, process multiple objects simultaneously, they were not designed with this possibility in mind. Thus, care should be taken in this regard. In particular, two issues mentioned in Section 7.2.2 should be addressed, namely considering more sophisticated shapes and explicitly modelling the object's boundaries.

Even more general would be to consider that the camera itself moves. This would entail modelling the camera motion as well. Several methods perform simultaneous rigid body segmentation and ego-motion estimation (Huang et al., 2019; Judd et al., 2018). However, the rigid body segmentation is performed at the object level, either by modelling the rigid body through semantics or motion consistency. Additionally, it is not clear whether such systems can achieve real-time performance. It would be interesting to have a real-time pipeline that could also automatically infer the objects' KS.

7.4 FINAL REMARKS

The KS of an articulated rigid body provides a compact and informative representation that fully discloses its motion capabilities. Having robust KS estimation methods is thus crucial for natural and fluid HRIs. The thesis advanced the state-of-the-art on KS estimation by developing novel methods for vision-based motion estimation and segmentation, and graph-based KS

generation, from frame-based and event-based data. While there are still challenges in deploying KS estimation methods for real-world applications, I hope the methods presented serve as small steps towards systems that, similarly to humans, are robust and work in real-time.

A

COMPONENTS, SENSORS AND SOFTWARE PACKAGES

In this work, several components and sensors were used, which are briefly described next. Open-source software packages that were developed on during this thesis are also outlined.

A.1 ROBOT OPERATING SYSTEM

The Robot Operating System (ROS) (Quigley et al., 2009) comprises an extensive collection of utilities for rapid development and integration of new programs that need to exchange data. At the time of writing of this thesis, it is the most popular middleware being used in robotics and computer vision. We used ROS to standardise the communication between different components and sensors, namely transfer data from sensors (e. g. frames, events from cameras) to a main device for further processing.

A.2 CAMERAS

We used two RGB-D cameras in this work, namely the Asus Xtion Pro and the Intel RealSense D415. The Asus Xtion Pro (Figure A.1, left) estimates the depth of the scene by relying on structured light information. The colour frames have 1280×1024 resolution, and the depth frames have 320×240 resolution, capturing reliable depth information between 0.8 and 3.5 meters. We used this camera to record the custom sequences in Chapter 4. The Intel RealSense D415 (Figure A.1, right) uses an active infrared stereo technology to estimate depth. The colour frames have 1920×1080 resolution, and the depth frames have 1280×720 resolution, capturing reliable depth information between 0.45 and 10 meters. We used this camera to record the event-based Kinematic Structure (KS) dataset presented in Chapter 6.

We also used the DAVIS346B event camera (Brandli et al., 2014) (Figure A.2) to record the event-based KS dataset presented in Chapter 6, and to show how event-based motion estimation via Dispersion Minimisation (DMin) can be achieved. The camera reports up to twelve million events per second, with a time resolution of $1\mu s$. An Inertial Measurement Unit (IMU) is also



Figure A.1: Asus Xtion Pro (left) and Intel RealSense D415 (right) RGB-D cameras.

integrated on the camera, which can provide 3-Degrees of Freedom (DOF) angular velocity and 3-DOF linear acceleration estimates at 200Hz.



Figure A.2: DAVIS346B event camera.

Additional specifications for all cameras are provided in Table A.1.

A.3 OPEN-SOURCE CONTRIBUTIONS

The following is a summary list of the open-source code repositories that are by-products of this work:

- **3DKSL**. Code repository for *Motion Segmentation of Articulated Rigid Objects from 3D Points*, presented in Chapter 3, and *Online Kinematic Structure Estimation of Articulated Rigid Objects from 3D Points*, presented in Chapter 4. Code publicly available here.
- **EventEMin**. Code repository for *Event-based Vision Motion Estimation by Dispersion Minimisation*, presented in Chapter 5. Code publicly available here.
- **event_emin_ros**. Code repository for *Event-based Vision Motion Estimation by Dispersion Minimisation* in real-time and incremental mode, presented in Appendix D.8. Code publicly available here.

Table A.1: Specifications of the cameras.

Sensors	Specifications		
DAVIS346B	Spatial resolution		346×260 APS+DVS
	Temporal resolution		1µs
	Max. throughput		12 MEPS
	Dynamic range		≈120 dB
RealSense D415	Depth	Spatial resolution	1280×720
		Frame rate	<90 FPS
		Field of view	64°×41°×72°
		Technology	Active IR Stereo
		Range	0.45 - 10 m
	RGB	Spatial resolution	1920×1080
		Frame rate	30 FPS
		Field of view	64°×41°×77°
		Technology	Rolling shutter
Xtion Pro	Depth	Spatial resolution	320×240
		Frame rate	60 FPS
		Field of view	58°×45°×70°
		Technology	Structured light
		Range	0.8 - 3.5 m
	RGB	Spatial resolution	1280×1024
		Frame rate	60 FPS
		Field of view	58°×45°×70°
		Technology	Rolling shutter

- **EventKSL**. Code repository for *Event-based Vision Incremental Kinematic Structure Estimation*, presented in Chapter 6. The code is not yet publicly available since the work is still in review at the time of writing.

B

MATHEMATICAL UTILITIES

Here, we provide a brief review of some probability distributions and processes that are useful for this work. We refer the reader to (Murphy, 2012) Chapters 2 and 25 for an in-depth exposition.

B.1 PROBABILITY DISTRIBUTIONS

B.1.1 *Gaussian Distribution*

The multivariate Gaussian distribution is a common joint probability density function for continuous variables. Its probability density function (pdf) in D dimensions is given by

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}(\mathbf{x}-\boldsymbol{\mu})\right]. \tag{B.1}$$

$\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}] \in \mathbb{R}^D$ is the mean vector and $\boldsymbol{\Sigma} = \mathrm{cov}[\mathbf{x}] \in \mathbb{R}^{D\times D}$ is the covariance matrix.

B.1.2 *Beta Distribution*

The Beta distribution has support over the interval $[0, 1]$, and is defined by

$$\mathrm{Beta}(x|a, b) = \frac{1}{B(a, b)} x^{a-1}(1-x)^{b-1}, \quad B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}, \tag{B.2}$$

where $a, b > 0$, $B(a, b)$ is the Beta function and $\Gamma(x)$ is the Gamma function

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du. \tag{B.3}$$

B.1.3 *Dirichlet Distribution*

The Dirichlet distribution is the multivariate generalisation of the Beta distribution. It has support over the K probability simplex, defined by

$$S_K = \left\{ \mathbf{x} : 0 \leqslant x_k \leqslant 1, \sum_{k=1}^{K} x_k = 1 \right\}, \tag{B.4}$$

and whose pdf is given by

$$\mathrm{Dir}(\mathbf{x}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} x_k^{\alpha_k - 1} \mathbb{I}(\mathbf{x} \in S_K). \tag{B.5}$$

$\boldsymbol{\alpha}$ are the distribution parameters, $\mathbb{I}(\cdot)$ represents the indicator function (i. e. 1 if its argument is true, and 0 otherwise), and $B(\boldsymbol{\alpha})$ is the generalisation of the Beta function to K variables

$$B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\alpha_0)}, \tag{B.6}$$

where $\Gamma(x)$ is the Gamma function and $\alpha_0 = \sum_{k=1}^{K} \alpha_k$.

Often a symmetric Dirichlet prior of the form $\alpha_k = \alpha/K$ is used, and thus the Dirichlet distribution is parameterised by a single parameter α.

B.1.4 *Inverse-Wishart Distribution*

The Inverse-Wishart distribution is a distribution over $D \times D$ positive definite matrices (e. g. covariances), parameterised by a parameter $\nu > D - 1$ and a positive semi-definite matrix $\mathbf{S}$. Its pdf is given by

$$\mathrm{IW}_\nu(\mathbf{X}|\mathbf{S}^{-1}) = \frac{1}{Z} |\mathbf{X}|^{-(\nu+D+1)/2} \exp\left\{ -\frac{1}{2} \mathrm{tr}(\mathbf{S}\mathbf{X}^{-1}) \right\}, \tag{B.7}$$

where Z is a normalisation constant

$$Z = \frac{|\mathbf{S}|^{\nu/2}}{2^{\nu D/2} \Gamma_D(\nu/2)}, \quad \Gamma_D(\nu/2) = \pi^{D(D-1)/4} \prod_{d=1}^{D} \Gamma\left(\frac{\nu + 1 - d}{2} \right). \tag{B.8}$$

B.1.5 *Normal-Inverse-Wishart Prior*

If we have a covariance matrix $\boldsymbol{\Sigma} \sim \mathrm{IW}_{\nu_0}(\boldsymbol{\Sigma}|\boldsymbol{\Lambda}_0^{-1})$ and a mean vector $\boldsymbol{\mu}|\boldsymbol{\Sigma} \sim \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}/\kappa_0)$, then the corresponding joint pdf is a Normal-inverse-Wishart distribution, given by

$$\begin{aligned} p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{\mu}_0, \kappa_0, \boldsymbol{\Lambda}_0, \nu_0) \\ &= \frac{1}{Z_0|\boldsymbol{\Sigma}|^{(\nu_0+D)/2+1}} \exp\left\{-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Lambda}_0\boldsymbol{\Sigma}^{-1}) - \frac{\kappa_0}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)\right\}, \end{aligned} \tag{B.9}$$

where Z_0 is a normalisation factor

$$Z_0 = \frac{2^{\nu_0 D/2}\Gamma_D(\nu_0/2)(2\pi/\kappa_0)^{D/2}}{|\boldsymbol{\Lambda}_0|^{\nu_0/2}}. \tag{B.10}$$

B.2 DIRICHLET PROCESS

A Dirichlet process (DP) is a distribution over probability measures G (where $G(x) \geqslant 0$ and $\int_X G(x)dx = 1$). It is implicitly defined by the requirement that $\big(G(x_1), G(x_2), \ldots, G(x_K)\big)$ has a joint Dirichlet distribution

$$\mathrm{Dir}\big(G(x_1), G(x_2), \ldots, G(x_K)|\alpha H(x_1), \alpha H(x_2), \ldots, \alpha H(x_K)\big) \tag{B.11}$$

for any finite partition $(x_1, x_2, \ldots, x_K)$, and we have that $G \sim \mathrm{DP}(\alpha, H)$, where α is the concentration parameter and H is the base measure.

B.2.1 *Stick Breaking Construction*

An infinite sequence of mixture weights $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^{\infty}$ derived from the process

$$\pi_k = \beta_k \prod_{i=1}^{k-1}(1-\beta_i) = \beta_k\left(1 - \sum_{i=1}^{k-1}\pi_i\right), \quad \beta_k \sim \mathrm{Beta}(1, \alpha), \tag{B.12}$$

is denoted by

$$\boldsymbol{\pi} \sim \mathrm{GEM}(\alpha). \tag{B.13}$$

This process terminates with probability 1, and the number of elements it generates increases with α, while the size of the π_k components decreases on average. If we define

$$G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\theta), \quad \boldsymbol{\pi} \sim \mathrm{GEM}(\alpha), \quad \theta_k \sim H(\boldsymbol{\lambda}), \tag{B.14}$$

then $G \sim \mathrm{DP}(\alpha, H)$.

B.2.2 *The Chinese Restaurant Process*

A more convenient approach to constructively define a DP is the Chinese restaurant process (CRP). Let z_i be a discrete variable that specifies which value of θ_k to use. Then, the predictive distribution of the next observation is given by

$$p(z_i = k | \mathbf{z}_{-i}, \alpha) = \begin{cases} \frac{N_{k,-i}}{\alpha + N - 1} & \text{if } k \text{ has been seen before} \\ \frac{\alpha}{\alpha + N - 1} & \text{if } k \text{ is a new cluster} \end{cases}, \tag{B.15}$$

where $\mathbf{z}_{-i} = \{z_j : z_j = k, j \neq i\}$, $N_{k,-i}$ is the number of data points assigned to cluster k, except for data point i and N is the number of data points.

B.2.3 *Dirichlet Process Mixture Model*

The DP is useful as a prior model for the parameters of a mixture model, when the number of mixtures (i. e. clusters) is unknown. This model can be constructed as follows:

$$\boldsymbol{\pi} \sim \mathrm{GEM}(\alpha), \tag{B.16}$$

$$z_i \sim \boldsymbol{\pi}, \tag{B.17}$$

$$\theta_k \sim H(\boldsymbol{\lambda}), \tag{B.18}$$

$$\mathbf{x}_i \sim F(\theta_{z_i}), \tag{B.19}$$

where $\boldsymbol{\pi} = \{\pi_k\}_{k=1}^{\infty}$ are the mixture weights, z_i specifies which value θ_k to use, θ_k are the parameters of the mixture k (with $\boldsymbol{\lambda}$ priors), and $\mathbf{x}_i$ corresponds to data point i.

