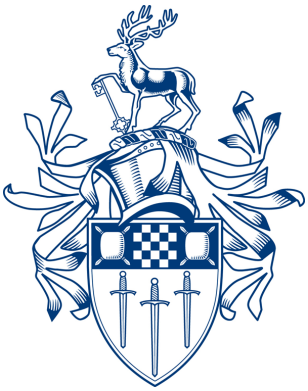
2D and 3D Tracking and Modelling

Karel Lebeda

Submitted for the Degree of
Doctor of Philosophy
from the University of Surrey



Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, GU2 7XH, United Kingdom

July 2016

Abstract

Visual tracking of unknown objects in unconstrained video-sequences is extremely challenging due to a number of unsolved issues. This thesis explores several of these and examines possible approaches to tackle them.

The unconstrained nature of real-world input sequences creates huge variation in the appearance of the target object due to changes in pose and lighting. Additionally, the object can be occluded by either parts of itself, other elements of the scene, or the frame boundaries. Observations may also be corrupted due to low resolution, motion blur, large frame-to-frame displacement, or incorrect exposure or focus of the camera. Finally, some objects are inherently difficult to track due to their (low) texture, specular/transparent nature, non-rigid deformations, *etc.*

Conventional trackers depend heavily on the *texture* of the target. This causes issues with transparent or untextured objects. Edge points can be used in cases where standard feature points are scarce; these however suffer from the *aperture problem*. To address this, the first contribution of this thesis explores the idea of *virtual corners*, using pairs of non-adjacent *line correspondences*, tangent to *edges* in the image. Furthermore, the chapter investigates the possibility of *long-term tracking*, introducing a re-detection scheme to handle occlusions while limiting drift of the object model. The outcome of this research is an edge-based tracker, able to track in scenarios including untextured objects, full occlusions and significant length. The tracker, besides reporting excellent results in standard benchmarks, is demonstrated to successfully track the *longest sequence* published to date.

Some of the issues in visual tracking are caused by suboptimal utilisation of the image information. The object of interest can easily occupy as few as ten or even one percent of the video frame area. This causes difficulties in challenging scenarios such as sudden *camera shakes* or *full occlusions*. To improve tracking in such cases, the next major contribution of this thesis explores relationships within the context of visual tracking, with a focus on *causality*. These include causal links between the tracked object and other elements of the scene such as the camera motion or other objects. Properties of such relationships are identified in a framework based on information theory. The resulting technique can be employed as a causality-based *motion model* to improve the results of virtually any tracker.

Significant effort has previously been devoted to rapid learning of object properties on the fly. However, state-of-the-art approaches still often fail in cases such as rapid *out-of-plane rotations*, when the appearance changes suddenly. One of the major contributions of this thesis is a radical rethinking of the traditional wisdom of modelling 3D motion as appearance change. Instead, 3D motion is modelled as 3D motion. This intuitive but previously unexplored approach provides new possibilities in visual tracking research.

Firstly, *3D tracking* is more general, as large out-of-plane motion is often fatal for 2D trackers, but helps 3D trackers to build better models. Secondly, the tracker's internal model of the object can be used in many different applications and it could even become the main motivation, with tracking supporting reconstruction rather than vice versa. This effectively bridges the gap between *visual tracking* and *Structure from Motion*. The proposed method is capable of successfully tracking sequences with extreme out-of-plane rotation, which poses a considerable challenge to 2D trackers. This is done by creating realistic 3D models of the targets, which then aid in tracking.

In the majority of the thesis, the assumption is made that the target's 3D shape is rigid. This is, however, a relatively strong limitation. In the final chapter, tracking and *dense* modelling of *non-rigid targets* is explored, demonstrating results in even more generic (and therefore challenging) scenarios. This final advancement truly generalises the tracking problem with support for long-term tracking of low texture and non-rigid objects in sequences with camera shake, shot cuts and significant rotation.

Taken together, these contributions address some of the major sources of failure in visual tracking. The presented research advances the field of visual tracking, facilitating tracking in scenarios which were previously infeasible. Excellent results are demonstrated in these challenging scenarios. Finally, this thesis demonstrates that 3D reconstruction and visual tracking can be used together to tackle difficult tasks.

Keywords: computer vision, visual tracking, 3D tracking, aperture problem, bundle adjustment, causality, edge, Gaussian process regression, line correspondence, long-term tracking, low texture, motion model, online modelling, out-of-plane rotation, structure from motion, transfer entropy, virtual corner.

Email: K.Lebeda@Surrey.ac.uk (university), Karel@Lebeda.sk (personal)

WWW: http://CVSSP.org/Personal/KarelLebeda/

Supersivors: Prof. Richard Bowden and Dr. Simon Hadfield

Acknowledgements

First of all, I would like to thank my supervisors, Richard Bowden and Simon Hadfield, for their supervision, leadership, inspiration and invaluable advice. To pick a particular example from the numerous things I owe to them, Rich has taught me to ask the question "If human brain can do this, why cannot we?" and to wonder how. Simon has greatly inspired me by his constant drive for self-improvement and often gave me hope when things seemed dark.

My second thankful thought goes to the members of the CVSSP, who accepted me, and who have been like a family to me over the past years. My thanks, on both the professional and personal level, belong especially to my current and past office colleagues. In particular I would like to thank to Fatemeh Tahavori, and also to Philip Krejov whose mastery in 3D modelling provided me with plenty of synthetic data.

I am further very thankful to all my friends, notably the dearest of them, Sabrina Tardio, who has been through all the joys and perils of a PhD "life" with me. I cannot forget to thank my family, for their moral support and for never having the slightest doubt in me, and to my girlfriend Rebecca Townsend, especially for coping with my mood swings during writing this thesis.

I would like to appreciate the work of my examiners, Adrian Hilton and Aleš Leonardis, whose comments helped improving this thesis significantly.

Finally, I would like to thank my funders. Most importantly it was the University of Surrey with my University Research Scholarship, giving me the freedom of independent research, and the EPSRC, supporting my research under the grant "Learning to Recognise Dynamic Visual Content from Broadcast Footage" (EP/I011811/1). My travel to the CVPR2015 and ICCV2015 conferences was covered by a scholarship awarded from the Rabin Ezra Fund. My travel to the ACCV2014 conference was supported by the BMVA Student Travel Bursary and my travel to the ICCV2013 conference by the PAMI-TC Student Travel Grant.

Declaration

Some elements of this work have appeared or will appear in the following publications. The symbol * marks publications where I was the primary author and are the key publications supporting this thesis. Contributions of this thesis and affiliation of particular publications to chapters are specified in Section 1.2.

- [129]* K. Lebeda, J. Matas, and R. Bowden. Tracking the Untrackable: How to Track When Your Object Is Featureless. *In Proceedings of the ACCV workshop on Detection and Tracking in Challenging Environments*, 2012.
- [127]* K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Long-Term Tracking Through Failure Cases. *In Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, 2013.
- [115] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Čehovin, G. Nebehay, G. Fernández, T. Vojíř, *et al*. The Visual Object Tracking VOT2013 challenge results. *In Proceedings of the ICCV workshop on Visual Object Tracking challenge*, 2013.
- [69] S. Hadfield, K. Lebeda and R. Bowden. Natural action recognition using invariant 3D motion encoding. *In Proceedings of the European Conference on Computer Vision*, 2014.
- [116] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, G. Fernández, *et al*. The Visual Object Tracking VOT2014 challenge results. *In Proceedings of the ECCV workshop on Visual Object Tracking challenge*, 2014.
- [121]* K. Lebeda, S. Hadfield, and R. Bowden. 2D or not 2D: Bridging the Gap Between Tracking and Structure from Motion. *In Proceedings of the Asian Conference on Computer Vision*, 2014.
- [123]* K. Lebeda, S. Hadfield, and R. Bowden. Exploring Causal Relationships in Visual Object Tracking. *In Proceedings of the International Conference on Computer Vision*, 2015.

- [122]* K. Lebeda, S. Hadfield, and R. Bowden. Dense Rigid Reconstruction From Unstructured Discontinuous Video. *In Proceedings of the ICCV workshop on 3D Representation and Recognition*, 2015.
- [114] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojíř, G. Häger, G. Nebehay, R. Pflugfelder, *et al*. The Visual Object Tracking VOT2015 challenge results. *In Proceedings of the ICCV workshop on Visual Object Tracking challenge*, 2015.
- [128]* K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Texture-Independent Long-Term Tracking Using Virtual Corners. *In IEEE Transactions on Image Processing*, 2016.
- [70] S. Hadfield, K. Lebeda and R. Bowden. Hollywood 3D: What are the best 3D features for Action Recognition? *In International Journal of Computer Vision*, 2016.
- [72] S. Hadfield, K. Lebeda and R. Bowden. Stereo reconstruction using top-down cues from urban environment. *In Computer Vision and Image Understanding*, 2016.
- [124]* K. Lebeda, S. Hadfield, and R. Bowden. Causal Relationships in Visual Tracking. *Under review for IEEE Transactions on Pattern Analysis and Machine Intelligence.*
- [126]* K. Lebeda, S. Hadfield, and R. Bowden. TMAGIC: A Model-free 3D Tracker. *Under review for IEEE Transactions on Image Processing.*
- [125]* K. Lebeda, S. Hadfield, and R. Bowden. Direct-from-Video: Unsupervised NRS*f*M. *Under review for the European Conference on Computer Vision*, 2016.
- [71] S. Hadfield, K. Lebeda and R. Bowden. PnP-HARD: PnP optimization using a hybrid approximate representation. *Under review for the European Conference on Computer Vision*, 2016.

Used Acronyms

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
300VW	300 Videos in the Wild
ACCV	Asian Conference on Computer Vision
AR	Augmented Reality
ASLA	Adaptive Structural Local Sparse Appearance [98]
AUC	Area Under Curve
BA	Bundle Adjustment
BALM	Bilinear modeling via Augmented Lagrange Multipliers
BC	Background Clutter
BMC	Background Motion Compensation
BMVA	British Machine Vision Association
BRIEF	Binary Robust Independent Elementary Features
CNN	Convolutional Neural Network
CONDENSATION	Conditional Density Propagation [95]
CVPR	(IEEE Conference on) Computer Vision and Pattern Recognition
DEF	Deformation
DGT	Dynamic Graph based Tracker [22]
DSST	Discriminative Scale Space Tracker [34]

DTAM	Dense Tracking And Mapping
EDFT	Enhanced Distribution Fields for Tracking [49]
EKF	Extended Kalman Filter [45, 59]
EM	Expectation Maximisation
EPSRC	Engineering and Physical Sciences Research Council
FAST	Features from Accelerated Segment Test
FLOTrack	Feature-Less Object Tracker [129]
FM	Fast Motion
FoF	Flocks of Features [89, 113]
FoT	Flock of Trackers [137, 199]
FPS	Frames Per Second
GP	Gaussian Process
GPR	Gaussian Process Regression
GPS	Global Positioning System
GT	Ground Truth
HOG	Histogram of Oriented Gradient
ICCV	International Conference on Computer Vision
IMU	Inertial Measurement Unit
IPR	In-Plane Rotation
IR	Infra-Red
IV	Illumination Variation
IVT	Incremental Visual Tracking
KCF	Kernelized Correlation Filter [83]
KDE	Kernel Density Estimation
KF	Kalman Filter [45, 101, 107]
KL	Kullback-Leibler (divergence)
KLT	Kanade-Lucas-Tomasi tracker [187]

LGT	Local-Global Tracker [24, 25]
LGT++	Enhanced Local-Global Tracker [209]
LIIP	Local Isometric and Infinitesimally Planar NRS*f*M
LK	Lucas-Kanade tracker [134]
LO-RANSAC	Locally Optimised RANSAC [30, 130]
LR	Low Resolution
LSD	Line Segment Detector
LT-FLOTrack	Long-Term Feature-Less Object Tracker (also LT-FLO) [127, 128]
LT-FLO	Long-Term Feature-Less Object Tracker [127, 128]
MB	Motion Blur
MDNet	Multi-Domain Convolutional Neural Networks
MILK	Mutual Information for Lucas-Kanade Tracking
NRSfM	Non-Rigid Structure from Motion
OCC	Occlusion
OPE	One-Pass Evaluation
OPR	Out-of-Plane Rotation
ORB	Oriented FAST and Rotated BRIEF
OV	Out-of-View
P3L	Perspective-3-Lines
P3P	Perspective-3-Points
PAMI	(IEEE Trans. on) Pattern Analysis and Machine Intelligence
PnL	Perspective-n-Lines
PnP	Perspective-n-Points
PLT	Pixel-based LUT Tracker [82]
ProFORMA	Probabilistic Feature-based On-line Rapid Model Acquisition
PTAM	Parallel Tracking And Mapping
RANSAC	Random Sample Consensus [53]

RBF	Radial Basis Function
RGB	Red-Green-Blue
RGBD	Red-Green-Blue-Depth
SAMF	Scale Adaptive and Feature Integration (sic) [131]
SCM	Sparsity-based Collaborative Model [216]
SIFT	Scale-Invariant Feature Transform [133]
SfM	Structure from Motion
SLAM	Simultaneous Localisation And Mapping
SRDCF	Spatially Regularised Discrete Correlation Filters [35]
SSD	Sum of Squared Distances
Struck	Structured Output Tracking with Kernels [75, 76]
SV	Scale Variation
TE	Transfer Entropy
TLD	Tracking-Learning-Detection [103, 104, 106]
TMAGIC	Tracking, Modelling And Gaussian-process Inference Combined [121]
UAV	Unmanned Aerial Vehicle
VOT	Visual Object Tracking challenge [115, 116]
VTB	Visual Tracker Benchmark [206, 207]

Used Symbols

$\mathcal{A}^t$	non-rigid shape mixing coefficient vectors up to the t-th frame
C^t	pose of the camera in the t-th frame
$\mathcal{C}^t$	poses of the camera in frames up to t
$\mathbf{C}^t$	position of the camera centre in the t-th frame
$\mathbf{c}^t$	vector of general camera parameters in the t-th frame
d_{G}	geometric distance of a pair of lines
d_{P}	position component of geometric distance
d_{A}	angular component of geometric distance
$\mathbf{d}_i^t$	i-th 2D dense feature in the t-th frame
$\mathcal{D}^t$	all 2D dense features in the t-th frame
$\mathbf{D}_i$	i-th 3D dense feature
$\dot{\mathcal{D}}$	all 3D dense features
E	expectation of a random variable
$\mathtt{E}_n$	$n \times n$ identity matrix
$e_{\mathbf{x}_i^t}$	i-th edge point fit (transformed to the next frame)
E^t	image evidence (average fit of edge points)
f	camera focal length in world units (*e.g.* millimetres)
$\mathbf{f}_i^t$	i-th 2D feature in the t-th frame
$\mathcal{F}^t$	all 2D features in the t-th frame
$\mathbf{F}_i$	i-th 3D feature

$\dot{\mathcal{F}}$	all 3D features
H	entropy ($H_{X\to Y}$ stands for transfer entropy from X to Y)
$\mathtt{I}^t$	t-th frame of a video-sequence
$\mathcal{I}^t$	inlier features in the t-th frame (a subset of $\mathcal{X}'^t$)
$\mathtt{K}^t$	camera calibration matrix (intrinsic parameters) in the t-th frame
$\mathbf{l}_i^t$	i-th 2D line feature in the t-th frame
$\mathcal{L}^t$	all 2D line features in the t-th frame
$\mathbf{L}_i$	i-th 3D line feature
$\dot{\mathcal{L}}$	all 3D line features
M	object model
$\mathbf{M}_i$	i-th point (3D) sampled on the model
$\dot{\mathcal{M}}$	all points (3D) sampled on the model
$\dot{\mathcal{N}}$	nearest neighbours in a (3D) feature cloud
$\mathbf{p}^t$	pose of the tracked object in the t-th frame
$\mathbf{p}'^t$	pose of a supporter object in the t-th frame
$\mathbf{p}_s^t$	pose of the tracked object in the t-th frame, in the scene coordinates
$\mathcal{P}$	poses of the tracked object in whole video-sequence (*i.e.* $\mathcal{P}^T$)
$\mathcal{P}^t$	poses of the tracked object in frames up to t
$\mathtt{P}^t$	camera projection matrix in the t-th frame
$\mathtt{Q}^t$	edge quality field in (after) the t-th frame
Q	edge quality score (computed from $\mathtt{Q}$)
$\mathbf{Q}$	Gaussian Process (GP) model query
$\bar{\mathbf{Q}}$	GP model query direction (a unit-length vector)
$r_{\bar{\mathbf{Y}}}$	(local) radius of the GP model in the direction $\bar{\mathbf{Y}}$
R	relative improvement in transfer entropy
$\mathtt{R}^t$	camera rotation matrix in the t-th frame
S	frame-to-frame transformation (final), particular case of τ

S', S''	frame-to-frame transformations (preliminary)
$\mathbf{s}_i^t$	i-th 2D supervision feature in the t-th frame
$\mathcal{S}^t$	all 2D supervision features in the t-th frame
$\mathbf{S}_i$	i-th 3D supervision feature
$\dot{\mathcal{S}}$	all 3D supervision features
t	frame index
T	total number of frames in a video-sequence
$\mathtt{T}$	object model texture
v	visibility indicating function
$\mathcal{V}$	set of cameras by which a point is visible
$\mathbf{V}$	vertex of the polygonal model
w_i^t	weight of the i-th feature in the t-th frame
$\mathbf{x}_i^t$	i-th 2D point feature in the t-th frame
$\mathcal{X}^t$	all 2D point features in the t-th frame
$\mathbf{X}_i$	i-th 3D point feature
$\dot{\mathcal{X}}$	all 3D point features
$\mathbf{Y}_i$	i-th centred training point of the GP model
$\bar{\mathbf{Y}}_i$	i-th training direction (normalised point) of the GP model
$\dot{\mathcal{Y}}$	all centred training points of the GP model
$\dot{\bar{\mathcal{Y}}}$	all training directions (normalised points) of the GP model
$\mathbf{Y}_\circ$	centre of the GP model
$\mathbf{Z}$	ray from the camera centre $\mathbf{C}$
$\alpha_{\textcircled{j}}^t$	j-th non-rigid shape mixing coefficient in the t-th frame
$\boldsymbol{\alpha}^t$	non-rigid shape mixing coefficient vector in the t-th frame
Γ	tracking hypothesis score

θ	threshold (multiple different used)
κ	kernel (multiple different used)
Λ	regularisation term (multiple different used)
$\boldsymbol{\mu}_{\Psi}^{t}$	mean of Ψ in (after) the t-th frame
$\boldsymbol{\mu}_{\mathbf{l}}$, $\boldsymbol{\nu}_{\mathbf{l}}$	end-points of line segment $\mathbf{l}$
Π^{t}	model to image projection function in the t-th frame
ρ	robust cost function (multiple different used)
$\sigma_{\bar{\mathbf{Y}}}$	(local) radius confidence of the GP model in the direction $\bar{\mathbf{Y}}$
Σ_{Ψ}^{t}	covariance of Ψ in (after) the t-th frame
τ	frame-to-frame tracking function
$\boldsymbol{\varphi}^{t}$	inner state of a tracker after the t-th frame
Φ	bank of stored tracker states
ϕ	learned predictive function
ϕ_a	window-based non-causal (autoregressive) predictive function
ϕ_s	time-based (sequential) non-causal predictive function
ϕ_t	time-based causal predictive function
ϕ_w	window-based causal predictive function
χ	surface indicator function
Ψ	probability distribution of $\mathbf{p}$
$\mathbf{1}_n$	vector of n ones

Contents

Chapter 1

Introduction

1.1 Motivation

Visual object tracking is an area of computer vision which has been studied extensively in the last few decades. The task of a *tracker* is, given a video-sequence and a user-chosen target object marked in the first frame, to output the *pose* of this object in all subsequent frames of the sequence. The nature of the pose is application-specific and can be as simple as an x-y location in the image, or as complex as a full pixel-wise segmentation (see Figure 1.1 for several examples).

Visual tracking has numerous applications such as video augmentation or annotation propagation (see Figure 1.2 for an example of a *shoppable video* where user-

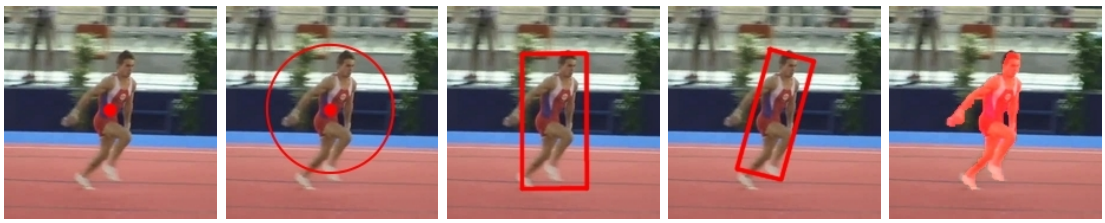


Figure 1.1: Different possible object *poses*, shown on the sequence Gymnastics. From left to right: centre location, centre and size, axis-aligned bounding box, general rectangular bounding box, pixel-level segmentation.

Figure 1.2: Example of a tracker application: annotation propagation in a shoppable video.

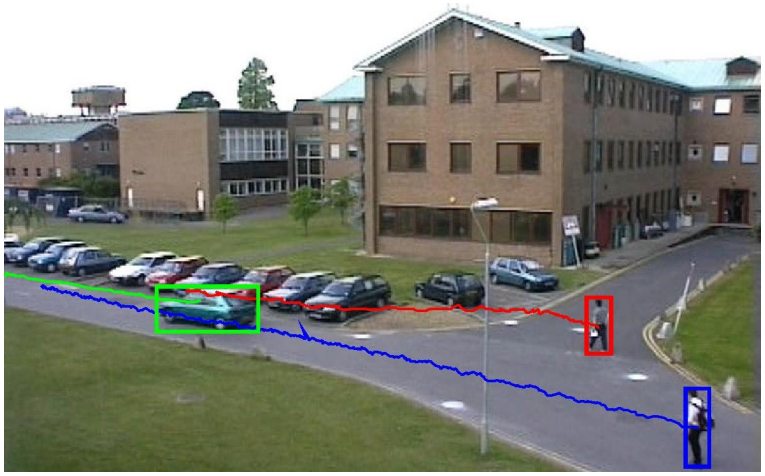


Figure 1.3: Example of a tracker application: tracked trajectories used for dynamic scene analysis.

annotated hyperlinks are tracked throughout the sequence and allow customers to buy the marked accessories). However, tracking is primarily used as a building block or a preprocessing step by techniques in other application areas – tracking is an *enabling technology*. As such, its major and most obvious output, the object *trajectory*, *i.e.* the location (or more generally pose) as a function of time, may be used in applications such as behaviour analysis, area monitoring, action recognition or human-computer interaction (see Figure 1.3 for an example). Secondly, successful tracking provides *object stabilisation*, *i.e.* object-centric spatial alignment of the sequence. This allows

Figure 1.4: Example of a tracker application: object stabilisation used for super-resolution. Top: selected frames of a video-sequence; bottom: detail of an input frame and of the output image.

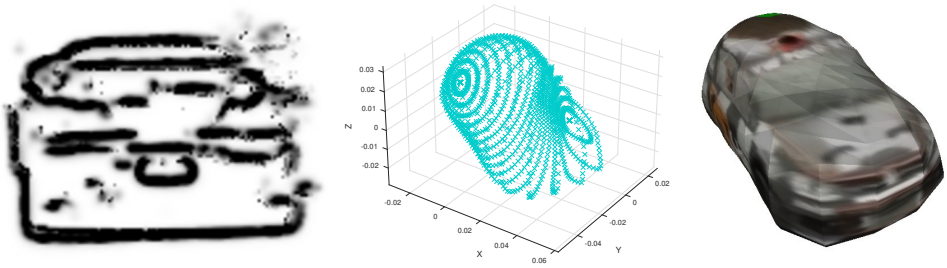
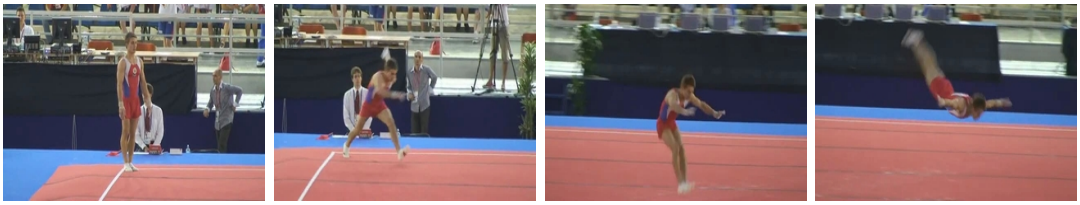


Figure 1.5: Examples of a trackers internal models. Left: an edge quality field of the 2D LT-FLO tracker (Chapter 3); middle: a Gaussian-Process implicit model of the 3D TMAGIC tracker (Chapter 5); right: the same model made explicit and textured.

extraction of richer information than could be provided by a single view and proves invaluable in subsequent tasks such as (object or person) recognition or super-resolution (see Figure 1.4 for an example).

Finally, most trackers employ some sort of internal *model* of the target object. While this model is primarily intended for the tracker's own usage, it often encapsulates interesting and/or exploitable information about the target object and can thus be regarded as one of possible tracker outputs. See Figure 1.5 for examples of internal

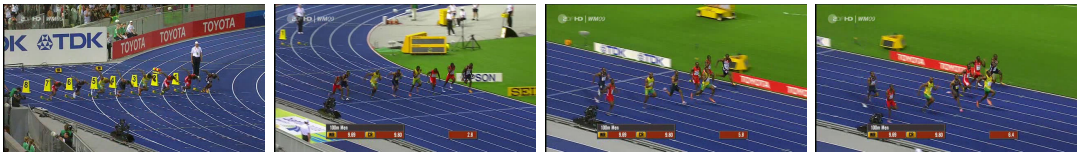


(a) shape variations in the GYMNASTICS sequence

(b) viewpoint variations in the RALLY-VW sequence

(c) illumination variations in the LIVERRUN sequence

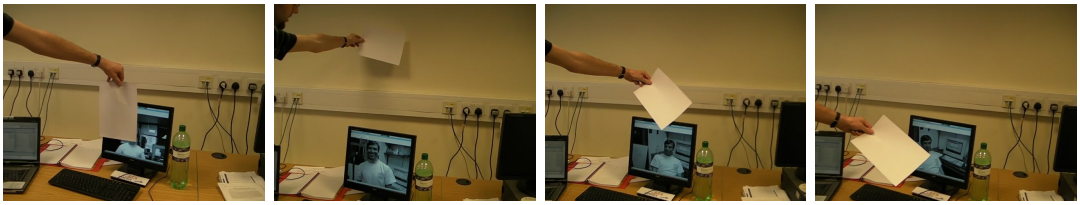
(d) size variations in the LIVERRUN sequence



(e) distractors in the BOLT sequence

(f) background clutter in the SINGER2 sequence

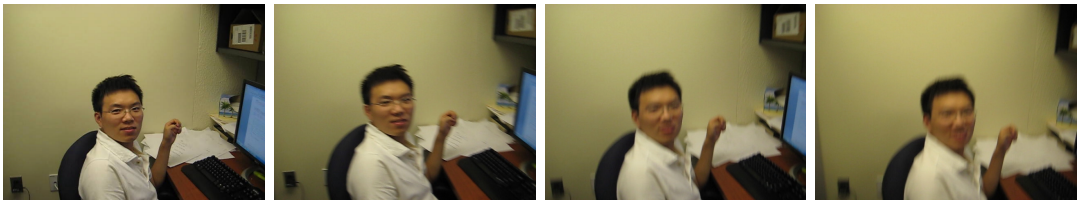
Figure 1.6: Examples of common challenges within visual tracking.



(a) low-textured target in the PAGE sequence

(b) transparent target in the SPACESHIP sequence

(c) full occlusions in the LIVERRUN sequence



(d) motion blur in the BLURFACE sequence

Figure 1.7: Examples of common challenges within visual tracking (continued).

models of 2D and 3D trackers presented in this thesis. A significant part of this thesis is devoted to 3D tracking, where this internal model is a probabilistic model of the 3D shape of the target object's surface. It can be easily extended to an explicit textured mesh model and may even become the major motivation for such a tracker. 3D shape models automatically extracted from videos can be employed in a broad range of applications, ranging from the entertainment industry (models for video-games, 3D film uplifting, *etc.*), through to 3D printing (as an accessible consumer-grade scanning technique) and robotics (object recognition, manipulation,*etc.*).

Visual object tracking is a difficult task, where state-of-the-art algorithms tend to fail in scenarios considered relatively simple by humans. Reliable results have been achieved only in simple settings and/or controlled conditions. There are many challenges that a tracker needs to overcome in order to provide robust, accurate and stable results. To name a few, the target may undergo huge variations in appearance (caused by change of its shape, viewpoint, illumination or distance), while the scene may contain other, possibly very similar, distractor objects, as well as background clutter. Of this potentially immense space of appearance variations, a tracker is given merely one example in a single frame, and a handful of prior assumptions. Additional challenges may include low-textured or transparent targets, camera shake and strong (even complete) occlusions. These challenges are illustrated in Figures 1.6 and 1.7. Furthermore, a tracker needs to be able to track the target for very long duration of time, ideally indefinitely. This work addresses several of these challenges, as described in the following section.

1.2 Contributions

This work aims to address several of the challenges mentioned above, which every long-term tracker needs to face. In this section, the contributions of each chapter are summarised, alongside supporting publications.

Chapter 2 brings an overview of tracking and reconstruction (including visual SLAM) literature. The most influential works of each field as well as recent publications are used to illustrate the background of this thesis and to state its relationship with the state of the art.

In Chapter 3, the problem of texture-independent long-term tracking is addressed. It is shown how line features based on edges in the image can be used for tracking. *Virtual corners* are defined by pairs of the lines. Correspondences of these virtual corners can then be employed to track a range of challenging objects, such as those with no texture (Figure 1.7a) or even transparency (Figure 1.7b). Furthermore, a

redetection scheme is introduced, highly resistant to drift, that facilitates tracking of extremely long sequences with strong appearance changes (Figures 1.6c and 1.6d) and full occlusions (Figure 1.7c). This chapter's main contributions were published in the following articles.

- [128] **K. Lebeda**, S. Hadfield, J. Matas, and R. Bowden. Texture-Independent Long-Term Tracking Using Virtual Corners. *In IEEE Transactions on Image Processing*, 2016.
- [127] **K. Lebeda**, S. Hadfield, J. Matas, and R. Bowden. Long-Term Tracking Through Failure Cases. *In Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, 2013.
- [129] **K. Lebeda**, J. Matas, and R. Bowden. Tracking the Untrackable: How to Track When Your Object Is Featureless. *In Proceedings of the ACCV workshop on Detection and Tracking in Challenging Environments*, 2012.

Some of the issues in visual tracking are caused by suboptimal utilisation of the image information. The object of interest can easily occupy as few as ten or even one percent of the video frame. This causes difficulties in challenging scenarios such as full occlusion (Figure 1.7c) or sudden camera shake causing large inter-frame displacement and motion blur (Figure 1.7d). In Chapter 4, causal relationships in the context of visual tracking are investigated to mitigate these problems. More concretely, the relationship between the tracked object and the camera motion is explored, as well as between different elements of the scene. An information-theoretic approach is taken to identify the presence and extent of such causal links and to measure their properties. It is then shown how these relationships can be exploited to causally predict the future trajectory of the target object, or the trajectory of a fully occluded object. The obtained information can be then supplied as a prior to any tracker, benefiting it by improving robustness. This chapter's main contributions were published in the following articles.

- [124] **K. Lebeda**, S. Hadfield, and R. Bowden. Causal Relationships in Visual Tracking. *Under review for IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

- [123] **K. Lebeda**, S. Hadfield, and R. Bowden. Exploring Causal Relationships in Visual Object Tracking. *In Proceedings of the International Conference on Computer Vision*, 2015.

Chapter 5 focuses on the issues of strong viewpoint variation (Figure 1.6b) in visual tracking, specifically out-of-plane rotation. The conventional approach is to treat such motion as an appearance change. Conversely, this chapter argues that an intrinsically 3D object in the 3D world should be modelled as such, and variation of viewpoint should be treated explicitly as the camera motion, in accordance with reality. Techniques from visual tracking, Structure from Motion, and Simultaneous Tracking and Modelling are brought together into a unified framework, the outcome of which is an approach able to track and model objects during strong out-of-plane rotation. This chapter's main contributions were published in the following articles.

- [126] **K. Lebeda**, S. Hadfield, and R. Bowden. TMAGIC: A Model-free 3D Tracker. *Under review for IEEE Transactions on Image Processing*, 2016.

- [121] **K. Lebeda**, S. Hadfield, and R. Bowden. 2D or not 2D: Bridging the Gap Between Tracking and Structure from Motion. *In Proceedings of the Asian Conference on Computer Vision*, 2014.

In Chapter 6, the problem of tracking and modelling non-rigid objects is studied (Figure 1.6a). It is shown how the (3D) shape of any target object, rigid or non-rigid, can be modelled based solely on the input video sequence, with minimal user input. The object segmentation and more flexible model allow the use of unconstrained in-the-wild videos as the input. Furthermore, it is shown how using dense features allows more general tracking (*e.g.* per-frame focal length estimation) and modelling (higher level of

detail). Finally, the issue of long-term tracking is revisited in the context of 3D tracking and unconstrained videos, providing tracking through shot cuts. This chapter's main contributions were published in the following articles.

- [125] **K. Lebeda**, S. Hadfield, and R. Bowden. Direct-from-Video: Unsupervised NRS*f*M. *Under review for the European Conference on Computer Vision*, 2016.
- [122] **K. Lebeda**, S. Hadfield, and R. Bowden. Dense Rigid Reconstruction From Unstructured Discontinuous Video. *In Proceedings of the ICCV workshop on 3D Representation and Recognition*, 2015.

Chapter 7 concludes this work. It summarises how some of the major failure cases in visual tracking can be addressed and how tracking and 3D reconstruction can be used together to tackle difficult tasks. Finally, possible directions for future work are shown, to which this thesis paves a way.

1.3 Used Notation

Symbol type	Typeface	Examples	Note
scalars	italics	t, E, ρ	
vectors	bold	$\mathbf{f}, \mathbf{X}, \boldsymbol{\mu}$	lowercase usually denote 2D vectors, uppercase 3D; all vectors are column-matrices
matrices	typewriter	$\mathtt{P}, \mathtt{E}, \mathtt{I}$	including bitmap images
sets	calligraphic	$\mathcal{C}, \mathcal{L}, \dot{\mathcal{X}}$	sets of 3D entities marked with a dot above
others	roman	$\mathrm{M}, \mathrm{E}, \Psi$	functions, probability distributions, structured entities, *etc.*

Table 1.1: General notation used in this work.

In this work, a general notation is used as tabulated in Table 1.1. Homogeneous entities are denoted by a tilde, *e.g.* $\widetilde{\mathbf{x}}$, and are considered normalised in mathematical operations. Points are normalised to a unit last component, lines and planes to a unit-length normal vector. A hat over a symbol denotes an estimate or an approximation of a variable, while a bar under it identifies a tentative or local entity,

Accent	Example	Explanation
tilde	$\widetilde{\mathbf{x}}$	homogeneous entity
hat	$\hat{\mathbf{x}}$	estimate or approximation
bar under	$\underline{\mathbf{x}}$	tentative or temporary entity
bar over	$\overline{\mathbf{x}}$	various (context-dependent)
dot above	$\dot{\mathcal{X}}$	3D set

Table 1.2: Accents used within this work.

Type	Letter	2D	3D	2D sets	3D sets
generic **F**eatures	F	$\mathbf{f}_i^t$	$\mathbf{F}_i$	$\mathcal{F}^t$	$\dot{\mathcal{F}}$
(sparse) points	X	$\mathbf{x}_i^t$	$\mathbf{X}_i$	$\mathcal{X}^t$	$\dot{\mathcal{X}}$
Dense points	D	$\mathbf{d}_i^t$	$\mathbf{D}_i$	$\mathcal{D}^t$	$\dot{\mathcal{D}}$
Lines	L	$\mathbf{l}_i^t$	$\mathbf{L}_i$	$\mathcal{L}^t$	$\dot{\mathcal{L}}$
Supervision	S	$\mathbf{s}_i^t$	$\mathbf{S}_i$	$\mathcal{S}^t$	$\dot{\mathcal{S}}$
Model points	M	–	$\mathbf{M}_i$	–	$\dot{\mathcal{M}}$

Table 1.3: Feature types and typefaces used within this work.

e.g. in $\mathbf{x} = \arg\min_{\underline{\mathbf{x}}} \mathrm{f}(\underline{\mathbf{x}})$. These are tabulated in Table 1.2. On several occasions, a bar over a symbol (*e.g.* $\bar{\mathbf{x}}$) is used. This notation is however context dependent and is explained whenever used.

Throughout this work, a number of *feature* types are used in different contexts. The same type of feature (*e.g.* point or line) is referred to using the same letter. 2D features are denoted by a lower-case vector (*i.e.* bold), 3D features by an upper-case vector. Sets of features are printed in a calligraphic typeface; sets of 3D features are marked with a dot above the symbol. These are all tabulated in Table 1.3 for clarity.

Chapter 2

Related Work

This chapter discusses literature related to the topic of the thesis. The most influential "seed" publications are examined as well as current state-of-the-art approaches. The chapter is divided into several sections, each covering one particular topic of interest. In Section 2.1, the area of 2D visual object tracking is explored and the main trends within the area discussed and compared. Since the problem of strong out-of-plane rotation is tackled using 3D tracking, Section 2.2 discusses this topic, both model-based and model-free. This is then accompanied by two sections focusing on areas from which 3D tracking emanated: both rigid and non-rigid Structure from Motion (S*f*M), and Simultaneous Localisation And Mapping (SLAM), in Sections 2.3 and 2.4, respectively. Finally, the problem of causal motion models is explored within this thesis. Therefore causality is examined in Section 2.5, both in general scientific use and within the context of computer vision.