We can estimate from which parameter θ_k data point $\mathbf{x}_i$ belongs to, by computing the full conditionals according to

$$p(z_i = k|\mathbf{z}_{-i}, \mathbf{x}, \alpha, \boldsymbol{\lambda}) \propto p(z_i = k|\mathbf{z}_{-i}, \alpha)p(\mathbf{x}_i|\mathbf{x}_{-i}, z_i = k, \mathbf{z}_{-i}, \boldsymbol{\lambda}). \tag{B.20}$$

The first term is given by Equation (B.15). The second term is the posterior predictive distribution for cluster k evaluated for data point $\mathbf{x}_i$. To compute the second term, we first partition the data $\mathbf{x}_{-i}$ into clusters based on $\mathbf{z}_{-i}$, such that $\mathbf{x}_{-i,k} = \{\mathbf{x}_j : z_j = k, j \neq i\}$ are the data assigned to cluster k. If $z_i = k$, then data point $\mathbf{x}_i$ is conditionally independent of all other data points except those assigned to cluster k, and thus the second term is given by

$$p(\mathbf{x}_i|\mathbf{x}_{-i}, z_i = k, \mathbf{z}_{-i}, \boldsymbol{\lambda}) = p(\mathbf{x}_i|\mathbf{x}_{-i,k}, \boldsymbol{\lambda}) = \frac{p(\mathbf{x}_i, \mathbf{x}_{-i,k}|\boldsymbol{\lambda})}{p(\mathbf{x}_{-i,k}|\boldsymbol{\lambda})}, \tag{B.21}$$

where

$$p(\mathbf{x}_i, \mathbf{x}_{-i,k}|\boldsymbol{\lambda}) = \int p(\mathbf{x}_i|\theta_k)H(\theta_k|\boldsymbol{\lambda}) \prod_{j \neq i : z_j = k} p(\mathbf{x}_j|\theta_k)d\theta_k \tag{B.22}$$

is the marginal likelihood of all the data assigned to cluster k, including i. If the value of z_i corresponds to a new cluster k^*, then the second term is given by the prior predictive distribution of a new cluster for data point $\mathbf{x}_i$

$$p(\mathbf{x}_i|\mathbf{x}_{-i}, z_i = k^*, \mathbf{z}_{-i}, \boldsymbol{\lambda}) = p(\mathbf{x}_i|\boldsymbol{\lambda}) = \int p(\mathbf{x}_i|\theta)H(\theta|\boldsymbol{\lambda})d\theta. \tag{B.23}$$

The incremental joint optimisation procedure described in Section 6.2 is based on the formalism above, whereby the appropriate probability distributions were substituted and additional assumptions were made.

C

DISPERSION MINIMISATION FRAMEWORK: APPLICATIONS AND MODELS

The proposed Dispersion Minimisation (DMin) framework is versatile and flexible, being able to tackle a wide range of event-based vision motion estimation problems. To demonstrate this, we provide details on how to use it to solve several vision-based motion-related estimation problems. The problems presented do not represent an exhaustive list.

C.1 2D TRANSLATIONAL MOTION ESTIMATION

The 2D translational motion model has 2-Degrees of Freedom (DOF) and can be parameterised by the 2D linear velocity $\theta = (v_x, v_y)^T$, being expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta) = \mathbf{y}_i - \Delta t_i \theta, \tag{C.1}$$

where $\mathbf{f}_i = \mathbf{x}'_i$ represents the image coordinates of the transformed events, $\mathbf{y}_i = \mathbf{x}_i$ represents the image coordinates of the original events and $\Delta t_i = t_i - t_{ref}$ (t_{ref} is some reference timestamp).

To use Equation (5.22), we need to specify a modified version of the model $\tilde{\mathcal{M}}(e_i; \theta)$, such that the transformed feature $\mathbf{f}_i$ can be written according to Equation (5.21). By inspection, Equation (C.1) is already in the form of Equation (5.21), where

$$\tilde{\mathbf{f}}_i = \mathbf{y}_i, \quad \mathbf{Y}_i = -\Delta t_i. \tag{C.2}$$

Substituting Equation (C.2) into Equation (5.22), we obtain

$$\theta^* = \frac{\sum_{i,j}^{N_e} w_{i,j} \Delta t_{i,j} \Delta \mathbf{y}_{i,j}}{\sum_{i,j}^{N_e} w_{i,j} (\Delta t_{i,j})^2}, \tag{C.3}$$

since $\frac{\partial \Delta \mathbf{f}_{i,j}^T}{\partial \theta} = -\Delta t_{i,j} \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

A similar least-squares equation was obtained by Zhu et al. (2017), as the M-step of an Expectation-Maximisation (EM) solution to optical flow estimation over a spatial neighbourhood. The E-step involved estimating the events' association probabilities, which boiled-down to estimating the weights $w_{i,j}$. Under the proposed framework, this step is skipped, since we

compute these association weights according to a Gaussian kernel given by Equation (5.1).

C.2 ROTATIONAL MOTION ESTIMATION

The rotational model has 3-DOF and can be parameterised by the 3D angular velocity $\theta = (w_x, w_y, w_z)^\top$, being expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta) = \frac{1}{d_i}\mathbf{P}\mathcal{R}^{-1}(t_i; \theta)\begin{pmatrix}\mathbf{y}_i \\ 1\end{pmatrix}, \tag{C.4}$$

where d_i is a normalisation factor, $\mathbf{y}_i = \mathbf{P}\mathbf{C}^{-1}\begin{pmatrix}\mathbf{x}_i^\top & 1\end{pmatrix}^\top = \begin{pmatrix}\bar{x}_i & \bar{y}_i\end{pmatrix}^\top$, $\mathbf{P}$ is a projection matrix and $\mathbf{C}$ is the camera intrinsic matrix. The rotation matrix $\mathcal{R} \in SO(3)$ can be written as a matrix exponential map of the angular velocity parameters θ as

$$\mathcal{R}(t_i; \theta) = \exp\left(\Delta t_i \hat{\theta}\right), \ \hat{\theta} = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix}, \tag{C.5}$$

where $\hat{\theta}$ represents the linear combination of the rotation generators (according to the Lie algebra).

To obtain the form of Equation (5.21), we note that we can subtract and add the linear combination of the rotation generators $\hat{\theta}$ (i. e. $\mathbf{G}(t_i; \theta) = \Delta t_i \hat{\theta}$) from the rotation matrix as

$$\begin{aligned}\mathbf{f}_i &= \frac{1}{d_i}\mathbf{P}\left(\mathcal{R}^{-1}(t_i; \theta) - \Delta t_i\hat{\theta} + \Delta t_i\hat{\theta}\right)\begin{pmatrix}\mathbf{y}_i \\ 1\end{pmatrix} \\ &= \underbrace{\frac{1}{d_i}\mathbf{P}\left(\mathcal{R}^{-1}(t_i; \theta) - \Delta t_i\hat{\theta}\right)\begin{pmatrix}\mathbf{y}_i \\ 1\end{pmatrix}}_{\tilde{\mathcal{M}}(e_i;\theta)} + \underbrace{\frac{\Delta t_i}{d_i}\mathbf{P}\hat{\theta}\begin{pmatrix}\mathbf{y}_i \\ 1\end{pmatrix}}_{\mathbf{G}(t_i;\theta)\mathbf{y}_i = \mathbf{Y}_i\theta}.\end{aligned} \tag{C.6}$$

After some algebraic manipulation, we obtain

$$\mathbf{Y}_i = \frac{\Delta t_i}{d_i}\begin{pmatrix} 0 & 1 & -\bar{y}_i \\ -1 & 0 & \bar{x}_i \end{pmatrix}. \tag{C.7}$$

For several motion estimation problems whose model can be described by Lie groups, this approach can be used and adapted to reach the form of Equation (5.21) from the general model given by Equation (5.2).

C.3 ISOMETRY TRANSFORMATION ESTIMATION

The isometry transformation has 3-DOF and can be parameterised by the 2D linear velocity $\mathbf{v} = (v_x, v_y)^\mathsf{T}$ and the angular velocity w, being expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta) = \mathbf{P}\mathcal{I}^{-1}(t_i; \theta) \begin{pmatrix} \mathbf{y}_i \\ 1 \end{pmatrix}, \tag{C.8}$$

where $\theta = (w, \mathbf{v}^\mathsf{T})^\mathsf{T}$ are the model parameters. The isometry matrix $\mathcal{I} \in \mathrm{SE}(2)$ can be written as a matrix exponential map of the model parameters θ as

$$\mathcal{I}(t_i; \theta) = \exp\left(\Delta t_i \hat{\theta}\right), \quad \hat{\theta} = \begin{pmatrix} 0 & -w & v_x \\ w & 0 & v_y \\ 0 & 0 & 0 \end{pmatrix}. \tag{C.9}$$

Applying the approach described in Appendix C.2, we obtain

$$\mathbf{Y}_i = \Delta t_i \begin{pmatrix} -\bar{y}_i & 1 & 0 \\ \bar{x}_i & 0 & 1 \end{pmatrix}. \tag{C.10}$$

C.4 SIMILARITY TRANSFORMATION ESTIMATION

The similarity transformation is an isometry transformation with an additional isotropic scaling $\lambda_i = 1 + \Delta t_i s$, being expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta) = \mathbf{P}\mathcal{S}^{-1}(t_i; \theta) \begin{pmatrix} \mathbf{y}_i \\ 1 \end{pmatrix}, \tag{C.11}$$

where $\theta = (w, s, \mathbf{v}^T)^T$ are the model parameters. The similarity matrix $\mathcal{S} \in \mathrm{Sim}(2)$ can be written as a matrix exponential map of the model parameters θ as

$$\mathcal{S}(t_i; \theta) = \exp\left(\Delta t_i \hat{\theta}\right), \quad \hat{\theta} = \begin{pmatrix} s & -w & v_x \\ w & s & v_y \\ 0 & 0 & 0 \end{pmatrix}. \tag{C.12}$$

Applying the approach described in Appendix C.2, we obtain

$$\mathbf{Y}_i = \Delta t_i \begin{pmatrix} -\bar{y}_i & \bar{x}_i & 1 & 0 \\ \bar{x}_i & \bar{y}_i & 0 & 1 \end{pmatrix}. \tag{C.13}$$

C.5 AFFINE TRANSFORMATION ESTIMATION

The affinity is a similarity transformation with non-isotropic scaling, which is accounted by two additional DOF, namely, the angle $\gamma_i = \Delta t_i w_\gamma$ that specifies the scaling direction and the ratio between the scaling parameters $\lambda_{1,i}/\lambda_{2,i} = (1 + \Delta t_i s_1)/(1 + \Delta t_i s_2)$. It is expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta) = \mathbf{P}\mathcal{A}^{-1}(t_i; \theta) \begin{pmatrix} \mathbf{y}_i \\ 1 \end{pmatrix}, \tag{C.14}$$

where $\theta = (w, w_\gamma, s_1, s_2, \mathbf{v}^T)^T$ are the model parameters. The affinity matrix $\mathcal{A} \in \mathrm{Aff}(2)$ can be written as a matrix exponential map of the model parameters θ as

$$\mathcal{A}(t_i; \theta) = \exp\left(\Delta t_i \hat{\theta}\right), \hat{\theta} = \begin{pmatrix} s_1 + s_2 & w_\gamma - w & v_x \\ w_\gamma + w & s_1 - s_2 & v_y \\ 0 & 0 & 0 \end{pmatrix}. \tag{C.15}$$

Applying the approach described in Appendix C.2, we obtain

$$\mathbf{Y}_i = \Delta t_i \begin{pmatrix} -\bar{y}_i & \bar{x}_i & \bar{x}_i & \bar{y}_i & 1 & 0 \\ \bar{x}_i & \bar{y}_i & -\bar{y}_i & \bar{x}_i & 0 & 1 \end{pmatrix}. \tag{C.16}$$

C.6 MOTION ESTIMATION IN PLANAR SCENES

Motion estimation in planar scenes can be achieved by using the homography model, which has 8-DOF. It can be parameterised by the 3D angular velocity $\mathbf{w} = (w_x, w_y, w_z)^\mathsf{T}$, the up to scale 3D linear velocity $\bar{\mathbf{v}} = \mathbf{v}/d = (\bar{v}_x, \bar{v}_y, \bar{v}_z)^\mathsf{T}$ and the normalised 3D normal vector $\mathbf{n} = (n_x, n_y, n_z)^\mathsf{T}$ of the inducing plane $\boldsymbol{\pi} = (\mathbf{n}^\mathsf{T}, d)^\mathsf{T}$, being expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i; \theta) = \frac{1}{d_i}\mathbf{P}\mathcal{H}^{-1}(t_i; \theta)\begin{pmatrix}\mathbf{y}_i \\ 1\end{pmatrix}, \tag{C.17}$$

where $\theta = (\mathbf{w}^\mathsf{T}, \bar{\mathbf{v}}^\mathsf{T}, \mathbf{n}^\mathsf{T})^\mathsf{T}$ are the model parameters and the homography matrix $\mathcal{H}$ can be written in function of the model parameters θ as

$$\mathcal{H}(t_i; \theta) = \mathcal{R}(t_i; \mathbf{w}) - \Delta t_i \bar{\mathbf{v}}\mathbf{n}^\mathsf{T}. \tag{C.18}$$

C.6.1 *Why There is No Trivial Procedure to Linearise the Parameters of Motion Estimation in Planar Scenes*

The approach described in Appendix C.2 to decompose Equation (5.2) such that it linearly depends on the motion parameters of interest can not be used for the case of motion estimation in planar scenes. Currently, we are unaware of any methodology to achieve that end and acknowledge that this is an interesting problem for future research. Next, we briefly explain why the approach described in Appendix C.2 does not work in this case.

Expanding Equation (C.18), we obtain

$$\begin{aligned}\mathcal{H}(t_i; \theta) &= \mathcal{R}(t_i; \mathbf{w}) - \Delta t_i \bar{\mathbf{v}}\mathbf{n}^\mathsf{T} \\ &= \mathcal{R}(t_i; \mathbf{w}) - \Delta t_i \begin{pmatrix}\bar{v}_x n_x & \bar{v}_x n_y & \bar{v}_x n_z \\ \bar{v}_y n_x & \bar{v}_y n_y & \bar{v}_y n_z \\ \bar{v}_z n_x & \bar{v}_z n_y & \bar{v}_z n_z\end{pmatrix},\end{aligned} \tag{C.19}$$

where the normal vector $\mathbf{n}$ can be written in spherical coordinates, according to

$$\begin{cases} n_x = \cos\psi \sin\phi \\ n_y = \sin\psi \sin\phi \\ n_z = \cos\phi \end{cases}, \quad \psi \in [0, 2\pi],\ \phi \in [0, \pi]. \tag{C.20}$$

We can see now that it is not straightforward to obtain a linear combination of the parameters ψ and ϕ since these are passed to non-linear functions, e. g. cos and sin.

However, even if we do not consider the normal vector as a parameter, it is not straightforward to employ the approach described in Appendix C.2. In this case, we again consider the homography model, where the normal vector is fixed, such that the direction of the linear velocity is always perpendicular to the camera plane, i. e. $\mathbf{n} = (0, 0, -1)^\mathsf{T}$. Substituting on Equation (C.19), we obtain

$$\mathcal{H}_n(t_i; \theta) = \mathcal{R}(t_i; \mathbf{w}) + \Delta t_i \begin{pmatrix} 0 & 0 & \bar{v}_x \\ 0 & 0 & \bar{v}_y \\ 0 & 0 & \bar{v}_z \end{pmatrix}. \tag{C.21}$$

Following Appendix C.2, we subtract and add the linear combination of the rotation generators plus the second term of Equation (C.19), which is denoted as $\hat{\theta}$, from the homography matrix as

$$\begin{aligned} \mathbf{f}_i &= \frac{1}{d_i}\mathbf{P}\left(\mathcal{H}_n^{-1}(t_i; \theta) - \Delta t_i \hat{\theta} + \Delta t_i \hat{\theta}\right)\begin{pmatrix} \mathbf{y}_i \\ 1 \end{pmatrix} \\ &= \underbrace{\frac{1}{d_i}\mathbf{P}\left(\mathcal{H}_n^{-1}(t_i; \theta) - \Delta t_i \hat{\theta}\right)\begin{pmatrix} \mathbf{y}_i \\ 1 \end{pmatrix}}_{\tilde{\mathcal{M}}(e_i; \theta)} + \underbrace{\frac{\Delta t_i}{d_i}\mathbf{P}\hat{\theta}\begin{pmatrix} \mathbf{y}_i \\ 1 \end{pmatrix}}_{\mathbf{G}(t_i; \theta)\mathbf{y}_i = \mathbf{Y}_i \theta}, \end{aligned} \tag{C.22}$$

where

$$\hat{\theta} = \begin{pmatrix} 0 & -w_z & \bar{v}_x + w_y \\ w_z & 0 & \bar{v}_y - w_x \\ -w_y & w_x & \bar{v}_z \end{pmatrix}. \tag{C.23}$$

After some manipulation, we obtain

$$\mathbf{Y}_i = \frac{\Delta t_i}{d_i}\begin{pmatrix} 0 & 1 & -\bar{y}_i & 1 & 0 & 0 \\ -1 & 0 & \bar{x}_i & 0 & 1 & 0 \end{pmatrix}. \tag{C.24}$$

The matrix $\mathbf{Y}_i$ given by Equation (C.24) is problematic since all the linear factors related to $\bar{v}_z$ are zero, i. e. the last column only contains zeros. This means that $\bar{v}_z$ is not accounted for during the estimation. The projection matrix $\mathbf{P}$ is responsible for this outcome since it eliminates the last row of the matrix given by Equation (C.23), and thus eliminates the dependency on $\bar{v}_z$ from the matrix $\mathbf{Y}_i$. We note that the choice of the matrix $\hat{\theta}$ is not

unique. However, we did not find any such matrix that solved the issue from Equation (C.24) and yielded stable estimates.

At the cost of interpretability, we could also consider another parameterisation from the literature on Lie groups. Such parameterisation can be viewed as an affine transformation (please refer to Appendix C.5 for additional details) with perspective changes, which are accounted by two additional DOF, denoted by p_1 and p_2. The homography matrix $\mathcal{H} \in \mathrm{SL}(3)$ can thus be written as a matrix exponential map of the model parameters $\theta = \left(w, w_\gamma, s_1, s_2, p_1, p_2, \mathbf{v}^\mathsf{T}\right)^\mathsf{T}$ as

$$\mathcal{H}(t_i;\theta) = \exp\left(\Delta t_i \hat{\theta}\right), \; \hat{\theta} = \begin{pmatrix} s_1 + s_2 & w_\gamma - w & v_x \\ w_\gamma + w & s_1 - s_2 & v_y \\ p_1 & p_2 & -2s_1 \end{pmatrix}. \tag{C.25}$$

We can immediately see that this parameterisation would also not work, the reason being that the last row would be eliminated due to the projection matrix $\mathbf{P}$, according to Equation (C.22). Consequently, the matrix $\mathbf{G}(t_i;\theta)$ would not depend on p_1 and p_2 since

$$\begin{aligned} \mathbf{G}(t_i;\theta) &= \frac{\Delta t_i}{d_i}\mathbf{P}\hat{\theta} \\ &= \frac{\Delta t_i}{d_i}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} s_1 + s_2 & w_\gamma - w & v_x \\ w_\gamma + w & s_1 - s_2 & v_y \\ p_1 & p_2 & -2s_1 \end{pmatrix} \\ &= \frac{\Delta t_i}{d_i}\begin{pmatrix} s_1 + s_2 & w_\gamma - w & v_x \\ w_\gamma + w & s_1 - s_2 & v_y \end{pmatrix}. \end{aligned} \tag{C.26}$$

C.7 6-DOF ESTIMATION

The 6-DOF can be parameterised by the 3D angular velocity $\mathbf{w} = (w_x, w_y, w_z)^\mathsf{T}$ and the 3D linear velocity $\mathbf{v} = (v_x, v_y, v_z)^\mathsf{T}$, being expressed as

$$\mathbf{f}_i = \mathcal{M}(e_i;\theta) = \mathcal{T}^{-1}(t_i;\theta)\mathbf{z}_i, \tag{C.27}$$

where $\mathbf{z}_i = \begin{pmatrix} \bar{x}_i & \bar{y}_i & \bar{z}_i \end{pmatrix}^\mathsf{T}$ and $\mathbf{f}_i$ represent the 3D coordinates of the original and transformed events augmented with depth information, respectively. The

matrix exponential map $\mathcal{T} \in SE(3)$ encodes the rigid body transformation and can be written in function of $\theta = (\mathbf{w}^T, \mathbf{v}^T)^T$ as

$$\mathcal{T}(t_i; \theta) = \exp\left(\Delta t_i \hat{\theta}\right), \ \hat{\theta} = \begin{pmatrix} 0 & -w_z & w_y & v_x \\ w_z & 0 & -w_x & v_y \\ -w_y & w_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{C.28}$$

Applying the approach described in Appendix C.2, we obtain

$$\mathbf{Y}_i = \Delta t_i \begin{pmatrix} 0 & \bar{z}_i & -\bar{y}_i & 1 & 0 & 0 \\ -\bar{z}_i & 0 & \bar{x}_i & 0 & 1 & 0 \\ \bar{y}_i & -\bar{x}_i & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{C.29}$$

D

DISPERSION MINIMISATION FRAMEWORK: ADDITIONAL RESULTS

D.1 IMPLEMENTATION DETAILS

The proposed Dispersion Minimisation (DMin) framework is implemented in C++. For the batch mode optimisation, we use the CG-FR algorithm of the scientific library GNU-GSL[1] for the optimisation and the Auto-Diff module of the Eigen library[2] for the (automatic) derivatives computation.

The current implementation is designed to favour usability and scalability, rather than performance due to developmental reasons (please refer to Appendix D.8 to achieve real-time performance using the incremental mode). In batch mode, the DMin framework converges on average in 4-5 line-searches for rotational motion estimation considering 20000 events. The time cost of evaluating the approximate entropy-based function and corresponding derivatives is ~15ms and ~25ms, respectively, on a single-threaded 3.7Ghz CPU. For comparison, our implementation of the Variance cost function (Gallego et al., 2018b) and corresponding derivatives takes ~12ms and ~21ms to be evaluated in the same settings, respectively. The framework is highly parallelisable and further speed-ups can be achieved. This includes the exact entropy, which can also achieve competitive computational costs for specific problems where a few ~ 1000 of events are processed.

D.2 OPTIMISATION PROCEDURE

Here, we briefly describe the optimisation procedure for the batch mode, namely the conjugate gradient Fletcher-Reeves (CG-FR) algorithm. We will then explain why this procedure is not suitable for the incremental mode optimisation, which will motivate the linearised optimisation approach described in Section 5.3. As noted in Section 5.3, although we could also employ the linearised optimisation approach for the batch mode, we instead use the CG-FR algorithm mainly due to legacy reasons since we started by using the CG-FR algorithm to optimise the batch-based measures. Another reason is

1 https://www.gnu.org/software/gsl
2 https://eigen.tuxfamily.org

that conjugate gradient optimisation methods have better properties, such as convergence, and thus typically obtain better estimates. The development of the dedicated optimisation procedure described in Section 5.3 was thus solely motivated by the need to have a method that was capable of incrementally optimise the motion parameters in an event-by-event fashion.

Briefly, the CG-FR optimisation algorithm works as follows. Given a function $f(\mathbf{x})$ to minimise w. r. t. $\mathbf{x}$, the best estimate $\mathbf{x}_n$ is iteratively updated according to

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{s}_n. \tag{D.1}$$

α_n is a scalar that adjusts the length of step $\mathbf{s}_n$ (e. g. in the neural networks community, this scalar is also referred as the *learning rate*). For the CG-FR algorithm this scalar is computed by performing a line search:

$$\alpha_n = \arg\min_{\alpha} f(\mathbf{x}_n + \alpha \mathbf{s}_n). \tag{D.2}$$

The step $\mathbf{s}_n$ is given by

$$\mathbf{s}_n = -\nabla_{\mathbf{x}} f(\mathbf{x}_n) + \beta_n \mathbf{s}_{n-1}, \tag{D.3}$$

where $\nabla_{\mathbf{x}}$ is the gradient operator and β_n is a scalar parameter. The CG-FR algorithm internally estimates the parameters α and β, and it requires the gradient and an initial estimate $\mathbf{x}_0$ to be provided. We provide the gradient using the Auto-Diff module of the Eigen library, and the initial estimate for each event batch is the solution found for the previous one (for the first batch, the initial estimate consists of zeros, i. e. $\mathbf{x}_0 = \mathbf{0}$). We note that most multidimensional minimisation algorithms work in similar ways, the only major difference being on how to select the adjustable step length α and the parameter β.