2.1 2D Visual Tracking

One of the most influential works in the field of visual tracking is undoubtedly the Lucas-Kanade (LK) tracker [134]. This technique, also called tracking by local optimisation, iteratively matches image patches by linearising the local image gradients

and minimising the sum of squared pixel errors within a tracked patch. It has been recently extended using segmentation-like object/background likelihoods [153]. Since the LK tracker tracks an image patch, rather than a concrete object, it is sometimes referred to as a *sparse optical flow* algorithm. During the iterations, a matrix of image gradients is inverted. To achieve good stability in tracking, this matrix needs to have large eigenvalues and the majority of commonly used image features fulfil this condition. However, a new type of features was developed in order to explicitly address this in the extended Kanade-Lucas-Tomasi (KLT) tracker [187], so-called *good features to track* [169].

While the LK tracker has become extremely popular, it suffers from several problems such as illumination variance (caused by the use of the Sum of Squared Distances (SSD) metric). An alternative formulation has been proposed, using *mutual information*: Mutual Information for Lucas-Kanade Tracking (MILK) [40]. This information-theoretic metric is invariant to illumination (allowing, for example, the registration of near-infrared and visible-light images) while keeping the computational costs close to the standard SSD formulation.

Visual tracking has significantly evolved since the LK and KLT trackers. Modern trackers can be roughly divided into two separate classes. Trackers from the first group, using *tracking by detection* employ advanced machine learning techniques to distinguish the object from the background. In the second class, *feature-based trackers* extend the idea using a group of semi-independent *tracklets* or *features* (small parts of the target) which are tracked using an LK or similar simple tracker, with an upper level managing them.

2.1.1 Tracking by Detection

All of the trackers mentioned above try to minimise the difference between the template and the image. In other words, they attempt to maximise generalised *correlation* between the two. Commonly used metrics include standard correlation, SSD and

mutual information. LK, as well as MILK, use *local optimisation* to predict tracker location in the new frame. Many correlation based trackers however take different approach, based on object scoring[21, 83]. Numerous object location hypotheses are sampled within the new frame and the one with the highest object score is chosen as a result. A simple version would be to use sliding window for generating these hypotheses. In state-of-the-art trackers, however, the sequential nature of tracking is exploited such that the sampled windows lie nearby the object pose in the previous frame (or, if a motion model is used, in the vicinity of the predicted move).

A special kind of tracking by detection is *particle filtering*. These approaches use a set of object pose hypotheses (the *particles*) to capture the possibly multimodal classifier objective. Similarly to the standard tracking by detection, the output is defined by the particle with the highest detection score. However, multiple hypotheses are kept as long as their score is sufficiently high, instead of keeping only the best one. One of the most notable is Conditional Density Propagation (CONDENSATION) [95], which brought into the field of visual tracking the use of particles for non-parametric modelling of a pose probability distribution. In their IVT tracker, Ross *et al.* [166] extended particle filtering by introducing *incremental learning* of an object appearance subspace that allowed the model to adapt to changes.

IVT is an example of a more recent approach to tracking: *classification* based tracking. Here tracking is defined as a classification task. There are two classes to be decided between: the object and the background. In the first frame, the classifier is trained on the positive sample from the initialisation (possibly augmented by synthetic training examples sampled from its near neighbourhood) and negative samples from the background. This model is usually updated online assuming the tracking so far has succeeded. The simplest, naïve solution is to add the tracked bounding box as a positive example after each frame and retrain the model. While this gives a tracker the ability to adapt to changes in the object appearance, it also provides opportunities for accumulation of error – *drift*. Different trackers employ different measures to address this issue, ranging from intelligent update conditions to advanced machine learning techniques.

For instance, Grabner *et al.* [62, 63] employ an *online boosting* method to update the appearance model while minimising error accumulation. Babenko *et al.* [10] use *multiple instance learning*, instead of traditional supervised learning, for a more robust tracker. Zheng *et al.* [215] address drift during tracking explicitly, using a dynamic set of basis classifiers, employing different basis classifiers for different problems. Since the object can get occluded by other elements of the scene, it is useful to detect this and prevent learning these occlusions as a part of the model. This is addressed in the ℓ_1-norm tracker [210], which learns a dictionary from local patches.

All of these trackers, however, suffer from the need to convert the estimated object position into a set of labelled training examples, and to couple the objective for the classifier (label prediction) to the objective for the tracker (object position estimation), which is difficult to perform optimally. This is solved in *regression* based trackers, predicting the location output directly to avoid the need for these intermediate steps. This is somewhat similar to the original LK tracker, where the estimated inter-frame shift is computed from the difference between the template and the target frame. Mayol and Murray address the question of regressing the transformation parameters in their work [139]. Another, more recent example is the successful Struck tracker [75], which addresses this by explicitly using *structured output prediction* of the target translation.

In recent years, a great deal of attention was paid by the computer vision community to Convolutional Neural Networks (CNNs), which provide great performance in many tasks. However, until recently the use of CNNs in visual tracking was limited by the lack of training data. This is further exacerbated by the fundamental difference between visual tracking and image classification, which is often used for pre-training networks before fine-tuning them for a particular task [144]). As the results of the latest Visual Object Tracking challenge (VOT) 2015 [114] show, this is not the case any more. The top two trackers in this benchmark were based on convolutional networks. At the top of the leader-board is the MDNet [144], using classification based tracking with a CNN. To mitigate the lack of training data, the network is divided into two parts: shared layers and domain specific layers. While shared layers are pre-trained

on a wide range of tracking sequences, domain specific (the uppermost) layers are trained online for each sequence. The final proposal is refined using bounding box regression [61]. The second place in the challenge [114] is taken by a version of the SRDCF tracker using convolutional features. SRDCF is a correlation-based tracker, removing the assumption of circular structure by spatial regularisation. Replacing its used features (such as the Histogram of Oriented Gradient (HOG)) by convolutional features improves performance as indicated in the challenge.

2.1.2 Feature-based Tracking

As a target is tracked through a video-sequence, it can undergo appearance variation for numerous reasons. The brings a challenge to every tracker in how and when the target representation (model) should be updated. This is the so called *template update problem* [138]. If the object model is not updated, it cannot sufficiently cover the space of possible appearance. On the other hand, naïve updates lead to the accumulation of small errors – drift. For this reason, trackers often employ an upper layer managing the update of the target model (template).

An upper, managing layer is also used in feature-based trackers, which employ a cloud (also called *flock*) of features. Each of the features is tracked (at least partially) independently and provides a hypothesis of the object motion, or a constraint on it in cases of more complex motion model (*e.g.* including scale change or rotation). The final resulting object pose is then reported as an agglomerate of these hypotheses/constraints. The upper level is responsible for management of the features – their creation, removal, re-detection, *etc.*

One of the first trackers built on this principle is the Flocks of Features (FoF) tracker by Kölsch and Turk [113]. This tracker, tracks human hands in a video from a head-mounted camera, using a *flock* of KLT features. To manage them, an approach was taken, inspired by flocks of birds. Without any bird in control, the flock still stays packed tightly together in a decentralised way, and the motion of the flock can be observed although the individual trajectories might differ significantly. The assumption

used is that good features (like birds) stay close together, but maintain a minimal distance (safe to fly) from any other feature. This provides robust tracking despite failure of any individual feature.

This principle has been extended by Kalal *et al.* in the Median Flow algorithm [105], where validation of features is performed based on *forward-backward error* [39]. Intuitively, this means that if a good feature is tracked to its new position in the new frame, and then back in the original frame, it returns to its original location. This principle was further improved in the more robust Flock of Trackers (FoT) [137, 199] by Vojíř *et al.* In FoT, features are additionally (in)validated based on a number of measures such as neighbourhood consistency or Markov chain probability.

Another recent example is the Local-Global Tracker (LGT) [25] by Cehovin *et al.*, using a coupled-layer visual model. The local layer consists of independently tracked visual patches, constraining the appearance of object components. The global layer models object features such as colour, shape or motion. The global model is learned from the local patches and it in turn constrains the addition of new patches. Consistency of local trackers is enforced, however changes in shape are possible, allowing LGT to track highly non-rigid objects. This makes LGT very robust as was made clear in the VOT challenges (see below).

On the boundary between these two directions (tracking by detection and using features), Kalal *et al.* combined tracking with detection in their Tracking-Learning-Detection (TLD) [106] framework, where (in)consistency of the tracker and detector helps to indicate tracking failure. While the (feature-based) *tracker* estimates the frame-to-frame motion, the *detector* treats every frame as independent (as in the tracking-by-detection scenario). Positive and negative examples are *learned* according to the (dis-)agreement of these two components, improving further detection. Explicit modelling of failures for both components coupled with independent detection makes this tracker suitable for long-term tracking, with inherent drift-resistance and redetection after full occlusions.

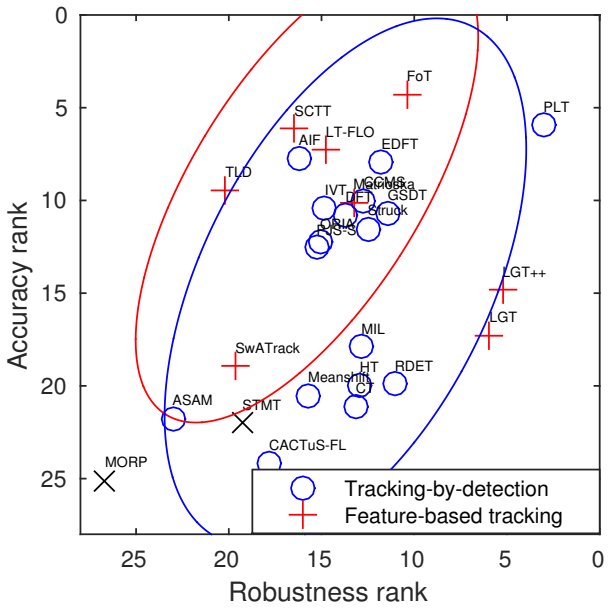


Figure 2.1: Accuracy-robustness plot of the Visual Object Tracking challenge (VOT) 2013 results. The 95th percentiles of fitted normal distributions are shown, in the feature-based case with the LGT trackers (both vanilla and enhanced) excluded.

2.1.3 Comparison of Tracking Techniques

In general, feature-based trackers attempt to estimate the exact frame-to-frame motion (transformation) of the bounding box. This requires the features within a cloud to be at least partially consistent with each other. On the other hand, in the tracking-by-detection scenario, the classifier decides whether the object is present in a sampled bounding box. Intuition is that this may lead to lower accuracy for tracking by detection. On the other hand, the inherent insensitivity (to intra-class variance) may increase robustness to the object appearance changes. As visualised in Figure 2.1, experimental results from VOT 2013 confirm this intuition: feature-based trackers show overall higher accuracy and lower robustness, according to the standard VOT measures [115]. The highly robust LGT [25] and LGT++ [209] trackers were excluded from the cluster when plotting the 95th percentile ellipses, due to the flexible bounds between the low-level features. With these included, the difference in robustness disappears; feature-based trackers however have on average still higher accuracy.

2.1.4 Insufficient Texture and Edge Features

A quality, common to all of these trackers, is their reliance on the object texture. In tracking by detection, the learned detector needs a consistent texture to distinguish between the object and the background, making it unsuitable for texture-less tracking. Similarly all the feature-based approaches are based on variants of point features, which renders them ineffective in cases where these are scarce, or unstable.

There have been previous attempts to decrease the reliance on object texture, including the use of edges. The aperture problem (see Figure 3.2) renders these spatially unstable, as neighbouring pixels along the direction of the edge are indistinguishable [78] and a unique motion cannot be determined. For this reason, conventional trackers often avoid edges. However, such features are still valuable: Tsin *et al.* [191] fit line segments modelling the object to detected edge-points. The same approach was used in [42] and [77], when searching for the pose of a wireframe-represented 3D object. It should be, however, noted that in all of these cases the line-based model was supplied by the user and there has been no previous use of line features for model-free tracking.

2.2 3D Visual Tracking

3D monocular tracking attempts to recover the trajectory of the object as well, however in a 3D world (relative to the camera) instead of in the image plane. Typically, such techniques employ 3D models of the object, which are either user-supplied or previously learnt. Examples of this are the tracking of the pose of human bodies [152, 172], vehicles in traffic scenes [33, 163, 203] or general *boxes* [110]. Model-based 3D tracking is not limited to a particular class of object; examples include planar trackers, generic model supplied as a mesh or wireframe, and tracking via fiducial markers.

Planar trackers use the assumption the target is planar in 3D and, since they do not attempt to learn the target shape on the fly, can be seen as a type of model-based tracker. These usually estimate homography between the 3D plane and its projected

image to obtain the camera trajectory. These find many applications, such as UAV control [141] or Augmented Reality (AR) [96, 174]. In some cases, the whole scene can be modelled as a set of planar patches [173] and the camera trajectory can then be estimated within the scene.

Generic 3D models are often supplied as wireframes. Then the contour of the object can be used for tracking, as mentioned above [42, 77], using either the gradient information directly, or explicitly extracting line features during image preprocessing. For textured objects, the model can be in the form of a polygonal mesh, which is fitted to the video frames through a reference image [193] (which has the camera pose set manually). This approach extends natually into the non-rigid case and is the base for many modern template-based Non-Rigid Structure from Motion (NRS*f*M) techniques (*e.g.* [213]).

Finally, fiducial markers are often used to track the camera pose relative to an environment [90, 112], usually for AR applications. While marker-based approaches tend to be very accurate and robust in general, they however require environment engineering and may be disruptive to the end user. This may disadvantage them against natural-featue based methods.

The focus of this thesis is on model-free tracking and there has been considerably less research in this field. While there have been relatively few attempts at learning 3D tracking models on the fly, such approaches are fundamental to online S*f*M. However, these assume the interest is in reconstructing the whole scene. In the area of visual object tracking, 3D based approaches must extract 3D shape and trajectory even for objects represented in only a minority of the video frame. For this reason, Kundu *et al.* [117] use motion segmentation to track and reconstruct moving bodies in their SLAM pipeline.

An example of a recent model-less 3D tracker is the work of Feng *et al.* [50], who introduced the idea of 3D monocular tracking with no offline modelling or training, using explicit colour-based segmentation. This allowed reconstruction of the segmented

object using visual SLAM techniques. However, they do not attempt to estimate the surface shape beyond just a cloud of points. Another similar approach is the work of Prisacariu *et al.* [162] who use level-set techniques for 3D modelling. This work is however not online, as it processes the video-sequence as a batch, which significantly limits possible applications of such a "tracker".

2.3 Structure from Motion

Structure from Motion (S*f*M) is an area of great interest for the computer vision community. Here the task is to simultaneously estimate the structure of a scene observed by multiple cameras, and the poses of these cameras. Images of the scene can be either unordered [60, 3] or ordered in a sequence [156, 160, 184] (in such a case this knowledge can be exploited). Great advances have been recently achieved in the scalability and automation of the S*f*M process by Snavely *et al.* in their Bundler system [3, 179], based on Bundle Adjustment (BA). Their approach is *sequential*, i.e. starting reconstruction with a smaller number of cameras and then adding progressively more. A different approach has been taken by Gherardi *et al.* [60], who merge smaller reconstructions hierarchically.

The main differences between the topic of this thesis (especially 3D tracking) and conventional S*f*M techniques is firstly its online nature and secondly the active background segmentation. While computing the overall scene geometry, S*f*M extracts the dominant motion and discards outliers. However, the object of interest can easily be as small as 1 % of the frame area. In such a case it is completely ignored by S*f*M algorithms as an outlier and the background scene is reconstructed instead. This is the exact opposite of what is sought in visual tracking, where the minority of the scene, the target, is of an interest, while the background needs to be *actively ignored*.

Regarding the online processing, there have been several exploratory works investigating online SfM. These mostly aim to provide feedback to a human operator, performing the scanning procedure. While ProFORMA by Pan *et al.* [156] builds a

partial coarse 3D model by tetrahedralisation of a cloud of sparse features to reveal unseen parts of a scanned object/scene, the work of Hoppe *et al.* [92] goes further and provides the operator with a redundancy map of the reconstructed area.

In the field of texture-independent reconstruction, there has been work on line-based reconstructions. The basics have been laid by Hartley in [79], using the *trifocal tensor* (although it was not named as such) to reconstruct a 3D scene from a triplet of images. The theory has been further developed by Zhang [214] who explored the properties of lines and line segments in the whole S*f*M pipeline – from matching through Perspective-*n*-Lines (PnL) to 3D reconstruction. The work of McLauchlan *et al.* [140] suggests using higher-level constraints arising from lines and planes in the S*f*M pipeline. This allows reconstruction of nearly texture-less scene from a sequence of images.

2.3.1 Non-rigid Structure-from-Motion

Non-Rigid Structure from Motion (NRS*f*M) is a subarea considering reconstruction of time-varying object shapes, usually from a video-sequence. Most approaches to NRS*f*M are based on factorization [188], as introduced by Bregler *et al.* [20]. To simplify the problem, the orthographic camera model is used [32, 154, 185]. This way, the 2D point locations (per frame) can be expressed as an affine function of the 3D locations, which are in turn a linear combination of *basis shapes*. The set of projection equations is then rewritten as a matrix-matrix multiplication, for each 3D point and video frame where it is visible. The projection multiplication is decomposed back to the factors, yielding the camera parameters (translation and rotation, for each frame), basis shape mixing parameters (i.e. coefficients of the linear combination, for each frame) and basis shape locations (for each point).

This problem is inherently ill-posed, having significantly more unknowns than equations. To render it solvable, additional constraints are applied. In the original paper [20], the low-rank constraint was applied, effectively setting/limiting the number of basis shapes. All following approaches use this constraint and apply additional con-

Property	Zollhofer [217]	Newcombe [146]	Garg [57]	Yu [213]	Chapter 6
Template-free		✓	✓		✓
Direct	✓	✓		✓	✓
Monocular RGB			✓	✓	✓
Perspective camera	✓	✓		✓	✓
Closed mesh w. self-occl. handling	✓			✓	✓

Table 2.1: Comparison of state-of-the-art approaches for reconstruction of generic dynamic shapes. Inspired by [213].

straints, priors, heuristics and regularizations. These include spatial smoothness of shape [12, 154, 159, 197] (the points lying close to each other in 2D tend to lie close to each other in 3D); temporal smoothness of shape [4, 12, 154] (the shape changes smoothly over time); temporal smoothness of camera poses [4, 154] (the camera trajectory is smooth in time); and inextensibility [159, 197] and other physics-based priors [4, 5].

One limitation of this formulation is that it is conditional on all 2D tracks spanning the length of the video. This condition is removed by either estimating the missing data [47] or using methods based on Bundle Adjustment (BA) [4, 12]. In this case matrix factorization is replaced with global optimization of the model parameters (basis shapes, mixing coefficients, camera trajectory). Another reason for the use of Bundle Adjustment, besides natural inclusion of missing data, is the ability to use more complicated camera models. Finally, BA-based techniques also scale well in terms of memory and computational time.

Table 2.1 compares the properties of selected state-of-the-art NRS*f*M approaches. Although there are many more works, this comparison captures general trends which can be observed in the field. All current techniques use either a template, a precomputed set of 2D trajectories, or an RGBD camera to address the task. There has been no previous approach to solve NRS*f*M which would be at the same time direct (*i.e.* using no precomputed tracks), template-free and using only a single RGB camera. The work presented in Chapter 6 addresses the task of simultaneous tracking and non-rigid reconstruction as indicated in the last column of the table.

	SfM	Tracking	Visual SLAM
Sequentiality	images may or may not be in sequence	online (preferably real-time)	online (preferably real-time)
2D / 3D	3D	2D or 3D	3D
Appl. focus	model	camera trajectory	model and camera
Target	scene	object of interest	scene (and possibly objects)
Input	monocular RGB (from definition)	RGB or RGBD	RGB, RGBD, stereo cameras, *etc.*

Table 2.2: Comparison of the studied problems: S*f*M, SLAM and visual tracking.

2.4 Simultaneous Localisation and Mapping

SLAM has been one of the fundamental challenges of robotics for decades, with an active research community. The task is to create a map of an unknown environment on-the-fly as it is traversed and at the same time to localise the robot within this environment. A number of sensors are used to gather the necessary information about the outer world, including video cameras. In principle there is no difference between localisation within an unknown environment and tracking a 3D object. Both can be essentially reduced to the problem of finding the camera pose relative to the map/model, based on 2D-to-3D feature correspondences. Similarly, mapping of such an environment is equivalent to object modelling. Again, in principle the task consists of finding 3D locations of features based on 2D observations from a video, and possibly building a higher-level map or model. This results in an interesting duality between the two problems, the only difference being the point of view – from the inside (SLAM) vs. from the outside (3D tracking). The problems (including S*f*M as well) are compared in Table 2.2. Clearly, there is an opportunity for an overlap of techniques used in SLAM and visual tracking, thus a review on (especially vision-based) SLAM is presented in this section.

Robot navigation and mapping were at first studied separately [43]. One of the first to treat them simultaneously was Smith *et al.* [177], who used an Extended Kalman Filter (EKF) [59] to capture and optimise the map and trajectory together,

taking correlation and covariances within and between these into account. Another great breakthrough was the discovery of convergent properties of SLAM in the work of Durrant-Whyte *et al.* [44]. This publication was also the first one to use the term (and acronym) "SLAM". Additional theory on convergence was developed and some of the first realistic experiments were carried out by Csorba [31]. Another vital part of any modern SLAM technique are *loop closures*, *i.e.* recognition of previously visited locations and associating them with the correct locations in the map. Studied in the work of Bar-Shalom [11], this still receives considerable interest within the community due to the importance of long-term consistency. Although most of the early SLAM works used radars, numerous sorts of sensors have been in use, such as sonars, laser scanners or video cameras (either RGB or IR [198]).

Video cameras are an extremely valuable type of sensor due to their low cost, passive operation and long-distance high-resolution measures of the environment's visual properties. They are, however, burdened by a higher computational cost necessary to process and use such data [9]. Classic SLAM techniques (*e.g.* based on scanning laser data) can use visual information alongside their regular methods. One such example is the work of Newman and Ho [150], who use image features to initialise *loop closures*. When a robot's estimated trajectory deviates grossly from its actual path, it may become impossible to connect the ends of a loop, *i.e.* to explain the current pose based on a previously seen part of the map. A global database of seen visual features can facilitate a drift-resistant linking mechanism: an attempt for a loop closure is commenced if there is a match between currently and previously seen features.

Image/video input can be, however, used as the main source of information. Since all the measurements are only angular and relative, the scene can be modelled by purely visual information only up to a similarity transform, *i.e.* the coordinate system is free to rotate, shift and undergo isotropic scaling. To overcome this and provide metric (in world units) information, additional sensors are often used, such as Inertial Measurement Units (IMUs) [73, 99], robot odometry [14, 15], depth sensor [7] or GPS [170]. Alternatively, the metric measurements can be provided knowing the physical set-up

of a fixed camera stereo rig on the robot, if this is used [56, 189]. Finally, it is possible to incorporate different kinds of higher knowledge, such as fixed dimensions of reconstructed common objects (such as human faces or an artificial calibration object). However, in the context of this thesis, the only available input is a monocular RGB video; hence monocular visual SLAM [36] is of primary interest. The idea of metric reconstruction is therefore abandoned and all models are of an arbitrary scale.

The first solely monocular vision-based SLAM technique was MonoSLAM developed by Davison *et al.* [36]. They used sparse probabilistic 3D mapping using point features of [169] to obtain real-time (30 Hz) mapping and navigation in a room-sized environment. Similarly to the depth-sensors based SLAM literature, an EKF is used. This system had to be nevertheless initialised manually; this was done by placing a known object in front of the camera at the beginning of the video-sequence. To this day, initialisation features as a part of the majority of SLAM techniques [142], preventing completely solving the *kidnapped robot* [46] scenario (when the robot is turned on in a completely unknown environment and may not use any prior knowledge of it).

This framework was later extended to work with straight lines [87, 176]. These algorithms (based on an EKF as well) were similarly able to run in real time and typically managed a very low number of features in both the point and line clouds. Using line features, the dependency on texture is significantly lowered, allowing employment in a standard indoor (*e.g.* corridor) environment, where straight lines are abundant and most surfaces are painted by a single colour.

More recently, there has been a shift from the EKF to BA [91]. There has been an observation that while filtering-based methods may still find their use in scenarios with very limited resources, methods based on optimisation (such as BA) provide better accuracy per unit of computing time [181].

In recent years, growing computational power and its more effective use has led to a transition from sparse to dense features. An example is the popular technique Parallel Tracking And Mapping (PTAM) [111]. Running the tracking and modelling parts of

the algorithm in parallel allows for real-time tracking to be performed independently of the more computationally intensive optimisation. Using such a separation provides real-time execution with maps of sizes up to thousands of features.

However, even denser maps can be achieved, ultimately leading to whole-image-alignment works [148]. A particularly interesting work is Dense Tracking And Mapping (DTAM) by Newcombe *et al.* [149], which can be seen as standing between S*f*M and SLAM. It offers real-time reconstruction of a static scene based on a monocular video stream, using a variational approach. One aspect, common to all of these methods is that they assume the modelled object covers the majority of the scene. When that is the case, the distractors can be avoided by a simple outlier rejection. As explained in the previous section, this is not the case for a large portion of video footage, which could be used for 3D reconstruction.

Modern approaches to monocular visual SLAM [183] try to remove this assumption. However, while increasingly focusing on dynamic scenes, the main aim within the scope of SLAM is increased robustness against occlusions. Although recent approaches such as ORB-SLAM [143] offer excellent performance in cases of strong occlusion, the requirement that the modelled target is the only part of the scene being preserved in time is restrictive. This clearly prevents direct use of SLAM techniques in 3D tracking (although the principle is similar) and emphasises the need for object/background segmentation.

There has been a significant amount of work in the area of online 3D modelling based on depth sensors. The most notable in this area is the work of Newcombe *et al.*, in particular Kinect Fusion [147] and Dynamic Fusion [146]. However, since this thesis focuses on tracking and modelling from strictly monocular RGB video, a more detailed review is omitted here. Recommended recent overviews for 3D modelling and tracking from RGBD are [26] and [180], respectively.

2.5 Causality

Chapter 4 of this thesis is focusing on motion models in visual tracking, especially causality-based. For this reason, literature concerning causality detection and its use is reviewed in this section. There have been numerous philosophical publications on causality in both ancient and modern times, originating from Aristotle [8] and being significantly influenced by Hume [94]. This section focuses more on applied rather than philosophical publications; a recent overview [158] is therefore recommended for further reading. It should be noted that it is impossible to reason about true causality without higher, semantic understanding of the scene. Therefore this thesis works with *predictive* causality instead, which reasons about apparent causal links instead of true causation.

One of the early uses of causality for time series analysis was done by Granger [65]. He proposed a statistical causality test, determining the presence of causal relationships between two normally distributed time-series. This approach has become known as *Granger causality* and has been successfully used in economics [84, 186], neurosciences [54, 74], and recently in computer vision [161, 211]. Although it has been revised and improved over the decades (*e.g.* Hacker and Hatemi-J [68] avoided the assumption of a normal distribution), Granger causality is suitable only for linear signals, since it is based on linear regression.

More recently, a novel concept of measuring causality has been proposed in the form of Transfer Entropy (TE) by Schreiber [168]. TE has since found its place in many areas, including again neurosciences [196], chemistry [13] and others [102], however it has not been previously used in computer vision. As the name suggests, it is based on information theory and is therefore able to detect arbitrary non-linear relationships. In this work, TE is used in Chapter 4 to measure causation in visual object tracking, capturing possibly complex relationships between the motion of the camera and the object.

Transfer entropy, as well as the basic measure of information, Shannon entropy, generally works with probabilities of states of discrete random variables (vectors in the multivariate case). However, both camera and object motions are inherently continuous processes, despite natural quantisation on a pixel grid. Fortunately, TE can also be calculated for continuous processes [88, 102], using an alternative formulation with probability densities and a *differential entropy*. Kaiser and Schreiber [102] proved that even though differential entropy *is not* invariant to coordinate change, continuous TE *is* invariant (to C^1-diffeomorphisms in general). This ensures consistent results regardless of image rescaling, cropping, etc.

As noted above, causal relationships have been previously examined in the area of computer vision for linear relationships. Fan *et al.* [48] used Granger causality to explore actions and the temporal dependencies between them in a surveillance scenario. This was then used to cluster and classify video-clips, according to the actions present. In a similar direction was the work on learning causal relationships between events in video-sequences, which has clear potential in action recognition and related tasks. An example is Fire and Zhu [52], who use *Causal And-Or Graphs* and Bayesian grammar models for inference about hidden effects, otherwise undetected. Gupta *et al.* [66] similarly use And-Or Graphs for automatic creation of descriptions for sport footage, while Sumioka *et al.* [182] use causality to learn joint attention for robots (in a human-robot interaction scenario). The work of Brand [19] explores the causal physics of the scene (how the mechanics of objects influence other objects). Similarly Mann *et al.* [135] derive a computational theory describing the kinematic and dynamic properties of a video-sequence.

Prabhakar *et al.* [161] use Granger causality on sequences of keywords (quantised spatio-temporal visual features) directly for the task of human action recognition. Yi and Pavlovic [211] perform the same task, but based on motion-capture data. They use Granger causality to infer the edges in a joint-influence graph of the human body, which improves the performance compared to fixed graphs. Narayan and Ramakrishnan [145] remove the need for motion-capture systems, using causal relationships between clusters

of dense trajectories. Finally, Zhou *et al*. detect causal links (using Granger causality) between pairs of trajectories obtained by a visual tracker to provide additional features for bi-actor action recognition.

However, there has been no previous work in the field of computer vision exploiting the modern TE approach to causality estimation, and no use of any type of causality for visual object tracking. Learning causal relationships between the motion of the camera and of the tracked object, or between multiple objects, can significantly help a tracker, when used in a causal motion model. In both scenarios of tracking by detection and feature-based tracking, the tracker can benefit not only by improved accuracy, but also by support during challenging events in the scene, such as camera shake or full occlusion.

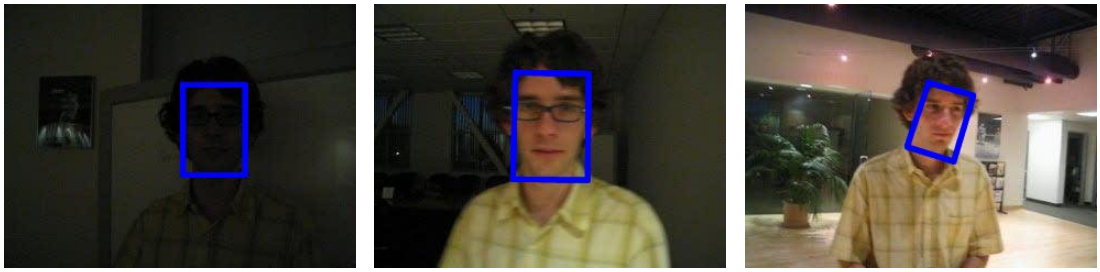
Scene context has been previously used to improve robustness of a tracker in the work of Grabner *et al*. [64]. Their approach uses the Generalised Hough Transform to find *supporters* within the video features, *i.e*. features which can be useful for determining the target object location. These are then used to help tracking in cases where the followed object might be occluded or lost. Another publication on a similar topic was by Dinh *et al*. [38], who added the notion of *distracters*: objects similar but unrelated to the target. The two approaches, however, use very simplistic models for both the discovery of the relationships and the location prediction, allowing only for a constant offset and no time delay between the target and its supporters. On the other hand, the use of transfer entropy and machine learning in this thesis allows complex, nonlinear relationships to be detected and employed.

Chapter 3

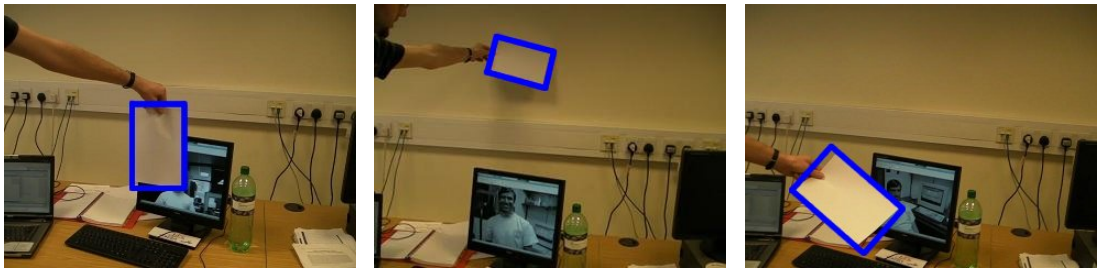
Long-Term 2D Tracking of Texture-less Objects

This chapter focuses on two common sources of failure in visual object tracking: low texture of the target and full occlusions in long-term scenarios. These are often difficult, as a tracker needs to learn and adapt a model of object appearance (commonly described by its texture) to follow the object or to redetect it after full occlusions. This chapter investigates solutions to these common sources of failure by abandoning more common point features and using lines instead; line features are more robust to lighting and are present even in cases of low-textured objects where point features are scarce.

Long-term trackers attempt to model and adapt to changes in object appearance over time, using multiple observations to enrich their representations. This in turn leads to drift (the accumulation of small frame-to-frame inaccuracies) due to the difficulty of unsupervised learning. Recent approaches overcome this through a combination of detection combined with local search and intelligent online update strategies, which can compensate for drift via redetection. As such, consistency of appearance is vital, because radical changes can cause tracking failure or corruption of the model. The notion of appearance typically relies on texture or other strong visual attributes. However, there are a whole range of scenarios where sufficient texture is either absent or



(a) frame #1 (b) frame #300 (c) frame #680

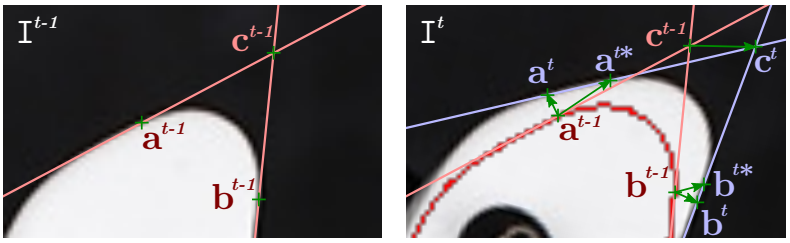


(d) frame #1 (e) frame #172 (f) frame #539

Figure 3.1: Examples of results on challenging sequences.

highly variable due to changes in pose or scene illumination. While this is only true for a subset of tracking sequences, it is a common source of tracking failure for most approaches.

Figure 3.1 shows example frames from two challenging sequences which typically lead to tracking failures. Figures 3.1a to 3.1c show images from the DAVID sequence of Ross *et al.* [166] where the initial target face (Figure 3.1a) is so dark that visual features are almost indistinguishable to the human eye. However, as shown by the bounding box, illumination invariant trackers are capable of tracking the entire sequence. Even the original authors do not track this sequence from the start, only beginning at the 300th frame (shown in 3.1b) where the visual appearance from lighting is more consistent with the remainder of the video. This sequence was recently used in a visual tracking benchmark [206] starting from the latter frame as well. Similarly Figures 3.1d–3.1f show a texture-less object (a single piece of white paper on a light background). Trackers which rely on texture or appearance fail on this sequence. Again the resulting bounding boxes show that it is possible for a texture-independent tracker to overcome this challenge.



(a) Frame $t-1$ (b) Frame t

Figure 3.2: Establishing line correspondences regardless of the aperture problem.

3.1 Exploiting Edge-based Line Features

The task of the tracker is to find a pose $\mathbf{p}^t$ (position, rotation, size) of an object in frame $\mathtt{I}^t$. In feature-based tracking, the tracked object is represented as a set of *features*. The commonly used point features are however scarce in scenarios involving low-textured objects, which are relatively common (*e.g.* painted man-made objects). Furthermore, they tend to lie near the object boundaries in such cases, and thus to be influenced by the background. Therefore, edge-point features are explored here instead, *i.e.* locations of locally maximal intensity gradient. The fundamental elements for tracking are tentative correspondences of lines tangential to edges in the image. In other words, to estimate the frame-to-frame motion of the target object, correspondences of lines are used, which are in turn defined by correspondences of edge-points.[1] This is a similar task to estimation of the image transformation from point correspondences.

Figure 3.2a shows two points $\{\mathbf{a}^{t-1}, \mathbf{b}^{t-1}\}$ identified on the contour of an object and their tangent lines. Attempting to locate the motion of these points in the next consecutive frame is ill-defined. Figure 3.2b shows that a local search normal to the edge direction incorrectly identifies correspondences $\{\mathbf{a}^t, \mathbf{b}^t\}$ which are shifted along the contour, instead of the true correspondence $\{\mathbf{a}^{t*}, \mathbf{b}^{t*}\}$. This is due to the well known aperture problem [85], which makes it impossible to detect the correct motion for points

[1] While an edge point is defined by its location and edge orientation, lines are in theory infinite, without any "anchor point". Hence while a line can be uniquely derived from an edge point, the same line can be derived from several (collinear and with equal orientation) edge points so this inference is only one-directional.

laying on edges. Should these incorrect corresponding pairs $\{(\mathbf{a}^{t-1}, \mathbf{a}^t), (\mathbf{b}^{t-1}, \mathbf{b}^t)\}$ be used directly as point-to-point corrsepondences, the estimated motion would be incorrect. However, under the assumptions of a small shift between two consecutive frames and a local linearity of the edges, these points generate the same tangent lines as the true correspondences. By using the intersection $\mathbf{c}^t$ of the tangent lines and its motion from the intersection in the previous frame ($\mathbf{c}^{t-1}$), transformations can be calculated using edge features while overcoming the aperture problem. The intersection can be seen as a *virtual corner point*, where the two edges forming the corner are allowed to be separated.