There are three main reasons this optimisation approach is not suitable to be applied in an event-by-event fashion. First, the global gradient of the function needs to be evaluated for each iteration, according to Equation (D.3), as well as the function itself, according to Equation (D.2). However, it is not clear how one would incrementally maintain the gradient and the function value, or even if such an approach would yield stable estimates. Second, even if we assume that the gradient and function could be incrementally maintained (possibly in an adapted manner similar to how the quantities $\boldsymbol{\psi}$ and $\boldsymbol{\Psi}$ in Equation (5.22) are maintained), the overhead introduced by the optimisation approach would be significantly higher compared to the computational cost of evaluating the function and respective gradient. For instance, for the batch

mode, the optimisation overhead introduced is negligible compared to the computational cost of evaluating the function and respective gradient. Third, the parameters α and/or β may introduce artificial estimation stability issues when optimising in online mode.

The optimisation approach described in Section 5.3 is tailored to overcome the possible issues encountered previously. Equation (5.22) is also in the form of Equation (D.1) since

$$\begin{aligned} \theta^* = \Psi^{-1}\psi \;\equiv\; \mathbf{x}_{n+1} &= \gamma_n \\ &= \mathbf{x}_n + \gamma_n - \mathbf{x}_n = \mathbf{x}_n + \mathbf{s}_n \end{aligned} \tag{D.4}$$

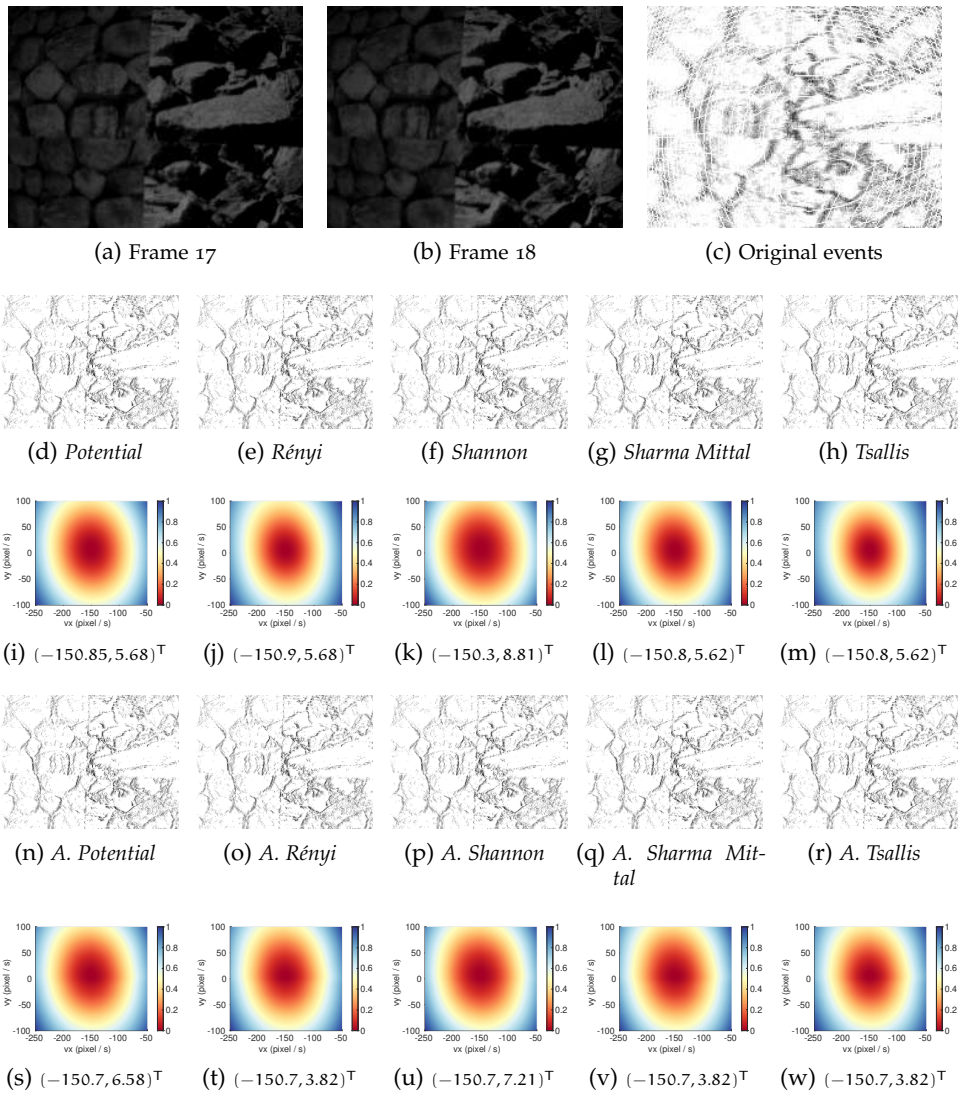
Distinctively, it does not depend on extra parameters since it implicitly adjusts the step length at each iteration. However, the motion parameters must be explicitly linearised, according to Equation (5.21).

D.3 ADDITIONAL RESULTS ON 2D TRANSLATIONAL MOTION ESTIMATION

Figure D.1 presents additional plots regarding 2D translational motion estimation using the proposed DMin framework. The 2D translational motion is estimated between two consecutive frames of the `poster_translation` sequence (Mueggler et al., 2017). In Figure D.1c, we show the original events, that were produced by the motion between the frames, projected onto the IWE without motion compensation. The resulting image is blurry exhibiting low contrast. By estimating the 2D translational motion model parameters, if we project the transformed events onto the IWE, we get a sharper image with higher contrast. Similar results are obtained using the Contrast Maximisation (CMax) framework (Gallego et al., 2018b). Distinctively, however, our framework does not directly optimise the contrast measure of the IWE. Instead, it minimises the dispersion of the transformed events in the feature space. Additionally, the measure profiles of the exact functions and respective approximations in function of the 2D translational motion parameters are almost identical.

D.4 ADDITIONAL RESULTS ON ROTATIONAL ESTIMATION

Table D.1 presents additional quantitative results on rotational motion estimation. The angular velocity errors for each component ($e_{w_x}, e_{w_y}, e_{w_z}$), their standard deviation (σ_{e_w}) and Root Mean Square (RMS) are presented



(a) Frame 17 (b) Frame 18 (c) Original events

(d) *Potential* (e) *Rényi* (f) *Shannon* (g) *Sharma Mittal* (h) *Tsallis*

(i) $(-150.85, 5.68)^T$ (j) $(-150.9, 5.68)^T$ (k) $(-150.3, 8.81)^T$ (l) $(-150.8, 5.62)^T$ (m) $(-150.8, 5.62)^T$

(n) *A. Potential* (o) *A. Rényi* (p) *A. Shannon* (q) *A. Sharma Mittal* (r) *A. Tsallis*

(s) $(-150.7, 6.58)^T$ (t) $(-150.7, 3.82)^T$ (u) $(-150.7, 7.21)^T$ (v) $(-150.7, 3.82)^T$ (w) $(-150.7, 3.82)^T$

Figure D.1: 2D translational motion estimation between frames 17 and 18 of the `poster_translation` sequence (Mueggler et al., 2017). The motion is dominated by the horizontal component: $\theta^* \approx (-150, 0)^T$. All methods tend to maximise the Image of Warped Events (IWE) contrast, which is reflected in the motion-compensated images (D.1d)-(D.1h) and (D.1n)-(D.1r). (D.1i)-(D.1m) Normalised profiles for the proposed *Potential* energy and exact entropy-based measures. (D.1s)-(D.1w) Normalised profiles for the proposed efficient approximate measures. The normalised profiles are plotted in function of the 2D translational motion parameters. We can see that all of the proposed functions produce similar results, both in terms of contrast measures and model parameters estimated.

in deg/s, w. r. t. Inertial Measurement Unit (IMU) measurements, as well as the RMS error compared to the maximum excursions in percentage (RMS %), considering batches of 20000 events. For the online measures, we instead maintain the most recent 10000 events and consider a 9×9 spatial neighbour-

Table D.1: Rotational motion estimation accuracy comparison on the `boxes_rotation` and `poster_rotation` sequences (Mueggler et al., 2017). The processing time per event T_e is also reported in micro-seconds. The best value per column is highlighted in bold.

	`boxes_rotation`						`poster_rotation`					
Function	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %
Variance	**7.51**	**6.43**	7.07	9.96	10.05	1.37	13.53	**8.64**	6.90	13.89	13.97	1.49
Potential	7.69	7.06	**6.49**	9.34	9.43	1.29	14.23	8.81	**6.48**	13.05	13.14	1.40
Rényi	7.91	7.53	6.50	9.48	9.56	1.30	14.55	9.30	6.54	13.38	13.45	1.44
Shannon	7.57	6.75	6.88	**9.28**	**9.40**	**1.28**	13.79	8.65	6.99	**12.94**	**13.08**	**1.40**
Sharma-Mittal	7.92	7.53	6.54	9.54	9.63	1.31	14.55	9.30	6.53	13.37	13.44	1.43
Tsallis	7.91	7.53	6.51	9.48	9.56	1.30	14.55	9.30	6.54	13.37	13.44	1.43
Approx. Potential	7.91	7.07	7.78	9.94	10.04	1.37	13.94	8.91	7.69	13.51	13.62	1.45
Approx. Rényi	8.13	7.41	7.92	10.25	10.34	1.41	14.19	9.25	7.84	13.82	13.90	1.48
Approx. Shannon	7.84	6.94	7.94	9.96	10.07	1.37	13.70	8.77	8.00	13.58	13.72	1.46
Approx. Sharma-Mittal	8.15	7.39	7.98	10.22	10.30	1.40	14.19	9.23	7.91	14.03	14.12	1.51
Approx. Tsallis	8.13	7.36	7.87	10.07	10.16	1.38	14.18	9.24	7.82	13.70	13.79	1.47
On. Potential	7.99	8.04	10.84	12.13	12.25	1.67	**11.42**	9.86	14.27	16.98	17.16	1.83
On. Tsallis	8.79	9.32	11.45	14.02	14.09	1.92	12.93	10.86	14.21	19.82	19.97	2.13

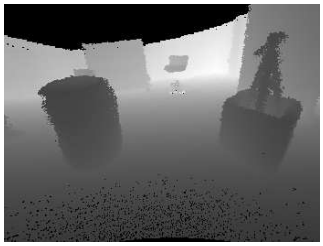
hood centred at the incoming event. The processing time per event T_e is also reported in micro-seconds for each measure. Overall, the exact measures are the most accurate, but more computationally demanding. The approximate versions are more computationally efficient without a significant loss in terms of accuracy. In online mode, the proposed measures are very computationally efficient (each event is processed in less than 3 micro-seconds), providing a valid trade-off between accuracy and computational runtime.

D.5 ADDITIONAL RESULTS ON 6-DOF ESTIMATION

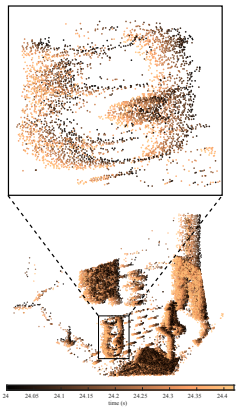
In Figure D.2, we qualitatively compare the estimated 6-DOF parameters and the resulting transformed 3D events according to the *Rényi* entropy-based measure and its approximate measure, on the `indoor_flying1` sequence (Zhu et al., 2018a). The resulting transformed 3D events are almost identical (see Figures D.2d and D.2e). This qualitatively corroborates that the proposed efficient approximate entropy-based measures provide a valid trade-off between accuracy and computational complexity.



(a) Greyscale image



(b) Depth image



(c) Original 3D events

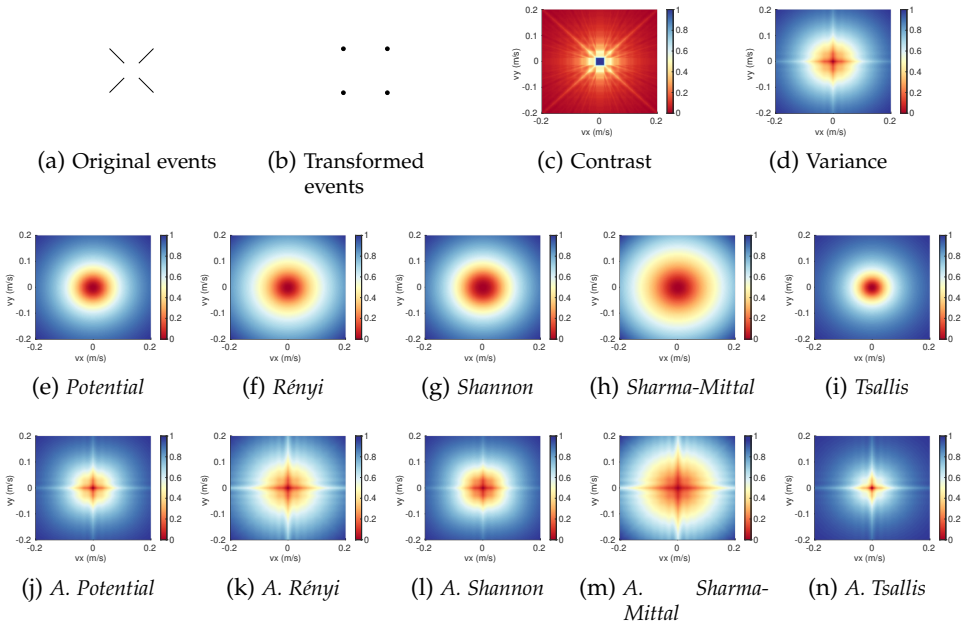


(d) 3D events transformed by *Rényi*



(e) 3D events transformed by *Rényi*

Figure D.2: 6-Degrees of Freedom (DOF) estimation in 3D space of the `indoor_flying1` sequence (Zhu et al., 2018a). (D.2a) Example greyscale image of the scene and (D.2b) corresponding depth. (D.2c) Original 3D events generated from the moving camera setup. (D.2d)-(D.2e) The events can be transformed directly in 3D to estimate the 6-DOF motion parameters.



(a) Original events (b) Transformed events (c) Contrast (d) Variance

(e) *Potential* (f) *Rényi* (g) *Shannon* (h) *Sharma-Mittal* (i) *Tsallis*

(j) *A. Potential* (k) *A. Rényi* (l) *A. Shannon* (m) *A. Sharma-Mittal* (n) *A. Tsallis*

Figure D.3: Projected normalised loss functions profiles at $\bar{v}_z = -1$ (m/s) and $\mathbf{n} = (0, 0, 1)^\top$ using the homography model on a synthetic data sequence. (D.3a) Original events projected onto the IWE, which simulate the corners of a square moving perpendicularly to the image plane with $v_z = -1$ (m/s). (D.3b) Transformed events projected onto the IWE, according to the optimal parameters $\bar{\mathbf{v}} = (0, 0, -1)^\top$ (m/s) and $\mathbf{n} = (0, 0, 1)^\top$. (D.3c) IWE contrast profile, which exhibits a clear peak at $\bar{\mathbf{v}} = (0, 0, -1)^\top$ (m/s); however, due to the image discretisation, the profile is not smooth and not suitable to be used in the optimisation framework. In contrast, the rest of the profiles shown are smooth and exhibit a clear minimum at the optimal motion model parameters. We can also observe that the profiles of the proposed approximate measures are not *as smooth as* the profiles of the respective exact measures. Interestingly, the (D.3d) variance profile (Gallego et al., 2018b) is similar to the profiles of the proposed approximate entropy-based measures, e. g. (D.3k) *A. Rényi*.

D.6 DISPERSION MINIMISATION OPTIMISATION PROFILES

Up to this point, all the proposed DMin measures profiles in function of the model parameters seem to suggest that there is a close similarity between the optimisation profiles of the exact measures and respective approximate measures. In Figure D.3, we show the projected normalised measures profiles for a synthetic data sequence, which simulates the corners of a square moving perpendicularly to the image plane. We can observe that, even though all profiles exhibit a clear minimum at the optimal model parameters, there is a clear distinction between the profiles of the exact measures and respective proposed approximations, i. e. the former exhibit a *smoother* landscape. Additionally, the variance loss function profile in Figure D.3d closely matches the proposed approximate entropy-based measures profiles, e. g. *A. Rényi* entropy-based measure profile shown in Figure D.3k.

D.7 TRUNCATED KERNEL SIZE

Table D.2: Performance influence of the truncated kernel size on the `boxes_rotation` and `poster_rotation` sequences (Mueggler et al., 2017), using the *Approx Potential* dispersion measure. The processing time per event T_e is also reported in micro-seconds. The best value per column is highlighted in bold.

	`boxes_rotation`							`poster_rotation`						
Size	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %	T_e	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %	T_e
3	7.91	7.07	7.78	9.94	10.04	1.37	**86.09**	13.94	8.91	**7.69**	13.51	13.62	1.45	**68.37**
5	7.80	6.85	**7.74**	**9.74**	**9.85**	**1.35**	168.11	13.66	8.85	7.80	13.54	13.67	1.46	170.59
7	7.80	**6.83**	7.79	9.76	9.88	1.35	278.67	**13.62**	8.85	7.74	13.28	13.41	1.43	283.01
9	**7.79**	**6.83**	7.79	9.78	9.89	1.35	420.84	13.64	**8.83**	7.77	13.32	13.45	1.44	433.31
11	7.81	6.84	7.78	9.75	9.87	1.35	603.46	**13.62**	8.84	7.75	**13.26**	**13.40**	**1.43**	617.73

The truncated kernel size affects how each event's influence is spread across the discretised space. For example, a larger size means that each event can influence a larger region in space. Table D.2 presents quantitative results regarding the performance of the proposed *Approx. Potential* measure, in function of the truncated kernel size. These results show that the truncated kernel size does not significantly influence the performance of the proposed optimisation framework using the proposed Approximate Dispersion Minimisation (ADMin). Thus, fixing it to 3 lowers the computational complexity of the overall framework, without performance degradation (the framework's

complexity using ADMin is $O(N_e \kappa^d)$, where κ is the truncated kernel size which may be fixed to 3).

D.8 REAL-TIME PERFORMANCE

Here, we show how the proposed DMin incremental mode can achieve real-time performance for general motion models estimation without a significant loss in terms of accuracy (please refer to Chapter 5 for the detailed framework's description).

D.8.1 *Implementation Description*

Similar to the description provided in Section 5.5.2 regarding practical considerations for the implementation of the incremental mode, we maintain a fixed number of the most recent events N_q, similar to a first-in-first-out queue policy. The components $\boldsymbol{\Psi}$ and $\boldsymbol{\psi}$ are incrementally computed by considering a 9×9 spatial neighbourhood centred at the incoming event. They are also decayed according to the factor $\exp\left(-(t_k - t_{k-1})n_{decay}\right)$, where t_k is the timestamp of the current event, t_{k-1} is the timestamp of the previous event and n_{decay} is incremented by one for each new event and is decayed according to the same factor.

The incremental mode has an almost constant processing time per event, which depends on the number of motion parameters. For example from Table 5.1, for the rotational motion estimation, it takes approximately $3\mu s$ to process one event.

To achieve real-time processing, we skip processing certain events. Conceptually, this can be thought of as *modifying the event's sequential index pattern* (e. g. to sequentially process one event at a time, a virtual *index variable* is incremented by one). Instead, we increment the index variable ξ by $\exp(a\Delta t)$, where $a = 5$ was set empirically, and Δt is the time difference between the current real time and the timestamp the current event being processed was generated. Thus, the event's index pattern adapts proportionally to the time delay.

This strategy (designated as S1) can still be improved since only modifying the index pattern as described previously may lead to an increase on the average time span between the first and last events on the queue, which may significantly decrease the overall performance. Figure D.4 illustrates this issue.

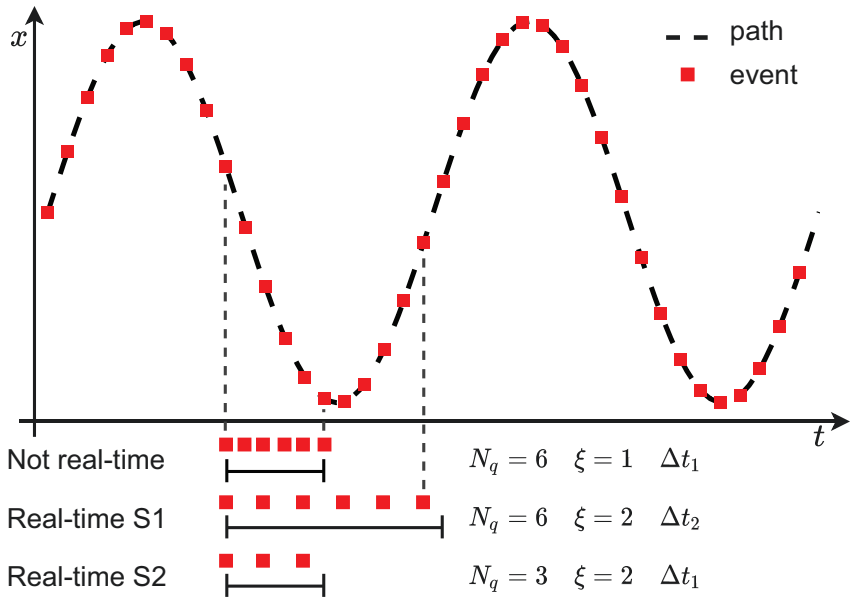


Figure D.4: Real-time implementation strategies. A particle moves up and down periodically according to the wave-like path plotted, generating the events illustrated as red squares. If no real-time strategy is employed, all events will be considered for the incremental optimisation procedure (i. e. the index variable $\xi = 1$), spanning some time Δt_1. According to strategy S1, events may be skipped, which in this example means that $\xi = 2$. However, since the number of events to be considered N_q is the same, the corresponding time span significantly increased (effectively to $\Delta t_2 > \Delta t_1$). This situation can be alleviated by employing strategy S2, which tries to conserve the time span as if no events were skipped, by adapting the number of events considered for the incremental optimisation procedure.

We address this problem by observing that the incremental DMin mode has a fixed throughput upper bound, defined as h_u. The throughput is given by

$$h = \frac{N_h}{\Delta t}, \tag{D.5}$$

where it reads that h is the number of events N_h processed by the procedure in some time interval Δt (note that by definition $h_u \geqslant h$). In the same time interval, consider also that there are N events being generated, and consequently the event rate r is given by

$$r = \frac{N}{\Delta t}. \tag{D.6}$$

We can also interpret r as the throughput of the incremental mode when considering all events (i. e. processing may not be in real-time), and thus $N = N_e$ (i. e. the number of events generated corresponds to the number of

events considered for the optimisation procedure). Since we want the time interval that is spanned by N_e events to be the same as the one spanned by N_h, we obtain

$$N_h = \frac{hN_e}{r} \leqslant h_u \frac{N_e}{r}. \tag{D.7}$$

By adapting the number of events considered during the optimisation procedure N_h according to Equation (D.7), we solve the problem identified for strategy S1. We designate this strategy by S2.

D.8.2 *Experimental Evaluation*

Table D.3: Quantitative comparison between real-time (RT1/2 for strategy S1/2) and not real-time (NRT) processing on the `boxes_rotation` and `poster_rotation` sequences (Mueggler et al., 2017).

	`boxes_rotation`						`poster_rotation`					
Approach	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %	e_{w_x}	e_{w_y}	e_{w_z}	σ_{e_w}	RMS	RMS %
NRT	7.99	8.04	10.84	12.13	12.25	1.67	11.42	9.86	14.27	16.98	17.16	1.83
RT1	19.34	21.23	30.44	43.78	43.81	5.97	16.46	18.95	40.86	44.35	44.67	4.77
RT2	10.09	10.17	16.77	17.47	17.64	2.41	11.53	11.24	19.23	21.56	21.62	2.31

Table D.3 reports the performance of the strategies employed for real-time event processing on two real sequences (Mueggler et al., 2017). Strategy S2 achieves significantly better quantitative performance than strategy S1. Additionally, the results are competitive compared to the NRT version, offering a valid trade-off between accuracy and real-time processing.

Figure D.5 depicts a possible use case setup for the incremental mode, whereby a DAVIS346B event camera is undergoing some motion and is connected to a standard laptop. The motion parameters estimates can then be plotted[3] or used for some other application downstream (e. g. Kinematic Structure (KS) estimation).