In other words, traditional techniques use point features (*e.g.* corners) to simultaneously provide the two orthogonal constraints needed to overcome the aperture problem. An alternative explored in this work is to use pairs of distant non-parallel lines instead, each providing one constraint, to define the virtual corners. These pairs of lines are assumed to be rigidly attached in 3D, which introduces the limitation of the approach to tracking (approximately) rigid objects. Virtual corners, besides being an interesting theoretical concept, are beneficial during the computation as they simplify the transformation equations compared to using lines directly.

It should be noted that some edge-points may slide too far along the contour for the lines to provide a valid correspondence, in cases where the assumption of local linearity and small shift is violated. This is pronounced especially in cases of strongly curved edges. However, since the presented approach uses a robust two-stage estimation scheme, it is robust to such violation.

Figure 3.3 shows an overview of the Long-Term Feature-Less Object Tracker (LT-FLOTrack) algorithm [128] which is built on these principles. It consists of two modules performing different tasks. The first is a *short-term tracker*, which finds line correspondences and estimates frame-to-frame transformations (block (i)). It then *updates* knowledge about the object in each frame (block (ii)), including the positions of good edges to track and observed edge stability (*edge quality field*). The short-term tracker is formalised in Algorithm 1.

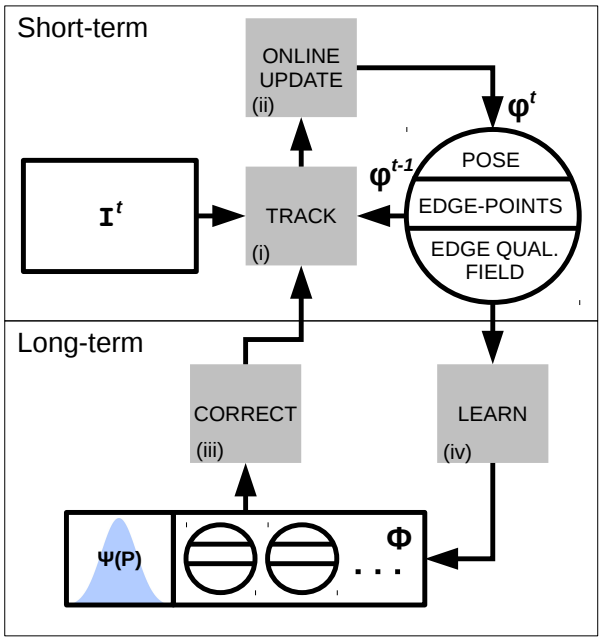


Figure 3.3: Overview of the LT-FLOTrack algorithm.

The second module maintains *long-term relations*. In cases of low confidence within the short-term tracker, a procedure to *correct* the pose is performed (block (iii)). Alternatively, if short-term tracking leads to a correct pose estimate, its state is stored (block (iv)) for future corrections. Block (iv) also includes updates to the observed pose probability distribution $\Psi(\mathbf{p})$ in every frame.

3.2 Short-term Edge-based Tracking

To find the pose $\mathbf{p}^t$, the short-term tracker estimates a (similarity) transformation S, such that $\mathbf{p}^t = \mathrm{S}(\mathbf{p}^{t-1})$. Algorithm 1 describes this short-term part of LT-FLOTrack, where T is the total number of frames to process. Lines 4 to 8 correspond to the block (i) and lines 3 and 9 refer to the block (ii) of Figure 3.3. In each frame, a set of edge-points $\mathbf{x}_i^{t-1} \in \mathcal{X}^{t-1}$ is generated (line 3). Successfully matched correspondences (inliers $\mathcal{I}^{t-1}$ to S^{t-1}) from the last frame are retained and new edge-points are generated to keep a stable number of correspondences. As this number has a significant impact

Algorithm 1 The Short-term Tracker

1: $\mathtt{Q}^1 \leftarrow$ initialise edge quality field (Sec. 3.2.4)
2: **for** $t = 2 \rightarrow T$ **do**
3: $\mathcal{X}^{t-1}, \mathcal{L}^{t-1} \leftarrow$ generate edge-points and lines $\mathtt{I}^{t-1}$ (Sec. 3.2.1&3.2.2)
4: $\mathcal{X}''^t, \mathcal{L}''^t \leftarrow$ find tentative correspondences ($\mathcal{X}^{t-1}$,$\mathtt{I}^t$) (Sec. 3.2.2)
5: $\mathrm{S}'' \leftarrow$ estimate transformation by LO-RANSAC ($\mathcal{L}^{t-1}$, $\mathcal{L}''^t$) (Sec. 3.2.3)
6: $\mathcal{X}'^t, \mathcal{L}'^t \leftarrow$ find tent. correspondences ($\mathcal{X}^{t-1}$,$\mathtt{I}^t$, init by S'') (Sec. 3.2.2&3.2.3)
7: $\mathrm{S}' \leftarrow$ estimate transformation by LO-RANSAC ($\mathcal{L}^{t-1}$, $\mathcal{L}'^t$) (Sec. 3.2.3)
8: $\mathrm{S} \leftarrow$ refine transformation (S', $\mathtt{Q}^{t-1}$) (Sec. 3.2.4)
9: $\mathtt{Q}^t \leftarrow$ update edge quality field ($\mathcal{I}^t, \mathrm{S}, \mathtt{Q}^{t-1}$) (Sec. 3.2.4)
10: **end for**

on execution times, it should be as low as possible. To estimate a suitable number of correspondences for a given sequence, a data-driven method is proposed, which accounts for the object's size and complexity; see Section 3.2.5 for details.

3.2.1 Good Edges to Track

The first step is to determine which edges are good for tracking and generate a set of new edge-points. These edges should be invariant to brightness changes and evenly distributed on the object, *i.e.* strong edges should be selected where possible and weaker edges only in regions of low contrast. LT-FLOTrack uses an iterative procedure, which searches for strong nearby edges. A point $\mathbf{q}_0^t$ is randomly drawn from inside the object bounding box (given by $\mathbf{p}^t$). The edge search is then performed along a line normal to the edge direction [77], starting from this point, to find $\mathbf{q}_1^t$. This is iterated until convergence ($\mathbf{x}_i^t = \mathbf{q}_n^t$ when $\mathbf{q}_n^t = \mathbf{q}_{n-1}^t$; the subscript n here does not mark a particular point, but rather an iteration number). Using only the gradient, this is dubbed an *unguided edge search*. The selected edge-points are defined as

$$\mathbf{q}_n^t = \mathbf{q}_{n-1}^t + \lambda \vec{\nabla} \mathtt{I}^t(\mathbf{q}_{n-1}^t)\,, \tag{3.1}$$

where the best distance λ along the line (in the direction if the image gradient $\vec{\nabla}\mathtt{I}$) is

$$\lambda = \arg\max_{\underline{\lambda}} ||\vec{\nabla}\mathtt{I}^t \left(\mathbf{q}_{n-1}^t + \underline{\lambda}\vec{\nabla}\mathtt{I}^t(\mathbf{q}_{n-1}^t)\right)|| \cdot \exp(-\underline{\lambda}^2/\sigma^2) \tag{3.2}$$

and σ is a scaling factor (proportional to the object size). Notice that since edge-point (and consequently line) correspondences between the previous and the current frame are sought: ($\mathcal{X}^{t-1}$, $\mathcal{X}'^t$) and ($\mathcal{L}^{t-1}$, $\mathcal{L}'^t$), respectively, the points $\mathcal{X}^{t-1} = \left\{\mathbf{x}_i^{t-1}\right\}$ used for tracking into $\mathtt{I}^t$ have been previously localised in $\mathtt{I}^{t-1}$.

3.2.2 Frame-to-Frame Correspondences

When establishing correspondences (line 4), another edge search is required, however the task is different. Instead of locally strong edges, edges similar to those from the previous frame are sought. As such, a *guided* (using information from the previous frame) *edge search* is used. This searches for positions with similar gradient angle and local appearance. The search starts at the locations of edge-points $\mathbf{x}_i'^t \in \mathcal{X}'^t$ from the previous frame:

$$\begin{aligned}\mathbf{x}_i''^t = \arg\max_{\underline{\mathbf{q}} \in \mathbf{x}_i^{t-1} + \lambda\vec{\nabla}\mathtt{I}^{t-1}(\mathbf{x}_i^{t-1})} \left(\frac{\cos(\Delta\alpha)+1}{2}\right) \cdot \delta(\mathbf{x}_i^{t-1}, \underline{\mathbf{q}}) \cdot \Lambda_{\mathbf{x}}(\lambda) \\ \text{s.t.} \quad \underline{\mathbf{q}} \text{ is a local maximum of gradient},\end{aligned} \tag{3.3}$$

where $\delta(\mathbf{x}_i^{t-1}, \underline{\mathbf{q}})$ measures similarity of local appearance of $\mathtt{I}^{t-1}$ around $\mathbf{x}_i^{t-1}$ and $\mathtt{I}^t$ around $\underline{\mathbf{q}}$, $\Delta\alpha$ is the difference between gradient angles at $\mathtt{I}^{t-1}(\mathbf{x}_i^{t-1})$ and $\mathtt{I}^t(\underline{\mathbf{q}})$ and $\Lambda_{\mathbf{x}}$ is a locality-preserving regularisation. This is defined as constant in the range of σ (as used above) and then linearly falling to zero with distance. Cosine is used as a measurement of angle error due to its tolerance (the flat region around zero), robustness (no outlier over-penalisation) and absence of problems with angle periodicity. It is scaled and shifted to the $[0;1]$ range. For the guided edge search, multiple search lines are used: normal to the edge in the last frame ($\vec{\nabla}\mathtt{I}^{t-1}(\mathbf{x}_i^{t-1})$) and offset by $\pm\frac{\pi}{10}$.

The tentative edge-point correspondences $(\mathbf{x}_i^{t-1}, \mathbf{x}_i''^t)$ are then transformed to line-to-line correspondences $(\mathbf{l}_i^{t-1}, \mathbf{l}_i''^t)$ using the gradient direction:

$$\mathbf{l}_i^{t-1} = \left(\mathbf{n}^\top, -\mathbf{n}^\top \mathbf{x}_i^{t-1}\right)^\top \tag{3.4}$$

(homogeneous), where $\mathbf{n}$ is normal to the gradient:

$$\mathbf{n} = \mathtt{J}\vec{\nabla}\mathtt{I}^{t-1}(\mathbf{x}_i^{t-1})\,, \tag{3.5}$$

using $\mathtt{J}$, a matrix of 2D counter-clockwise rotation,

$$\mathtt{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{3.6}$$

The same process is performed for edge-points $\mathbf{x}_i''^t$ to get lines $\mathbf{l}_i''^t$ based on $\vec{\nabla}\mathtt{I}^t$.

The set of inliers $\mathcal{I}^t(\mathrm{S})$ is defined as a subset of correspondences $(\mathbf{l}_i^{t-1}, \mathbf{l}_i'^t)$ having low geometric error d_G (see Section 3.2.6 for details) with respect to the estimated transformation S:

$$\mathcal{I}^t(\mathrm{S}) = \left\{ (\mathbf{l}_i^{t-1}, \mathbf{l}_i'^t) \,\middle|\, \sqrt{d_\mathrm{G}\left(\mathbf{l}_i'^t, \mathrm{S}(\mathbf{l}_i^{t-1})\right)^2 + d_\mathrm{G}\left(\mathbf{l}_i^{t-1}, \mathrm{S}^{-1}(\mathbf{l}_i'^t)\right)^2} \leq \theta_d \right\}. \tag{3.7}$$

This can be equivalently seen as a set of point-to-point correspondences $(\mathbf{x}_i^{t-1}, \mathbf{x}_i'^t)$, since lines are defined by edge-points (as used in Equation (3.12)). The subset of edge-point inliers $\mathcal{X}'^t$ (*i.e.* defining lines belonging to the inlier set) is retained for use in the next frame as $\mathcal{X}^t$. Additional edge-points are then generated as described above (line 3 of Algorithm 1), which encloses the short-term tracking loop. The relationship between the points $\mathcal{X}^{t-1}$, $\mathcal{X}'^t$ and $\mathcal{I}^t$ is visualised in Figure 3.4.

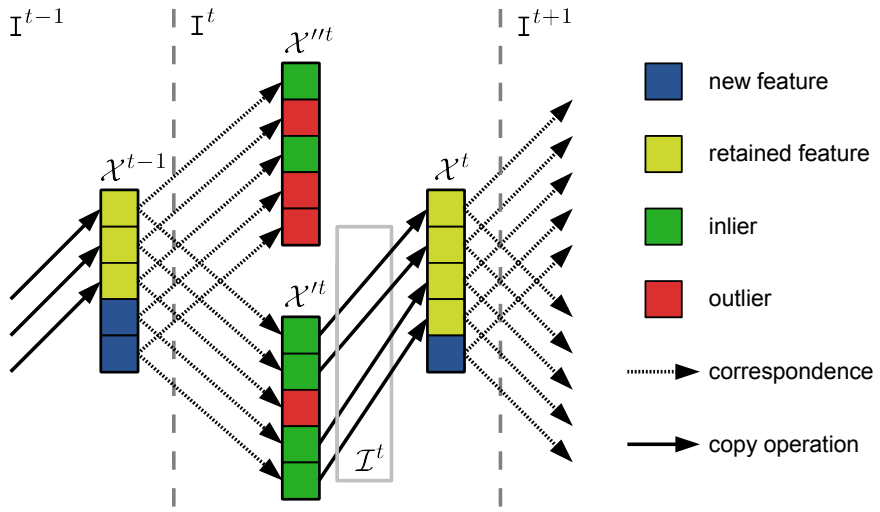


Figure 3.4: Relationships and transitions between features.

3.2.3 Frame-to-Frame Transformation

The Locally Optimised RANSAC (LO-RANSAC) [130] uses the correspondences $(\mathbf{l}_i^{t-1}, \mathbf{l}_i''^{t})$ to estimate an initial geometric transformation S'' between the two frames, maximising *image evidence* E^t (Algorithm 1, line 5). The similarity transformation is used within this chapter. This is defined as the average fit $e_{\mathbf{x}_i^{t-1}}$ of edge-points from $\mathtt{I}^{t-1}$ to the edges in $\mathtt{I}^{t}$. The computation of image evidence is based on an *oriented Chamfer distance* [151, 171] as

$$E^t(\mathrm{S}'') = \Lambda_E \cdot \frac{1}{|\mathcal{X}^{t-1}|} \sum_{i=1}^{|\mathcal{X}^{t-1}|} e_{\mathbf{x}_i^{t-1}} \,, \tag{3.8}$$

$$e_{\mathbf{x}_i^{t-1}} = \frac{1}{1 + \mathrm{d}(\mathrm{S}''(\mathbf{x}_i^{t-1}))} \cdot \frac{\cos(\Delta\alpha) + 1}{2} \,, \tag{3.9}$$

where $\mathrm{d}(\cdot)$ is Euclidean distance of a point to the nearest Canny's edge [23], $\Delta\alpha$ is the difference between the gradient angle of a point and its nearest edge (taking rotation induced by S'' into account) and Λ_E is a regularisation term penalising large changes between $\mathbf{p}^{t-1}$ and $\mathbf{p}^t$ (smoothness enforcing prior). The minimal sample for Random


Figure 3.5: Examples of an image and its edge quality field after several frames of tracking.

Sample Consensus (RANSAC) is a triplet of lines, whose intersections (the virtual corner points) are used to generate a transformation hypothesis. While two points would suffice for the similarity transformation, two lines do not provide sufficient constraints and thus a triplet is necessary. These lines are sampled in a way to ensure sufficient angles between any two of them. This provides good location of virtual corners and thus numerical conditioning. In this "triangle sample", the vertices (the virtual corners) can be seen as dual to the edges (lines).

To get a more precise transformation with a higher number of inliers, the process of correspondence search and transformation estimation is repeated using the transformation estimate S'' as initialisation (lines 6 and 7). In this second iteration, the new locations of correspondences $\mathcal{X}'^t = \{\mathbf{x}_i'^t\}$ and $\mathcal{L}'^t = \{\mathbf{l}_i'^t\}$ are computed. LO-RANSAC is then executed again using the new correspondences to obtain S'.

3.2.4 Beyond Frame-to-Frame Tracking

The estimated transformation S' is usually more accurate than S'', however it may still be noisy, which would ultimately result in tracker drift. It is therefore necessary to stabilise the estimation relative to previous frames. In LT-FLO this is done by learning the locations of edges, which have previously predicted a correct transformation. This knowledge is stored as an *edge quality field* $\mathtt{Q}^t$, giving an estimate of object structure (stable edges, see Figure 3.5 for an example). The corresponding edges in the new frame are expected to fit to this model of previously stable edges w.r.t. the estimated

transformation (line 8 of Algorithm 1):

$$\mathrm{S} = \arg\max_{\underline{\mathrm{S}}} Q(\underline{\mathrm{S}}) \tag{3.10}$$

with

$$Q(\mathrm{S}) = \sum_{i=1}^{|\mathcal{X}'^t|} \mathtt{Q}^{t-1}(\mathrm{S}^{-1}(\mathbf{x}_i'^t))\,; \tag{3.11}$$

maximised using Nelder-Mead iterative optimisation of the transformation parameters. In every frame, $\mathtt{Q}^t$ is updated as follows:

$$\mathtt{Q}^t = \omega \cdot \mathrm{S}(\mathtt{Q}^{t-1}) + \bigcup_{(\mathbf{x}_i^{t-1}, \mathbf{x}_i'^t) \in \mathcal{I}^t} e_{\mathbf{x}_i^{t-1}}\,, \tag{3.12}$$

where ω is a forgetting factor (line 9 of Algorithm 1). The field $\mathtt{Q}^1$ is initialised taking all the edge-points from the first frame as reliable (line 1).

To allow reference to the short-term tracker in a compact form, we adopt the following notation. The complete *state* of the tracker (or model of the object) will be referred to as $\boldsymbol{\varphi}^t$ and includes information about the object pose $\mathbf{p}^t$, edge-points $\mathcal{X}^t$ and the *edge quality field* $\mathtt{Q}^t$. The short-term tracker can then be seen as a series of consecutive calls to a tracking function, performing a local search started at $\mathbf{p}^{t-1}$:

$$\mathbf{p}^t = \tau(\mathbf{p}^{t-1} | \boldsymbol{\varphi}^{t-1}, \mathtt{I}^t)\,. \tag{3.13}$$

Similarly the update of the current state comprises of updating $\mathtt{Q}^{t-1}$ to $\mathtt{Q}^t$ according to Equation (3.12), estimating $\mathbf{p}^t = \mathrm{S}(\mathbf{p}^{t-1})$ and generating new points $\mathbf{x}_i^t$, which can be concisely summarised as

$$\boldsymbol{\varphi}^t = \mathrm{update}(\boldsymbol{\varphi}^{t-1})\,. \tag{3.14}$$

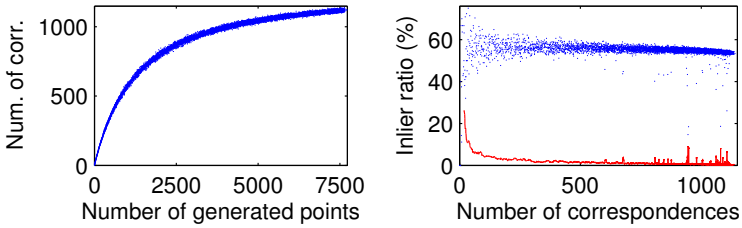


Figure 3.6: Dependences of the number of correspondences and inliers on the number of generated points. The red line in the right image shows the standard deviation in bins of size 20. In this particular case (beginning of the DUDEK sequence), one can expect stable behaviour with about 350 correspondences, thus it is enough to generate 500 new points.

3.2.5 On the Number of Generated Edge-points

To estimate a sufficient number of correspondences for successful tracking of a given sequence, an approach that accounts for the object's size and complexity is employed. The number of final correspondences after the edge search is usually lower than the number of generated edge-points. Furthermore, this dependency is strongly non-linear and saturates (see Figure 3.6).

In LT-FLOTrack, this saturation level is found in the first frame by initialising with a high number of random points. Then the number of generated points is set proportional to the saturation level. This is adjusted according to the observed scale changes in subsequent frames.

3.2.6 Geometric Error of Line Correspondences

Previously, we have worked with terms such as "inliers", or "consistent line correspondence", which require a measure of distance between projected and measured line features (an equivalent to the projection error for point correspondences). But what does it mean for two lines in an image $\mathbf{l}$ and $\mathbf{l}'$ to be close to each other? Hartley [79] stated that distance (or geometric error) of lines has to be measured with respect to some point of interest. He suggested to use the distance between a line and line seg-

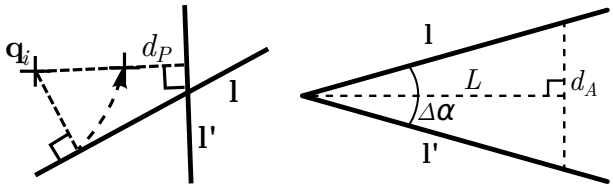


Figure 3.7: Geometric meaning of d_P and d_A.

ment, as in general there is rarely any interest in infinite lines. This approach yields usable results. However, lines in LT-FLOTrack are not defined by segments, and are infinite in extent, since their real endpoints are unknown, and possibly invisible due to (self-)occlusion. It is therefore necessary to calculate intersections between the lines and all four sides of the tracked object bounding box. Computational complexity is then prohibitively large. Furthermore, this approach introduces a bias, giving lower errors to lines near borders of the tracked area, as their intersections with the borders lie closer to each other.

A more feasible approach is to see the distance as two independent components – difference of angles d_A and difference of position d_P with respect to a given *point of interest* $\mathbf{q}_i$ (e.g. centre of gravity of the tracked object), as can be seen in Figure 3.7. The error of position is defined as the difference between the distances from the lines to $\mathbf{q}_i$, in normalised homogeneous coordinates as:

$$d_\mathrm{P} = \left|\tilde{\mathbf{q}}_i^\top \tilde{\mathbf{l}}\right| - \left|\tilde{\mathbf{q}}_i^\top \tilde{\mathbf{l}}'\right| . \tag{3.15}$$

The angular error is defined as the divergence of the lines at some specified distance from their intersection (in 2D). This distance is a constant L, which can be derived from the size of the tracked object, or set manually.

$$d_\mathrm{A} = 2 \cdot L \cdot \tan\frac{\Delta\alpha}{2} , \tag{3.16}$$

where $\Delta\alpha$ is the angle between the lines. The choice of L can be used to control the relative strength of the two components (d_P and d_A). Selection around half the size of

the tracked bounding box will ensure approximately equal weights. The value of 50 px was used in all experiments throughout this chapter. Finally, the geometric error term is computed as

$$d_{\mathrm{G}} = \sqrt{d_{\mathrm{P}}^2 + d_{\mathrm{A}}^2}\,. \tag{3.17}$$

This technique gives errors similar to Hartley's approach in significantly lower time (10-fold speed-up with correlation coefficient 0.9). It should be noted that d_{P} is strongly underestimated in the case of $\mathbf{q}_i$ laying *between* the lines. However, as we are usually concerned with the distance of lines that are close to each other, this condition appears rarely (the correspondences are incorrectly classified as inliers less than one percent of the time).

3.3 Drift and Long-term Relationships

Using line-correspondences for short-term tracking works well for short sequences. However, for longer sequences it suffers from error accumulation (drift) and is not robust to severe occlusions.

The long-term module of Long-Term Feature-Less Object Tracker (LT-FLO) continuously checks the *image evidence* score E^t, as this is a good indicator of the quality of the estimated transformation. When E^t decreases suddenly, this indicates a problem (the confidence in the current solution is low). On such an occasion, the short-term tracker may experience difficulties and may need *correction* (block (iii) of Figure 3.3). The desired property of the (local) correction is that it can, given the last known pose of the tracked object, estimate its new pose regardless of any drift in the short-term tracker. Furthermore, it should identify a disappearance of the object (either because of a tracker failure or a full occlusion) and start a global *redetection*.

A correction procedure is proposed, which fulfils these requirements and works both on a local level and in a redetection scenario. The long-term module has several *states* of the short-term tracker stored – a set Φ (initialised as $\Phi = \{\varphi^1\}$). When

a tracking failure is detected, the short-term tracker is initialised using each of the stored states $\varphi^u \in \Phi$ at the last known valid pose $\mathbf{p}^{t-1}$. A process analogous to the standard short-term tracking function τ is performed, yielding several *correcting* hypotheses in addition to the *current* one (using φ^{t-1}). Each hypothesis is assigned a score Γ, based on transformation quality, temporal consistency and the inlier ratio (Equations (3.8, 3.11&3.7)):

$$\Gamma\left(\tau(\mathbf{p}|\varphi, \mathtt{I})\right) = E(\mathrm{S}_\varphi) \cdot Q(\mathrm{S}_\varphi) \cdot \sqrt{\frac{|\mathcal{I}(\mathrm{S}_\varphi)|}{|\mathcal{X}'|}}\,, \tag{3.18}$$

where S_φ is the estimate of the transformation given by tracking from a state φ. The best correcting hypothesis is selected as:

$$\varphi^* = \arg\max_{\varphi \in \Phi} \Gamma\left(\tau(\mathbf{p}^{t-1}|\varphi, \mathtt{I}^t)\right)\,, \tag{3.19}$$

where all the corrections (local search) are started at the position $\mathbf{p}^{t-1}$. It is then used for *correction* where appropriate, replacing the current estimate in the short-term tracker.

If the best *correcting* hypothesis estimated is an object pose similar to the *current* one, $||\mathbf{p}(\varphi^*) - \mathbf{p}(\varphi^{t-1})|| < \theta_2$, it is a signal that the original estimated pose was correct and the current state is not replaced as it is expected to be better adapted to the current object appearance. In the extreme case, when the agreement of the estimated pose is (almost, up to θ_1) exact, the current state is stored for use in future corrections (block (iv) of Figure 3.3).

See Figure 3.8 for a schema of these situations. The estimated pose $\mathbf{p}$ and score Γ of φ^* are compared with the ones obtained before the correction (using φ^{t-1}). One of the following four cases can happen. 1) If the tracking so far is considered good enough (up to θ_2), the current tracking is not corrected. 2) The same happens if the current estimate is better (has higher score Γ) than the best correction. 3) If the pose from tracking so far is even closer to the best correction (up to θ_1), the current tracking is

not corrected and additionally its state φ^{t-1} is learned. 4) Finally, if the current pose is considered worse than the best correction (lower score Γ and significantly different pose $\mathbf{p}$), the correction is applied.

The correction procedure can be formalised as:

$$\begin{aligned} \mathrm{S} &= \begin{cases} \mathrm{S}_{\varphi^*} & \text{if } \Gamma(\varphi^*) > \Gamma(\varphi^{t-1}) \bigwedge ||\mathbf{p}(\varphi^*) - \mathbf{p}(\varphi^{t-1})|| \geq \theta_2 \\ \mathrm{S}_{\varphi^{t-1}} & \text{otherwise}\,, \end{cases} \\ \varphi^t &= \begin{cases} \mathrm{update}(\varphi^*) & \text{if } \Gamma(\varphi^*) > \Gamma(\varphi^{t-1}) \bigwedge ||\mathbf{p}(\varphi^*) - \mathbf{p}(\varphi^{t-1})|| \geq \theta_2 \\ \mathrm{update}(\varphi^{t-1}) & \text{otherwise}\,, \end{cases} \end{aligned} \tag{3.20}$$

where $\mathbf{p}(\varphi^*)$ is a shorthand for $\tau(\mathbf{p}^{t-1}|\varphi^*, \mathtt{I}^t)$ and $\Gamma(\varphi^*)$ for $\Gamma\left(\tau(\mathbf{p}^{t-1}|\varphi^*, \mathtt{I}^t)\right)$ (and analogously for φ^{t-1}; these are also used for conciseness in the text). The thresholds θ_1 and θ_2 are calculated from the object size. The learning can be formalised as:

$$\Phi_{\text{new}} = \begin{cases} \Phi \cup \varphi^{t-1} & \text{if } ||\mathbf{p}(\varphi^*) - \mathbf{p}(\varphi^{t-1})|| < \theta_1 \\ \Phi & \text{otherwise}\,. \end{cases} \tag{3.21}$$

The corrections may consume a significant portion of the execution time. As the asymptotic time complexity is $O(T \cdot |\Phi|)$, it is not feasible to keep all observed states. Therefore a method is needed to maximise the diversity of the learned states to cover as much variation in object appearance as possible in a fixed space and time. A limit is therefore placed on the cardinality of Φ (in experiments within this chapter, $|\Phi|_{\max} = 5$), and when it is reached, the state used least often in recent corrections is replaced, as this is a good indicator of state's usefulness in the future:

$$\Phi_{\text{new}} = \Phi \setminus \arg\min_{\underline{\varphi} \in \Phi} \Upsilon(\underline{\varphi})\,, \tag{3.22}$$

where Υ is the number of occasions in the past when $\underline{\varphi}$ was selected according to Equation (3.19). This pruning is carried out before expanding the set Φ in the former option of Equation (3.21).

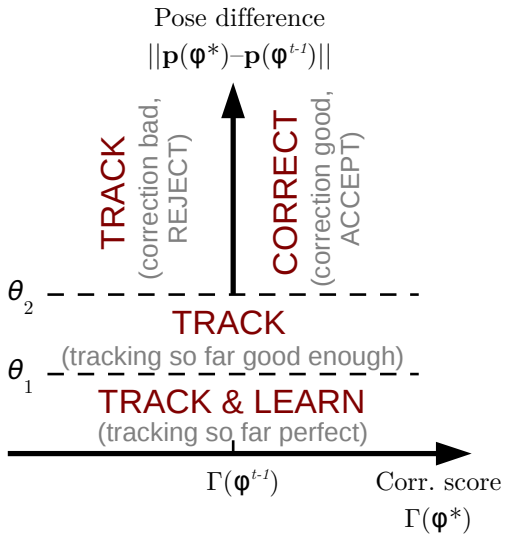


Figure 3.8: Possible situations during correction. Thresholds θ_1 and θ_2 for the decisions are calculated from the object size.

3.4 Tracking Failure and Redetection

In a significant portion of real-world scenarios, the object of interest undergoes strong (sometimes full) occlusion by either other objects in the frame, or the video frame boundaries. It is therefore necessary for a long-term tracker, to handle such occlusions and to redetect the object afterwards. We call these *global redetections*, or corrections, as opposed to the previous section's corrections which are local to $\mathbf{p}^{t-1}$.

3.4.1 Disappearance or Failure Discovery

The first step of the redetection procedure is to identify object disappearances and/or tracking failures. Since LT-FLOTrack is edge-based, loss of tracking is closely related to the observed gradient. By definition, the edge-points are extracted in areas of strong, reliable gradient, therefore if this is not true after tracking, it indicates a problem.

More specifically, the proposed approach to failure discovery is based on the following observation. In the correction stage of LT-FLO (block (iii) of Figure 3.3), the points

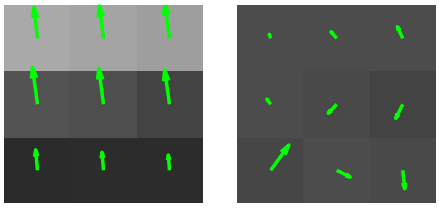


Figure 3.9: Agreement of gradient direction in a pixel with its neighbourhood. Left image: region of strong, stable gradient, right image: noisy gradient with low magnitude.

$\mathcal{X}^{\underline{t}}$ from a stored state $\varphi^{\underline{t}}$ (from frame $\mathtt{I}^{\underline{t}}$, as used in Equation (3.19)) are applied to the previous frame $\mathtt{I}^{t-1}$, to search for correspondences $\mathcal{X}'^{t}$. Since this is a local search, the projected edge-points ($\mathbf{q} = \mathrm{S}^{\underline{t}\to t-1}(\mathbf{x}_i^{\underline{t}})$) are expected to lie near edges (*i.e.* areas of strong gradient) in the current frame $\mathtt{I}^{t}$. When they are moved to an area of weak or noisy gradient (see Figure 3.9), the direction of the gradient at a particular pixel is not in accordance with its neighbourhood:

$$\text{accordance}(\mathbf{q}) = \begin{cases} \text{true} & \left|\text{atan}(\mathtt{I}_x(\mathbf{q}), \mathtt{I}_y(\mathbf{q})) - \text{atan}(\bar{\mathtt{I}}_x(\mathbf{q}), \bar{\mathtt{I}}_y(\mathbf{q}))\right| < 90° \\ \text{false} & \text{otherwise}\,, \end{cases} \tag{3.23}$$

where $\mathtt{I}_x, \mathtt{I}_y$ are components of the image gradient $\vec{\nabla}\mathtt{I}$ (i.e. $\mathtt{I}_x = \frac{\partial \mathtt{I}}{\partial x}, \mathtt{I}_y = \frac{\partial \mathtt{I}}{\partial y}$) and $\bar{\mathtt{I}}$ indicates the average over the 3×3 neighbourhood of $\mathbf{q}$. The number of points *not* in accordance $\left|\,\{\mathbf{q}|\text{accordance}(\mathbf{q}) = \text{false}\}\,\right|$ is recorded. A failure is identified, when there is an unusually high number of such points across all the corrections. The threshold is set using the 99-th percentile of the fitted normal distribution for that sequence, such that sequences with different contrast levels are handled appropriately.

Object disappearance and tracking failures are also detected, based on the geometric properties of the object pose. Specifically, this happens when the tracked rectangle is too small, as sizes under 10 px render the similarity function δ (as used in Equation (3.3) in Section 3.2.2) unreliable, objects are thus considered too distant to track. Furthermore, it happens when the target bounding box is larger than the image or when a large portion of it is outside the image (more than three quarters of the object

is out of the scene). Finally, a failure is indicated when there are no inliers to the estimated transformation S.

3.4.2 Object Redetection

Once the tracker detects that the object is lost, a redetection occurs. Firstly, a local correction is performed (Section 3.3, Equation (3.19)). This is followed by a global redetection, employing the stored states Φ. Instead of initialisation by $\mathbf{p}^{t-1}$, the global redetection is initialised at several random poses $\mathbf{p}_{\text{rand}}$ (discussed in the next section):

$$\varphi^* = \underset{\varphi \in (\Phi \cup \varphi^{t-1})}{\arg\max} \; \Gamma\left(\tau(\mathbf{p}_{\text{rand}} | \varphi, \mathcal{I}^t)\right) . \tag{3.24}$$

The best hypothesis is used for correction in the same way as φ^* in Equation (3.20). After a failure, this global redetection is performed in every frame, until a good pose is found.

3.4.3 Object Pose Distribution

In the context of LT-FLOTrack, a sliding-window redetection approach would mean restarting tracking (see below) many times, in numerous different positions, rotations and scales. This would be excessively computationally demanding. For many sequences, however, the object-and-camera system does not use the entire parameter space of object poses (induced by rigid transformations). Instead, only a significantly smaller subspace is occupied. An example would be the appearance of a car, followed from the rear. In such a scenario, there is almost no rotation and scale is correlated with the y-coordinate (related to the actual distance between the cars), see Figure 3.10.

This property of the sequences can be exploited to obtain prior information about the object pose $\mathbf{p}^t$. The distribution Ψ of the object pose is modelled as a multivariate Gaussian distribution ($\Psi = \mathcal{N}(\boldsymbol{\mu}_\Psi, \Sigma_\Psi)$, log-normal for the scale component). This models the distribution well despite the fact that the true distribution is often not

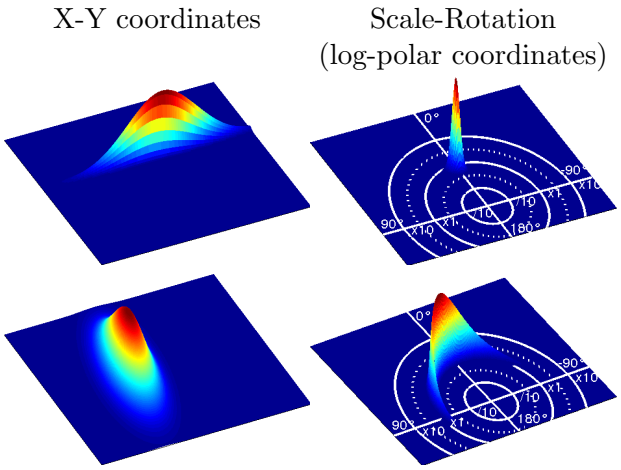


Figure 3.10: Examples of probability distribution Ψ (scale relative to the initial size). Top row: tracking a car, change only in the x-coordinate (visible steps in the upper left image indicate how narrow Ψ is). Bottom row: tracking the PAGE sequence with large variation in rotation.

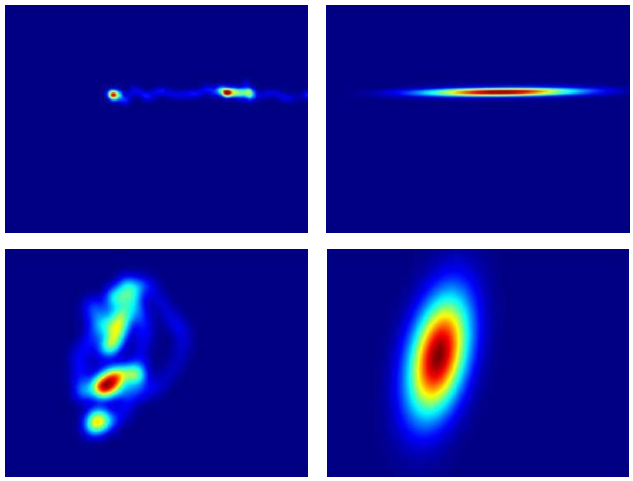


Figure 3.11: Examples of probability distribution Ψ, over x- and y-coordinates (for the same sequences as in Figure 3.10). Although the distributions on the left (estimated by KDE) are not unimodal, they are well modelled by the fitted Gaussian distribution (right column).

unimodal. See Figure 3.11 for an illustration of this phenomenon. For instance, for the PAGE sequence, the Kernel Density Estimation (KDE) and Gaussian distributions have a Bhattacharyya coefficient (approximate overlap measure) of 0.92 and KL-divergence of 1.1 bits (both distributions have an entropy of around 15 bits).