3 We use the PlotJuggler library - https://github.com/facontidavide/PlotJuggler

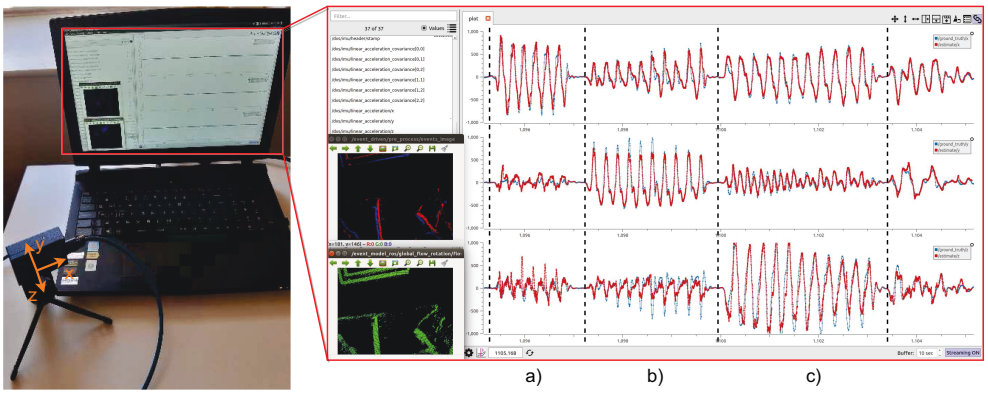
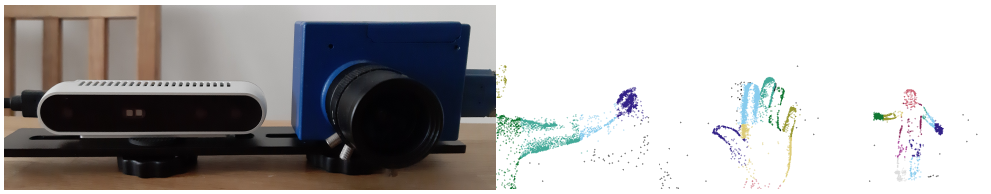


Figure D.5: Real-time DMin incremental mode. (Left) The setup consists of a DAVIS346B event camera that undergoes some motion and is connected to a standard consumer laptop. (Right) The motion estimates are plotted in red and the ground-truth angular velocities provided by the camera IMU are also plotted in blue. This plot exemplifies an event camera undergoing large rotational motions (up to ~ 1000 deg/s) around the (a) x-axis, (b) y-axis and (c) z-axis. Overall, the incremental motion estimation method follows the IMU measurements. The resultant global optical flow is also shown, as well as the corresponding generated events by accumulating them onto the image plane (bottom left corner).

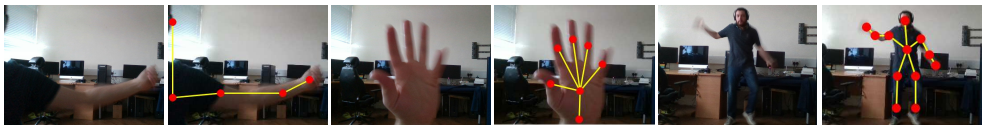
E

EVENT-BASED VISION INCREMENTAL KINEMATIC STRUCTURE ESTIMATION: EKS DATASET

To the best of our knowledge, we provide the first event-based vision dataset for Kinematic Structure (KS) estimation of arbitrary rigid objects: the Event-based KS (EKS) Dataset. Its main purpose is to facilitate the research and development of event-based motion segmentation and KS estimation methods.



(a) Camera setup (b) Ground-truth - each cluster is colour coded



(c) `arm_fast` sequence (d) `hand_fast` sequence (e) `human_fast` sequence

Figure E.1: EKS Dataset overview.

E.1 CAMERA SYSTEM

The dataset was recorded by a stationary system composed of a DAVIS346B event camera and a frame-based RGB-D Intel RealSense D415 camera, as depicted in Figure E.1a, and whose specifications are provided in Table A.1. The cameras intrinsic and extrinsic parameters were estimated by moving an AprilTag (Olson, 2011) in front of the system, using the Kalibr toolbox[1] (Furgale et al., 2012, 2013; Maye et al., 2013).

E.2 DATASET SEQUENCES

The dataset is composed of three challenging sequences that exhibit very rapid and diverse motions. The frame-based data exhibit massive blur effects

1 https://github.com/ethz-asl/kalibr

due to the fast motions presented, as depicted in Figures E.1c to E.1e, making frame-based estimation using commercially available cameras unfeasible. For each sequence, the number of motion clusters corresponds to the body parts of the respective articulated rigid object, ranging from five to twelve. Per-event labelling was manually done and is also provided, as depicted in Figure E.1b.

`arm_fast`: The simplest sequence since it only contains five distinct segments, there is no occlusion, and the generated events represent all body parts relatively well. Topologically, the kinematic structure is just composed of sequentially linked nodes (i. e. segments).

`hand_fast`: This sequence presents two additional challenges, namely: 1) brief occlusions (e. g. between the fingers) and 2) the palm and wrist are not well represented by the generated events (e. g. very few events are triggered inside of the palm, as seen in the middle plot in Figure E.1b). We consider each finger as one segment since no significant distinct motion is observed between the phalanges. Topologically, the kinematic structure resembles a tree whose root corresponds to the palm.

`human_fast`: The sequence containing more segments (twelve) and whose object's kinematic structure is the most complex. No significant occlusion occurs, and all segments are relatively well represented by the generated events. However, the resolution of each segment is reduced compared to the other sequences.

F

EVENT-BASED VISION INCREMENTAL KINEMATIC STRUCTURE ESTIMATION: ADDITIONAL RESULTS

F.1 PER-EVENT EVALUATION

Table F.1: Quantitative results on the 3DKS and EKS Datasets, in terms of average precision (P %), recall (R %) and f-measure (F %), comparing the estimated per-event labelling against the ground-truth. Higher values of P, R and F are better.

Dataset	Sequence	P	R	F
3DKS	donkey	97.6	99.7	98.6
	lamp	98.1	90.0	93.9
	pipe 1/2	94.4	95.4	94.9
	pipe 3/4	93.9	98.6	96.2
EKS	arm_fast	91.2	88.3	89.7
	hand_fast	92.8	79.0	85.4
	human_fast	87.0	69.9	77.5

By directly comparing the estimated per-event labelling against the ground-truth, we can also compute the precision, recall and f-measure metrics. However, this approach requires per-event ground-truth labelling, which may be expensive to obtain and thus may not be readily available (e. g. HKS Dataset). Table F.1 presents the results obtained by following this evaluation procedure for the sequences where per-event ground-truth labelling is available. Overall, the results obtained are in line with the ones provided in Tables 6.2 and 6.4, although slightly better.

F.2 CONSIDERING ONLY SHAPE INFORMATION

We also evaluate the proposed method by only considering the shape information during the optimisation (i. e. no motion information, according to Equations (6.8) to (6.12)). The results are provided in Table F.2. Overall, the method's performance is degraded since motion is a crucial source of information for the segmentation (please compare with Tables 6.2 and 6.4).

However, we note that the incremental approach as presented does not work if only motion is considered, during the optimisation (i. e. no shape

Table F.2: Quantitative results on the HKS, 3DKS and EKS Datasets, in terms of average precision (P %), recall (R %), f-measure (F %), normalised number of segments (N) and normalised computation time (T), considering shape information only. Higher values of P, R and F are better. Values of N closer to 1 are better, and values of T below 1 correspond to real-time processing.

Dataset	Sequence	P	R	F	N	T
HKS	Baxter	60.5	63.4	61.9	1.55	11.52
	dancing	77.7	79.9	78.8	1.00	2.56
	head	81.3	94.6	87.4	1.00	0.30
	iCub_body	80.9	75.8	78.3	0.86	0.90
	iCub_hand	64.7	64.1	64.4	1.00	0.72
	robot_arm	66.5	61.2	63.7	1.25	1.04
	yellow_crane	76.6	76.9	76.7	1.50	0.33
3DKS	donkey	89.3	87.2	88.2	1.00	6.99
	lamp	97.8	97.9	97.9	1.00	61.60
	pipe 1/2	93.1	75.4	83.3	2.00	5.13
	pipe 3/4	97.7	87.0	92.0	1.50	9.10
EKS	arm_fast	85.6	79.6	82.5	1.00	3.97
	hand_fast	72.5	72.2	72.4	1.14	16.78
	human_fast	81.1	57.4	67.2	0.83	2.50

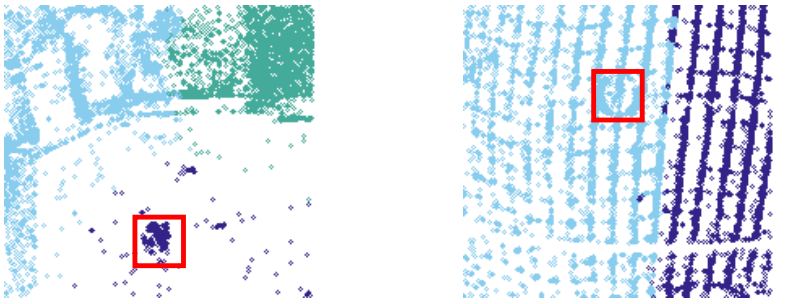
information). The reason for this comes from the fact that we include the responsibilities onto the dispersion score, given by Equation (6.6), we perform a hard assignment, and we do not consider any additional methodology for initialisation. Thus, the method struggles to find more clusters than one since the responsibility associated with the first cluster will be greater than the second cluster for most of the cases.

Naively, we could remove the responsibilities from the dispersion score and/or perform a soft assignment. These tentative fixes however create another problem since there is no guarantee of spatio-temporally coherent cluster assignments. This is the main reason we include shape information directly onto the optimisation, as proposed. We acknowledge that this approach is not desirable for general-purpose motion segmentation problems. However, it works well in the context of event-based incremental Kinematic Structure (KS) estimation since we also need to estimate and track the object's spatial topology.

F.3 QUALITATIVE RESULTS ON EED

We qualitatively tested the proposed incremental joint motion-shape segmentation approach on EED (Mitrokhin et al., 2018). Figure F.1 presents

the segmentation results for two distinct sequences, namely `fast_drone` and `what_is_background`.



(a) `fast_drone` (b) `what_is_background`

Figure F.1: Qualitative results on the EED. Each estimated cluster is colour coded and the object to be detected is highlighted with a red box.

The `fast_drone` sequence presents a drone moving (highlighted with a red box), while the event-based camera also moves. For most of the sequence, the drone is clearly visible since there is no significant occlusion, and the proposed approach can thus segment it successfully. When there is significant occlusion, as it occurs in the `what_is_background` sequence, the proposed approach fails to properly segment the object (i. e. a ball). Additionally, the proposed segmentation approach assumes that the underlying segments can be approximately described by a Gaussian-like shape and it fails when this condition is not met. For example, in the `what_is_background` sequence, even though the ball can be described by a Gaussian-like shape, the rest of the scene cannot.

BIBLIOGRAPHY

Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast High-Dimensional Filtering Using the Permutohedral Lattice. *Computer Graphics Forum*, 29(2):753–762, 2010. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2009.01645.x. (page 80).

Antonio Agudo and Francesc Moreno-Noguer. Combining Local-Physical and Global-Statistical Models for Sequential Deformable Shape from Motion. *International Journal of Computer Vision*, 122(2):371–387, 2017. ISSN 1573-1405. doi: 10.1007/s11263-016-0972-8. (page 56).

Antonio Agudo and Francesc Moreno-Noguer. A Scalable, Efficient, and Accurate Solution to Non-rigid Structure from Motion. *Computer Vision and Image Understanding*, 167:121–133, 2018. ISSN 1077-3142. doi: 10.1016/j.cviu.2018.01.002. (page 56).

Himanshu Akolkar, Sio Hoi Ieng, and Ryad Benosman. Real-time High Speed Motion Prediction Using Fast Aperture-Robust Event-Driven Visual Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3010468. (page 32).

Mohammed Almatrafi, Raymond Baldwin, Kiyoharu Aizawa, and Keigo Hirakawa. Distance Surface for Event-Based Optical Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(7):1547–1556, 2020. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.2986748. (pages 32, 84, 85, 94).

Patrick Bardow, Andrew J. Davison, and Stefan Leutenegger. Simultaneous Optical Flow and Intensity Estimation from an Event Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 884–892, 2016. doi: 10.1109/CVPR.2016.102. (page 32).

Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous Frameless Event-based Optical Flow. *Neural Networks*, 27:32–37, 2012. ISSN 0893-6080. doi: 10.1016/j.neunet.2011.11.001. (page 32).

Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-Based Visual Flow. *IEEE Transactions on Neural Networks*

and Learning Systems, 25(2):407–417, 2014. ISSN 2162-2388. doi: 10.1109/TNNLS.2013.2273537. (page 32).

Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme. Interactive Perception: Leveraging Action in Perception and Perception in Action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017. ISSN 1941-0468. doi: 10.1109/TRO.2017.2721939. (pages 19, 38).

T. E. Boult and L. Gottesfeld Brown. Factorization-based Segmentation of Motions. In *IEEE Workshop on Visual Motion*, 1991. doi: 10.1109/WVM.1991.212809. (page 30).

Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014. ISSN 1558-173X. doi: 10.1109/JSSC.2014.2342715. (pages 84, 123).

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, 2017. doi: 10.1109/CVPR.2017.143. (page 20).

Hyung Jin Chang and Yiannis Demiris. Unsupervised Learning of Complex Articulated Kinematic Structures Combining Motion and Skeleton Information. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3138–3146, 2015. doi: 10.1109/CVPR.2015.7298933. (page 36).

Hyung Jin Chang and Yiannis Demiris. Highly Articulated Kinematic Structure Estimation Combining Motion and Skeleton Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2165–2179, 2018. ISSN 1939-3539. doi: 10.1109/TPAMI.2017.2748579. (pages 20, 36, 40, 60, 104, 106, 109, 111, 117).