The distribution's parameters are learned online from observed object poses, according to [16]. This has negligible computational cost; in contrast, the complexity of KDE would be quadratic in the number of frames. In the t-th frame, the update is calculated as:

$$\boldsymbol{\mu}_\Psi^t = \boldsymbol{\mu}_\Psi^{t-1} + \frac{\mathbf{p}^t - \boldsymbol{\mu}_\Psi^{t-1}}{t} \tag{3.25}$$

and

$$\Sigma_\Psi^t = \left(\Sigma_\Psi^{t-1} + \frac{\left(\mathbf{p}^t - \boldsymbol{\mu}_\Psi^{t-1}\right)\left(\mathbf{p}^t - \boldsymbol{\mu}_\Psi^{t-1}\right)^\top}{t} \right) \frac{t-1}{t} \,. \tag{3.26}$$

Given this probability distribution Ψ, one can make assumptions about the object's pose. For example, false "corrections" where the resulting pose has extremely low probability can be rejected. This adds the following constraint to the optimisation in Equation (3.19):

$$\text{s. t.} \quad \Psi(\tau(\mathbf{p}^{t-1}|\underline{\varphi}, \mathfrak{I}^t)) > \theta_\Psi \,. \tag{3.27}$$

An example of this constraint's value is a rectangular object, which looks very similar when rotated. However, a low probability for this pose may help to identify such a situation as incorrect.

Finally, the pose distribution is used to guide the search during redetection (Equation (3.24), $\mathbf{p}_{\text{rand}} \sim \Psi$). This is more efficient than dense sampling of the whole parameter space. To improve the ability to generalise to possible, but previously unseen or rare poses, the search space is extended by multiplying the covariance matrix Σ_Ψ by a constant factor during subsequent redetection.

3.5 Experimental Evaluation of LT-FLOTrack

This section provides experimental evaluation of edge-based tracking and the proposed redetection scheme. Firstly LT-FLOTrack is evaluated on several recent benchmarks (VOT 2013 and 2014, and Visual Tracker Benchmark (VTB) 1.0). Then the behaviour is evaluated in detail and compared to several state-of-the-art trackers on

	grayscale			region_noise		
Name	**Acc.**	**Rob.**	**Speed**	**Acc.**	**Rob.**	**Speed**
Bicycle	0.58	1.73	4.52	0.57	1.60	4.10
Bolt	0.48	5.27	3.16	0.47	4.87	2.99
Car	0.44	2.00	2.37	0.44	1.33	3.43
Cup	0.87	0.00	8.65	0.79	0.00	9.09
David	0.64	0.07	6.54	0.63	0.33	5.70
Diving	0.38	1.53	3.06	0.37	1.67	2.72
Face	0.82	0.00	5.46	0.72	0.07	5.32
Gymnastics	0.53	1.33	2.95	0.50	1.20	2.45
Hand	0.45	4.93	3.89	0.45	4.67	4.38
Iceskater	0.39	2.47	2.51	0.43	1.80	2.26
Juice	0.89	0.00	6.74	0.78	0.07	6.87
Jump	0.52	0.27	3.55	0.52	0.20	3.61
Singer	0.60	0.53	0.67	0.60	0.20	0.76
Sunshade	0.68	1.40	5.33	0.63	1.47	5.73
Torus	0.58	2.27	4.19	0.55	1.87	4.62
Woman	0.51	6.33	3.10	0.49	5.53	3.18

Table 3.1: Results of LT-FLOTrack in the VOT2013 benchmark. The tabulated values are accuracy, robustness and speed in FPS.

selected sequences in both short-term and long-term scenarios. These contain examples of completely texture-less and even transparent objects. Finally, the contribution of different components of the algorithm is explored.

3.5.1 VOT Challenge 2013 Results

In this section, the performance of the proposed tracker on the standard VOT2013 dataset is reported, in the VOT benchmark scenario [115]. The results are tabulated in Table 3.1, using *accuracy*, *robustness* and *speed* in FPS. Accuracy is defined as the average overlap between the tracked and ground-truth bounding boxes and robustness as the number of tracker failures per sequence, when the overlap dropped to zero. Results are averaged over 15 runs. The `region_noise` experiment is carried out with perturbed initialisation. Only the results of experiments 2 and 3 are reported, since LT-FLOTrack performs equally well on both coloured and greyscale sequences (due to the use of gradient information). The performance of LT-FLOTrack is the best on

Tracker	Acc. Rank	Rob. Rank	Combined Rank
PLT	5.9	3.0	4.5
FoT	4.3	10.4	7.3
EDFT	7.9	11.8	9.8
LGT++	14.8	5.2	10.1
LT-FLO	**7.3**	**14.8**	**11.0**
GSDT	10.7	11.4	11.1
SCTT	6.1	16.5	11.3
CCMS	10.0	12.7	11.4
LGT	17.3	5.9	11.6
Matrioska	10.1	13.3	11.7
AIF	7.7	16.2	12.0
Struck	11.5	12.5	12.0
DFT	10.8	13.7	12.2
IVT	10.4	14.8	12.6
ORIA	12.2	15.1	13.7
PJS-S	12.6	15.3	13.9
TLD	9.5	20.2	14.8
MIL	17.9	12.8	15.4
RDET	19.9	11.1	15.5
HT	20.0	12.9	16.5
CT	21.1	13.2	17.1
Meanshift	20.5	15.7	18.1
SwATrack	18.9	19.6	19.3
STMT	22.0	19.3	20.6
CACTuS-FL	24.2	17.8	21.0
ASAM	21.7	23.0	22.4
MORP	25.1	26.7	25.9

Table 3.2: Overall results of the VOT Challenge 2013 [115]. The trackers are sorted according to combined performance.

sequences where a rigid object is tracked (Cap, Juice). On the other hand, sequences with highly non-rigid objects (Diving, Hand) score lower, as well as cases of strong out-of-plane rotation (Car).

Table 3.2 brings an overall comparison of all trackers competing in the VOT Challenge 2013. In this highly competitive challenge, LT-FLOTrack performed favourably compared to other state-of-the-art trackers and finished fifth place in the competition – out of 27. See Figure 3.12 for a visual comparison of trackers' performance in the challenge. Rankings in both accuracy and robustness are plotted, the higher and fur-

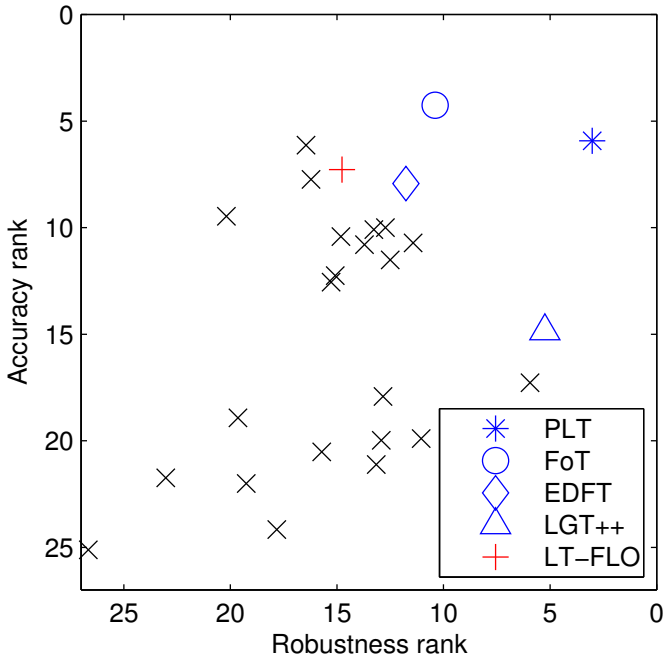


Figure 3.12: Accuracy-robustness plot of the VOT Challenge 2013 results. For each tracker, the accuracy and robustness rankings were computed as a mean over all three experiments. Legend shown only for the best five trackers, see [115] for a full report.

ther right a tracker is, the better its rank. LT-FLOTrack, while having only average robustness (mainly due to its difficulties with articulated objects), excels in the accuracy ranking, where its drift-resistant nature proves invaluable. However, it should be noted that the VOT Challenge does not test properties where LT-FLO is the strongest, in particular low texture and long sequences (possibly with full occlusions), but it does provide evidence that it remains extremely effective on these simpler sequences.

3.5.2 VOT Challenge 2014 Results

The same experiments were carried out on the VOT2014 [116] dataset – see Table 3.3 for results. Among the best scoring are, unsurprisingly, rigid-object sequences such as Sphere or Car (not to be confused with the VOT2013 sequence of the same name). Although the tracked object in the Surfing sequence is a person, LT-FLOTrack per-

	baseline			region_noise		
Name	**Acc.**	**Rob.**	**Speed**	**Acc.**	**Rob.**	**Speed**
Ball	0.38	4.07	5.01	0.38	4.27	5.09
Basketball	0.52	4.60	2.04	0.48	4.80	2.12
Bicycle	0.58	1.67	5.05	0.57	1.60	4.41
Bolt	0.50	4.40	3.20	0.46	5.07	3.23
Car	0.75	0.87	4.27	0.68	0.87	4.62
David	0.68	0.00	7.60	0.62	0.20	6.51
Diving	0.24	3.27	2.68	0.28	3.53	3.02
Drunk	0.59	0.93	1.77	0.45	0.73	2.07
Fernando	0.32	1.27	1.49	0.32	1.40	1.21
Fish1	0.36	6.80	4.85	0.33	7.20	4.37
Fish2	0.27	6.27	2.61	0.23	6.13	2.73
Gymnastics	0.56	1.07	3.13	0.51	2.20	2.57
Hand1	0.48	3.80	3.70	0.47	4.67	4.77
Hand2	0.40	9.00	3.63	0.35	9.13	4.21
Jogging	0.68	1.13	5.46	0.63	1.07	5.57
Motocross	0.61	1.67	2.43	0.55	1.47	2.56
Polarbear	0.63	0.00	5.19	0.54	0.07	4.77
Skating	0.40	1.73	4.17	0.44	1.60	4.26
Sphere	0.81	0.00	4.48	0.76	0.00	4.50
Sunshade	0.70	1.33	5.28	0.69	1.20	5.45
Surfing	0.80	0.07	9.54	0.72	0.07	9.13
Torus	0.58	1.73	4.78	0.58	2.13	4.12
Trellis	0.62	2.13	5.58	0.60	1.67	5.93
Tunnel	0.60	0.60	2.68	0.55	0.67	2.05
Woman	0.55	5.53	3.57	0.50	5.40	3.72

Table 3.3: Results of LT-FLOTrack in the VOT2014 benchmark. The tabulated values are accuracy, robustness and speed in FPS.

forms well, since the person does not move his limbs such that body articulation does not create a problem. On the other side of the spectrum is highly articulated Diving (as in VOT2013), and Ball, which, while rigid, contains strong out-of-plane rotation.

In Table 3.4 and Figure 3.13 the performance of LT-FLO with other trackers competing in the VOT2014 challenge is compared. As can be seen from the accuracy-robustness plot, the proposed tracker again performs well in terms of accuracy, being placed near the top of the vertical axis. However, the weaker performance in robustness prevents it from outperforming the main body of the competing trackers, placing it in

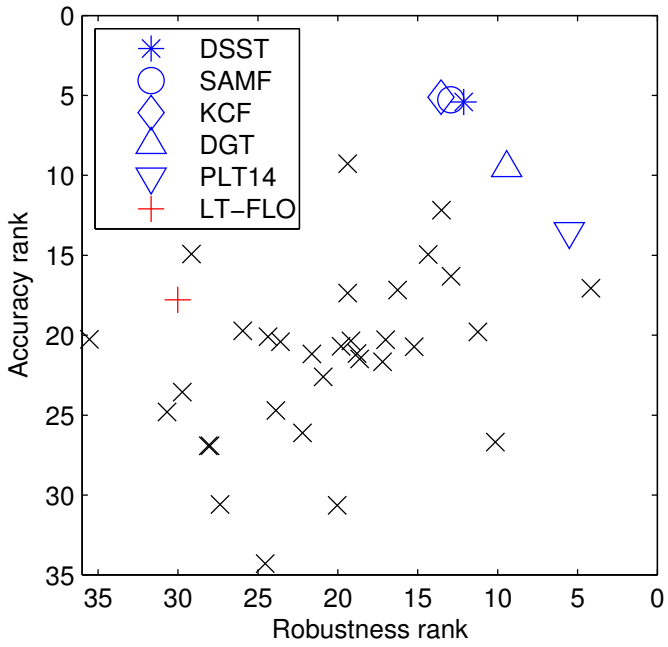


Figure 3.13: Accuracy-robustness plot of the VOT Challenge 2014 results. For each tracker, the accuracy and robustness rankings were computed as a mean over the two experiments. Legend shown only for the best five trackers, see [116] for a full report.

the lower half of the leader-board. It should again be emphasised that the benchmark has a very different focus than this chapter and it fails to test our contributions: texture-less objects, transparencies and long-term sequences. However, it is useful to show the long-term and texture-less adaptations do not preclude competitive performance on standard sequences.

3.5.3 Visual Tracker Benchmark 1.0 Results

The proposed tracker was additionally evaluated in the context of the Visual Tracker Benchmark (VTB) 1.0 [206], also known as the CVPR tracker benchmark. This reports results of a different (mostly disjoint with VOT) set of trackers and a more extensive dataset (although with a significant overlap). The dataset contains sequences with longer durations and larger, even full, occlusions. The results can be seen in Table 3.5. Following common practice, Area Under Curve (AUC) of the success plots on the

	baseline			region_noise			
Tracker	**Acc.R.**	**Rob.R.**	**Comb.**	**Acc.R.**	**Rob.R.**	**Comb.**	Σ
DSST	5.4	11.9	8.7	5.4	12.3	8.9	8.77
SAMF	5.3	13.6	9.4	5.2	12.3	8.8	9.10
KCF	5.0	14.6	9.8	5.2	12.5	8.8	9.33
DGT	10.8	9.1	9.9	8.3	9.7	9.0	9.48
PLT14	13.9	6.2	10.0	13.1	4.8	9.0	9.51
PLT13	17.5	3.7	10.6	16.6	4.7	10.6	10.62
eASMS	13.5	13.3	13.4	10.9	13.7	12.3	12.85
HMM-TxD	9.4	19.9	14.7	9.1	18.8	14.0	14.33
MCT	15.9	13.5	14.7	16.8	12.3	14.5	14.61
ACAT	13.0	14.5	13.7	16.9	14.2	15.6	14.65
MatFlow	21.2	8.5	14.9	18.3	14.0	16.2	15.51
ABS	19.7	17.9	18.8	14.6	14.7	14.6	16.72
ACT	20.1	15.9	18.0	21.4	14.5	17.9	17.97
qwsEDFT	16.6	18.5	17.6	18.1	20.2	19.1	18.37
LGT	28.1	11.2	19.7	25.2	9.1	17.2	18.42
VTDMG	20.8	17.7	19.2	19.8	16.3	18.1	18.65
BDF	22.4	17.1	19.8	20.9	17.3	19.1	19.44
Struck	20.1	20.3	20.2	20.6	18.1	19.3	19.77
DynMS	21.5	18.8	20.1	20.8	18.8	19.8	19.97
ThStruck	21.7	19.4	20.5	21.3	17.9	19.6	20.06
aStruck	21.4	18.4	19.9	20.0	21.2	20.6	20.24
Matrioska	21.1	19.9	20.5	21.2	23.4	22.3	21.40
SIR-PF	23.6	20.1	21.9	21.6	21.7	21.7	21.76
EDFT	19.4	23.8	21.6	21.4	23.4	22.4	22.00
OGT	13.8	29.1	21.4	16.1	29.2	22.6	22.04
CMT	18.9	24.6	21.8	21.3	24.1	22.7	22.23
FoT	18.5	25.7	22.1	21.0	26.2	23.6	22.84
LT-FLO	**16.0**	**29.8**	**22.9**	**19.6**	**30.2**	**24.9**	**23.90**
IPRT	26.7	21.7	24.2	25.5	22.7	24.1	24.16
IIVTv2	24.8	24.8	24.8	24.6	23.0	23.8	24.29
PT+	32.0	20.7	26.4	29.2	19.4	24.3	25.34
FSDT	23.6	31.2	27.4	23.6	28.3	25.9	26.65
IMPNCC	25.6	27.7	26.6	28.3	28.3	28.3	27.45
IVT	27.2	28.9	28.1	26.6	27.3	26.9	27.51
FRT	23.4	30.4	26.9	26.2	31.0	28.6	27.74
NCC	17.7	34.2	26.0	22.8	36.8	29.8	27.90
CT	31.5	27.8	29.6	29.7	26.9	28.3	28.98
MIL	34.0	24.2	29.1	34.6	24.9	29.7	29.41

Table 3.4: Overall results of the VOT Challenge 2014 [116]. The trackers are sorted according to overall performance.

Tracker	IV	OPR	SV	OCC	DEF	MB	FM	IPR	OV	BC	LR	All
SCM	0.473	0.470	0.518	0.487	0.448	0.298	0.296	0.458	0.361	0.450	0.279	0.499
Struck	0.428	0.432	0.425	0.413	0.393	0.433	0.462	0.444	0.459	0.458	0.372	0.474
LT-FLO	**0.422**	**0.398**	**0.453**	**0.430**	**0.374**	**0.326**	**0.315**	**0.409**	**0.456**	**0.412**	**0.369**	**0.444**
TLD	0.399	0.420	0.421	0.402	0.378	0.404	0.417	0.416	0.457	0.345	0.309	0.437
ASLA	0.429	0.422	0.452	0.376	0.372	0.258	0.247	0.425	0.312	0.408	0.157	0.434
CXT	0.368	0.418	0.389	0.372	0.324	0.369	0.388	0.452	0.427	0.338	0.312	0.426
VTS	0.429	0.425	0.400	0.398	0.368	0.304	0.300	0.416	0.443	0.428	0.168	0.416
VTD	0.420	0.434	0.405	0.403	0.377	0.309	0.302	0.430	0.446	0.425	0.177	0.416
CSK	0.369	0.386	0.350	0.365	0.343	0.305	0.316	0.399	0.349	0.421	0.350	0.398
LSK	0.371	0.400	0.373	0.409	0.377	0.302	0.328	0.411	0.430	0.388	0.235	0.395
DFT	0.383	0.387	0.329	0.381	0.439	0.333	0.320	0.365	0.351	0.407	0.200	0.389
L1APG	0.283	0.360	0.350	0.353	0.311	0.310	0.311	0.391	0.303	0.350	0.381	0.380
MTT	0.305	0.362	0.348	0.342	0.280	0.274	0.333	0.395	0.342	0.337	0.389	0.376
OAB	0.300	0.358	0.370	0.368	0.351	0.324	0.358	0.345	0.414	0.341	0.304	0.370
LOT	0.286	0.364	0.335	0.378	0.345	0.312	0.331	0.355	0.467	0.385	0.189	0.367
MIL	0.311	0.350	0.335	0.335	0.369	0.282	0.326	0.340	0.382	0.373	0.153	0.359
IVT	0.306	0.323	0.344	0.325	0.281	0.197	0.202	0.330	0.274	0.291	0.238	0.358
CPF	0.271	0.372	0.335	0.384	0.371	0.250	0.307	0.341	0.400	0.303	0.151	0.355
TM-V	0.291	0.323	0.320	0.325	0.308	0.362	0.347	0.340	0.429	0.312	0.214	0.352
Frag	0.269	0.332	0.289	0.360	0.366	0.253	0.281	0.300	0.319	0.325	0.149	0.352
RS-V	0.302	0.342	0.314	0.350	0.364	0.305	0.299	0.318	0.368	0.359	0.221	0.346
ORIA	0.321	0.346	0.317	0.327	0.256	0.193	0.221	0.362	0.302	0.275	0.180	0.333
SemiT	0.273	0.303	0.285	0.313	0.337	0.296	0.300	0.297	0.319	0.317	0.330	0.332
KMS	0.333	0.317	0.312	0.333	0.328	0.326	0.321	0.287	0.400	0.321	0.211	0.326
BSBT	0.278	0.319	0.266	0.322	0.295	0.289	0.289	0.313	0.401	0.275	0.229	0.322
PD-V	0.296	0.293	0.263	0.316	0.347	0.276	0.264	0.255	0.231	0.260	0.165	0.308
CT	0.295	0.297	0.302	0.321	0.345	0.269	0.298	0.282	0.359	0.273	0.120	0.306
VR-V	0.212	0.256	0.234	0.273	0.301	0.232	0.214	0.223	0.217	0.264	0.125	0.268
SMS	0.214	0.237	0.259	0.268	0.233	0.221	0.248	0.191	0.298	0.183	0.151	0.229
MS-V	0.225	0.215	0.212	0.200	0.166	0.234	0.230	0.203	0.305	0.212	0.121	0.212

Table 3.5: Results (AUC on the OPE test) on the VTB1.0 benchmark [206]. The trackers are sorted according to overall performance.

One-Pass Evaluation (OPE) test is reported, for all the sequences and broken down by the scene attributes (Illumination Variation (IV), Out-of-Plane Rotation (OPR), Scale Variation (SV), Occlusion (OCC), Deformation (DEF), Motion Blur (MB), Fast Motion (FM), In-Plane Rotation (IPR), Out-of-View (OV), Background Clutter (BC), Low Resolution (LR)).

The longer sequences with stronger occlusion allow the long-term module of LT-FLO to prove its value. It placed third (*i.e.* in the top 10 %) in the overall leader-board. Unsurprisingly, the results on sequences with occlusions (the OCC column of Table 3.5) are even better (the second place with AUC 0.43). On the other hand, in sequences with out-of-plane rotation (OPR) the performance is not as good, as virtual corners assume edges are rigidly attached in 2D. This shows an opportunity for improvement based on 3D lines; this idea is explored in Chapter 5.

3.5.4 Short-term Tracking

In the previous experiments, the value of LT-FLOTrack was demonstrated on multiple standard benchmarks. In this section, its properties are examined in detail on several selected sequences.

The performance of the LT-FLOTrack algorithm is first evaluated in a short-term tracking scenario, against a number of competitive state-of-the-art approaches – Tracking-Learning-Detection (TLD) [106], Local-Global Tracker (LGT) [25] and Flock of Trackers (FoT) [137] using both standard and low-textured sequences. Properties of the sequences are summarised in Table 3.6 and selected frames with overlaid initialisation are shown in Figure 3.14. The same settings were used for all the sequences. Although the global redetection was enabled during all the experiments, it was not required in the short-term tracking scenario and only local corrections were employed by the method.

The speeds are shown in Table 3.7. In most cases LT-FLOTrack is faster than both FoT and LGT (everywhere except Spaceship, where the competing trackers fail). It should be noted that while the trackers are generally implemented as a compiled (C/C++) core with Matlab front-end, the FoT tracker is written completely in C++. The tests were carried out on a computer with the Intel i7-2600 processor (3.4 GHz, single core used). LT-FLO takes around 300 MB of RAM.

Figures 3.15 to 3.17 show the quantitative and qualitative results, respectively. Performance is measured using *location error* (distance of the bounding box centre from its ground truth position) and *scale error* (logarithm of the ratio of the estimated object size to its ground truth size, 0 means no error). The values were averaged over 20 executions.

For the Dog sequence, the most challenging part is between frames 700 and 1200, with a strong scale change and occlusion by the image boundary. While LT-FLO has no major problems and FoT experiences only light scale drift, LGT and TLD have severe problems, both in localisation and scale estimation.

The largest challenge of the Dudek sequence comes around the 210$^{\text{th}}$ frame, when

Name	Resolution	Frames
Dog [27]	320×240	1 353
Dudek [97]	720×480	1 145
Mug$^{\text{N}}$	640×480	737
Page$^{\text{N}}$	640×480	539
Spaceship$^{\text{N}}$	640×360	360

Table 3.6: Experimental video sequences for short-term tracking. Sequences marked by $^{\text{N}}$ are new and are available online with ground truth [118].

Name	LT-FLO	TLD	LGT	FoT
Dog	6.3	4.3	3.4	405.5
Dudek	3.3	2.0	2.1	131.7
Mug	4.8	2.6	2.8	230.6
Page	3.1	2.7	2.5	212.4
Spaceship	2.5	5.0	4.6	112.9

Table 3.7: Speed comparison of different trackers (in FPS).

Figure 3.14: Short-term tracking dataset (initialisation overlaid). From top to bottom: DOG [27], DUDEK [97], MUG^{N}, PAGE^{N} and SPACESHIP^{N}.

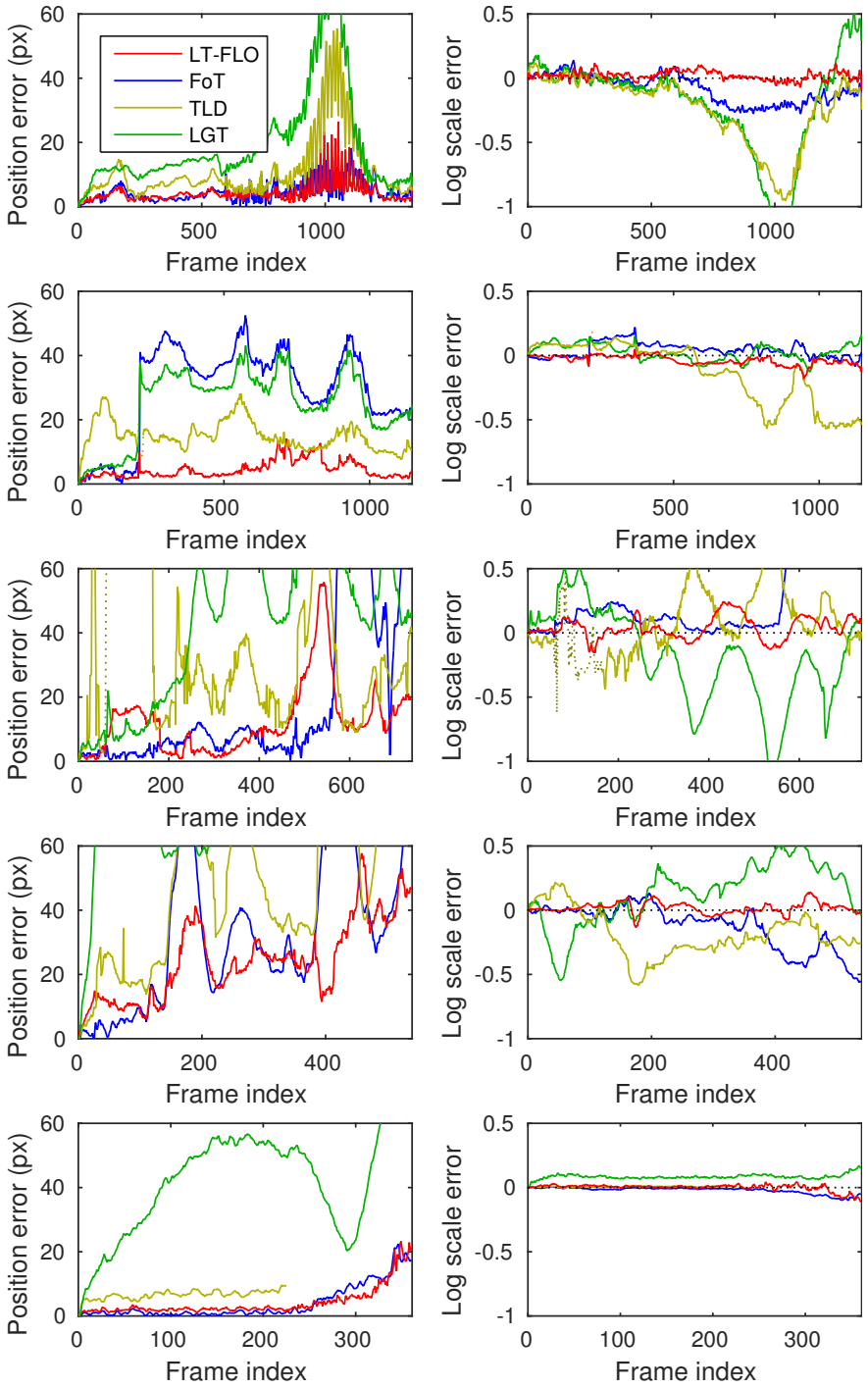


Figure 3.15: Results of short-term tracking evaluation. From top to bottom: Dog, Dudek, Mug, Page and Spaceship.

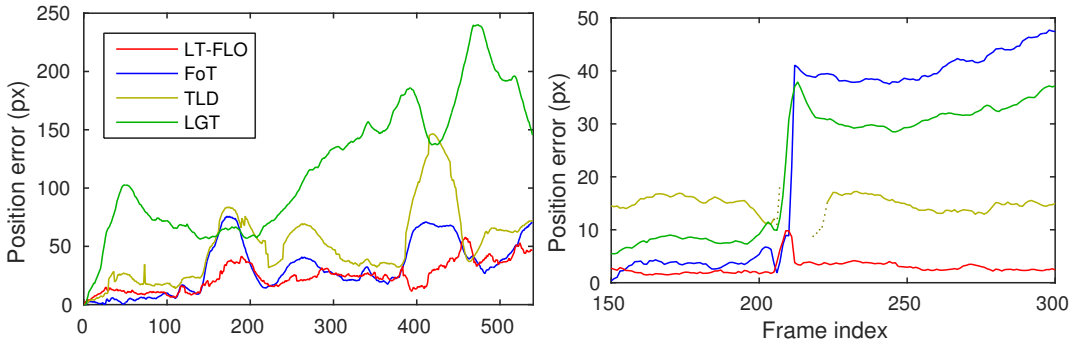


Figure 3.16: Results of short-term tracking evaluation: the PAGE sequence and a detailed view of the most challenging part of the DUDEK sequence.

the face is occluded by the right hand. While LT-FLO's pose is corrected in several frames, TLD requires a significantly longer time and the other trackers never fully recover (see Figure 3.16 for details). LT-FLO also experiences difficulties around frame 800, where background points influence tracking and cause drift. Nevertheless, LT-FLO recovers in every run.

Due to non-existent texture in the MUG scene, LGT is unable to track this sequence; the points simply drift off the mug and stay at the person's wrist. TLD often suffers from under- or overestimation of the object size and sometimes loses it completely. We mark these frames, where the object was (incorrectly) reported as missing in more than half the runs, by a dotted line. Frames where the object is always lost have no yellow plot at all. FoT works well in this sequence until around frame 550, where it fails. LT-FLO is comparable up to around frame 500, at which point tracking is lost in some of the runs, resulting in a poorer average score. However, the object is successfully re-detected before frame 600.

For PAGE, LGT performs similarly to the MUG sequence, all the points stabilise at the person's hand and wrist. FoT uses only features from fingers and TLD often loses tracking and rarely re-detects even when the paper returns to a pose similar to the initial one. LT-FLO experiences difficulties, but still significantly outperforms all the other trackers. Due to higher errors observed in this sequence for all the trackers, a rescaled plot of the position error is shown in Figure 3.16.

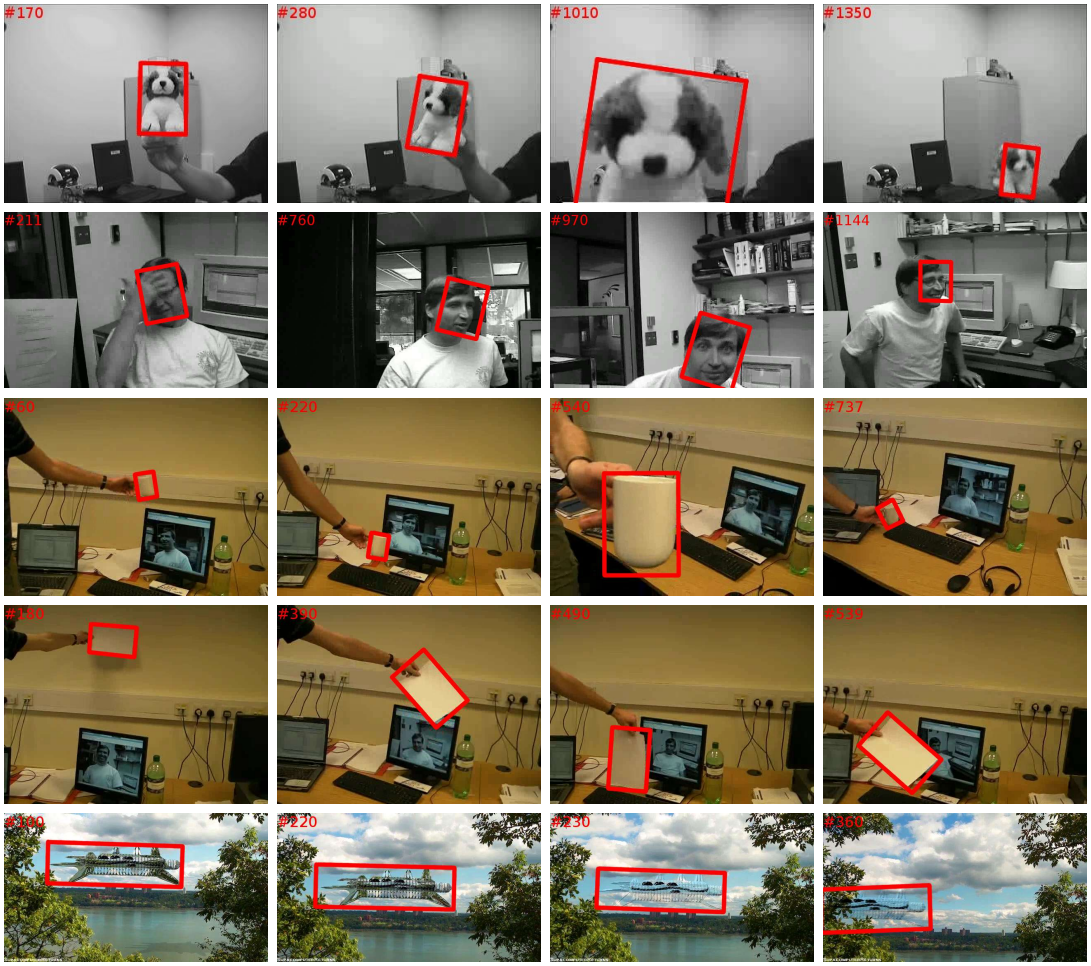


Figure 3.17: Short-term tracking qualitative results. From top to bottom: Dog, Dudek, Mug, Page and Spaceship.

Spaceship is an *augmented reality* sequence (a computer generated object on a real background). The tracked object becomes mostly transparent with only the outline roughly visible. TLD reflects this by confidently reporting it as missing. LGT fails to separate the target from the background, leading to high errors from the beginning and a failure later when it starts moving. Both FoT and LT-FLO track successfully throughout the sequence until the end when the object is partially occluded by trees. Their performance is comparable; FoT having slightly lower position error and LT-FLO lower scale error.

Name	Resolution	Frames
CarChase [106]	290×217	9 928
Panda[106]	312×233	3 000
Volkswagen[106]	640×480	8 576
LiverRun^{N}	320×240	**29 700**
Nissan^{N}	640×480	3 800

Table 3.8: Experimental video sequences for long-term tracking. Sequences marked by ^{N} are new and are available online with ground truth [118].

3.5.5 Long-term Tracking

In the short-term tracking scenario, LT-FLOTrack's performance is comparable or superior to state-of-the-art trackers. However, it obviously still loses track in cases of full occlusion/disappearance, thus it is necessary to test robustness of trackers to these conditions. For the evaluation of long-term tracking, different sequences are necessary. Not only should they be longer, but more importantly they must include full occlusions, background clutter and scale and illumination changes [106]. A common source of such sequences are traffic scenes such as car chases, when a vehicle is followed by another (possibly aerial) vehicle. The majority of sequences in this evaluation are therefore of this type. These sequences explore redetection capability and predisposition to drift. Their properties are tabulated in Table 3.8 and selected frames with overlaid initialisation are shown in Figure 3.18. Qualitative results of LT-FLOTrack on selected sequences are shown in Figure 3.19.

In a long-term tracking scenario, it is necessary to check whether the detection and/or tracking is precise (when the overlap with the ground truth bounding box is higher than 50 %) as well as to check for successful detection of object disappearance. Kalal *et al.* [106] applied the precision/recall/F-measure comparison. To check sensitivity of trackers to the initialisation, the experiments are run multiple times with the bounding box in the first frame shifted by 5 % in both horizontal and vertical direction and also scaled up and down by 5 % (averaging the results over all 7 possibilities). The concept of partial occlusion is also introduced into the evaluation process. When the object is not fully visible, but not fully occluded, the tracker can receive a score for either overlap of bounding boxes or for reporting disappearance.