Hyung Jin Chang, Tobias Fischer, Maxime Petit, Martina Zambelli, and Yiannis Demiris. Learning Kinematic Structure Correspondences Using Multi-Order Similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2920–2934, 2018. ISSN 1939-3539. doi: 10.1109/TPAMI.2017.2777486. (pages 20, 36, 38).

D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. ISSN 1939-3539. doi: 10.1109/34.1000236. (page 103).

João Paulo Costeira and Takeo Kanade. A Multibody Factorization Method for Independently Moving Objects. *International Journal of Computer Vision*, 29(3):159–179, 1998. ISSN 1573-1405. doi: 10.1023/A:1008000628999. (page 30).

Ehsan Elhamifar and René Vidal. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2013.57. (pages 20, 30, 31, 40, 43, 44, 45, 46, 47, 48, 49, 51).

Ali Elqursh and Ahmed Elgammal. Online Motion Segmentation Using Dynamic Label Propagation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2008–2015, 2013. doi: 10.1109/ICCV.2013.251. (pages 20, 59).

Michael D. Escobar and Mike West. Bayesian Density Estimation and Inference Using Mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995. ISSN 0162-1459. doi: 10.1080/01621459.1995.10476550. (page 101).

João Fayad, Chris Russell, and Lourdes Agapito. Automated Articulated Structure and 3D Shape Recovery from Point Correspondences. In *IEEE International Conference on Computer Vision (ICCV)*, pages 431–438. IEEE, 2011. ISBN 978-1-4577-1102-2 978-1-4577-1101-5 978-1-4577-1100-8. doi: 10.1109/ICCV.2011.6126272. (page 36).

Paul Furgale, Timothy D. Barfoot, and Gabe Sibley. Continuous-time Batch Estimation Using Temporal Basis Functions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095, 2012. doi: 10.1109/ICRA.2012.6225005. (page 153).

Paul Furgale, Joern Rehder, and Roland Siegwart. Unified Temporal and Spatial Calibration for Multi-Sensor Systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280–1286, 2013. doi: 10.1109/IROS.2013.6696514. (page 153).

Guillermo Gallego and Davide Scaramuzza. Accurate Angular Velocity Estimation With an Event Camera. *IEEE Robotics and Automation Letters*, 2(2):632–639, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2016.2647639. (pages 33, 73, 75, 79, 80, 83).

Guillermo Gallego, Jon E.A. Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-Based, 6-DOF Camera Tracking

from Photometric Depth Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2402–2412, 2018a. ISSN 1939-3539. doi: 10.1109/TPAMI.2017.2769655. (page 33).

Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3867–3876, 2018b. doi: 10.1109/CVPR.2018.00407. (pages 21, 33, 73, 75, 76, 79, 80, 83, 85, 86, 87, 88, 89, 141, 143, 147).

Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus Is All You Need: Loss Functions for Event-Based Vision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12272–12281, 2019. doi: 10.1109/CVPR.2019.01256. (pages 33, 83).

Yixing Gao, Hyung Jin Chang, and Yiannis Demiris. Iterative Path Optimisation for Personalised Dressing Assistance using Vision and Force Information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4398–4403, 2016. doi: 10.1109/IROS.2016.7759647. (pages 20, 38).

Jiahui Huang, Sheng Yang, Zishuo Zhao, Yu-Kun Lai, and Shi-Min Hu. ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5875–5884, 2019. (page 118).

Xiaoxia Huang, Ian Walker, and Stan Birchfield. Occlusion-aware Multi-view Reconstruction of Articulated Objects for Manipulation. *Robotics and Autonomous Systems*, 62(4):497–505, 2014. ISSN 0921-8890. doi: 10.1016/j.robot.2013.12.006. (page 40).

Yan Huang, Jingliang Peng, C.-C. Jay Kuo, and M. Gopi. A Generic Scheme for Progressive Point Cloud Coding. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):440–453, 2008. ISSN 1941-0506. doi: 10.1109/TVCG.2007.70441. (page 61).

Bastien Jacquet, Roland Angst, and Marc Pollefeys. Articulated and Restricted Motion Subspaces and Their Signatures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1506–1513, 2013. (page 36).

Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. A Primal-Dual Framework for Real-Time Dense RGB-D Scene

Flow. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 98–104, 2015. doi: 10.1109/ICRA.2015.7138986. (pages 40, 43, 57).

Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. ScrewNet: Category-Independent Articulation Model Estimation From Depth Images Using Screw Theory. *arXiv:2008.10518*, 2021. URL https://arxiv.org/abs/2008.10518. (page 37).

Kevin M. Judd, Jonathan D. Gammell, and Paul Newman. Multimotion Visual Odometry (MVO): Simultaneous Estimation of Camera and Third-Party Motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956, 2018. doi: 10.1109/IROS.2018.8594213. (pages 20, 32, 40, 118).

Heechul Jung, Jeongwoo Ju, and Junmo Kim. Rigid Motion Segmentation using Randomized Voting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1210–1217, 2014. (pages 36, 40).

Dov Katz, Moslem Kazemi, J. Andrew Bagnell, and Anthony Stentz. Interactive Segmentation, Tracking, and Kinematic Modeling of Unknown 3D Articulated Objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5003–5010, 2013. doi: 10.1109/ICRA.2013.6631292. (page 37).

Daniel R. Kepple, Daewon Lee, Colin Prepsius, Volkan Isler, Il Memming Park, and Daniel D. Lee. Jointly Learning Visual Motion and Confidence from Local Patches in Event Cameras. In *Computer Vision – ECCV*, Lecture Notes in Computer Science, pages 500–516. Springer International Publishing, 2020. ISBN 978-3-030-58539-6. doi: 10.1007/978-3-030-58539-6_30. (page 34).

Margret Keuper, Bjoern Andres, and Thomas Brox. Motion Trajectory Segmentation via Minimum Cost Multicuts. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3271–3279, 2015. doi: 10.1109/ICCV.2015.374. (pages 30, 31).

Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In *Computer Vision – ECCV*, Lecture Notes in Computer Science, pages 349–364. Springer International Publishing, 2016a. ISBN 978-3-319-46466-4. doi: 10.1007/978-3-319-46466-4_21. (pages 33, 75).

Youngji Kim, Hwasup Lim, Sang Chul Ahn, and Ayoung Kim. Simultaneous Segmentation, Estimation and Analysis of Articulated Motion from Dense

Point Cloud Sequence. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1085–1092, 2016b. doi: 10.1109/IROS.2016.7759184. (page 37).

A. C. Koivunen and A. B. Kostinski. The Feasibility of Data Whitening to Improve Performance of Weather Radar. *Journal of Applied Meteorology and Climatology*, 38(6):741–749, 1999. ISSN 1520-0450, 0894-8763. doi: 10.1175/1520-0450(1999)038<0741:TFODWT>2.0.CO;2. (page 83).

Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: 10.1002/nav.3800020109. (page 50).

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning (ICML)*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001. ISBN 978-1-55860-778-1. (page 76).

Xavier Lagorce, Cédric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1710–1720, 2015. ISSN 2162-2388. doi: 10.1109/TNNLS.2014.2352401. (page 78).

Xiaolong Li, He Wang, Li Yi, Leonidas J. Guibas, A. Lynn Abbott, and Shuran Song. Category-Level Articulated Object Pose Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3706–3715, 2020. (pages 37, 118).

Yaron Lipman, Raif M. Rustamov, and Thomas A. Funkhouser. Biharmonic Distance. *ACM Transactions on Graphics*, 29(3):1–11, 2010. ISSN 0730-0301. doi: 10.1145/1805964.1805971. (pages 37, 41, 60, 97, 98, 105, 115).

Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust Recovery of Subspace Structures by Low-Rank Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.88. (pages 31, 51).

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. (page 39).

Xuequan Lu, Honghua Chen, Sai-Kit Yeung, Zhigang Deng, and Wenzhi Chen. Unsupervised Articulated Skeleton Extraction From Point Set Sequences Captured by a Single Depth Camera. *AAAI Conference on Artificial Intelligence*, 32(1), 2018. ISSN 2374-3468. (pages 20, 37).

Steven N. Maceachern and Peter Müller. Estimating Mixture of Dirichlet Process Models. *Journal of Computational and Graphical Statistics*, 7(2):223–238, 1998. ISSN 1061-8600. doi: 10.1080/10618600.1998.10474772. (page 101).

Roberto Martín Martín and Oliver Brock. Online Interactive Perception of Articulated Objects with Multi-level Recursive Estimation based on Task-specific Priors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2494–2501, 2014. doi: 10.1109/IROS.2014.6942902. (page 37).

Roberto Martín-Martín, Sebastian Höfer, and Oliver Brock. An Integrated Approach to Visual Perception of Articulated Objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5091–5097, 2016. doi: 10.1109/ICRA.2016.7487714. (page 37).

Jérôme Maye, Paul Furgale, and Roland Siegwart. Self-Supervised Calibration for Robotic Systems. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 473–480, 2013. doi: 10.1109/IVS.2013.6629513. (page 153).

Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-Based Moving Object Detection and Tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018. doi: 10.1109/IROS.2018.8593805. (pages 33, 34, 73, 75, 80, 156).

Anton Mitrokhin, Chengxi Ye, Cornelia Fermüller, Yiannis Aloimonos, and Tobi Delbruck. EV-IMO: Motion Segmentation Dataset and Learning Pipeline for Event Cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6105–6112, 2019. doi: 10.1109/IROS40897.2019.8968520. (page 34).

Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermüller, and Yiannis Aloimonos. Learning Visual Motion Segmentation Using Event Surfaces. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14402–14411, 2020. doi: 10.1109/CVPR42600.2020.01442. (page 34).

Clément Moulin-Frier, Tobias Fischer, Maxime Petit, Grégoire Pointeau, Jordi-Ysard Puigbo, Ugo Pattacini, Sock Ching Low, Daniel Camilleri, Phuong

Nguyen, Matej Hoffmann, Hyung Jin Chang, Martina Zambelli, Anne-Laure Mealier, Andreas Damianou, Giorgio Metta, Tony J. Prescott, Yiannis Demiris, Peter Ford Dominey, and Paul F. M. J. Verschure. DAC-h3: A Proactive Robot Cognitive Architecture to Acquire and Express Knowledge About the World and the Self. *IEEE Transactions on Cognitive and Developmental Systems*, 10(4):1005–1022, 2018. ISSN 2379-8939. doi: 10.1109/TCDS.2017.2754143. (pages 19, 38).

Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM. *The International Journal of Robotics Research*, 36(2):142–149, 2017. ISSN 0278-3649. doi: 10.1177/0278364917691115. (pages 84, 86, 87, 88, 89, 116, 143, 144, 145, 148, 151).

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. ISBN 978-0-262-30432-0. (pages 102, 127).

Sundaram Muthu, Ruwan Tennakoon, Tharindu Rathnayake, Reza Hoseinnezhad, David Suter, and Alireza Bab-Hadiashar. Motion Segmentation of RGB-D Sequences: Combining Semantic and Motion Information Using Statistical Inference. *IEEE Transactions on Image Processing*, 29:5557–5570, 2020. ISSN 1941-0042. doi: 10.1109/TIP.2020.2984893. (page 117).

Andrew Ng, Michael Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an Algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 14:849–856, 2002. (pages 30, 46, 59).

Urbano Miguel Nunes and Yiannis Demiris. 3D Motion Segmentation of Articulated Rigid Bodies Based on RGB-D Data. In *British Machine Vision Conference (BMVC)*. British Machine Vision Association (BMVA), 2018. (pages 23, 54).

Urbano Miguel Nunes and Yiannis Demiris. Online Unsupervised Learning of the 3D Kinematic Structure of Arbitrary Rigid Bodies. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3808–3816, 2019. doi: 10.1109/ICCV.2019.00391. (pages 23, 72).

Urbano Miguel Nunes and Yiannis Demiris. Entropy Minimisation Framework for Event-based Vision Model Estimation. In *Computer Vision – ECCV*, Lecture Notes in Computer Science, pages 161–176. Springer International Publishing, 2020. ISBN 978-3-030-58558-7. doi: 10.1007/978-3-030-58558-7_10. (pages 23, 96).

Urbano Miguel Nunes and Yiannis Demiris. Live Demonstration: Incremental Motion Estimation for Event-Based Cameras by Dispersion Minimisation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1322–1323, 2021a. (pages 24, 96).

Urbano Miguel Nunes and Yiannis Demiris. Robust Event-based Vision Model Estimation by Dispersion Minimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021b. ISSN 1939-3539. doi: 10.1109/TPAMI.2021.3130049. (pages 23, 96).

Peter Ochs and Thomas Brox. Object Segmentation in Video: A Hierarchical Variational Approach for Turning Point Trajectories into Dense Regions. In *International Conference on Computer Vision (ICCV)*, pages 1583–1590, 2011. doi: 10.1109/ICCV.2011.6126418. (page 31).