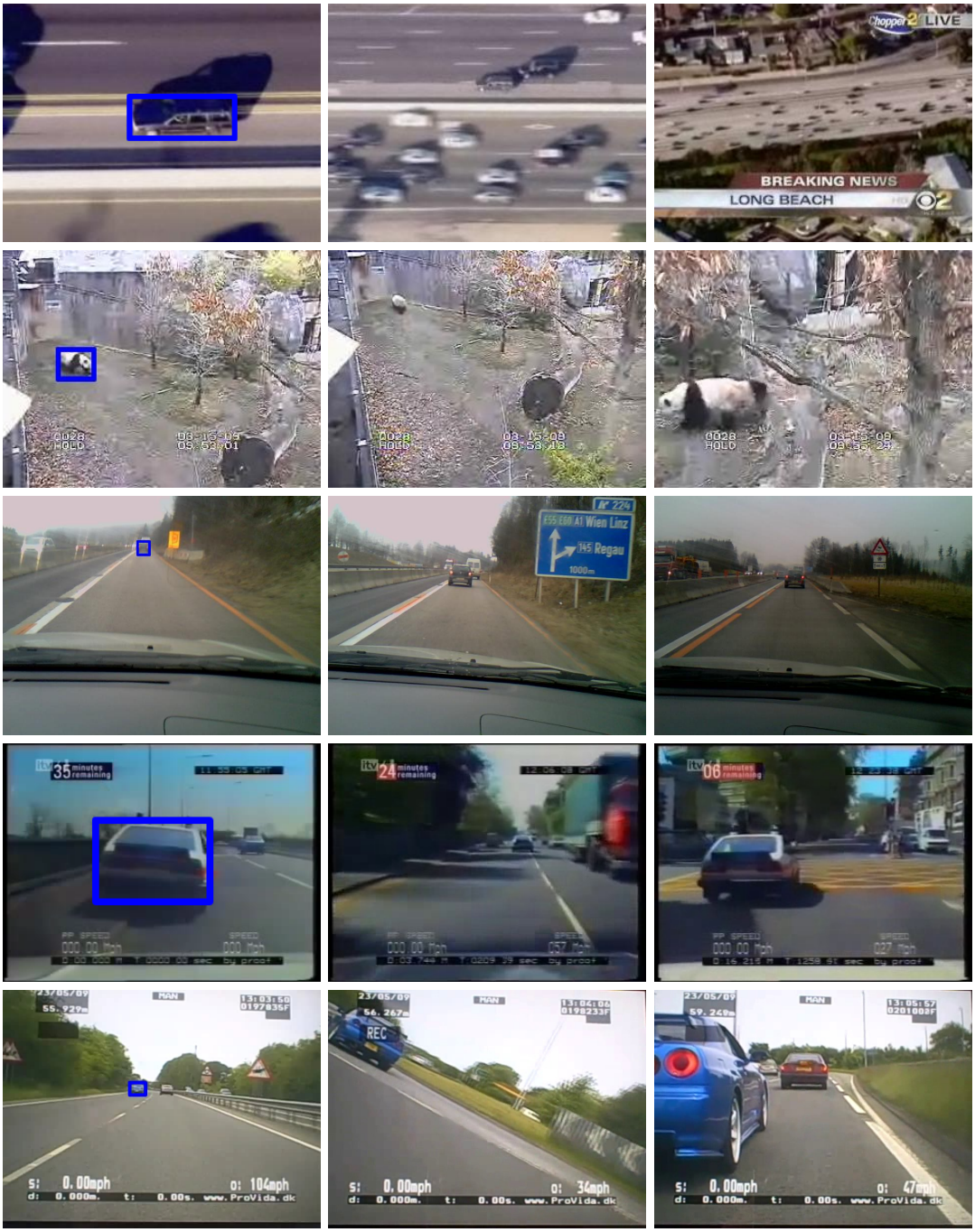


Figure 3.18: Long-term tracking dataset (initialisation overlaid). From top to bottom: CarChase [106], Panda [106], Volkswagen [106], LiverRun^{N} and Nissan^{N}.

Figure 3.19: Long-term tracking qualitative results. From top to bottom: LiverRun and Nissan.

The proposed LT-FLO tracker is compared with TLD [106], the first explicitly long-term tracker. The results can be found in Table 3.9. The trackers were initialised by a tight bounding box. It should be noted that the experiments of TLD on their own dataset have significantly different results if their own initialisation is used (better on the CarChase and Panda sequences and worse on the Volkswagen sequence, relating to F-measures of 0.45, 0.49 and 0.57 respectively). This indicates high sensitivity to the initialisation.

The results show that LT-FLO is capable of long-term tracking with competitive performance. In three out of five of the tested sequences it performed significantly better than TLD and comparably on one sequence. Also the average performance is better than TLD. The worst results of LT-FLO were acquired on the Panda sequence. This again highlights sensitivity to highly non-rigid objects with out-of-plane rotation.

Name	LT-FLO	TLD
CarChase	**.42**±.07 (3.2)	.15±.08 (11.3)
Panda	.17±.04 (4.1)	**.23**±.07 (12.0)
Volkswagen	.51±.20 (3.0)	**.62**±.13 (5.4)
LiverRun	**.56**±.20 (6.1)	.29±.28 (4.0)
Nissan	**.88**±.14 (4.1)	.63±.14 (4.4)
mean	**.53**±.17 (4.9)	.36±.22 (6.0)

Table 3.9: Results of long-term tracking. Tabulated values are in the format: *F-measure (speed in FPS)*. The mean values (in the last row) are weighted by the number of frames.

Experiments were carried out with the short-term trackers as well, but results of these are not tabulated as they lost tracking at or before the first full occlusion (*e.g.* frame 440 for LiverRun or 1860 for Nissan). These include FoT, LGT and LT-FLO without *global correction*, which proves its importance in long-term tracking.

3.5.6 Contributions of Algorithm Components

Local Corrections

As all of the short-term trackers failed, a similar experiment was designed, to explore the behaviour of LT-FLOTrack without *local* corrections. To test the contribution of local corrections, only the short-term tracker (as introduced in Section 3.2) was executed. See the results in Figure 3.20. First, the short-term tracker was run on the Dog sequence. The results are comparable to those of LGT or TLD, with errors almost an order of magnitude worse than those of LT-FLO. The high errors can, however, be attributed to drift only – the tracker did not lose the object completely.

The Page sequence is more difficult. Without the corrections, the short-term tracker failed completely several times, usually after approximately 100 frames. At this point (when the bounding box overlap decreased below 10 %) the tracking was restarted from GT, hence the low error after each discontinuity. This is similar to the VOT challenge evaluation protocol. In this case, the short-term tracker was not able to

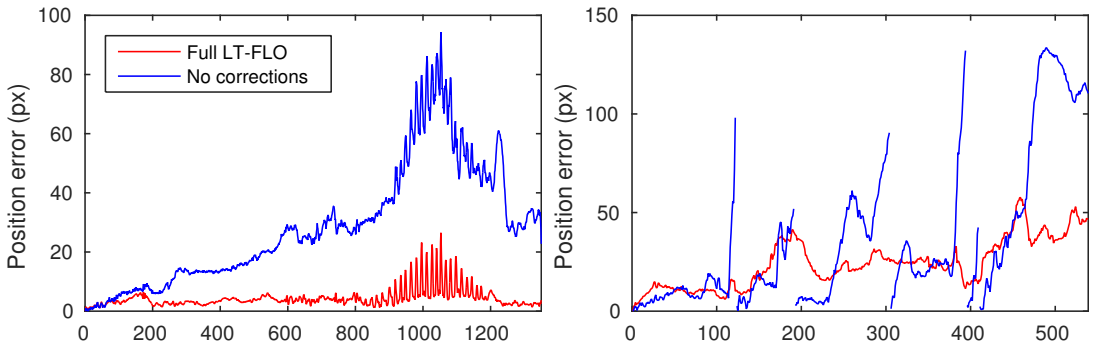


Figure 3.20: Tracking without local corrections. Left: Dog, right: Page.

track the object throughout the sequence, with high errors reported very shortly after each manual intervention.

Transformation Quality Measures

To test the contribution of different components of the transformation quality measure (E, Q and $\mathcal{I}$ in Equation (3.18)), each of them was removed and the final performance of the tracker was tested without them. Table 3.10 summarises the effect of the elements in both the short-term (Dog, Page) and long-term (Panda, Nissan) scenarios. While in simple sequences (Dog) the variation is low, more challenging scenarios have significant differences in performance.

The full technique performs the best in all cases. From the results it is not clear which of the individual measures performs the best. It can however be observed, that the combination of edge quality field fit with the number of inliers obtains the highest score of all the two-component combinations. This can be probably attributed to the fact that the image evidence E is maximised explicitly and is thus high in all hypotheses compared using the score Γ.

Name	E	Q	$\mathcal{I}$	$E \cdot Q$	$E \cdot \mathcal{I}$	$Q \cdot \mathcal{I}$	$E \cdot Q \cdot \mathcal{I}$
Dog	0.76	0.73	0.74	0.73	0.76	0.74	**0.77**
Page	0.18	0.13	0.27	0.20	0.23	0.32	**0.61**
Panda	0.14	0.10	0.08	0.12	0.10	**0.21**	**0.21**
Nissan	0.20	0.42	0.31	0.25	0.41	0.78	**0.88**

Table 3.10: Effect of elements in Equation (3.18): short-term tracking (mean bounding box overlap tabulated) and long-term tracking (F-measure tabulated).

Thresholds Selection

Finally, the effect of changing the algorithm parameters and its robustness to such selection is tested. The most important parameters (besides the number of generated edge-points discussed in Section 3.2.5) are the position error thresholds θ_1 and θ_2 as specified in Section 3.3, which control the behaviour of the redetection framework. These are defined relative to the object size, as a portion of the bounding box diagonal. In our implementation, $\theta_1 = 0.03$ and $\theta_2 = 0.06$ are used (see the vertical cyan lines).

Figure 3.21 shows the performance of LT-FLOTrack as a function of these thresholds. Both graphs are relatively flat, indicating low sensitivity. However, certain observations can be made. In the case of θ_1, a good range to choose from is between 1 and 5 %. Setting the threshold too low causes the learning to be too strict, preventing the tracker's ability to adapt. On the other hand, too high a setting renders the tracker too adaptive and thus drift-prone. In the case of θ_2, the lower values in general provide better performance. This indicates that it is advantageous to accept even very similar corrections, as long as they have higher score Γ.

3.6 Closing Remarks on LT-FLOTrack

In this chapter, virtual corners are explored as a solution to low texture tracking. A two module approach is used, where the short-term tracker finds the frame-to-frame transformations, using correspondences of lines tangent to edges. This works for textured, low-textured and even transparent objects, only requiring rigid sets of edges,

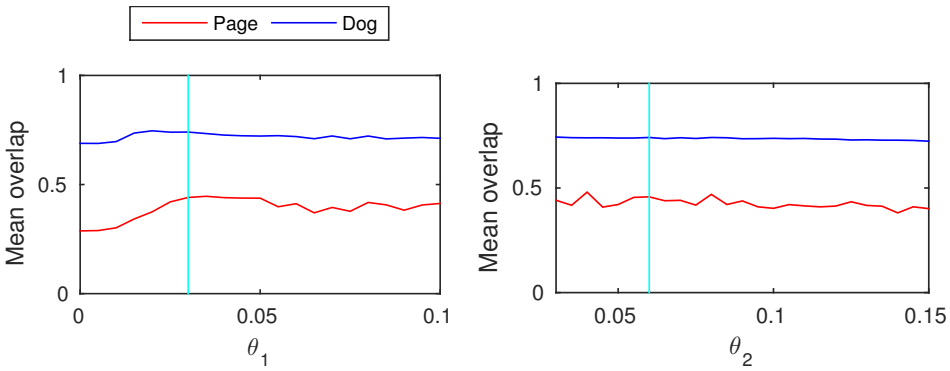


Figure 3.21: Sensitivity of the method to parameters: the learning threshold θ_1 (left) and the correction threshold θ_2 (right).

not uniquely localisable corners. The long-term module facilitates learning of common modes of appearance for the tracked object, recovery from failures and redetection after full occlusions. Strict learning rules make it robust to drift. This, in conjunction with a guided redetection framework, renders LT-FLOTrack capable of tracking long sequences robustly. This is demonstrated, among others, by its excellent performance on the LiverRun sequence of nearly 30 000 frames, the longest sequence published so far.

LT-FLOTrack covers several challenging cases, where traditional trackers often fail. However, it does not attain the highest score on every sequence. As the experiments show, there are several failure cases. Firstly, edge-distorting compression artefacts typically cause the edge-correspondence search to fail. Typically, performance is also degraded for highly non-rigid or articulated objects (*e.g.* in the Panda sequence), as non-rigid objects break the definition of virtual corners (assuming that distant edges are rigidly attached). LT-FLO can cover these only partially, using its multiple models. The same reasoning applies to cases of strong out-of-plane rotation, which will be explained in more detail in Chapter 5.

Chapter 4

Causality-based Motion Models in Visual Tracking

Until lately, the evaluation protocols varied wildly between different publications on visual tracking. In the recent years, however, several widely-used benchmarks have set standard measures allowing for a broad comparison of large numbers of trackers. Such a comparison, besides being beneficial to the tracking community, brought several unexpected observations. Conclusions can be drawn when looking in detail at, for instance, the Visual Object Tracking challenge (VOT) 2013 [115]. Some sequences, which should be tracked easily (and are trivial for a human observer), are actually surprisingly difficult. An example is the DIVING sequence, with an extreme case of a *central bias*. Others should be trivial, like the static scene in the JUICE sequence (where the entire motion is due to the camera being moved). Finally, there are sequences, which have isolated moments where many approaches fail, such as BICYCLE.

In all of these cases, the motion of the object is different than motion commonly considered when designing motion models of trackers. The motion of the camera plays a significant role with artefacts such as the mentioned centre bias, or sudden camera shake. In this chapter, the relationship between the camera and object motion, as well as between the motion of multiple objects within the scene is explored. Automated

identification of such relationships and ways to exploit them are investigated. This can lead to improved motion models, which take the context of the scene into consideration.

4.1 Tracking within Scene Context

An example of a relationship, which can be observed (and exploited) in the area of visual tracking, is the relationship between the motions of the camera and an object. There are different possible relationships. For instance, the motion of the camera instantly causes motion of the object in the image frame. An abrupt movement of the camera (*e.g.* a shake) can cause a tracker to fail even in otherwise simple tracking scenarios. A particular example can be seen in the performance of all submitted trackers on the BICYCLE sequence in the ICCV Visual Object Tracking challenge (VOT) 2013 [115]. While this sequence is relatively easy to track in general, there are two challenging moments (see Figure 4.1, showing the number of failed trackers per frame in the VOT Challenge). Many tracking failures are present around frame 180, caused by a strong occlusion, and around frame 140, stemming from an abrupt camera shake. If these were detected and accounted for, many of the failures could be prevented, regardless of the tracker.

Another interesting relationship often arises when the motion of the object causes changes of the camera motion. If there is a human in the loop, *e.g.* a cameraman, they are partially tracking an object by definition. A similar conclusion would hold for an automatically-controlled camera, tracking the object. When the object moves towards the edge of the image, the cameraman is likely to move the camera such that the object does not disappear from the scene. An extreme case of this is the satirical Zero-order Tracker [136], shown to successfully track a challenging sequence by simply returning a bounding box on a constant location in the image. As illustrated in Figure 4.2, here the cameraman kept the diver in the centre of the image frame for almost whole sequence. However, even in less extreme cases, the commonly assumed centre bias can be detected, measured and exploited.

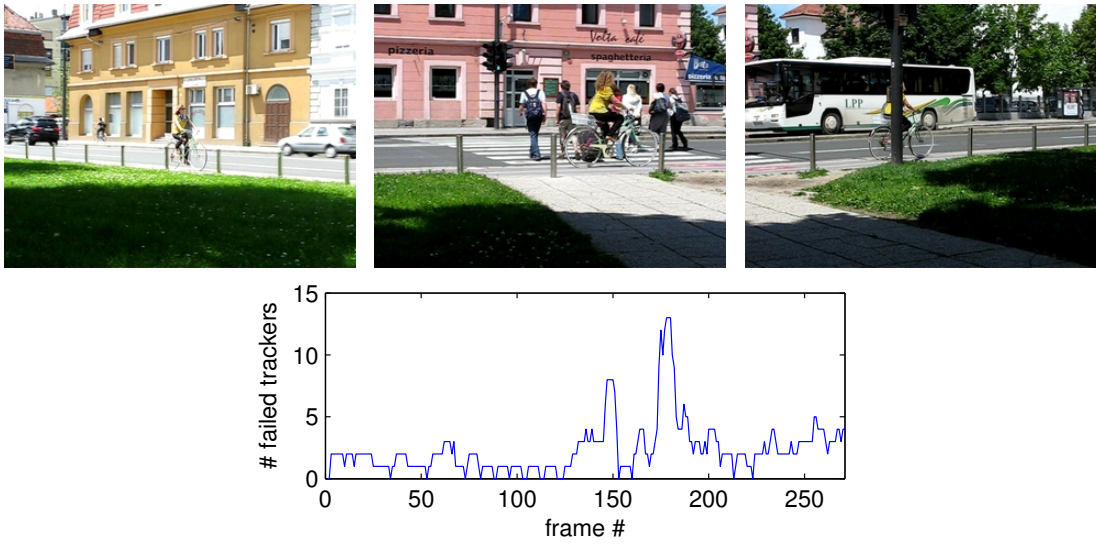


Figure 4.1: Top: selected frames of the BICYCLE sequence in the VOT Challenge (1, 140&173). Bottom: number of trackers from the challenge, which failed on particular frames. Notice the two challenging moments, a strong occlusion around frame 180 and an abrupt camera shake around frame 140.

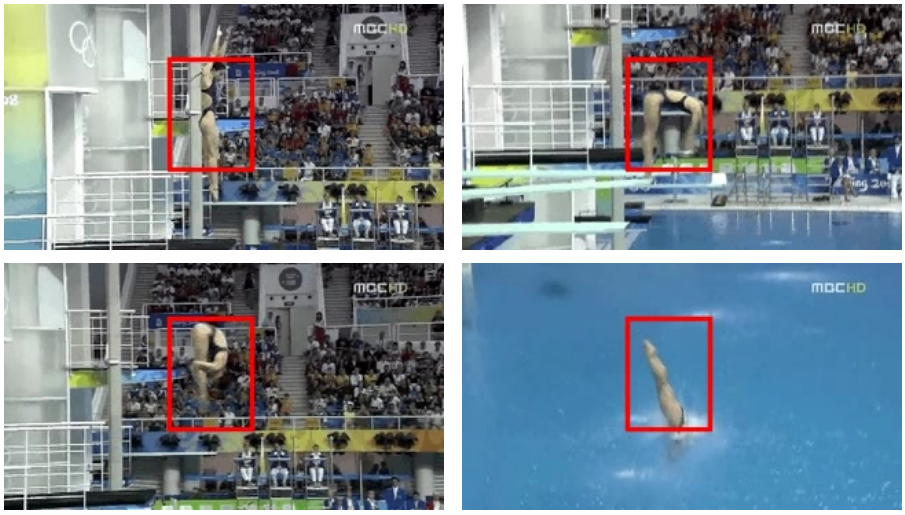


Figure 4.2: Selected frames from the DIVING sequence, challenging for many trackers, with overlaid "results" of the Zero-order Tracker [136].

It should be emphasised that the work presented in this chapter does not *assume* any kind of high-level oracle (*e.g.* a human operator) driving the camera motion. In cases where a relationship exists, its influence can be discovered and measured. However, if

none is present, no link is found and no further action (such as object motion prediction) is performed. This even extends to changes in behaviour within a sequence. This means the motion model can be applied to any existing tracking framework to improve results. To demonstrate this, its effect on two state-of-the-art trackers FoT [199] and Struck [76], as well as on the LT-FLO tracker presented in the previous chapter, are examined on two benchmark datasets. It should be noted that while the prediction helps the tracker, it does not replace it and the result is thus still limited by the abilities of the tracker.

However, relationships in the context of visual object tracking are not limited to the camera↔object links. Different elements of the scene may simultaneously be moving along their own trajectories, which may also be linked. These links are explored and described in a way similar to the object-camera relationship. They can be analogously exploited to make predictions about the object even when it is completely occluded.

4.2 Measuring Causal Relationships: Transfer Entropy

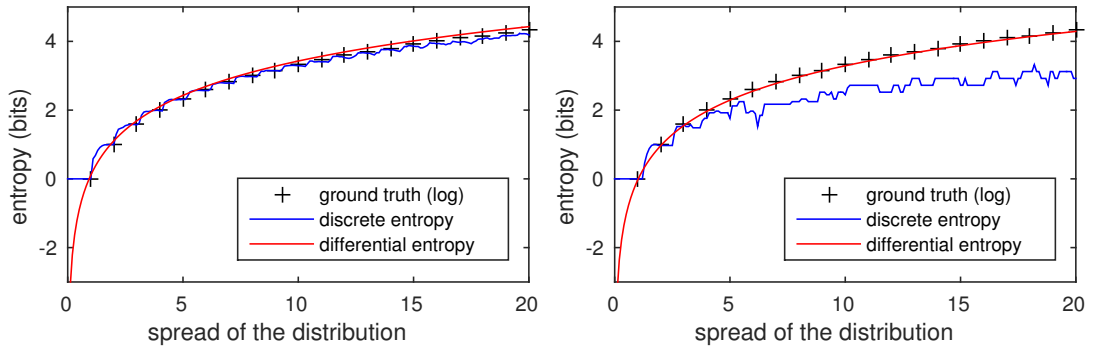
In this chapter, the relationships are identified as *causal relationships*. It should be however noted that it is impossible to reason about true causality without higher, semantic understanding of the scene. Therefore this chapter works with *predictive causality* instead, which reasons about apparent causal links instead of true causation (*i.e.* it does not distinguish between causation and delayed correlation).

Causality is a relation between two events, a *cause* (source) and an *effect* (consequence). In general terms, we say that an event causes another event (its effect), if it precedes the effect in time and it increases the probability of the effect happening. Although causality has been studied by philosophers for millennia, it received little attention from the scientific community before the twentieth century. Recently, theoretical advances have brought practical progress in the analysis of time series in many scientific areas.

We can measure the degree of causality between the camera and the object motion using various mathematical tools. *Granger causality* has become a standard approach used in numerous applications. However, as discussed in Chapter 2, it is limited to linear relationships despite its recent improvements. Approaches based on information theory are beneficial, as they impose few assumptions about the signals. *Mutual information*, a measure of general statistical dependence, has been used. While removing the assumption of linearity, it is symmetric, *i.e.* does not distinguish between a cause and an effect. In this work, Transfer Entropy (TE) is used, a directional information-theoretic formulation employing (differential) entropies (see Sections 4.2.1 and 4.2.2). To discover if the relationship is significant, a statistical significance analysis can then be employed (Section 4.2.3). This can be executed in each frame, until such a relationship is found. In the case it is not, it can be concluded that the motions are unrelated (static camera or independent motion) and no information can be supplied to the tracker (uniform prior), possibly until the end of the sequence. In the case where a statistically significant causal relationship is found, its parameters can be estimated (Section 4.2.4). This information may then be used to predict future object motion (Section 4.3); such information is supplied to the tracker.

4.2.1 Differential Entropy

Histogram-based methods are usually employed to estimate the entropy of a random process [40, 41]. However, in the visual tracking scenario this has two major disadvantages. Firstly, there is an arbitrary choice of bin size for the histograms (for quantisation of continuous signals). Secondly, the number of bins grows exponentially with the number of dimensions. This causes the histograms to be very sparse (and thus not representative of the distribution). Furthermore it requires immense computational cost even for a small number of bins per dimension. Discrete entropy measures also tend to introduce artefacts to the estimation process, which need to be taken into account [41]. The effect of these can be seen by the periodic oscillations of the discrete estimate in Figure 4.3a.



(a) 100 points used (b) 10 points used

Figure 4.3: Comparison of discrete and differential entropy. Points were uniformly sampled from an interval $(0;x)$ (x varies along the horizontal axis). The histogram bins for discrete entropy computation were fixed at integer positions. Notice how stable differential entropy is, even with sparsely distributed points (no interpolation used).

Therefore, differential entropy is used instead, which operates directly on the continuous variables (see Figure 4.3 illustrating the advantages of differential entropy). In this work the Kernel Density Estimation (KDE) [88] approach to compute differential entropy is employed, which only requires a choice of kernel (a Gaussian kernel with full covariance is used). The differential entropy of a continuous random process X is

$$H(X) = -\int_X p(x) \log p(x) \, \mathrm{d}x \,, \tag{4.1}$$

similar to its discrete counterpart. For a finite sample set $\mathcal{S}$ it is approximated using KDE by:

$$\hat{H}(X) = -\frac{1}{|\mathcal{S}|} \sum_{x_i \in \mathcal{S}} \log \hat{p}(x_i) = -\frac{1}{|\mathcal{S}|} \sum_{x_i \in \mathcal{S}} \log \left(\frac{1}{|\mathcal{S}| - 1} \sum_{x_j \in \mathcal{S} \setminus x_i} \kappa_\Sigma (x_i - x_j) \right) , \tag{4.2}$$

where κ_Σ is a Gaussian kernel with covariance Σ. In this formulation, the probability $p(x_i)$ outside the logarithm is approximated by the distribution of the samples from $\mathcal{S}$ (*i.e.* assuming $\mathcal{S}$ was drawn according to $p(x)$). The kernel covariance matrix Σ is computed using an EM algorithm. This removes the necessity to choose a bandwidth for KDE manually. In related work, it has been suggested that over-relaxation may

speed the estimation up [88], however, this proved unstable with high-dimensional data (when the number of data points is less than roughly two orders of magnitude over the number of dimensions) and was not used here.

Even without using overrelaxation, the results are often unstable when the number of data points is of the same order of magnitude as the number of dimensions. However, it is possible to emulate having more data. Since the motion of both camera and object are continuous (and can be assumed to be smooth between frames), their positions can be interpolated at non-integer time moments (between the frames). This makes the entropy estimation robust even at the beginning of the sequence, where only a few samples are available.

4.2.2 Transfer Entropy

Differential entropy describes the amount of information within a random process. However, in this chapter, the interest lies in relationships between several processes. For this purpose, Transfer Entropy (TE) is used, a measure of directed influence flow between two processes ($X \to Y$, with windows[1] of length n and lag Δt, see Figure 4.4). For continuous signals it is defined as:

$$H_{X\to Y} = \iiint p(y^t, \mathbf{y}_n^t, \mathbf{x}_n^{t-\Delta t}) \log \frac{p(y^t|\mathbf{y}_n^t, \mathbf{x}_n^{t-\Delta t})}{p(y^t|\mathbf{y}_n^t)} \,\mathrm{d}y^t \,\mathrm{d}\mathbf{y}_n^t \,\mathrm{d}\mathbf{x}_n^{t-\Delta t}\,, \tag{4.3}$$

with time windows defined as $\mathbf{y}_n^t = (y^{t-n}, ..., y^{t-1})$. It can be reformulated (using differential entropies from Section 4.2.1) as the difference of two information gains:

$$H_{X\to Y} = \big(H(Y^t, \mathbf{Y}_n^t) - H(\mathbf{Y}_n^t)\big) - \big(H(Y^t, \mathbf{Y}_n^t, \mathbf{X}_n^{t-\Delta t}) - H(\mathbf{Y}_n^t, \mathbf{X}_n^{t-\Delta t})\big)\,. \tag{4.4}$$

Intuitively, this tells us that if there is an information transfer between X and Y with the correct direction and lag, then adding knowledge about Y^t brings more information to a system which does not know X, than to one which does (as X can partially predict it).

[1] Although the window lengths are equal for X and Y throughout this work, they do not necessarily need to be such.

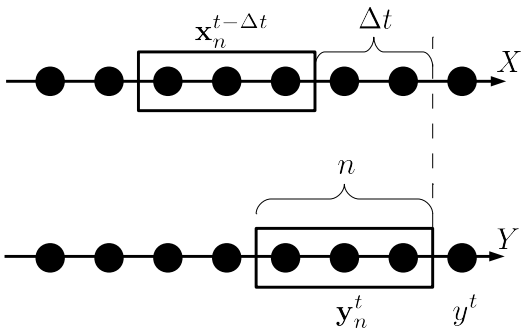


Figure 4.4: Illustration of information transfer between two processes X and Y.

TE is used here as the measure of the strength of a causal link. It is estimated on every frame, using the observed history (all the $\mathbf{x}_n^{t-\Delta t}$ and $\mathbf{y}_n^t$ windows) thus far. This creates a time series, which can be investigated using statistical methods, as detailed in the next section.

4.2.3 Statistical Significance Analysis

Once the transfer entropy between the motion of the camera and the object is known, it needs to be decided if this relationship is significant enough to make predictions of the object movement. Tests of statistical significance are preferred rather than comparing to a fixed threshold. They offer theoretically founded decisions with probabilistic thresholds and explicitly cope with the inherent uncertainty caused by insufficient data.

T-tests are one of the best established statistical methods. In this framework, a *null hypothesis* is formulated as an opposite of what is sought to be confirmed. Then a probability of the null hypothesis being valid is computed and compared to a threshold. An extremely unlikely null hypothesis can be disproved and therefore the original idea confirmed. If the null hypothesis cannot be rejected, it can mean either that it is valid, or that there was not enough data.

The null hypothesis, sought to be disproved (and hence the opposite hypothesis proved) in this case could be formulated simply as $\mathrm{E}(H_{X\to Y}) \leq 0$. In other words, the

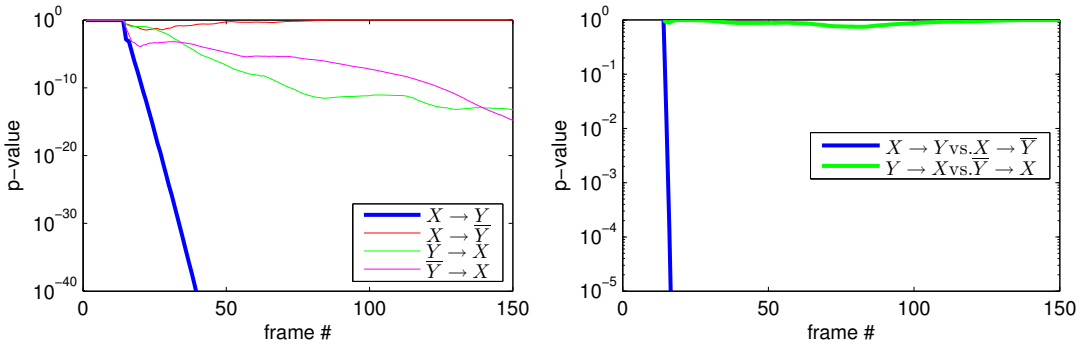


Figure 4.5: Left: Results of t-test for information transfer on synthetic data. Notice that very high statistical significance (p-value beyond any commonly used threshold) was reached even for random/invalid signal, due to non-independent samples. Right: Results of Welch's t-test using surrogate data.

expectation (*i.e.* the mean of the underlying distribution) of the TE is negative or zero, meaning X is not informative about Y. This will however often fail as the samples of TE are not independent (a large portion of the data is overlapping in consecutive calculations) and many false positives appear (see Figure 4.5 for an example). Therefore one needs to compare the distribution of $H_{X\to Y}$ with known false *surrogate data* (a synthetic baseline).

Welch's test, a more robust and reliable adaptation of Student's t-test, is used for such comparison. This test, also known as *unequal variances t-test* is a statistical test to compare the means of two random distributions (assumed normal). No assumptions are taken regarding the numbers of samples given or the variances. It can be, however, applied to distributions with equal variances without any significant disadvantage against more common (such as Student's) tests.

To provide a sequence-specific baseline with no causal relationship the target time series Y is shuffled to remove any causality while still retaining the appropriate distribution of amplitudes. The shuffled signal is denoted as $\bar{Y}$. Then a Welch's t-test is performed, to obtain a p-value indicating the probability of the null hypothesis $\mathrm{E}(H_{X\to Y}) \leq \mathrm{E}(H_{X\to\bar{Y}})$: that the information gain between the signals $\mathrm{E}(H_{X\to Y})$ is weaker than the information gain of the shuffled signals $\mathrm{E}(H_{X\to\bar{Y}})$ (the opposite of

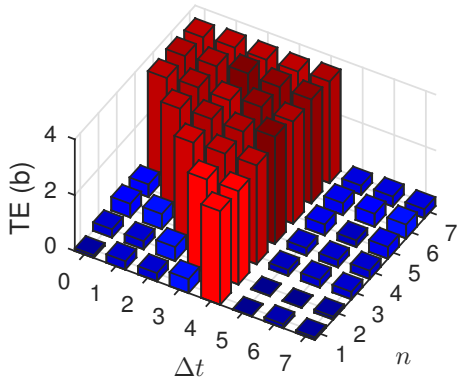


Figure 4.6: Dependence of TE on the time lag and the size of the time window.

what could be expected if there was a causal link). Figure 4.5 shows the advantage of using the surrogate-data test, as opposed to a standard t-test. When the observed causal relationship is statistically more significant than what is likely to arise by chance given the signal distributions, it is concluded that it can be used for object motion prediction. This approach is shown to be successful in 15 out of 16 sequences from the VOT2013 dataset.

4.2.4 Finding the Optimal Parameters

When a causal relationship has been confirmed, an attempt can be made to predict the object motion from the overall movement of the scene (which is dual to the camera motion). Since different processes have, in general, different causal relationships, each particular sequence will have unique properties. These may even change within a sequence, the process allowing this change is described in Section 4.4.1. In other words, an optimal set of parameters needs to be identified for this prediction. These parameters are the time delay and the length of the time window, which can be seen as a mean and variance of the lag Δt. These should be chosen such that TE is maximal:

$$(\Delta t^*, n^*) = \arg\max_{\Delta t, n} H_{X \to Y}(\Delta t, n)\,, \tag{4.5}$$

where $H_{X\to Y}(\Delta t, n)$ relates to TE parameterised with a particular window length and lag. Unfortunately, optimising the window length n is not as simple as optimising Δt. The transfer entropy stays high even when the window length is overestimated. Notice the characteristic triangular shape of the area with consistently high TE in Figure 4.6: when the time window already containing the most relevant information is extended, no significant information is gained or lost. However, even though the longer windows still contain all the important information from the smaller windows, they are more computationally expensive. Furthermore, non-discriminative features are likely to degrade performance, particularly with small training sets [67]. Therefore a relative improvement measure $R(\Delta t, n)$ from adding an additional frame to the window length is defined, visualised by the column colours in Figure 4.7. This is required to be higher than a given threshold θ_R. This leads to a constrained optimisation problem:

$$(\Delta t^*, n^*) = \arg\max_{\Delta t, n} H_{X\to Y}(\Delta t, n) \quad \text{s. t.} \quad R(\Delta t, n) > \theta_R\,, \tag{4.6}$$

with R defined as

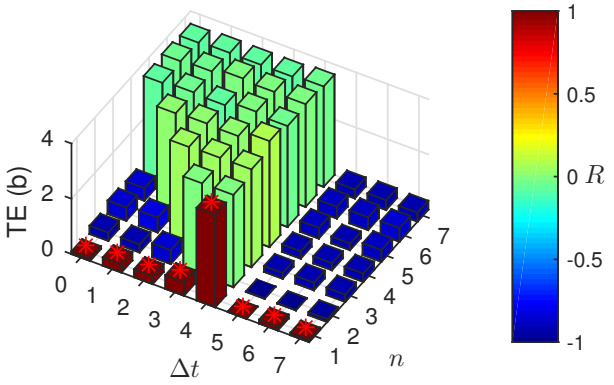
$$R(\Delta t, n) = \frac{H_{X\to Y}(\Delta t, n) - H_{\max}(n-1)}{H_{\max}(n)}\,, \tag{4.7}$$

where $H_{\max}(n)$ is the maximal entropy achievable with a window of length (at most) n:

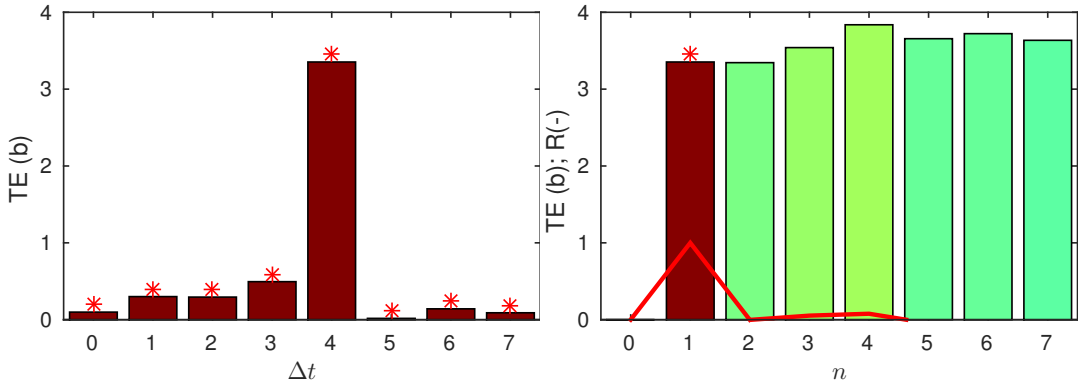
$$H_{\max}(n) = \max_{\underline{\Delta t};\ \underline{n}\leq n} H_{X\to Y}(\underline{\Delta t}, \underline{n}) \tag{4.8}$$

For experiments in this work, $\theta_R = 10\,\%$ was used.

Figure 4.7 shows the effect of the parameters Δt and n. Note that in Figure 4.7b the decrease of TE is slower when moving from the optimal lag to the future (lower lag), than to the past (higher lag). This is caused by future data providing more information than the more redundant information from the past, as the past information can be already recovered from the Y signal.



(a) Joint parameter space.



(b) Slice with fixed $n = 1$.

(c) Slice with fixed $\Delta t = 4$.

Figure 4.7: Dependence of TE on the time lag and the size of the time window. The column colours and the red line in (c) visualise the relative improvement R. The red stars denote all combinations of n and Δt with $R > \theta_R$ (*i.e.* where the condition from Eq. (4.6) is satisfied).

4.3 Exploiting Causal Relationships: Causal Predictions

The knowledge of the relationships between the camera motion and the object motion is interesting in itself, and can be useful, for instance, in the task of behaviour analysis [18]. However, since the causal relationships were formulated in the domain of visual object tracking, it is natural to apply them to this task and explore their effectiveness in helping to improve tracking results. The global motion of the camera can be estimated robustly using the inter-frame shift of the whole image, with higher reliability than the object tracking. Therefore, any discovered relationship can be used

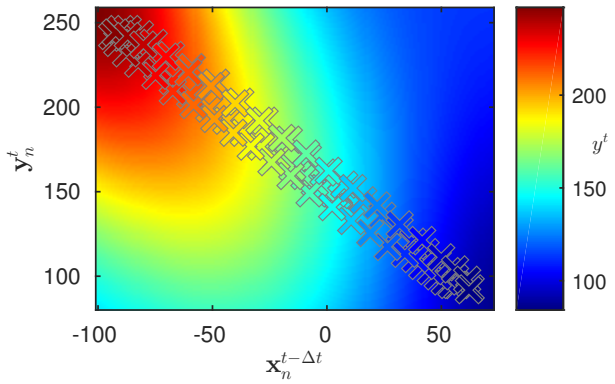
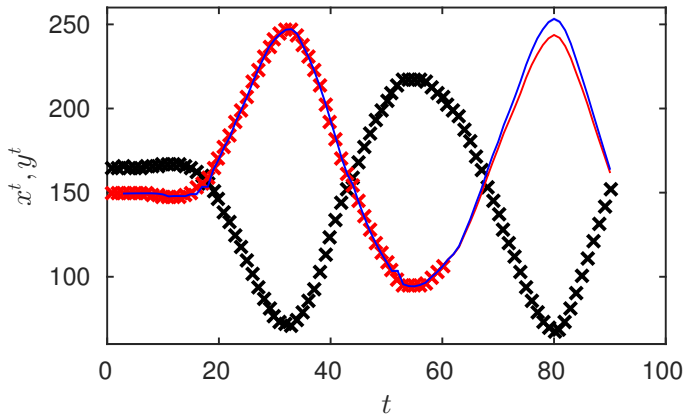


Figure 4.8: Window-based prediction function ϕ_w for the JUICE sequence. The training data are denoted by crosses and their associated colour, the background colour illustrates prediction of y^t (inter- and extrapolation from the training data), given $\mathbf{y}_n^t$ and $\mathbf{x}_n^{t-\Delta t}$, by the learned function ϕ_w. In this case $n = 1$, however in general $\mathbf{y}_n^t$ and $\mathbf{x}_n^{t-\Delta t}$ can be high-dimensional.

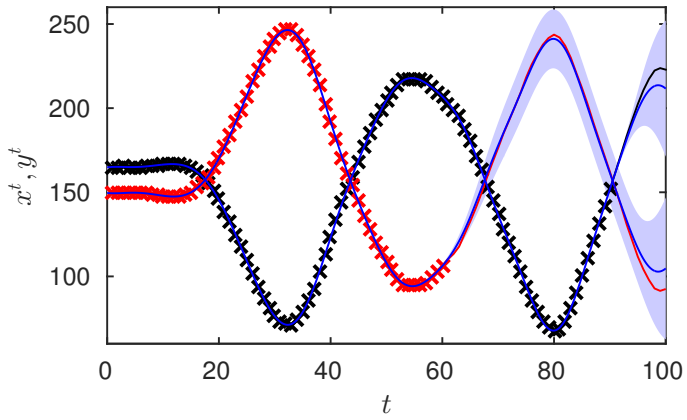
to transfer information from one (reliably estimated) signal to the other. In this chapter, this information transfer is seen as the estimation of a distribution of possible poses for an object, based on its history Y and additionally on its relation to the information from the image signal X. This distribution can be supplied to a tracker as prior information to guide the tracking process. Two different approaches to object position prediction are examined; the following sections describe these.

4.3.1 Window-based Prediction

In the first case, a window-based prediction is used, similar to a non-linear autoregressive model. This approach is intuitively similar to the formulation of transfer entropy as described in Section 4.2.2. In autoregression, the current state is estimated (predicted) using a learned autoregressive function ϕ_a from its own history: $y^t = \phi_a(\mathbf{y}_n^t)$. This is similar to Kalman Filter (KF), which predicts future state of the filtered process based on previous observations, assuming that the process and observation noises are Gaussian. As is the case with autoregression, KF uses only the observations of the predicted signal. Knowing there is a causal relationship between



(a) window-based prediction function ϕ_w



(b) time-based prediction function ϕ_t

Figure 4.9: Predictions for the JUICE sequence. Black and red: the training data (crosses) and ground truth (in the extrapolated regions) of the X and Y signals, respectively; blue: mean and 95 % confidence intervals (too narrow to be seen in (a)) of the prediction. The learned relationships ensure predictions of Y for frames 60–90 have higher accuracy and confidence than would be possible with simple extrapolation.

the two signals, the prediction can be improved using the other signal:

$$y^t = \phi_w(\mathbf{y}_{n^*}^t, \mathbf{x}_{n^*}^{t-\Delta t^*}) \tag{4.9}$$

This window-based regressive function ϕ_w can be learned. In other words, a set of all windows from the history is taken as training data and a regression (mapping) from the known part ($\mathbf{y}_n^t$ or both $(\mathbf{y}_{n^*}^t, \mathbf{x}_{n^*}^{t-\Delta t^*})$ knowing the optimal parameters) to the current

pose y^t is learned. This learned structure proves beneficial in experiments, compared to the one of KF, which is fixed.

The principle of this is visualised in Figures 4.8 and 4.9a. In Figure 4.8, the colour is visualised as a function of the image coordinates. In Figure 4.9a, both the training data and predictions are shown as a function of time. Normally, the image pose is predicted only one step in advance, for the extrapolation beyond frame 61, the last prediction of Y was used along with the measured X value.

4.3.2 Time-based Prediction

In the second case, both the object position and the image position are modelled as functions of time X^t and Y^t. A sequential version of the autoregressive function can be learned, using the information about the data sequentiality: $y^t = \phi_s(t \mid y^{1..t-1})$. This strenghtens the influence of nearby training points, rather than treating all of them as equal.

Exploiting the causal knowledge, the X signal is shifted forward by the lag found as described in Section 4.2.4, Equation (4.6), to create $X^{t-\Delta t^*}$ (aligning the signals). Then, the relationship between the two time-aligned signals is learned to predict the future changes of Y. A (time-based) regressive function ϕ_t is defined such that

$$y^t = \phi_t(t \mid y^{1..t-1}, x^{1..t-\Delta t^*}, n^*) \ . \tag{4.10}$$

The window length n^* is used as a measure of uncertainty in the timing of X, *i.e.* how large a part of X must be taken into account during the prediction. In other words, both X and Y are modelled as related functions of time with one guiding prediction of the other in areas of insufficient data. The time-based function ϕ_t is visualised in Figure 4.9b.

Comparing the graphs in Figure 4.9, several observations can be made. The window-based method is very confident about its (inaccurate) prediction – the confidence in-

tervals cannot be seen in Figure 4.9a as they are too narrow. The time-based method predicts the signal more accurately in the extrapolated region (frames 60–90), the ground truth values are always within the 95 % confidence intervals. Additionally, the time-based prediction is able to extrapolate beyond the extent of the training inputs in both the X and Y domains (frames 90–100).

4.4 Application to Visual Tracking

4.4.1 Causality Detection

When using real data, the signals are defined as follows. Two time-series are assumed, $\mathbf{c}$ for the camera (image) and $\mathbf{p}$ for the object pose. Multivariate signals are used: x and y coordinates and size (bounding box diagonal length), but additional dimensions are possible (such as rotation). Within this work a simple approach based on feature matching and Random Sample Consensus (RANSAC) is used for camera tracking (the inter-frame shift), but a more complicated method (e.g. based on tracklets like in Flock of Trackers (FoT) [199]) may be used in challenging scenarios. For the camera, the measured quantity is the image position relative to the first image. This is expressed in pixels, and is defined by the accumulated inter-frame motion $(\Delta x^t, \Delta y^t, \Delta s^t)^\top$, *i.e.*:

$$\mathbf{c}^t = \left(\sum_{u=2}^{t} \Delta x^u, \sum_{u=2}^{t} \Delta y^u, s^1 \prod_{u=2}^{t} \frac{\Delta s^u}{s^{u-1}} \right)^\top . \tag{4.11}$$

The meaning of these camera parameters is visualised in Figure 4.10. The object pose $\mathbf{p}$ is used as given by the tracker, *i.e.* $\mathbf{p}^t = \tau(\mathbf{p}^{t-1})$ with $\mathbf{p}^1$ supplied by the user as target selection.

The complexity of the overall relationship can be significant, rendering it difficult to model. For the object coordinates $\mathbf{p}^t$, the camera motion creates instantaneous feedback, an apparent object motion in the opposite direction (as seen in the the Juice sequence of Figure 4.13). This clearly can be measured and exploited. It can be

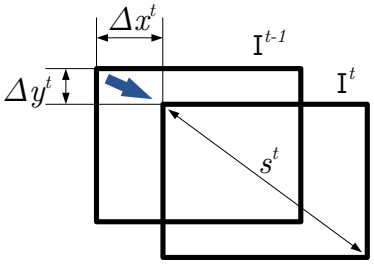


Figure 4.10: The meaning of camera parameters $\mathbf{c}^t$.

particularly valuable in the case of static objects and camera ego-motion. On the other hand, for autonomous objects (such as in GYMNASTICS), the causal relationship is often in the opposite direction: the camera follows the moving object. However, the previously noted relationship is still present, creating a feedback loop with different latencies in each direction.

To address this, the dominant relationship is isolated by testing the object position in both image ($\mathbf{p}$) and scene coordinates ($\mathbf{p}_s$, computed using the inter-frame motion, *i.e.* with the camera motion compensated). Both are independently tested for causal relationships, and the subsequent prediction is performed in the coordinate frame where the first (strongest) causal link was found. Then the information transfer from the image motion to this object motion is sought. Since it is more likely that the cameraman will move the camera based on the object motion than vice versa, the area of negative Δt is given more attention. While the other direction is theoretically possible (*e.g.* if a person was deliberately trying to escape out from the camera's view), this does not occur in any of the VOT sequences and is not evaluated here to reduce computational time. It is worth reiterating that this chapter works with predictive (apparent) causality instead of true, semantic causality.

On these signals the causality analysis is performed, using $H_{\mathbf{c}\to\mathbf{p}}$; the $\mathbf{c}$ signal takes the role of X as used in Section 4.2.2 while $\mathbf{p}$ represents Y. To obtain reliable measures of a time series, one should use multiple realisations of the random process (*i.e.* observations drawn from the distribution). However, since the causal relationships are not

consistent between sequences, only one realisation of the signal is available. Nevertheless, it is possible to use multiple samples from the same signal as realisations under the assumption the signal is stationary (*i.e.* that the distribution of the signal does not change over time). In this work a less stringent assumption is used, that the signal is *approximately locally stationary*, *i.e.* its distribution does not change significantly in a limited time window (where the properties of the signals are analysed). It is, however, allowed and expected to change in the longer term, allowing different causal relationships to be found and measured in different parts of the sequence.

For an initial coarse estimation of the signals lag, several overlapping windows with fixed length are used and TE with its statistical significance is computed for each of them in each frame, using Equation (4.4). When the statistical significance of any window exceeds a specified significance level α, a causal relationship is assumed and the optimal set of parameters is found according to Equation (4.6). If no window is significant enough, it is assumed there is currently no causal relationship between the camera and the object motion. In the experiments, $n = 4$ and $\Delta t \in \{-4, -7, -10, \dots\}$ are used, such that the immediate feedback, caused by camera motion, is covered and that there is a one-frame overlap between neighbouring windows. A conservative significance level (in literature often denoted by α) $\theta_p = 0.01\,\%$ is used.

4.4.2 Causal Predictions

The prediction was carried out as described in Sections 4.3.1 and 4.3.2. For the machine-learning stage, Gaussian Process Regression (GPR) was employed [165], as a probabilistic non-parametric regression approach, robust to overfitting. In all cases a combination of an RBF and a bias kernel was used. Since different video-sequences in general do not share their causal properties, all the predictions were made using online sequence-specific learning. In other words, a tracker is required to track successfully for some time at the beginning of the sequence and this initialisation is used to learn the properties of the sequence (this implies that early tracking failures may lead to

incorrect causal relationships, which would then be rejected as not significant). The prediction would be then supplied to the tracker as prior information and its tracking result would be added to the history data for the next prediction. The duration of the sequence history was limited to 100 frames, partially for reasons of speed, and partially to avoid the assumption of stationarity of the signal (as described in Section 4.4.1).

For the window-based autoregression ϕ_a, windows containing a short history (3 frames: $\mathbf{p}_3^t$) of the position signal $\mathbf{p}$ were taken as features to predict the position in the consecutive frame $\mathbf{p}^t$. Any other temporal information (inter-window relationships) was discarded, treating the data as a bag of equally important training inputs. The function ϕ_a was then learned and queried with the current history window $\mathbf{p}_3^t$ to obtain the prediction. The window-based causal prediction was done in a similar manner. The history windows $\mathbf{p}_{n^*}^t$ and $\mathbf{c}_{n^*}^{t-\Delta t^*}$ were concatenated together into $(3 \times 2 \times n^*)$-dimensional features ($3\times$ because of both $\mathbf{c}^t$ and $\mathbf{p}^t$ being $(x, y, s)^\top$ vectors), and the function ϕ_w was trained on the available history.

In the case of the time-based autoregressive (non-causal) prediction via ϕ_s, the independent features are simply the frame indices and the dependent features the coordinates. For the causal prediction (the ϕ_t function), a method is needed to tie the two signals together in an *a priori* unknown relationship. This can be achieved using a *coregionalisation* in the GPR. Coregionalisation is a technique which can model both signals $\mathbf{p}^t$ and $\mathbf{c}^{t-\Delta t*}$ as functions of time with a hidden relationship. Knowing the shape of one of the signals ($\mathbf{c}$) then guides the prediction of the other one ($\mathbf{p}$) even in locations distant from any training points of $\mathbf{p}$, as shown in Figure 4.9b.

Applying the prediction to a tracker is a non-trivial task. Ideally, a tracker should be designed with the causal prediction in mind as its internal motion model. In the experiments presented below, the prediction was applied to state-of-the-art trackers at the top of the recent VOT and VTB benchmarks. Incorporating a new motion model would essentially mean redesigning the trackers. For this reason, the causal prediction was used as a pre-processing step, applied to the input video. Only the mean of the

predicted distribution was used. Every following frame was shifted and scaled such that the predicted object position matched the object position reported from the previous frame, before being given to the tracker.

4.5 Experimental Evaluation of Causality Detection

Firstly, the proposed approach to detecting causal relationships is validated. This is done on both synthetic data with known Ground Truth (GT) and real data from a standard VOT benchmark dataset [115]. The evaluation of causal prediction follows in Section 4.6.

4.5.1 Synthetic Experiments

For the purpose of synthetic-data evaluation, a number of 1D *stationary autoregressive processes* (random walks with exponential fading) were generated as input signals X. The dependent signals Y were created by delaying X, applying transformations (ranging from an identity to complex non-linear transformations) and adding Gaussian noise. The transfer entropy between the real signals is compared with random signals $\bar{Y}$. For comparison, the measurements are also included for the signals with cause and effect reversed. Then the significance analysis is performed, and finally the optimal parameters for prediction are found.

See Figure 4.12 for results, showing the generated and computed signals described in Section 4.2 (the last column uses the same legend as Figure 4.7) with the different transformations. In all cases, TE of the correct signal is consistently much higher than that of the other signals. The correct signal is also the only statistically significant relationship. In all the cases, the true parameters ($\Delta t^* = 4$ and $n^* = 1$) were revealed by the method. The results for the derivative signal are different, since at least two frames are needed for its estimation (and therefore $\Delta t^* = 3$ and $n^* = 2$). Since the integral transformation uses information from a longer time interval (in fact from

Sequence	Length	Length ratio	Δt^*	n^*
Bicycle	271	81.2 %	-3	7
Bolt	350	64.3 %	-1	2
Car	374	64.7 %	-10, -14	7, 5
Cup	303	53.5 %	-3, -3	4, 8
David	770	94.9 %	-2, -1	2, 1
Diving	231	40.3 %	-11	8
Face	415	91.3 %	-1	1
Gymnastics	207	81.6 %	-1	1
Hand	244	0.0 %	NA	NA
Iceskater	500	92.4 %	-11, -2	3, 7
Juice	404	90.6 %	-1	1
Jump	228	0.0 %	NA	NA
Singer	351	82.3 %	-17, -13, -11	6, 1, 1
Sunshade	172	59.3 %	-8	8
Torus	264	0.0 %	NA	NA
Woman	597	49.6 %	-8, -3, -3	5, 8, 1
Average	355	59.1 %		

Table 4.1: Causal detections on the VOT2013 dataset – detected durations and properties.

the whole history), there is a noticeable transport of information even in the reversed direction.

The results clearly show, that transfer entropy is an excellent approach for analysing time series with complex causal relationships. While transfer entropy of the correct relationship gains statistical significance after only a small number of frames, the significance of the reversed causal relationship stays low for the duration of the sequences. Performance under even highly non-linear transformations shows the value of the information-theoretic approach over simpler ones (such as Granger causality [65]).

4.5.2 Real-Data Experiments

For further evaluation, the ICCV VOT Challenge 2013 [115] dataset was used (16 sequences, each containing between 172 and 770 frames). See Table 4.1 for the results. In the third column, the fraction of the sequence marked as containing a significant

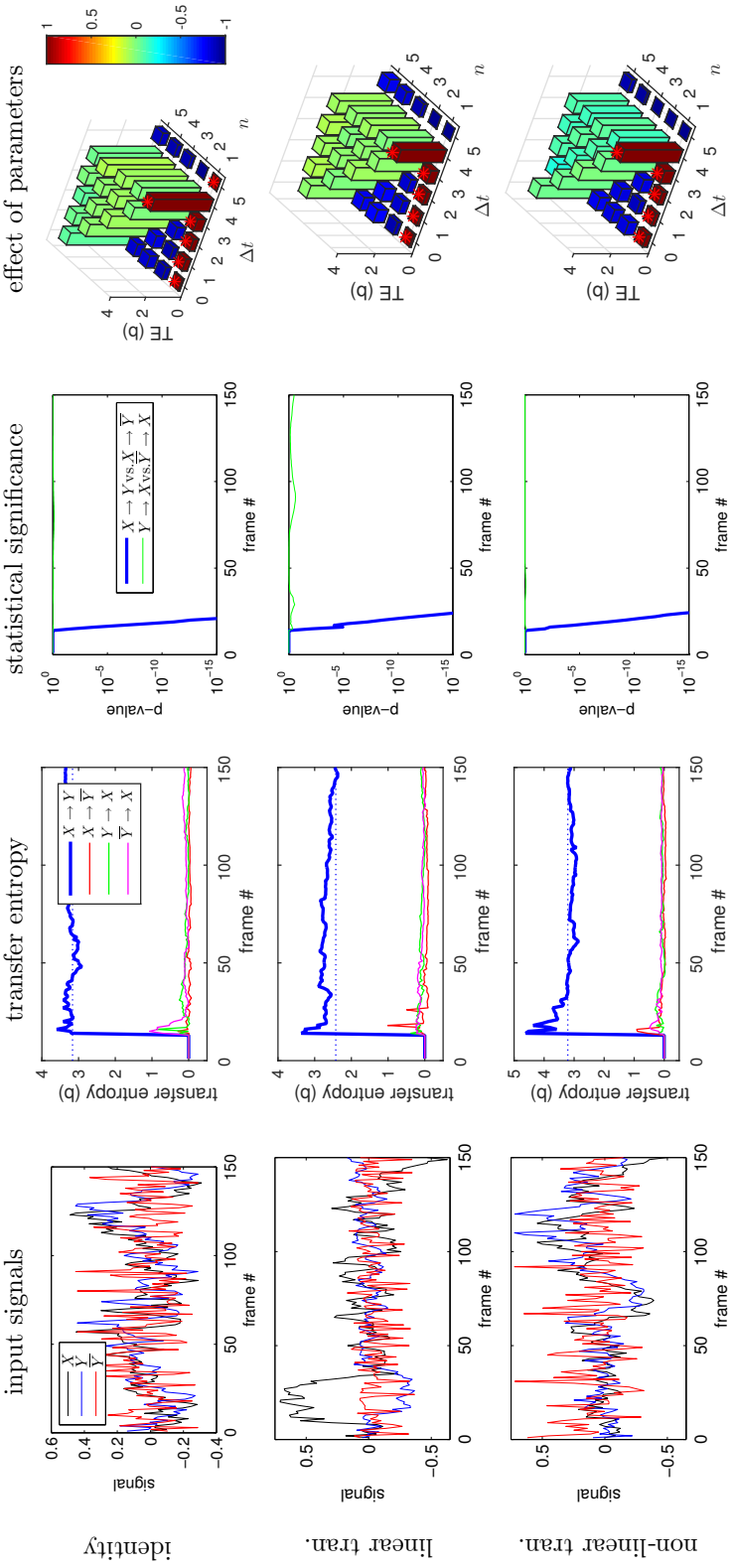


Figure 4.11: Results on synthetic data.

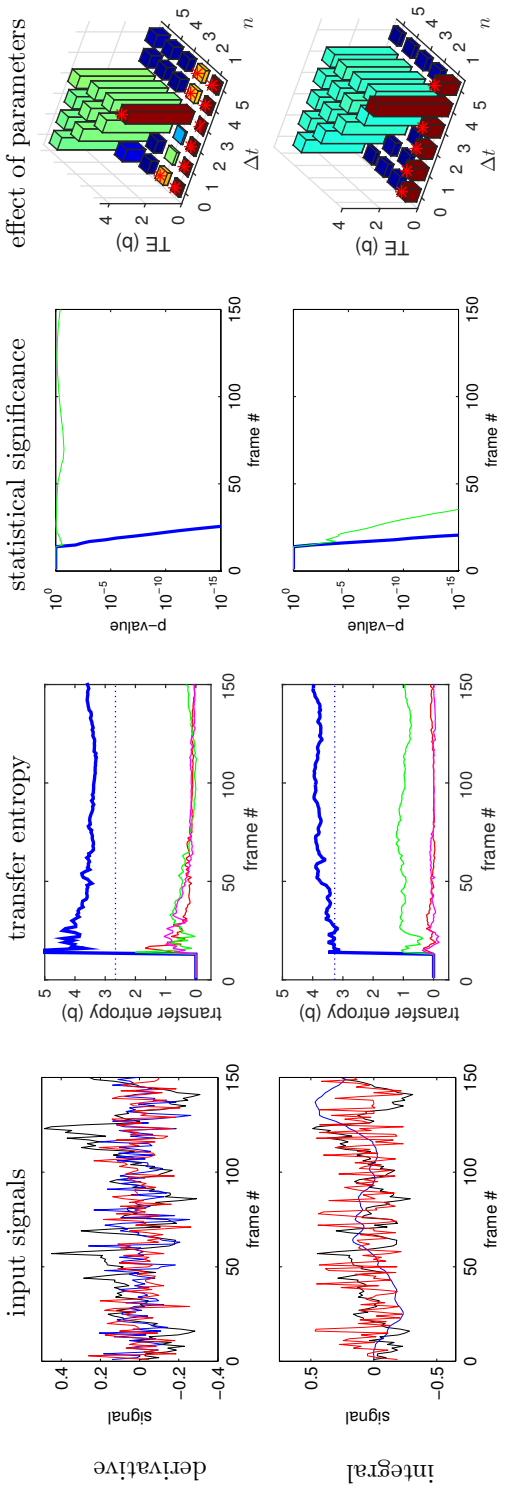


Figure 4.12: Results on synthetic data (continued). A ground truth delay of 5 frames ($\Delta t = 4$ for $n = 1$) is used. The dotted lines in the second column denote approximate ground truth for $H_{X \to Y}$ (this approximation would be exact if the X and Y signals were normally distributed).

causal relationship is shown. Then the optimal parameters for prediction are shown; in the case of different relationships for different time periods there are multiple parameter sets (*e.g.* 1 for BICYCLE, none for HAND or 3 for SINGER).

There are no "ground-truth causal relationships" that could be used to measure the quality of detection on the sequences (with the exception of zero relationship in the case of a static camera). However, the detected relationships are consistent with intuitive understanding of the scene dynamics and the optimal prediction parameters fulfill human expectations. This shows that it is possible to use information-theory based measures to discover and quantify relationships between signals in real sequences for the task of visual object tracking.

In the case of the sequence JUMP, none of the detected causal relationships were statistically significant. However, this is not necessarily an error as although the camera is not static, it is not known if a true relationship exists between object and camera motion. There are three sequences with a static camera (constant zero $\mathbf{c}$) and therefore no causal relationships, these are marked in grey in the tables. For two of them, this was correctly detected using TE. For the CAR sequence, a causal relationship was incorrectly discovered due to inaccurate estimation of $\mathbf{c}$. However, this means that causality detection only failed in 1 out of 16 sequences (and this failure had a negligible effect on the tracking results). It is also worth pointing out the relatively common occurrence of the $(-1, 1)$ pair, indicating the immediate causal effect of a moving camera on the apparent motion of a static object.

See Figure 4.13 for detailed results on selected sequences. The same measures as in the previous section were taken. In the first row, the static object and ego-motion (see the relation between $\mathbf{c}$ and $\mathbf{p}$) JUICE sequence is shown. Naturally, the feedback between the camera motion and apparent object motion (in image coordinates) is immediate, identified within the $(\Delta t = -4, n = 4)$ window in the second and third column and refined (to $\Delta t = -1, n = 1$) in the search of the 4th column. In the parameter set optimisation, the expected triangular shape of the high-TE area can

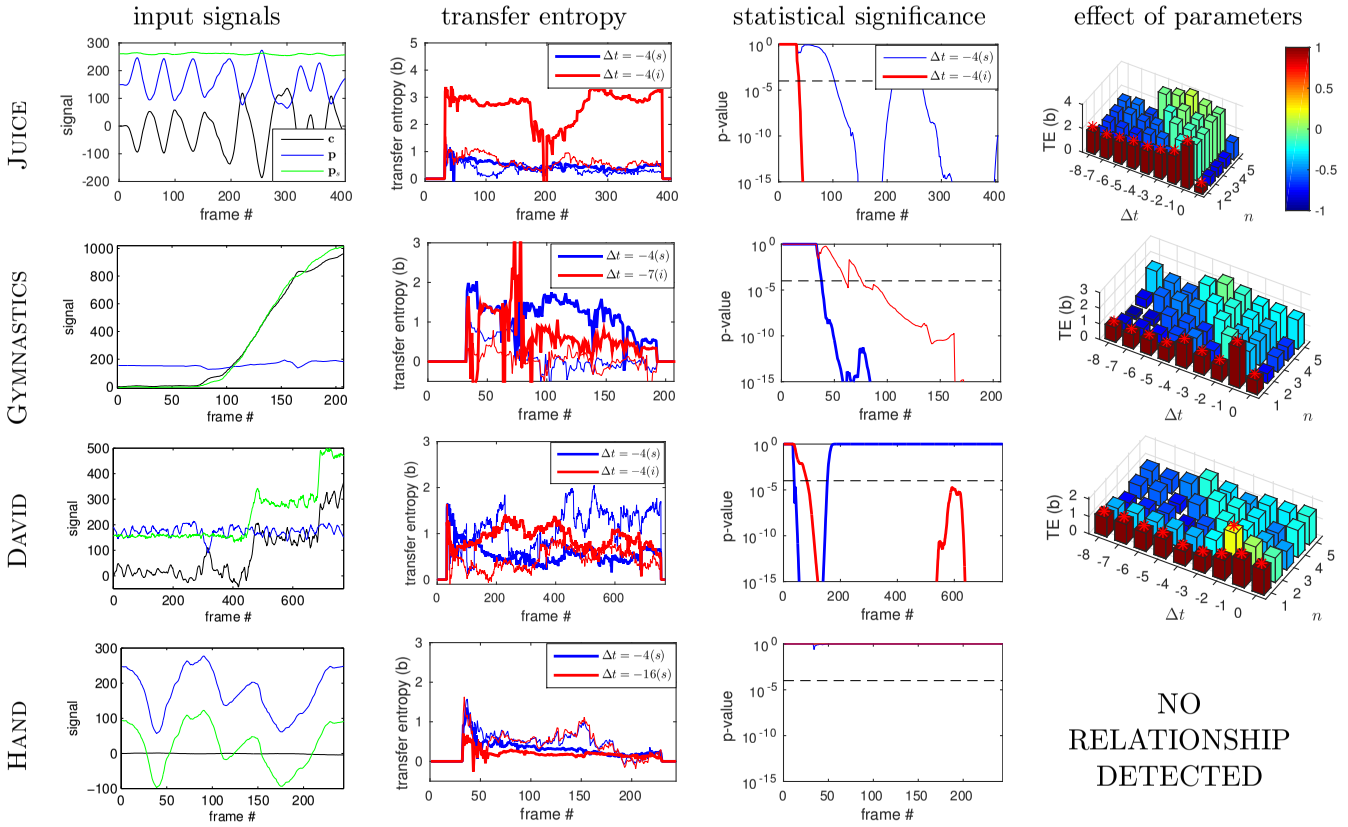


Figure 4.13: Results on real data. The input signals $\mathbf{c}$, $\mathbf{p}$ and $\mathbf{p}_s$ are the image position and the object position in the image and scene coordinate systems, respectively (only x-coordinate shown). TE and statistical significance using a range of lags are shown: (i) and (s) for image and scene coordinates, respectively. TE from the input signal Y is denoted by the bold lines while the shuffled ($\bar{Y}$) variants are denoted by the thin lines. For clarity, only x-signals in the case of the inputs, and only a small number of lags in the case of entropy and significance, are shown. Notice the different TE levels of the random signals, based on different input data. See the text for discussion.

be seen and the correct window length and lag were recovered. The signal containing the most relevant lag has significantly higher TE and reaches the statistical significance after several frames. However, while the other signals are not optimal, they still achieve statistical significance, as they contain relevant information.

The GYMNASTICS sequence is characterised by its strong *centre bias* (object being tracked by the cameraman, similarly to Figure 4.2). This is visible from the strong link between $\mathbf{c}$ and $\mathbf{p}_s$. Here the TE of the $\mathbf{p}$ signal is lower than of $\mathbf{p}_s$, and significantly noisier, due to nearly constant levels of $\mathbf{p}$. The optimal parameter set is, however, identical to JUICE: $\Delta t = -1$ and $n = 1$ (although in the scene coordinates, where the camera motion has been compensated for).

The DAVID scene shares properties of both the previous sequences. Both $\mathbf{p}$ and $\mathbf{p}_s$ signals contain recognisable elements of $\mathbf{c}$, each in a different manner. Intuitively, this can be related to $\mathbf{c}$ sharing high-frequency components with $\mathbf{p}$ (image movement induced by camera shakes) and lower frequencies with $\mathbf{p}_s$ (the cameraman follows the general motion of the target). Indeed, both show significant levels of TE, however in different periods of the sequence. At the very beginning of the sequence, the TE of $\mathbf{p}_s$ gains sufficient statistical significance and therefore prediction is started in the scene coordinates, with optimal parameters $\Delta t = -2$ and $n = 2$ (a slightly delayed and less consistent relationship). After the significance of this signal drops (at frame #155), the prediction is stopped and the optimal parameter set is re-estimated, using the window with the most significant TE (denoted in red in the second and third column). In the case none exceeds θ_p, it would be assumed there is no relationship. By constantly monitoring TE, this allows changes of the causal relationships during the sequence. In this particular case, the prediction continues as an immediate image-coordinates feedback from camera shake, *i.e.* using the $\mathbf{p}$ signal with $\Delta t = -1$ and $n = 1$.

Finally, the HAND sequence was recorded using a stationary camera. This is obvious from the constant signal $\mathbf{c}$. Therefore TE for both $\mathbf{p}$ and $\mathbf{p}_s$ is negligible and purely caused by noise and as such it is not statistically significant. In such a case, it is

impossible to exploit causality to help the tracker using the (non-existent) camera motion, the approach correctly detects this and no prediction takes place.

Summarised, all these results show that it is possible to use information-theory based measures to discover and quantify relationships between signals in real sequences in the task of visual object tracking. Using these, one can measure if there is a causal relationship between a camera and an object in a video-sequence, in which parts of the sequence, and then measure its properties. Additionally, variations of these properties in time can be detected and the prediction parameters consequently updated. The final question that remains is, can detected causal relationships be employed to increase tracking robustness?

4.6 Experimental Evaluation of Causal Predictions

In this section, the performance of causal prediction and its effect on tracking results is evaluated. The prediction is computed and supplied to the tracker as detailed in Section 4.4.2. Figure 4.14 shows an example of such a prediction on synthetic data (with the linear transform). The knowledge of the $\mathbf{c}$ signal, allowed by the discovered causal relationship, gives the predictor ϕ_t better accuracy and lower variance compared to ϕ_s.

4.6.1 Qualitative Prediction Evaluation

The task given during the prediction stage is to estimate the distribution of the possible object positions to be supplied to the tracker as a prior. The performance measure should demonstrate how well the GT position is represented by this distribution. While simple distance between the distribution mean and the GT indicates the prediction accuracy, it does not take uncertainty into account. In particular, if two predictors predict the same correct position, the one with high confidence is of most benefit to the tracker. The opposite is true as well, for a misprediction it is better to report

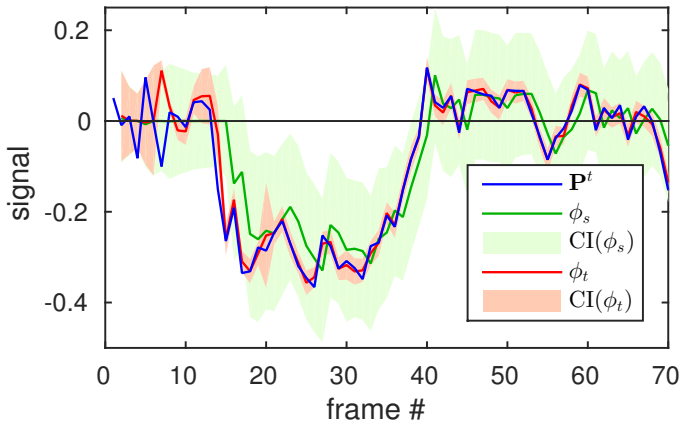
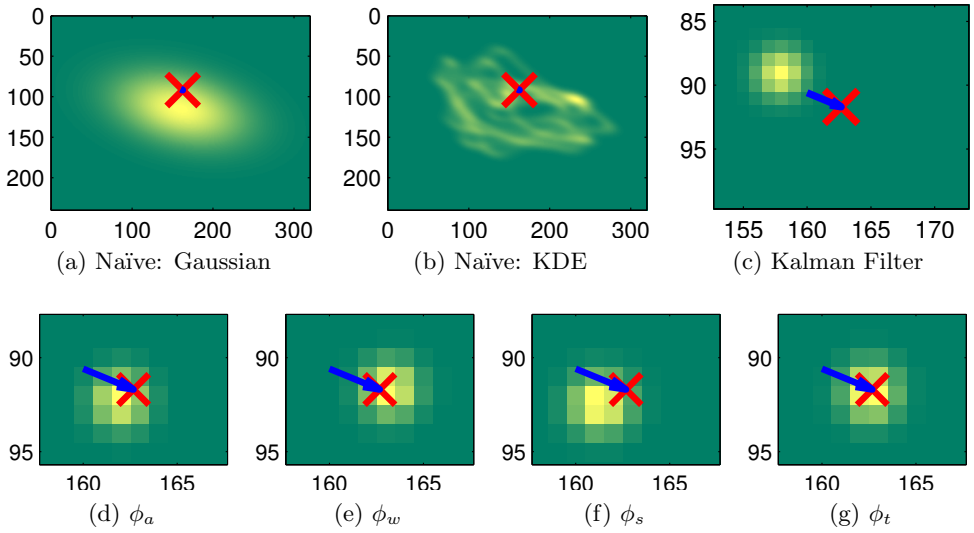


Figure 4.14: Example of the time-based prediction on synthetic data. Notice how ϕ_t predicts $\mathbf{p}$ more accurately than ϕ_s. The narrower 95 % confidence interval CI(ϕ_t) also indicates a lower variance for the ϕ_t predictor.



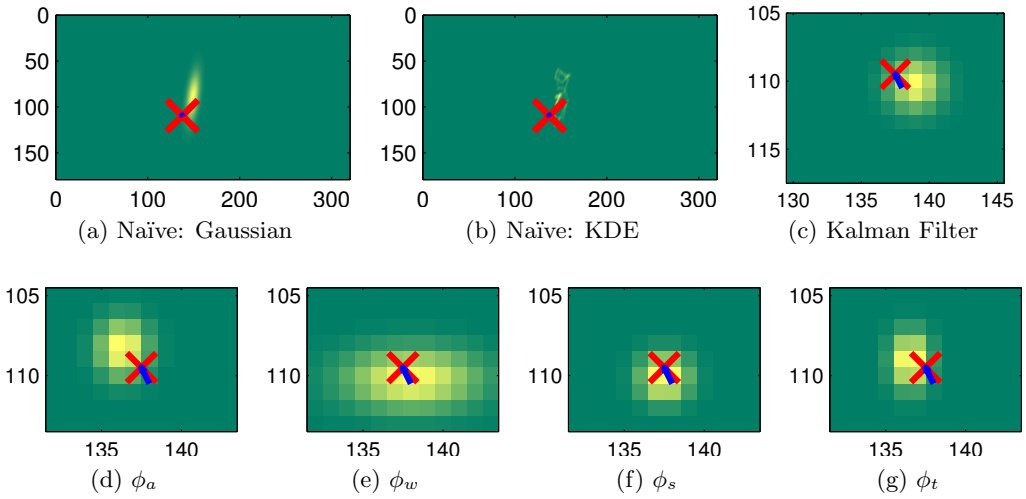
(a) Naïve: Gaussian (b) Naïve: KDE (c) Kalman Filter

(d) ϕ_a (e) ϕ_w (f) ϕ_s (g) ϕ_t

Figure 4.15: Qualitative prediction results on the JUICE sequence, frame #400. Predicted distributions shown with the ground truth position (in red) and ground truth inter-frame shift (blue) overlaid. See the text for discussion.

lower certainty. For qualitative evaluation, the probability densities of the predicted distributions were compared on selected examples from the VOT 2013 dataset.