Peter Ochs and Thomas Brox. Higher Order Motion Models and Spectral Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 614–621, 2012. doi: 10.1109/CVPR.2012.6247728. (page 30).

Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of Moving Objects by Long Term Video Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, 2014. ISSN 1939-3539. doi: 10.1109/TPAMI.2013.242. (pages 31, 50, 64, 108).

Edwin Olson. AprilTag: A Robust and Flexible Visual Fiducial System. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407, 2011. doi: 10.1109/ICRA.2011.5979561. (page 153).

Chethan M. Parameshwara, Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. 0-MMS: Zero-Shot Multi-Motion Segmentation With A Monocular Event Camera. *arXiv:2006.06158*, 2020. URL http://arxiv.org/abs/2006.06158. (page 34).

Elzbieta Pekalska, Pavel Paclik, and Robert P. W. Duin. A Generalized Kernel Approach to Dissimilarity-based Classification. *The Journal of Machine Learning Research*, 2:175–211, 2002. ISSN 1532-4435. (pages 58, 62, 63).

Samunda Perera and Nick Barnes. Maximal Cliques Based Rigid Body Motion Segmentation with a RGB-D Camera. In *Computer Vision – ACCV*, Lecture Notes in Computer Science, pages 120–133. Springer, 2013. ISBN 978-3-642-37444-9. doi: 10.1007/978-3-642-37444-9_10. (pages 31, 37, 58, 64, 65, 70).

Maxime Petit and Yiannis Demiris. Hierarchical Action Learning by Instruction Through Interactive Grounding of Body Parts and Proto-actions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3375–3382, 2016. doi: 10.1109/ICRA.2016.7487513. (page 38).

Maxime Petit, Tobias Fischer, and Yiannis Demiris. Lifelong Augmentation of Multimodal Streaming Autobiographical Memories. *IEEE Transactions on Cognitive and Developmental Systems*, 8(3):201–213, 2016. ISSN 2379-8939. doi: 10.1109/TAMD.2015.2507439. (page 54).

Sudeep Pillai, Matthew R. Walter, and Seth Teller. Learning Articulated Motions From Visual Demonstration. In *Robotics: Science and Systems*, 2014. doi: 10.15607/RSS.2014.X.050. (page 37).

Ewa Piątkowska, Ahmed Nabil Belbachir, Stephan Schraml, and Margrit Gelautz. Spatiotemporal Multiple Persons Tracking Using Dynamic Vision Sensor. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 35–40, 2012. doi: 10.1109/CVPRW.2012.6238892. ISSN: 2160-7516. (page 34).

Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS : An Open-Source Robot Operating System. In *IEEE International Conference on Robotics and Automation Workshop on Open Source Software*, 2009. (page 123).

Shankar R. Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion Segmentation via Robust Subspace Separation in the Presence of Outlying, Incomplete, or Corrupted Trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. ISBN 1063-6919. doi: 10.1109/CVPR.2008.4587437. (page 30).

Henri Rebecq, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2016.2645143. (page 33).

Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. EMVS: Event-Based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time. *International Journal of Computer Vision*, 126(12):1394–1414, 2018a. ISSN 1573-1405. doi: 10.1007/s11263-017-1050-6. (page 33).

Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. ESIM: an Open Event Camera Simulator. In *Conference on Robot Learning*, pages 969–982. PMLR, 2018b. (pages 99, 106).

David A. Ross, Daniel Tarlow, and Richard S. Zemel. Learning Articulated Structure and Motion. *International Journal of Computer Vision*, 88(2):214–237, 2010. ISSN 1573-1405. doi: 10.1007/s11263-010-0325-y. (page 36).

Martin Rünz and Lourdes Agapito. Co-Fusion: Real-time segmentation, Tracking and Fusion of Multiple Objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4471–4478, 2017. doi: 10.1109/ICRA.2017.7989518. (page 31).

Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Continuous-Time Intensity Estimation Using Event Cameras. In *Computer Vision – ACCV*, Lecture Notes in Computer Science, pages 308–324. Springer International Publishing, 2018. ISBN 978-3-030-20873-8. doi: 10.1007/978-3-030-20873-8_20. (page 117).

Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Asynchronous Spatial Image Convolutions for Event Cameras. *IEEE Robotics and Automation Letters*, 4(2):816–822, 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2893427. (pages 80, 85).

Bhudev D. Sharma and Dharam P. Mittal. New Non-Additive Measures of Entropy for Discrete Probability Distributions. *Journal of Mathematical Science*, 10:28–40, 1975. (pages 77, 78).

Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. ISSN 1939-3539. doi: 10.1109/34.868688. (pages 30, 59).

Timo Stoffregen and Lindsay Kleeman. Simultaneous Optical Flow and Segmentation (SOFAS) using Dynamic Vision Sensor. In *Australasian Conference on Robotics and Automation (ACRA)*, pages 52–61. Australian Robotics and Automation Association (ARAA), 2017. (pages 33, 34).

Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. Event-Based Motion Segmentation by Motion Compensation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7243–7252, 2019. doi: 10.1109/ICCV.2019.00734. (pages 21, 34, 40).

Jurgen Sturm, Christian Plagemann, and Wolfram Burgard. Unsupervised Body Scheme Learning Through Self-perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3328–3333, 2008. doi: 10.1109/ROBOT.2008.4543718. (page 38).

Jurgen Sturm, Cyrill Stachniss, and Wolfram Burgard. A Probabilistic Framework for Learning Kinematic Models of Articulated Objects. *Journal of Artificial Intelligence Research*, 41:477–526, 2011. ISSN 1076-9757. doi: 10.1613/jair.3229. (pages 20, 36, 37).

Jürgen Sturm, Christian Plagemann, and Wolfram Burgard. Body Schema Learning for Robotic Manipulators from Visual Self-perception. *Journal of Physiology-Paris*, 103(3):220–231, 2009. ISSN 0928-4257. doi: 10.1016/j.jphysparis.2009.08.005. (page 38).

Jörg Stückler and Sven Behnke. Efficient Dense Rigid-Body Motion Segmentation and Estimation in RGB-D Video. *International Journal of Computer Vision*, 113(3):233–245, 2015. ISSN 1573-1405. doi: 10.1007/s11263-014-0796-3. (page 39).

Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. Mean Curvature Skeletons. *Computer Graphics Forum*, 31(5):1735–1744, 2012. ISSN 1467-8659. doi: 10.1111/j.1467-8659.2012.03178.x. (page 37).

Carlo Tomasi and Takeo Kanade. Shape and Motion From Image Streams Under Orthography: A Factorization Method. *International Journal of Computer Vision*, 9(2):137–154, 1992. ISSN 1573-1405. doi: 10.1007/BF00129684. (page 30).

P. Tresadern and I. Reid. Articulated Structure from Motion by Factorization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1110–1115, 2005. doi: 10.1109/CVPR.2005.75. (page 36).

Dimitrios Tzionas and Juergen Gall. Reconstructing Articulated Rigged Models from RGB-D Videos. In *Computer Vision – ECCV Workshops*, Lecture Notes in Computer Science, pages 620–633. Springer International Publishing, 2016. ISBN 978-3-319-49409-8. doi: 10.1007/978-3-319-49409-8_53. (pages 37, 49, 64, 106).

Markus Unger, Manuel Werlberger, Thomas Pock, and Horst Bischof. Joint Motion Estimation and Segmentation of Complex Scenes with Label Costs

and Occlusion Modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1878–1885, 2012. doi: 10.1109/CVPR.2012.6247887. (page 40).

R. Vidal, Yi Ma, and S. Sastry. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27 (12):1945–1959, 2005. ISSN 1939-3539. doi: 10.1109/TPAMI.2005.244. (page 31).

Ulrike von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395–416, 2007. ISSN 1573-1375. doi: 10.1007/s11222-007-9033-z. (page 30).

Ziwei Wang, Yonhon Ng, Cedric Scheerlinck, and Robert Mahony. An Asynchronous Kalman Filter for Hybrid Event Cameras. *arXiv:2012.05590*, 2021. URL http://arxiv.org/abs/2012.05590. (page 117).

Yan Wu and Yiannis Demiris. Hierarchical Learning Approach for One-shot Action Imitation in Humanoid Robots. In *International Conference on Control Automation Robotics Vision*, pages 453–458, 2010a. doi: 10.1109/ICARCV.2010.5707349. (page 38).

Yan Wu and Yiannis Demiris. Towards One Shot Learning by Imitation for Humanoid Robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2889–2894, 2010b. doi: 10.1109/ROBOT.2010.5509429. (page 38).

Yan Wu, Yanyu Su, and Yiannis Demiris. A Morphable Template Framework for Robot Learning by Demonstration: Integrating One-shot and Incremental Learning Approaches. *Robotics and Autonomous Systems*, 62(10): 1517–1530, 2014. ISSN 0921-8890. doi: 10.1016/j.robot.2014.05.010. (page 38).

J. Yan and M. Pollefeys. A Factorization-based Approach to Articulated Motion Recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 815–821, 2005. doi: 10.1109/CVPR.2005.27. (page 36).

Jingyu Yan and Marc Pollefeys. A Factorization-Based Approach for Articulated Nonrigid Shape, Motion and Kinematic Chain Recovery From Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):865–877, 2008. ISSN 1939-3539. doi: 10.1109/TPAMI.2007.70739. (pages 20, 36, 44, 60, 104).

Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A. Yorke, and Yiannis Aloimonos. Unsupervised Learning of Dense Optical Flow, Depth and Egomotion from Sparse Event Data. *arXiv:1809.08625*, 2018. (pages 33, 91, 92).

Lihi Zelnik-manor and Pietro Perona. Self-Tuning Spectral Clustering. *Advances in Neural Information Processing Systems (NIPS)*, pages 1601–1608, 2004. (pages 31, 43, 44, 47, 59).

Vicky Zeng, Timothy E. Lee, Jacky Liang, and Oliver Kroemer. Visual Identification of Articulated Object Parts. *arXiv:2012.00284*, 2021. URL http://arxiv.org/abs/2012.00284. (page 37).

Fan Zhang, Antoine Cully, and Yiannis Demiris. Personalized Robot-Assisted Dressing Using User Modeling in Latent Spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3603–3610, 2017. doi: 10.1109/IROS.2017.8206206. (page 38).

Fan Zhang, Antoine Cully, and Yiannis Demiris. Probabilistic Real-Time User Posture Tracking for Personalized Robot-Assisted Dressing. *IEEE Transactions on Robotics*, 35(4):873–888, 2019. ISSN 1941-0468. doi: 10.1109/TRO.2019.2904461. (pages 20, 38).

Quanshi Zhang, Xuan Song, Xiaowei Shao, Ryosuke Shibasaki, and Huijing Zhao. Unsupervised Skeleton Extraction and Motion Capture from 3D Deformable Matching. *Neurocomputing*, 100:170–182, 2013. ISSN 0925-2312. doi: 10.1016/j.neucom.2011.11.032. (pages 20, 37).

Yin Zhang and Zhi-Hua Zhou. Non-Metric Label Propagation. In *International Joint Conference on Artificial Intelligence*, pages 1357–1362, 2009. (page 58).

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with Local and Global Consistency. In *Advances in Neural Information Processing Systems (NIPS)*, pages 321–328, 2004. (pages 30, 57, 59).

Yi Zhou, Guillermo Gallego, Xiuyuan Lu, Siqi Liu, and Shaojie Shen. Event-based Motion Segmentation with Spatio-Temporal Graph Cuts. *arXiv:2012.08730*, 2021. URL http://arxiv.org/abs/2012.08730. (page 34).

Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based Feature Tracking with Probabilistic Data Association. In *IEEE International*

Conference on Robotics and Automation (ICRA), pages 4465–4470, 2017. doi: 10.1109/ICRA.2017.7989517. (pages 33, 133).

Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018a. ISSN 2377-3766. doi: 10.1109/LRA.2018.2800793. (pages 84, 90, 91, 92, 93, 145, 146).

Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. In *Proceedings of Robotics: Science and Systems*, 2018b. doi: 10.15607/RSS.2018.XIV.062. (pages 33, 94).

Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 989–997, 2019. doi: 10.1109/CVPR.2019.00108. (pages 21, 33, 76, 91, 92).

Vasileios Zografos, Reiner Lenz, Erik Ringaby, Michael Felsberg, and Klas Nordberg. Fast Segmentation of Sparse 3D Point Trajectories Using Group Theoretical Invariants. In *Computer Vision - ACCV*, Lecture Notes in Computer Science, pages 675–691. Springer International Publishing, 2015. ISBN 978-3-319-16817-3. doi: 10.1007/978-3-319-16817-3_44. (page 32).