The causality-based prediction was compared with several alternative approaches as follows. Firstly, two naïve approaches are examined, treating all historical states as



(a) Naïve: Gaussian (b) Naïve: KDE (c) Kalman Filter

(d) ϕ_a (e) ϕ_w (f) ϕ_s (g) ϕ_t

Figure 4.16: Qualitative prediction results on the DIVING sequence, frame #200.

equally important, based on the assumption that the object stays in an approximately stable location. One models the distribution as a Gaussian, as used in the re-detector of the LT-FLO tracker (Chapter 3), while the other one uses a Kernel Density Estimation (KDE) to model the probability density with higher accuracy. These can be seen as implementations of the central bias and stable location priors. Visual tracking algorithms often use a Kalman Filter (KF), or its extension, as their internal motion model [45, 101, 107, 212]. Therefore, the KF is a natural alternative to causal prediction. Finally, the autoregressive functions ϕ_a and ϕ_s and the causal predictions ϕ_w and ϕ_t (as defined in Section 4.3) are evaluated.

The results are visualised in Figures 4.15 and 4.16. In both cases, the GT is inside the naïvely predicted distributions. However, these distributions are spread over large portions of the image and therefore the probability density is relatively low at the correct position. In the case of the KF, knowledge of recent poses leads to a tighter distribution, however the predictions lag behind the true signal somewhat, causing mispredictions. The autoregression given by ϕ_a and ϕ_s helps significantly with much tighter distributions; ϕ_s shows better performance than ϕ_a, particularly evident in Figure 4.16. Window-based causal prediction ϕ_w gives accurate modes for the distribution, although

the long window in the case of DIVING ($n^* = 8$) results in a low confidence and thus a lower density. The time-based method performed the best of all the tested predictors; ϕ_t predicted positions close to the GT while having an appropriate confidence.

4.6.2 Quantitative Prediction Evaluation

Similarly to the qualitative evaluation, the probability density is used as a performance measure in the experiments. Additionally, the error is integrated over the entire probability density support region to obtain the expectation for the prediction error. The mean values of both across each sequence are used as quantitative measures. These two error types are mutually complementary: while the mean probability density says how likely a prediction is to be wrong, the error expectation shows how wrong a wrong prediction can be.

They are similar to other complementary error measures used within this work. For instance, in the VOT measures, the robustness (similarly to the mean probability density) indicates the probability of failure and the accuracy the average quality of a returned solution. Similarly, the often used precision and recall measures show both how likely one is to get a good answer (recall or probability density) and how good an average answer is (precision or error expectation).

There are periods in the sequences, where no causal relationship was detected, and therefore no prediction parameters exist. In such places, the causal prediction is replaced by the appropriate autoregressive function: ϕ_w by ϕ_a and ϕ_t by ϕ_s. This explains the identical results for the sequences without causality during quantitative experiments in Section 4.6.2 (HAND, JUMP and TORUS).

See Table 4.2 for the results. In the CAR sequence, the tracked car stops for a large part of the sequence in one location, the KDE predicted very high probability for this location, which is reflected by a very high mean density. A similar phenomenon can be found in GYMNASTICS, with the tracked person standing in one place for part of the sequence. In the BOLT and ICESKATER sequences, the $\mathbf{c}$ signal estimation failed for

	Expectation of error (px)							Mean probability density (-)						
Sequence	**Gaussian**	**KDE**	**KF**	ϕ_a	ϕ_w	ϕ_s	ϕ_t	**Gaussian**	**KDE**	**KF**	ϕ_s	ϕ_w	ϕ_s	ϕ_t
Bicycle	22.6	22.8	3.8	3.4	19.3	3.2	**3.1**	0.001	0.002	0.025	0.028	0.010	0.031	**0.036**
Bolt	64.1	64.7	**3.5**	3.9	63.5	4.2	6.8	0.001	0.004	**0.030**	0.022	0.013	0.020	0.015
Car	52.2	64.6	2.3	1.5	1.5	1.4	**1.0**	0.006	**0.577**	0.060	0.123	0.124	0.130	0.524
Cup	22.5	22.8	3.5	1.8	2.2	1.7	**1.5**	0.012	0.015	0.022	0.086	0.065	0.093	**0.121**
David	25.4	25.3	**5.2**	5.8	5.5	5.4	5.3	0.001	0.001	**0.013**	0.009	0.009	0.010	0.011
Diving	18.7	19.1	2.5	1.8	1.9	1.8	**1.7**	0.007	0.009	0.040	0.085	0.081	0.089	**0.096**
Face	16.3	4.6	2.0	1.5	1.6	1.4	**1.3**	0.002	0.013	0.068	0.118	0.108	0.124	**0.206**
Gymnastics	18.2	22.6	4.2	3.5	12.5	**2.5**	3.7	0.021	**0.193**	0.044	0.069	0.045	0.070	0.109
Hand	81.0	80.6	9.0	**5.6**	**5.6**	6.1	6.1	0.002	0.002	0.001	**0.010**	**0.010**	**0.010**	**0.010**
Iceskater	31.2	31.1	3.9	**2.5**	648.4	2.7	14.3	0.001	0.001	0.022	**0.061**	0.006	0.053	0.008
Juice	71.7	72.0	10.7	3.1	2.1	2.5	**2.0**	0.006	0.009	0.001	0.039	0.083	0.045	**0.088**
Jump	40.2	38.6	2.5	**1.7**	**1.7**	1.8	1.8	0.001	0.003	0.041	**0.102**	**0.102**	0.096	0.096
Singer	77.2	69.8	**3.2**	3.6	20.1	3.6	5.4	0.000	0.001	**0.030**	0.022	0.010	0.021	0.014
Sunshade	56.4	56.3	10.1	9.8	41.4	4.5	**3.9**	0.000	0.001	0.003	0.012	0.005	0.018	**0.021**
Torus	63.2	62.7	6.0	3.3	3.3	**3.1**	**3.1**	0.004	0.003	0.007	**0.031**	**0.031**	0.026	0.026
Woman	34.8	35.2	3.9	3.4	6.3	3.0	**2.9**	0.000	0.001	0.027	0.059	0.046	0.060	**0.069**
Average	43.5	43.3	4.8	3.5	52.3	**3.1**	4.0	0.004	0.052	0.027	0.055	0.047	0.056	**0.091**

Table 4.2: Quantitative results of the prediction on the VOT2013 dataset. The best and second best results are denoted by a bold typeface and underlining respectively (separately for error expectation and mean probability density; multiple columns highlighted in cases of equal values).

one region of each sequence due to very low texture of the background. This renders the relationship between the camera and object motions unstable and therefore the ϕ_w and ϕ_t predictors have lower performance in these sequences. This is more noticeable in the case of error expectation, where these outliers mean the ϕ_t prediction does not have the lowest average error, despite being the lowest on the majority of sequences.

In general, disregarding these outliers, several statements can be made about the performance of the compared predictors. Both global probability distributions have an image-wide spread and therefore a very low density. Prediction using KF is better localised and has therefore significantly better performance, although still worse than the learned regressive functions. For the learned functions, the time-based ones (ϕ_s and ϕ_t) in general perform better than the window-based ϕ_a and ϕ_w. This is attributed to the lower dimensionality of the signals in time-based prediction and the usage of the ordering information. Regressive function ϕ_w performs slightly worse than its non-causal counterpart ϕ_a, due to the lower confidence of the prediction (higher variance and therefore lower probability density). The time-based causal function ϕ_t was shown as the best predictor, beating the second best by a large margin (62 %). In addition, it performs more than three times better than a KF, which is a commonly used motion model.

4.6.3 Effect on Tracking Results

In the previous section, it was demonstrated that causal prediction provides a better estimate of the future target pose than common motion models such as autoregression or Kalman Filter. This can be incorporated in a tracker to improve its tracking performance. This section shows how two state-of-the-art trackers Flock of Trackers (FoT) [199] and Structured Output Tracking with Kernels (Struck) [76] can benefit from such a prediction. These were chosen as the highest scoring methods in the VOT 2013 and VTB 1.1 benchmarks, respectively. Additionally, the effect on Long-Term Feature-Less Object Tracker (LT-FLOTrack) presented in the previous chapter

is examined. In all cases, the tabulated values are the VOT accuracy/robustness metrics [115] – mean bounding box overlap (higher is better) and number of failures per sequence (lower is better).

Table 4.3 shows the effect of causal prediction on tracking performance on the VOT 2013 dataset. The causal prediction is compared against vanilla trackers and a simple Background Motion Compensation (BMC), using the image context but no temporal causal relationships (this was incorporated the same way as the causal prediction, as described in Section 4.4.2). While the simple camera motion information does not prove useful, supplying the tracker prior information from causality-based prediction improves its performance significantly. In general, robustness is affected only slightly, while the main improvement is in the accuracy domain. For FoT on the ICESKATER sequence, there is a marginal drop in accuracy, which is more than balanced by a dramatic increase in robustness, lowering the number of failures by 60 %. For comparison, the same experiments were carried out with the Zero-order Tracker (as described in Section 4.1). While it works in some cases, the mean performance is significantly poorer: accuracy of 0.34 and robustness 6.25.

Finally, in Table 4.4, the results of the Long-Term Feature-Less Object Tracker (LT-FLO) (as introduced in the previous chapter) are shown. In this case, the causality-based motion model does not improve the tracking results: there is no significant difference between the first and third column. This is probably due to the inherent robustness of LT-FLO to the issues the causal prediction aims to prevent (such as camera shake). Such robustness is likely caused by the larger basin of convergence, which is (automatically) chosen for edge searching instead of being limited by the image content which is the case for local optimisation approaches.

Additionally, further experiments were carried out on the much larger Visual Tracker Benchmark (VTB) 1.1 [207].[2] The Struck tracker is at the head of the leader-board.

[2]Background Clutter (BC), Deformation (DEF), Fast Motion (FM), In-Plane Rotation (IPR), Illumination Variation (IV), Low Resolution (LR), Motion Blur (MB), Occlusion (OCC), Out-of-Plane Rotation (OPR), Out-of-View (OV), Scale Variation (SV).

Sequence	FoT	FoT_{BMC}	FoT_{ϕ_t}	Struck	$\text{Struck}_{\text{BMC}}$	Struck_{ϕ_t}
Bicycle	0.70/ **1**	0.70/ **1**	**0.71**/ **1**	0.43/0.3	0.39/0.2	**0.54**/**0.0**
Bolt	0.46/14	**0.59**/**13**	0.52/**13**	**0.76**/**3.7**	0.58/8.5	0.72/5.4
Car	0.55/ **1**	0.53/ **1**	**0.59**/ **1**	0.40/**0.0**	**0.42**/**0.0**	0.38/**0.0**
Cup	0.81/ **0**	0.80/ **0**	**0.82**/ **0**	0.78/**0.0**	**0.83**/**0.0**	0.82/**0.0**
David	**0.76**/ **0**	0.59/ **0**	0.75/ **0**	0.67/0.7	0.60/**0.5**	**0.70**/0.9
Diving	0.25/ 5	**0.32**/ **3**	0.25/ 5	**0.39**/**1.0**	0.36/**1.0**	0.36/**1.0**
Face	0.74/ **0**	**0.84**/ **0**	0.78/ 1	**0.83**/**0.0**	0.80/**0.0**	**0.83**/**0.0**
Gymnastics	**0.63**/ 6	0.60/ **4**	0.61/ 6	0.55/**2.3**	**0.59**/3.9	0.56/4.0
Hand	**0.40**/ 4	0.38/ **3**	0.38/ 4	**0.52**/**4.1**	**0.52**/4.6	**0.52**/**4.1**
Iceskater	0.43/10	**0.45**/10	0.38/ **4**	**0.62**/**0.0**	0.32/9.4	0.54/0.7
Juice	0.88/ **0**	**0.93**/ **0**	0.90/ **0**	0.65/**0.0**	**0.91**/**0.0**	0.89/**0.0**
Jump	0.62/ 1	0.71/ **0**	**0.72**/ **0**	0.56/**0.0**	**0.57**/**0.0**	**0.57**/**0.0**
Singer	**0.74**/ **0**	0.65/ **0**	**0.74**/ **0**	0.30/**0.0**	**0.41**/1.0	0.33/**0.0**
Sunshade	0.59/ 2	0.57/ **1**	**0.76**/ 2	**0.77**/**0.0**	**0.77**/**0.0**	0.74/**0.0**
Torus	0.73/ **0**	**0.75**/ 1	0.72/ **0**	0.49/**4.3**	0.55/5.2	**0.56**/5.1
Woman	0.61/ **0**	0.12/ 1	**0.71**/ 5	**0.75**/**0.0**	0.65/**0.0**	0.74/**0.0**
Average	0.62/2.8	0.60/**2.4**	**0.65**/2.6	0.59/**1.0**	0.58/2.1	**0.61**/1.3

Table 4.3: Tracking results on the VOT2013 benchmark. The best and second best results are highlighted separately for the FoT/Struck families of trackers and for accuracy/robustness.

As shown in Table 4.5, using the causal predictions further improves this — by more than the current difference between the first two trackers — leading to a new state-of-the-art on this benchmark. However, it is worth noting that a new state-of-the-art tracker was not the objective of this work, but rather to demonstrate that causal prediction can be used to assist an existing tracker.

4.7 Tracking Through Occlusion

The relationships between the object and camera motion can help tracking when they are related. The target may interact with (*e.g.* get occluded by) other objects in the scene, which can pose a challenge for a tracker. However, other elements of the scene can also help tracking – it is possible to use the scene context to infer the object location even in cases of full occlusion. This section extends the previous discussion of relationships in visual tracking beyond the scope of the camera motion.

Sequence	LT-FLO	LT-FLO$_{\text{BMC}}$	LT-FLO$_{\phi_t}$
Bicycle	0.62/1.6	**0.64**/ 3.8	0.54/**0.9**
Bolt	**0.49**/4.7	0.13/25.9	0.47/**4.5**
Car	0.44/1.8	**0.48**/ 1.3	0.42/**0.7**
Cup	**0.87**/**0.0**	0.86/ **0.0**	**0.87**/**0.0**
David	**0.73**/0.2	0.64/ 0.9	**0.73**/**0.0**
Diving	0.38/1.8	0.34/ 3.5	**0.39**/**1.7**
Face	**0.83**/**0.0**	0.82/ **0.0**	**0.83**/**0.0**
Gymnastics	**0.54**/**1.3**	0.46/ 9.3	0.53/1.5
Hand	0.45/**4.1**	**0.46**/ 4.5	0.45/**4.1**
Iceskater	0.38/**1.5**	0.26/24.3	**0.44**/3.8
Juice	0.88/**0.0**	**0.89**/ **0.0**	0.88/**0.0**
Jump	**0.59**/**0.1**	0.52/ 1.1	0.52/0.3
Singer	0.67/0.2	0.41/ 0.5	**0.69**/**0.1**
Sunshade	0.67/0.7	**0.70**/ 2.1	0.68/**0.1**
Torus	**0.64**/**1.3**	0.61/ 1.7	0.59/1.6
Woman	**0.54**/5.4	0.38/10.3	0.50/**5.1**
Average	**0.61**/**1.5**	0.54/ 5.6	0.59/**1.5**

Table 4.4: Tracking results on the VOT2013 benchmark with LT-FLOTrack.

Cat.	ASLA[98]	SCM[216]	Struck[76]	Struck	Struck$_{\text{BMC}}$	Struck$_{\phi_t}$
BC	0.59/3.0	0.61/2.9	0.59/3.3	0.60/1.9	0.55/1.9	**0.61**/**1.7**
DEF	0.51/4.5	0.52/4.8	0.52/4.6	0.55/2.4	0.55/2.6	**0.60**/**2.2**
FM	0.42/6.5	0.43/6.5	0.56/3.8	0.53/3.2	0.51/3.3	**0.57**/**2.5**
IPR	0.52/4.1	0.52/4.3	0.57/3.4	0.53/2.6	0.50/3.0	**0.55**/**2.0**
IV	0.60/3.0	0.61/3.1	0.59/3.3	0.58/2.1	0.51/1.9	**0.60**/**1.6**
LR	0.59/2.3	0.62/2.5	0.59/3.9	0.51/1.4	0.48/1.1	**0.56**/**1.0**
MB	0.45/5.9	0.45/5.9	0.60/2.8	0.53/3.0	0.51/2.9	**0.56**/**2.0**
OCC	0.56/3.8	0.57/3.8	0.56/4.1	0.55/2.5	0.54/2.9	**0.59**/**2.0**
OPR	0.56/3.7	0.57/3.8	0.57/3.7	0.55/2.3	0.54/2.7	**0.59**/**1.9**
OV	0.55/4.3	0.56/4.5	0.59/3.4	0.55/3.0	**0.58**/2.7	0.55/**2.5**
SV	0.54/3.9	0.56/3.9	0.58/3.6	0.52/2.3	0.49/2.7	**0.57**/**1.9**
All	0.53/4.1	0.54/4.1	0.57/3.6	0.55/2.4	0.51/2.6	**0.59**/**1.9**

Table 4.5: Tracking results on VTB1.1. Results in the last three columns were obtained using the VOT evaluation criteria, using the VTB criteria would improve the accuracy even further.

Thus far, the focus of this chapter was on the relationship between the motion of the camera and the tracked object ($H_{\mathbf{c}\to\mathbf{p}}$). In this section, this focus is changed slightly, exploring the relationships between multiple elements of the scene: $H_{\mathbf{p}'\to\mathbf{p}}$. The most

Figure 4.17: Where is the mug? Causality-based prediction indicates its position even after the person turned away and completely occluded it.

notable example of this idea in the literature is the context tracker of Grabner *et al.* [64]. This uses features from across the image as *supporters*, helping to predict the object position in cases of full occlusion. In this section, the proposed causality-based approach is used to achieve a similar result: to recognise which parts of the image are moving with some relationship to the tracked object, and to use them for motion prediction. Similarly Sivic *et al.* [175] used affine factorisation to group together parts belonging to an object. However, unlike these simpler methods, the presented approach allows complex non-linear relationships to be used. See Figure 4.17 for an example of prediction for an object under full occlusion.

To achieve this, windows $\mathbf{p}'$ are sampled on a regular grid across the whole video frame and each of them is tracked independently using FoT [199]. The causal relationships to the target object $\mathbf{p}$ can be inspected: whether the null hypothesis $\mathrm{E}(H_{\mathbf{p}'\to\mathbf{p}}) \leq \mathrm{E}(H_{\mathbf{p}'\to\bar{\mathbf{p}}})$ can be disproved (using the Welch's test as described in Section 4.2.3). The results are visualised in the upper-left corners of Figures 4.18 and 4.19. Windows which were tracked successfully but had no causal relationship are denoted in red (windows unable to be tracked were not used and are not shown). Windows with significant causal relationship detected and with an overlap with the tracked object are denoted in yellow. Windows, which are not overlapping the object, but are causally related to it, are considered supporters and are denoted in green. Once this set of supporters is found, object pose can be predicted from them:

$$\mathbf{p}^t = \underset{\mathbf{p}'}{\mathrm{median}} \left(\phi(\mathbf{p}'^{t-\Delta t}_n) \right) \tag{4.12}$$

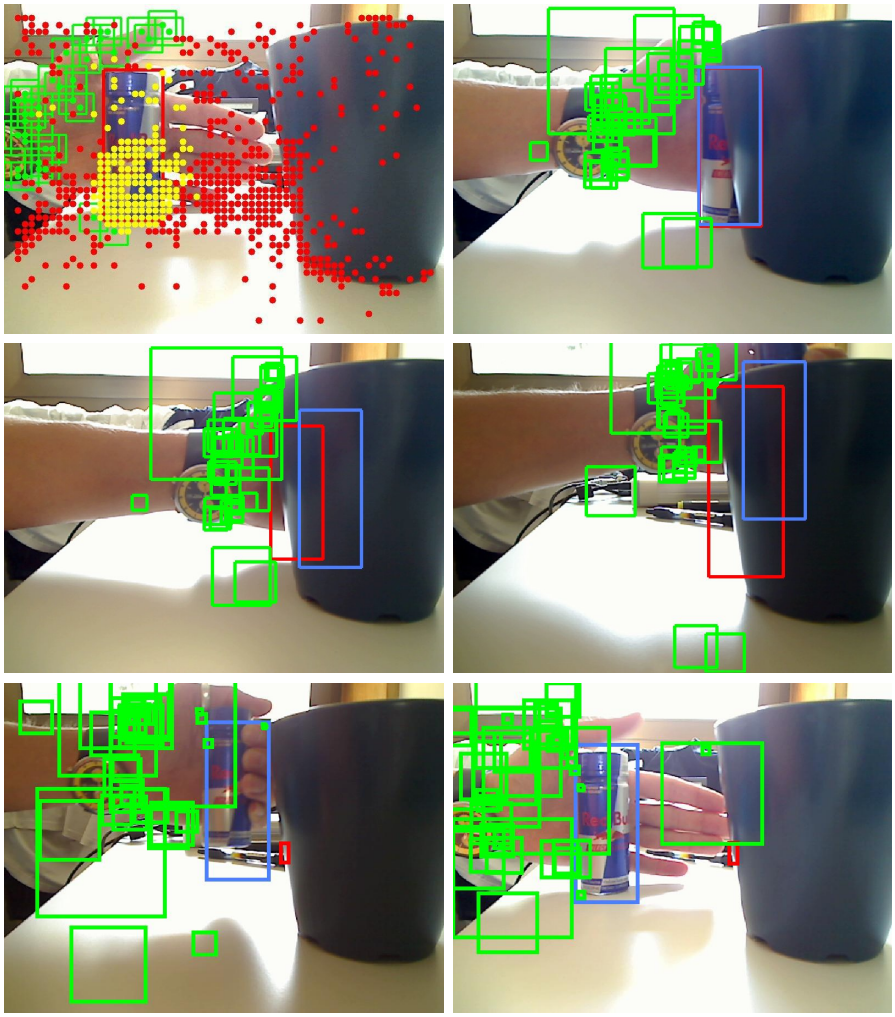


Figure 4.18: Robust prediction during tracking failure on the ETH-Cup sequence of Grabner *et al.* [64].

where ϕ is the causal prediction function (learned using Gaussian Process Regression (GPR)). Inconsistency between the predictions and the tracked location of the target using a standard tracker may be used to indicate failure of the object tracker, *e.g.* due to occlusion. At this point, the causal prediction can take over.

See Figures 4.18 and 4.19 for the results. The upper left images show the first frames with windows to eventually become supporters in green. The upper right images visualise frames where an inconsistency between tracking and prediction was detected. The remaining rows show selected frames after causal prediction takes over. During

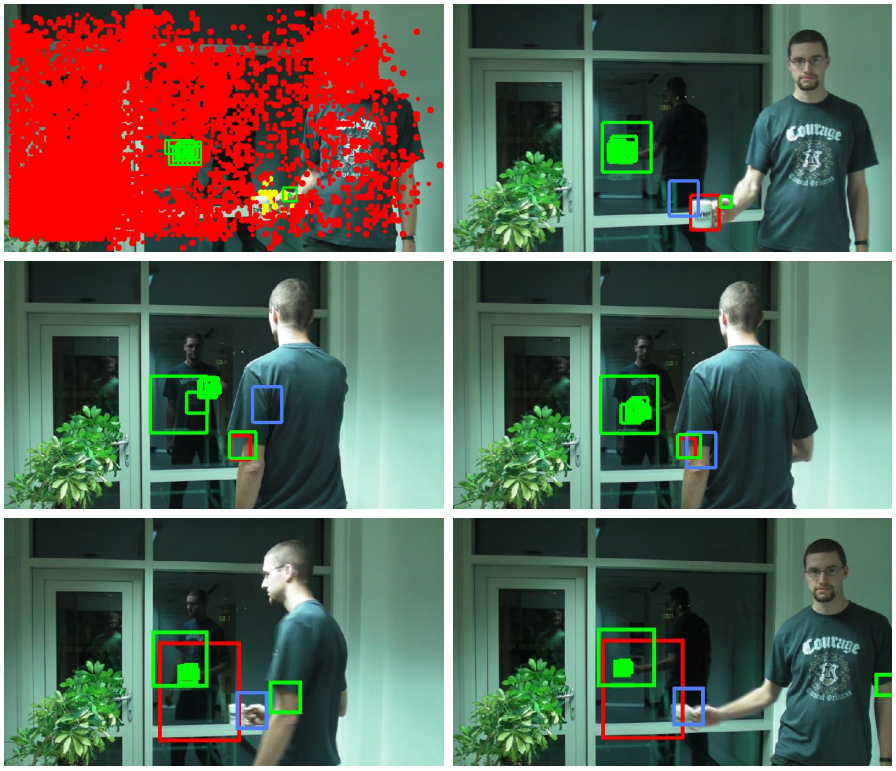


Figure 4.19: Robust prediction during tracking failure on the CVMPMug sequence.

successful tracking, the tracked (red) and causally predicted (blue) trajectories are almost identical. However, after the occlusion of the object, whilst the causal prediction still correctly indicated its position, the tracker has failed.

The results are similar to those of Grabner *et al.* on the ETH-Cup sequence. However, the proposed causality-based approach can handle more complex relationships, such as the highly non-linear scale changes in the CVMPMug sequence. Here it learns that a complex relationship between object motion and the reflection in the window can be used to predict the object pose even under full occlusion. See the relationships visualised in Figure 4.20, which could not be modelled as a constant offset (such as in [64]) or even with a linear relationship, especially for the y-coordinate, where the learned causal prediction proves necessary. Additionally, the proposed approach even discovered two complex supporters on the shadow of the hand (mirrored and scaled y-component, see Figure 4.18) that [64] was not able to use. The ultimate result is

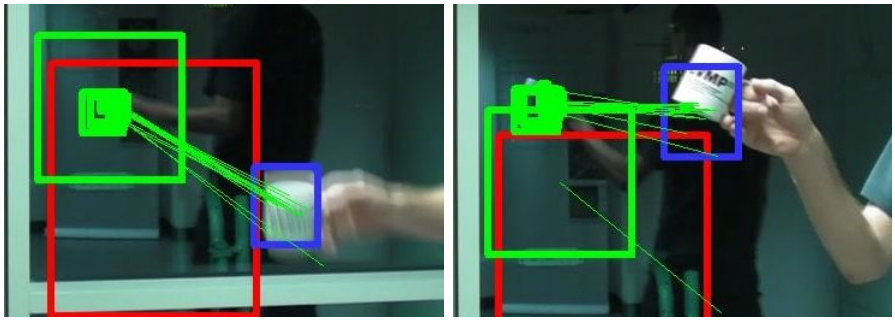


Figure 4.20: Detail of the causal prediction, with individual windows contributions visualised.

then tracking the object pose without actually being able to see it, using solely cues from related objects in the video. After the occlusion, the tracker could be automatically reinitialised; the occlusion detection and causal prediction prevent drift of the appearance model in such cases.

Figure 4.21 shows quantitative results, compared to [64]. The error (measured by deviation from the mean human-annotated position) was reduced by more than one third, from 41.8 ± 70.3 to 27.5 ± 23.1 pixels. In [64], the points of failure, returned with lower (however still higher than 50 %) confidence are discarded. These can be seen as the blue spikes in Figure 4.21. Even when ignoring these, the proposed approach still significantly improves results over the sequence.

4.8 Closing Remarks on Causal Relationships

In this chapter, causal relationships between object and camera motions were explored. An approach was proposed to discover and quantify this relationship using transfer entropy, a statistical tool which has not been used in any previous publication in the area of computer vision. It was also shown that it is possible to find the optimal time window for prediction of the object position based on the global image motion even for complex non-linear relationships. Finally, these causality-based motion predictions were evaluated on a range of standard tracking sequences, and shown to offer signifi-

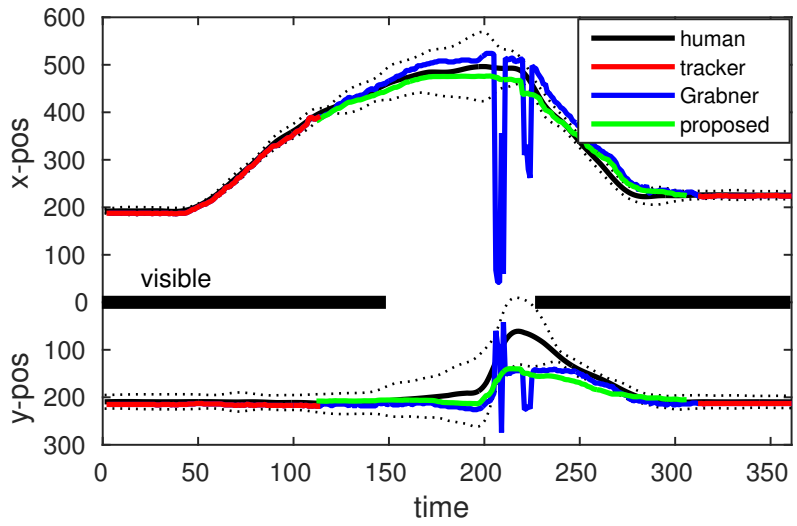


Figure 4.21: Tracking and prediction results on the ETH-Cup sequence. Mean and confidence intervals (three standard deviations in dotted line) shown for human annotations.

cant improvements (increasing average prediction accuracy by 62 % and improving the top performing tracker on VTB1.1 by 7 % in accuracy and 22 % in robustness), with particular robustness to camera shakes and fast motion. However, the improvements are variable, indicating the shortcomings of not fully integrating the prediction into the motion model of the trackers. Employing the prediction directly inside the tracker, or alternatively designing a tracker with this prediction in mind would certainly improve the results further. This would be a very interesting topic for future research.

Additionally, causal relationships between different elements of the scene help in cases of strong (even full) occlusion. These are typically the greatest source of errors in modern tracking, as shown in recent benchmarks, and thus the proposed techniques, which were made publicly available, could provide an invaluable addition to any tracking algorithm.

Chapter 5

Between Tracking and Structure from Motion

One of the challenges mentioned in the Introduction chapter is *out-of-plane rotation*, caused by variations in viewpoint. This is a hard problem, and a significant cause of failure for most state-of-the-art trackers. The LT-FLO track introduced in Chapter 3 is, unfortunately, not an exception. Out-of-plane rotations are one of its common failure cases, *e.g.* the rather poor performance on the Car sequence from the VOT2013 dataset in Table 3.1. Similarly in the VTB benchmark (Table 3.5), LT-FLOTrack dropped from its excellent overall third position to the ninth position, when considering only sequences with strong out-of-plane rotation (column OPR). It is obvious that any tracker needs to address this kind of appearance change, which comes from the object pose and cannot be modelled by a simple planar transformation.

Many approaches have been proposed to overcome variations of appearance. Online approaches typically assume that the tracking has thus far succeeded, using this to enrich the representation of the object over time. The object is usually represented as a 2D patch [76, 166], a cloud of 2D points [25, 113] or a combination of these [106]. Unfortunately, variations of viewpoint lead to rapid changes in appearance – see Figure 5.1 for an example. This causes problems for 2D trackers which do not have sufficient observations to confidently update their object representation.

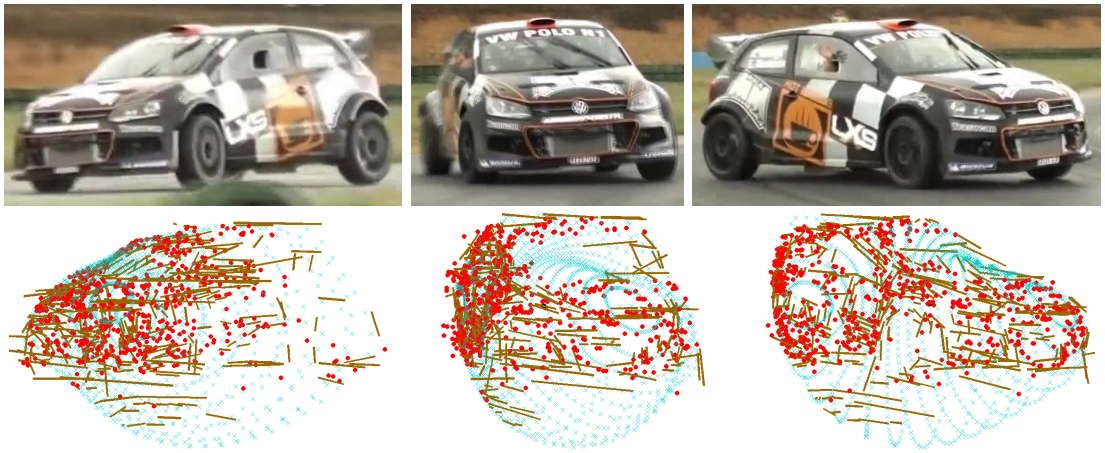


Figure 5.1: Out-of-plane rotations change the object appearance significantly, here is a complete change in just 50 frames. First row: original images. Second row: feature cloud and final model returned by the tracker. Notice the bottom and back side of the car, which have not been observed yet, so the point cloud does not reach there and the model is smoothly extrapolated.

In this chapter, the conventional approach of treating appearance changes resulting from viewpoint variation as object appearance diversity is challenged. Instead, it is argued that an intrinsically 3D object in the 3D world should be modelled as such. This is similar to the case of model-based 3D tracking (reviewed in Section 2.2), where the pose of the camera is sought relatively to the model (which may be user-supplied [77, 193], assumed to be a plane [96, 174], or induced by fiducial markers [90, 112]). Variations of viewpoint should then be treated explicitly as the camera motion, in accordance with reality. By doing so, the negative effects of out-of-plane rotation are not only mitigated; it actually proves beneficial, as it improves the numerical conditioning (wider baseline).

The 3D shape of the object is estimated online using techniques developed in the fields of Structure from Motion (S*f*M) and Simultaneous Localisation And Mapping (SLAM). As such, this approach can be seen as a bridge between visual tracking and S*f*M/SLAM, combining 2D feature tracking and object segmentation with camera pose and 3D point/line estimation, while avoiding the need for initialisation in SLAM [142]. Another difference from S*f*M/SLAM is object/background segmentation, where only a small portion of the image is used (*e.g.* 10 %, but this could be as low as 1 %). This can

easily become a significant issue as S*f*M/SLAM techniques would attempt to model the scene (background) while features on the object would be rejected as outliers. However, exactly the opposite behaviour is desired. Therefore it is vital to use the tracking to provide object/background segmentation to focus on the target object and *actively ignore* the background.

To achieve this, a novel approach to modelling the object's 3D shape using a Gaussian Process (GP) is explored. This model helps to distinguish which parts of the image belong to the projection of the object and which are background, allowing intelligent detection of new features. In addition, the GP shape model 1) provides an initialisation of the 3D positions for newly detected 2D features, 2) mitigates the sparsity of features that would lead to failure of techniques such as PTAM [111], 3) the surface normals of the GP indicate which points on the object may be visible to a particular camera to aid redetection and loop closure and 4) the GP offers a model of the surface for visualisation and subsequent tasks, such as dense reconstruction or robot navigation. For instance, the model, which is learned online as the video-sequence is processed, can be shown to the user as immediate feedback (analogous to [92, 156] with hand-held or remote-controlled aerial recording) on how successful the modelling has been thus far.

5.1 Simultaneous 3D Tracking and Reconstruction

The objective is tracking a 3D object throughout a sequence and learning a model of appearance on the fly. However, unlike most online tracking approaches, ideas from both S*f*M and SLAM are employed in this chapter, to form a 3D representation of the model that can cope with out-of-plane rotation. The program and data flow of the proposed Tracking, Modelling And Gaussian-process Inference Combined (TMAGIC) tracker is illustrated in Figure 5.2. It consists of two parts: *tracking* and *modelling* (see the subsequent sections for full descriptions of the individual elements).

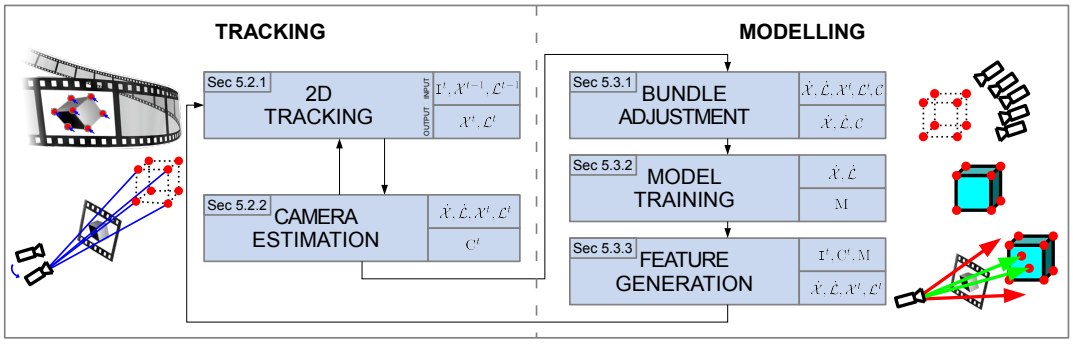


Figure 5.2: Overview of the TMAGIC tracker. Symbols on the right side of each step indicate inputs and outputs, as labelled for the case of 2D tracking. While the tracking loop is repeated in every frame, modelling only runs when necessary (according to Equation (5.4)). The numbers in upper left corners denote sections covering individual steps.

The *tracking loop* is performed on every frame. 2D features (points and line segments in the image) are tracked in the new frame (Section 5.2.1), yielding the sets of features currently visible. Using these, the new camera pose can then be estimated (a linear perspective camera model is used, Section 5.2.2), while keeping the corresponding 3D features (points and lines in the real world) fixed. Without an outer reference, the world coordinate system is not fixed. Therefore it can be, without loss of generality, safely assumed that the camera is moving around a stationary object. The tracking loop is repeated until a change in viewpoint necessitates an update of the 3D features, using the modelling subsystem.

The first step of *modelling* is a Bundle Adjustment (BA) (Section 5.3.1). This refines the positions of 3D features and the camera, using the 2D observations. The updated features are subsequently used to retrain the shape model (Section 5.3.2), which can be exploited in two ways. The model defines regions of the image which are eligible to detect new 2D features. Secondly, it provides an initialisation of the corresponding backprojected 3D features (Section 5.3.3). Features, successfully extracted using the current frame, camera pose and the shape model, then enrich the 2D and 3D sets for use in future tracking. The TMAGIC tracker assumes a single rigid object is selected for

tracking. Tracking and modelling of non-rigid objects is the subject of the next chapter. Multiple object tracking with advanced correlation/occlusion reasoning is beyond the scope of this thesis, but would be an interesting topic for future work.

5.2 Camera Pose Tracking

5.2.1 2D Features and Tracking

The TMAGIC algorithm uses two types of features: points and line segments. The main advantages of point features are that they form readily available unique descriptors (patches), are localised precisely and have intuitive and simple projective properties. On the other hand, line features, which provide complementary information about the image, have different virtues. Lines encode a higher level of structural information [140, 214], *e.g.* constraining the orientation of the surface. They can be not only texture-based, but also stemming from the shape of the object [129]. Therefore in man-made environments they appear in situations where point features are scarce [176] (such as in a low texture scenario). This observation was used extensively in the LT-FLO tracker presented in Chapter 3.

Features of both types go through the same life cycle, visualised in Figure 5.3. Firstly, they are extracted from the image, in areas belonging to (*i.e.* segmented as) the tracked object. This is denoted as S1. Newly created features are regarded as *active*. These features may be deactivated (and their 2D counterparts removed), if they cannot be tracked, or are deemed to be on an invisible part of the object. This transition to the *invisible* state is marked as S2. On the other hand, *invisible* features may be redetected and thus return to the *active* state (S3). Finally, features considered invalid (*e.g.* laying on the background) are discarded (S4). Particular techniques used are described in the following paragraphs and the role of the online learned model is detailed in Section 5.3.3.

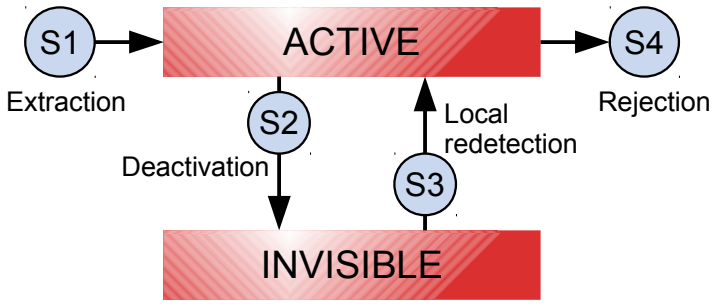


Figure 5.3: Life cycle of features used for 3D tracking.

The 2D point features $\mathbf{x}_i^t \in \mathcal{X}^t$ are extracted (S1) using two techniques: Difference of Gaussians (*i.e.* SIFT points) and Hessian Laplace [195]. These features are tracked independently by a Lucas-Kanade (LK) tracker from frame $\mathtt{I}^{t-1}$ to $\mathtt{I}^t$. Features, which do not converge are removed from the 2D feature cloud (S2). The tracked features generate a set of correspondences $\{(\mathbf{x}_i^{t-1}, \mathbf{x}_i^t)\}$, which are subsequently verified by Locally Optimised RANSAC (LO-RANSAC) [30, 130]. Outliers to RANSAC, *i.e.* correspondences inconsistent with a global epipolar geometry model, are removed (S4), as well as their respective 3D features, as these are likely to lie on the background.

The 2D line features $\mathbf{l}_i^t \in \mathcal{L}^t$ are extracted (S1) using the Line Segment Detector (LSD) [200] approach with false-positive detection control. Lines are tracked as follows. Firstly, the LSD is executed on $\mathtt{I}^t$ to obtain a set of candidate segments $\mathbf{l}_j'^t \in \mathcal{L}'^t$. Along each of the previous line segments $\mathbf{l}_i^{t-1}$ a number of edge points are then sampled. Each of these is tracked independently, using the *guided edge search* as in Section 3.2, leading to a new edge point in the current frame. If this new point belongs to a line segment in $\mathcal{L}'^t$, this point votes for the segment. The segment $\mathbf{l}_j'^t$ with the most votes becomes the new feature $\mathbf{l}_i^t$. As there is no way to validate line correspondences w.r.t. an epipolar geometry [80], the features are validated using a threshold on the minimum number of votes (3 out of 5 in all the experiments in this chapter). See Figure 5.4 for an example.

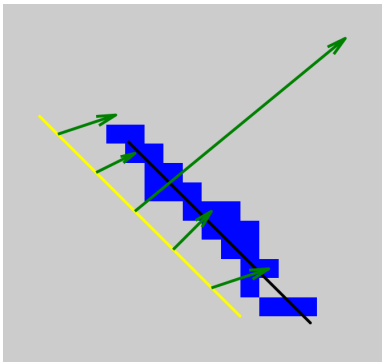


Figure 5.4: An example of line tracking. Yellow: line feature $\mathbf{l}_i^{t-1}$, black: line feature $\mathbf{l}_j'^t$, blue: pixels belonging to $\mathbf{l}_j'^t$ (as given by LSD), green: edge-to-edge correspondences. Since 3 out of the 5 sampled edge points converged, feature $\mathbf{l}_j'^t$ is validated and will become $\mathbf{l}_i^t$.

5.2.2 Camera Estimation

The camera C^t with a projection matrix $\mathtt{P}^t$ is defined by the rotation $\mathtt{R}^t$ and position $\mathbf{C}^t$ of the projection centre in the world coordinate frame, given by the decomposition

$$\mathtt{P}^t = \frac{1}{f}\mathtt{K}\mathtt{R}^t \left[\mathtt{E}_3 | -\mathbf{C}^t\right] , \tag{5.1}$$

where f is a focal length (in world units, e.g. millimetres) and $\mathtt{K}$ is a calibration matrix of intrinsic camera parameters [80], while $\mathtt{E}_3$ stands for the 3×3 identity matrix. Since $\mathtt{P}^t$ is a homogeneous entity, f can be neglected in the computation. For simplicity, a general projection function Π is defined, such that 3D lines are projected as $\mathbf{l}_i^t = \Pi(\mathbf{L}_i|\mathrm{C}^t)$ and 3D points as $\mathbf{x}_i^t = \Pi(\mathbf{X}_i|\mathrm{C}^t)$.

Assuming a cloud of 3D features (points $\mathbf{X}_i \in \dot{\mathcal{X}}$ and line segments $\mathbf{L}_i \in \dot{\mathcal{L}}$, which are defined by their end-points, specified in Section 5.3.1) and their projections ($\mathcal{X}^t$, $\mathcal{L}^t$) is given, it is possible to estimate a pose (in particular $\mathtt{R}^t, \mathbf{C}^t$) of the camera C^t. The calibration matrix $\mathtt{K}$ of the camera is not computed exactly, instead an estimate

based on image dimensions [160] is used within this chapter:

$$\mathtt{K} = \begin{bmatrix} w+h & 0 & w/2 \\ & w+h & h/2 \\ & & 1 \end{bmatrix}, \tag{5.2}$$

where w and h are the width and height of the image respectively. This formula would not suffice for cases of strong zoom, wide-angle cameras, or cropped videos (shift of the principal point and narrowed viewing angle). However, Pollefeys *et al.* [160] show that images obtained by standard cameras are generally well approximated by this formula.

Estimation of calibrated camera pose given 2D to 3D point correspondences (the so-called Perspective-n-Points (PnP) problem) is a standard textbook problem, *e.g.* solved by a P3P-RANSAC [80]. However, although research has been done in the P3L/PnL area for lines [214], combining these two is not straightforward. Therefore an optimisation approach is used to solve for camera pose:

$$\mathrm{C}^t = \arg\min_{\underline{\mathrm{C}}} \left(\sum_{i=1}^{|\mathcal{X}^t|} \rho_1 \left(||\mathbf{x}_i^t - \Pi(\mathbf{X}_i|\underline{\mathrm{C}})|| \right) + \sum_{i=1}^{|\mathcal{L}^t|} \left(\rho_1 \left(\widetilde{\boldsymbol{\mu}}_{\mathbf{l}_i^t}^\top \Pi(\mathbf{L}_i|\underline{\mathrm{C}}) \right) + \rho_1 \left(\widetilde{\boldsymbol{\nu}}_{\mathbf{l}_i^t}^\top \Pi(\mathbf{L}_i|\underline{\mathrm{C}}) \right) \right) \right), \tag{5.3}$$

using both point and line features in a unified framework and exploiting the sequential nature of tracking [71]; ρ_1 is a robust cost function to provide outlier tolerance (similar to [190]). The error function consists of an error term for each point and line feature (in the first and second summation, respectively). For points, this is just a norm of the projection error. For line features, the error terms are defined as orthogonal distances of the end-points of the segment $\mathbf{l}_i$ ($\widetilde{\boldsymbol{\mu}}_{\mathbf{l}_i}$, $\widetilde{\boldsymbol{\nu}}_{\mathbf{l}_i}$, in homogeneous coordinates), to the projection of the 3D line $\Pi(\mathbf{L}_i|\underline{\mathrm{C}})$ (homogeneous, normalised to the unit length of the normal vector). Note that since the line may not be fully visible (and is theoretically infinite), only perpendicular distances are used, in order to cope with the *aperture problem* (see Section 3.1, also [176]). This minimisation is initialised at the pose in the previous frame (C^{t-1}). During experimentation, it was found that due to the smooth nature of

the derivatives of (5.3), the basin of convergence for this optimisation is several orders of magnitude larger than typical inter-frame difference.

In the first frame, the world coordinate frame is chosen as follows. The object, initialised as a sphere, is centred at the origin and camera centre $\mathbf{C}^1$ is at $(0,0,1)^\top$. The rotation $\mathtt{R}^1$ is set such that the origin is projected to the centre of the user-given bounding box and the y axis of the camera coordinate system is parallel to the y-z plane of the world coordinate system (see Figure 5.9). This is, however, not fixed, and changes freely during BA, which optimises both the camera trajectory and feature positions.

5.3 Online Modelling 3D Shapes

5.3.1 Bundle Adjustment

After initialisation, 2D tracking is performed until the distance between camera centres exceeds a specified threshold $\theta_{\mathbf{C}}$ [156]:

$$||\mathbf{C}^t - \mathbf{C}^{t'}|| > \theta_{\mathbf{C}}\,, \tag{5.4}$$

where t' is the time of the last BA. Theoretically, $\theta_{\mathbf{C}}$ could be set to 0 such that BA is executed on every frame, however that would be excessively time-consuming. Requiring a baseline of sufficient width (non-negligible camera motion) between two consecutive BA runs creates well-timed on-demand execution on keyframes characterized by equidistant camera poses. When the condition (5.4) is satisfied, the modelling part of the algorithm is performed. Firstly, the Bundle Adjustment refines the positions of 3D features $\dot{\mathcal{X}}$, $\dot{\mathcal{L}}$ and cameras $\mathcal{C}$. For the purposes of BA, we define $\mathcal{C}$ as the set of previous cameras $\mathrm{C}^1,\ldots,\mathrm{C}^t$. If speed is an issue, one may limit the BA to take only the last k cameras into account, *i.e.* to define $\mathcal{C} = \{\mathrm{C}^u | u = \max(1, t-k), \ldots, t\}$, however, in experiments within this chapter this did not prove necessary (thus setting $k = \infty$

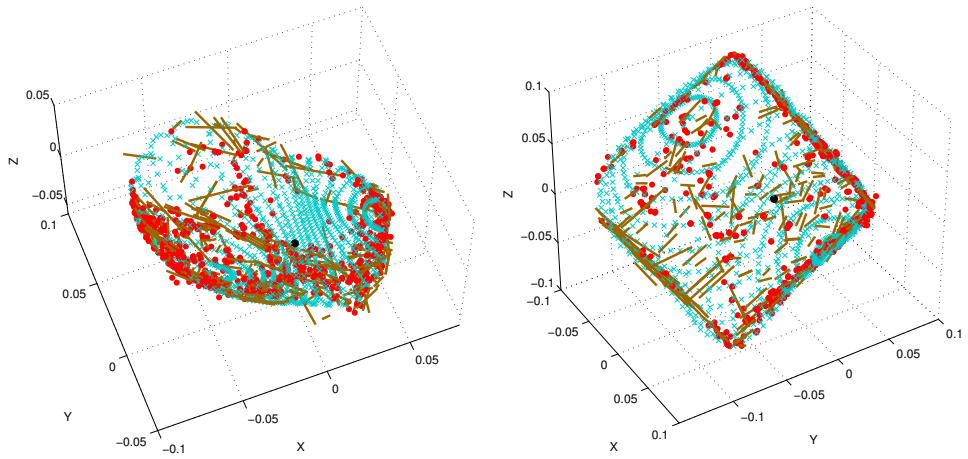


Figure 5.5: Examples of 3D feature clouds and GP-learned smooth models. Notice the unseen parts of the objects, which are without features and where the model is extrapolated (*i.e.* the rear side of the car and the "pole" on the left side of the cube).

was employed). The BA [2] minimises a similar error to (5.3):

$$\begin{aligned}\underset{\dot{\mathcal{X}},\dot{\mathcal{L}},\mathcal{C}}{\arg\min} \sum_{u=1}^{t} \Bigg(\sum_{i=1}^{|\mathcal{X}^u|} \rho_2(||\mathbf{x}_i^u - \Pi(\mathbf{X}_i|\mathrm{C}^u)||) + \\ \sum_{i=1}^{|\mathcal{L}^u|} \left(\rho_2 \left(\widetilde{\boldsymbol{\mu}}_{\mathbf{l}_i^u}^{\top} \Pi(\mathbf{L}_i|\mathrm{C}^u) \right) + \rho_2 \left(\widetilde{\boldsymbol{\nu}}_{\mathbf{l}_i^u}^{\top} \Pi(\mathbf{L}_i|\mathrm{C}^u) \right) + \Lambda_i \right) \Bigg),\end{aligned} \tag{5.5}$$

where the added term Λ_i is a regularisation term, which ensures that the lengths of 3D line segments are close to those observed. The robust cost function ρ_2 employed here is the Cauchy loss, as provided by the Ceres Solver [2]. Note also that every point from $\mathcal{X}^t$ and $\mathcal{L}^t$ has a correspondence in $\dot{\mathcal{X}}$ and $\dot{\mathcal{L}}$, respectively, for every t, but not necessarily vice-versa, due to features which are not currently visible. After the 3D feature positions have been refined, they are used to train the shape model.

5.3.2 Gaussian Process Modelling

As discussed previously in this chapter, the object shape is modelled as a Gaussian Process (GP) [1, 165]. This allows us to infer a fully dense 3D model from the finite collection of discrete observations $\dot{\mathcal{X}}$ and $\dot{\mathcal{L}}$. Using the GP in this manner can be seen as

estimating a distribution over an infinite number of possible shapes. The expectation of such a distribution (the most probable shape) can be used to model the object, while the variance at any point represents confidence.

For online target/background segmentation, a Gaussian Process (GP) is trained as a coarse, probabilistic model. The representation can be chosen such that every point $\mathbf{X}$ on the surface is represented in spherical coordinates (θ, φ) relative to the object centre $\mathbf{Y}_\circ$ as:

$$\mathbf{X} = \mathbf{Y}_\circ + r \cdot (\sin\theta\,\cos\varphi, \sin\theta\,\sin\varphi, \cos\theta)^\top \tag{5.6}$$

(for more details on the choice of $\mathbf{Y}_\circ$ see below). For any pair of angles, the radius would then be modelled as:

$$r = \mathrm{GP}(\theta, \varphi|\kappa)\,, \tag{5.7}$$

where κ is the *kernel* of the GP, relating to the surface properties of the modelled object, and may be any positive definite two-parameter function. This function is learned during tracking, such that the likelihood of the training data is maximised.

This is a natural, minimal (*i.e.* using exactly 2 parameters for the two-dimensional space of 3D directions) parameterisation. However, it suffers from singularities at the "poles", where the whole range of azimuths represent a single point. For this reason, the unit-vector parameterisation is used within this work instead. A 3D point is first re-expressed as a vector from the centre $\mathbf{Y}_\circ$ of the object:

$$\mathbf{X} = \mathbf{Y}_\circ + \mathbf{Y}\,. \tag{5.8}$$

These are then modelled such that the unit-length normalised vectors $\bar{\mathbf{Y}}_i = \mathbf{Y}_i/||\mathbf{Y}_i||$ represent the independent variable and the radii $r_{\bar{\mathbf{Y}}_i} = ||\mathbf{Y}_i||$ the dependent variable, *i.e.*:

$$\mathbf{X} = \mathbf{Y}_\circ + r \cdot \bar{\mathbf{Y}}\,, \tag{5.9}$$

$$r = \mathrm{GP}(\bar{\mathbf{Y}}|\kappa)\,. \tag{5.10}$$

As it does not suffer from a singularity in any direction, this parameterisation was found superior to alternatives such as the spherical coordinates, despite its higher dimensionality.

The observed 3D points $\dot{\mathcal{Y}} = \{\mathbf{Y}_i\}$ (point features $\dot{\mathcal{X}}$ and end-points[1] of line features $\dot{\mathcal{L}}$, in both the *active* and *invisible* states) are used as training data for the Gaussian Process (GP). It could be argued that the 3D parameter space in this new representation is not sufficiently covered by training data. It is true that only training points laying on the unit sphere (a 2D manifold) are provided and the parameter space outside the manifold is unconstrained. However, this does not create a problem in practice, since only query points laying on the unit sphere (*i.e.* direction vectors) are queried.

Without loss of generality, we can assume that the centre $\mathbf{Y}_\circ$ coincides with the origin of the world coordinate system. In this case, for a query direction $\bar{\mathbf{Q}}$ (where $||\bar{\mathbf{Q}}|| = 1$), the resulting 3D point $\mathbf{Q}$ is predicted as [165]:

$$\mathbf{Q} = \bar{\mathbf{Q}} \quad \Big(\kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}})^\top \; \kappa(\dot{\bar{\mathcal{Y}}}, \dot{\bar{\mathcal{Y}}})^{-1} \; \mathbf{r}_{\dot{\bar{\mathcal{Y}}}} \\ \pm\kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}})^\top \; \kappa(\dot{\bar{\mathcal{Y}}}, \dot{\bar{\mathcal{Y}}})^{-1} \; \kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}}) \Big) \;, \tag{5.11}$$

or more succinctly as

$$\mathbf{Q} = \bar{\mathbf{Q}} \;\; \left(r_{\bar{\mathbf{Q}}} \pm \sigma_{\bar{\mathbf{Q}}} \right) \;, \tag{5.12}$$

where $r_{\bar{\mathbf{Q}}}$ is the predicted radius and $\sigma_{\bar{\mathbf{Q}}}$ is the confidence. The notation $\mathbf{r}_{\dot{\bar{\mathcal{Y}}}}$ represents a vector of norms of all vectors in the training set $\dot{\mathcal{Y}}$, *i.e.* $\mathbf{r}_{\dot{\bar{\mathcal{Y}}};i} = r_{\bar{\mathbf{Y}}_i} = ||\mathbf{Y}_i||$.

Intuitively, Equation (5.11) shows that the predicted radius at any point is defined by the training radii while accounting for the spatial relationships between the data points. The influence of any particular element of the training data is quantified by $\kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}})$, while $\kappa(\dot{\bar{\mathcal{Y}}}, \dot{\bar{\mathcal{Y}}})^{-1}$ removes any correlation within the training data.

This kind of representation means that the object shape is modelled by an *implicit non-parametric function*. While one can query the surface in any direction, there is no discrete "set of vertices" marking the shape. Instead, for visualisation the model is

[1] It is possible to sample more points along line features which have high confidence.

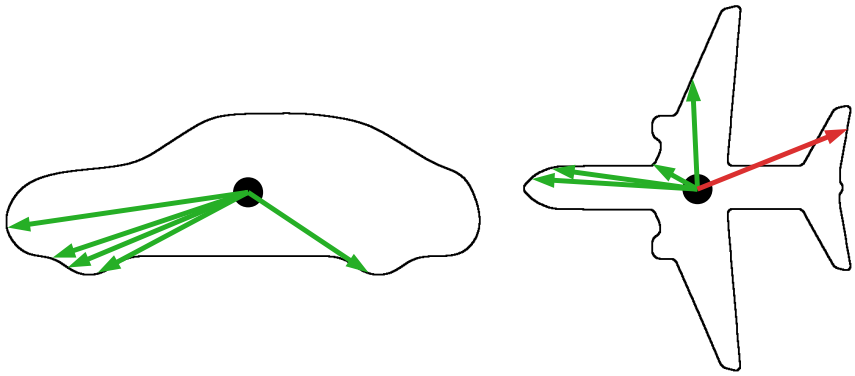


Figure 5.6: Star domain example in 2D. Left: Every region of the car shape can be reached from the *centre* without crossing the boundary, *i.e.* its shape is a star domain. Right: Since there are regions unreachable by a straight line, it is not a star domain.

queried at regularly sampled positions (see Figure 5.5 for an example). This, however, is not an obstacle for its use.

As the Gaussian Process (GP) shape model is fully probabilistic, not only a shape estimate is provided, but a whole distribution of shapes (radius functions). From this distribution, the mean (*i.e.* the most probable) shape is used as the estimate, and the variance as the uncertainty at any given point of the object surface. The probabilistic nature of the GP model further prevents overfitting through an implicit "Occam's-razor" effect, that favours models which are both simple and which explain the observations well. Other beneficial properties of GP modelling include smooth interpolation and extrapolation in regions without training inputs. Several alternatives for the surface shape modelling were tested, such as a mesh-based model, a parametric probability distribution and a spherical-coordinate model with a different machine-learning technique (such as nearest neighbour regression, neural network, random regression forest or support vector regression). However, none had the properties required.

The non-parametric nature of the model used makes it possible to model a wide range of object shapes and resolutions without the need for reparameterisation. To specify rigorously what class of objects can be modelled, one needs to consider the properties of the spherical representation. Since the radius for any given direction

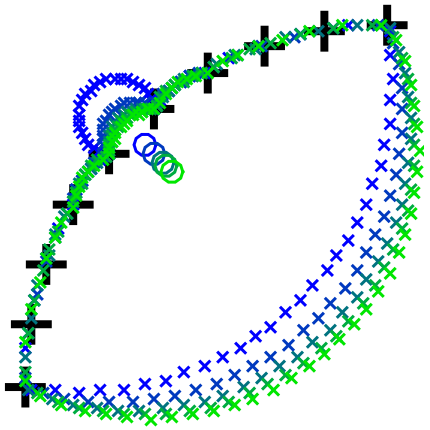


Figure 5.7: Iterative search for the shape centre. Black: training data. Coloured: shape centre (∘) and sampled points (×), iterating from the centre of mass (blue) to convergence (green).

angle must be unique, there must exist a point inside the object, the *shape centre*, such that the line segments connecting it to all the points on the shape surface lie inside the object. This class of objects is known in computational geometry as *star shapes* or *star domains* (of a Euclidean space). See Figure 5.6 for 2D examples of objects which are and are not a star domain. While this choice of parameterisation might seem overly limiting, in practice most "compact" objects (without deep concavities or long, extended parts) are of approximately star shape. Furthermore, it is not necessarily harmful when the online model smooths over minor regions which break this assumption.

However, attention must be paid to the selection of the *shape centre* $\mathbf{Y}_\circ$. While using the centre of mass is a viable solution for many shapes, this can sometimes lie too close to the object surface, which leads to unwanted artefacts (see the blue samples in Figures 5.7 and 5.8). Therefore a data-driven shape centre is found as follows. The centre of mass of the training points is used only as an initialisation and the centre is subsequently shifted towards the midpoint between this and the centre of mass of the sampled points (trained with the previous centre). The sampled points change with

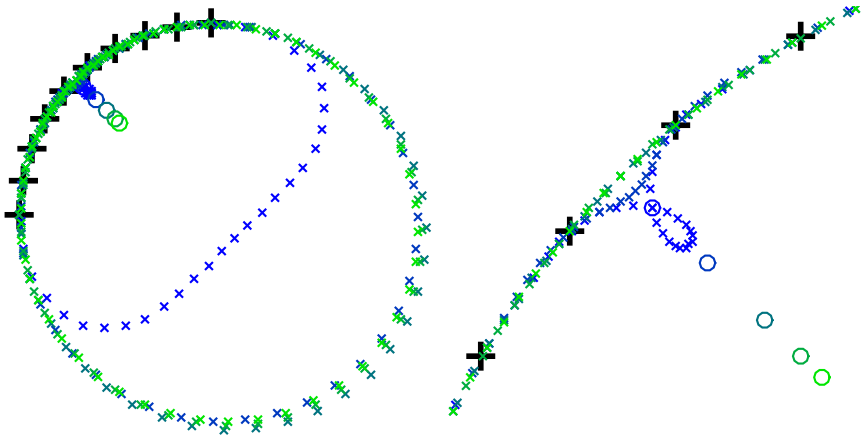


Figure 5.8: Iterative search for the shape centre with the RBF kernel; legend is the same as in Figure 5.7. Right: detail of regression near the shape centre.

the shift of the centre, so this needs to be iterated:

$$\mathbf{Y}_\circ^{\text{new}} = \gamma \frac{\frac{1}{|\dot{\mathcal{Y}}|}\sum_{\dot{\mathcal{Y}}} \mathbf{Y}_i + \frac{1}{|\dot{\mathcal{M}}|}\sum_{\dot{\mathcal{M}}} \mathbf{M}_i}{2} + (1-\gamma)\mathbf{Y}_\circ\,, \tag{5.13}$$

where γ is a learning factor and $\mathbf{M}_i$ is a point sampled on the surface model (in regular angular intervals, visualised as $\times$ throughout this chapter). The factor γ was set to 0.5 in the experiments throughout this thesis. See Figure 5.7 for a 2D illustration of the convergence. It is necessary to repeat this search for the shape centre every time the training data changes (after every bundle adjustment). However, it is *not* necessary to repeat all steps to full convergence. Instead, only one step is performed after each bundle adjustment, which converges eventually as the relative magnitude of updates of the training data decreases.

One of the most important choices while designing a technique using a GP is the choice of kernel or a combination of kernels. The kernel choice represents prior knowledge about properties of the modelled function (in this case surface shape), especially smoothness and differentiability. Formally the only condition on a function to be used as a kernel is that it is positive definite. A sum or a product of positive definite functions are positive definite as well, thus a kernel can be created as an additive or multiplicative combination of sub-kernels.

One of the most commonly used kernels is the RBF kernel (also called Gaussian kernel), which is infinitely differentiable and therefore induces smooth shape contours. This, however, causes artefacts in the shape, as the overly smooth gradient extrapolates too far from the training data and exaggerates rapid radius changes (see Figure 5.8). The same holds for the Matérn kernel (of both common orders 3/2 and 5/2). For this reason, the *exponential* kernel is employed, which allows fast changes in both the radius and its gradient and thus models sharp edges (the gradient can even be discontinuous as the kernel is only once differentiable). This kernel is (additively) combined with a *bias* kernel to avoid the assumption of zero-centred data and with a *white-noise* kernel to gain robustness against outliers:

$$\kappa_{\mathrm{GP}} = \lambda_{\mathrm{Exp}}\kappa_{\mathrm{Exp}} + \lambda_{\mathrm{B}}\kappa_{\mathrm{B}} + \lambda_{\mathrm{W}}\kappa_{\mathrm{W}}\,. \tag{5.14}$$

Besides this choice of kernel, there are no other parameters in the GP modelling.

There are, however, several *hyper-parameters* $\boldsymbol{\lambda}$, which are learned from the training data (hence *not* algorithm parameters). These include the weights λ from Equation (5.14) as well as the *length* parameter of the exponential kernel (controlling the input scaling). The hyper-parameters are optimised to maximise the likelihood of the training data $\dot{\mathcal{Y}}$. This provides the previously mentioned implicit "Occam's razor" principle [164], preventing overfitting, a unique feature of Gaussian Process modelling.

It should be noted, that at the beginning of the video-sequence (*i.e.* before the first Bundle Adjustment), there is no depth estimate and therefore no 3D information. As an initialisation of the model, a sphere is used, with dimensions inferred from the user-given initial bounding box (see Figure 5.9). While this model would be useless as an output, it gives sufficient information to track for a limited time. The model is then trained as soon as the 3D feature positions are refined.

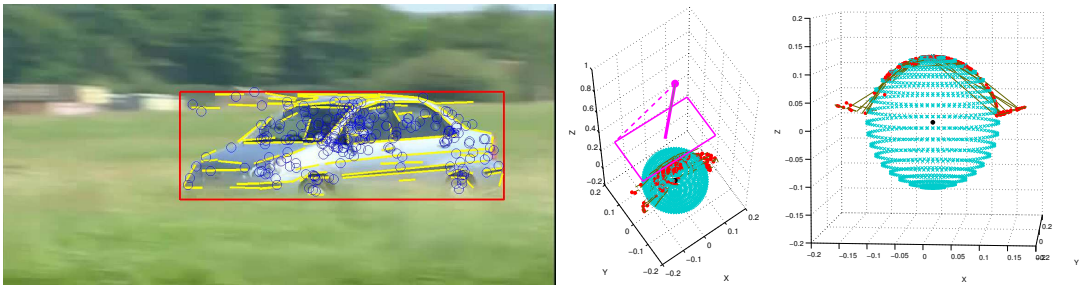


Figure 5.9: State of the proposed TMAGIC tracker after the first frame. Left: a bounding box, $\mathcal{X}^1$ and $\mathcal{L}^1$. Right: $\dot{\mathcal{X}}$, $\dot{\mathcal{L}}$, initial model M (the blue sphere visualises sampled points $\dot{\mathcal{M}}$) and C^1 (magenta). For the camera, the visualisation shows the projection centre $\mathbf{C}$, principal direction and image plane. The upper left corner of the image plane is indicated by the dashed line.

5.3.3 Feature generation

For camera pose estimation (Sec. 5.2.2), it was assumed that the 3D feature clouds $\dot{\mathcal{X}}$ and $\dot{\mathcal{L}}$ are known and fixed. In this section, the issue of feature generation and localisation is addressed. The assumption is made that a shape model of the object is known (trained according to the previous section).

In the first frame $\mathtt{I}^1$, initial sets of 2D features $\mathcal{X}^1$ and $\mathcal{L}^1$ are generated inside a user-supplied bounding box. When generating a 3D feature $\mathbf{X}_i$ for a new 2D feature $\mathbf{x}_i^t$ (in the case of line features, both end-points must lie on the surface), the process is as follows. Firstly, the corresponding ray $\mathbf{Z}$ from the camera centre is generated, parameterised by α:

$$\mathbf{Z}(\alpha) = \mathbf{C}^t + \alpha(\mathtt{K}\mathtt{R}^t)^{-1}\widetilde{\mathbf{x}}_i^t \qquad (\alpha > 0)\,, \tag{5.15}$$

where $\widetilde{\mathbf{x}}_i^t$ is the homogeneous representation of $\mathbf{x}_i^t$. Then a search for the (nearest to the camera) intersection between the ray and the shape is performed:

$$\mathbf{X}_i = \mathbf{Z}(\alpha)\,, \tag{5.16}$$

$$\alpha = \underset{\underline{\alpha}>0}{\arg\min}\, \underline{\alpha} \qquad \text{s.t.} \quad ||\mathbf{Z}(\underline{\alpha})|| = r_{\bar{\mathbf{Z}}(\underline{\alpha})}\,. \tag{5.17}$$

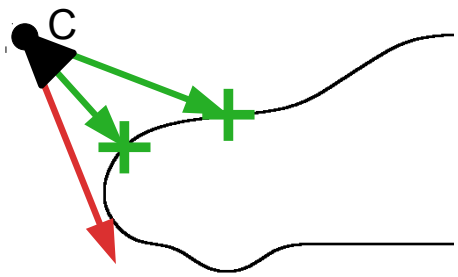


Figure 5.10: Model intersection search example. The intersection points (+) are initial 3D locations of the newly generated features. **C** marks the camera centre.

If the minimisation of (5.17) has no solution, it means that the ray does not intersect the *mean surface* given by the GP (see Figure 5.10). The use of the mean surface corresponds to a threshold such that there is an equal probability of false positives and false negatives. A false positive happens when a detection on the background is added to the feature set, while a rejected point on the object surface is a false negative. If there is prior knowledge about the respective robustness of other system components, this can be exploited by adding the appropriate factor (multiple of $\sigma_{\bar{\mathbf{Z}}(\underline{\alpha})}$) to $r_{\bar{\mathbf{Z}}(\underline{\alpha})}$ in Equation (5.17).

It should be noted that as in parts of the previous section, the shape centre $\mathbf{Y}_\circ$ is assumed to be the origin of the coordinate system, to keep the notation uncluttered. Therefore the (finite-length) ray can be simply expressed, similarly to Equation (5.12), as cast from the origin: $\mathbf{Z} = r_{\bar{\mathbf{Z}}} \cdot \bar{\mathbf{Z}}$. This, again, does not limit generality, as reintroducing $\mathbf{Y}_\circ$ back to the equations is trivial.

Thus far, the process has been the same both for initialisation in the first frame and for adding new features after model retraining in the subsequent frames. However, there are several differences. Firstly, if a ray does not intersect the surface during the generation of new features in frame $t > 1$, it is not used: features are detected over the whole image but only those covering the target object are used. However, in the first frame, all the 2D features will lie inside the user-specified bounding box. In this case, their 3D positions are reconstructed such that even when they do not intersect, they

minimise the distance to the mean surface:

$$\alpha = \arg\min_{\underline{\alpha}>0}(||\mathbf{Z}(\underline{\alpha})|| - r_{\bar{\mathbf{Z}}(\underline{\alpha})})^2 \ . \tag{5.18}$$

This can lead to the characteristic fringe seen in Figure 5.9. Generation of new features in $t > 1$ has one further condition. Since adding new features increases the time complexity of all other computations, new features are added only into uncertain regions of the object, with variance greater than a specified threshold:

$$\sigma_{\bar{\mathbf{Z}}(\alpha)} > \theta_\sigma \ . \tag{5.19}$$

As previously mentioned, some of the 3D features may be temporarily occluded, *i.e.* without 2D correspondences (in the *invisible* state). Surface normals, given by the shape model, provide a tool to determine which parts of the object are visible from a particular direction. This is done by sampling several points from the GP in close proximity to the location of interest (*i.e.* several orders of magnitude below the object dimensions) and locally fitting a tangent plane. TMAGIC uses this information in two complementary ways. Firstly, if a 3D feature is deemed not visible, but it has a 2D correspondence, it can be removed (transition S2 in Figure 5.3, e.g. when it adheres to an object contour). On the other hand, if a 3D feature has no 2D correspondence, but is on a surface which is seen by C^t under an angle close to normal, the 2D feature can be redetected (S3). This is performed by projecting it into $\mathtt{I}^t$ by Π^t and then tracking it in 2D using a stored appearance patch. *Loop closures* (as termed in the SLAM literature) are thus possible when a number of previously seen features are redetected.

5.4 Experimental Evaluation of TMAGIC

In all experiments, the parameters were fixed as follows: $\theta_{\mathbf{C}} = 10\,\%$, $\theta_\sigma = 0.5\,\%$, relative to the scene size. The tested proof-of-concept implementation (Matlab framework with several parts in C++) currently runs at several seconds per frame. However,

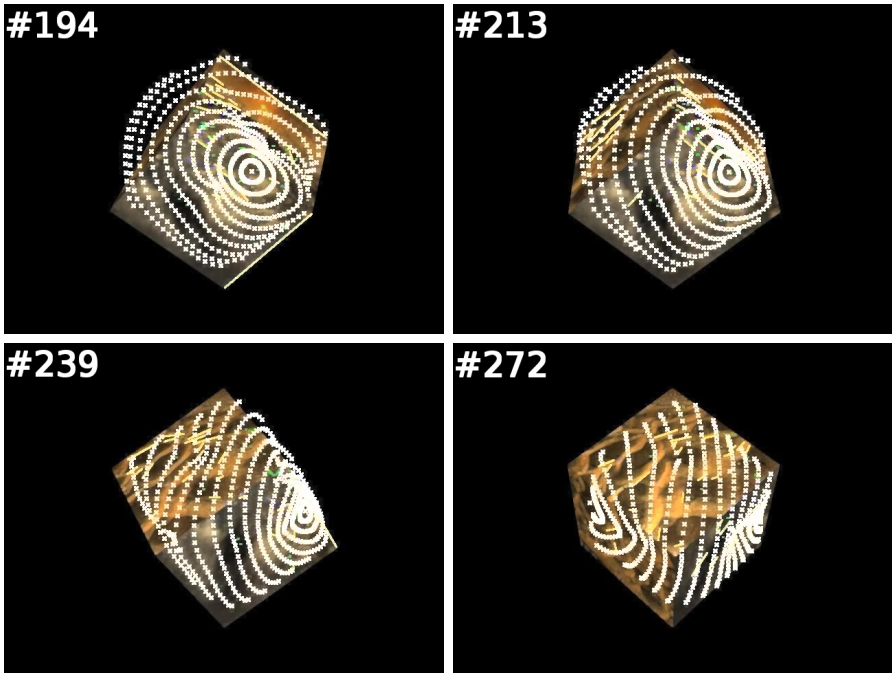


Figure 5.11: Selected frames from the CUBE1 sequence. Notice, how TMAGIC learns the new face of the cube. #194: unknown shape, the surface is smoothed over. #213: first features detected, shape roughly estimated. #239: more features identified, shape refined. #272: Final state, model in agreement with the object.

there are possibilities for trivial technical improvements and for parallelisation, allowing real-time application.

5.4.1 Synthetic data

Firstly, results on a synthetic sequence CUBE1 are shown (Figure 5.11). This has been rendered to have the following properties. It contains a cube, rotating with speed 1°/frame. Some of the sides are rich in texture, some are weakly textured. From the point of view of the TMAGIC tracker, the camera circles around the fixed cube with a perfect circular trajectory (see Figure 5.12). However, since the world coordinate frame is defined only up to a similarity transform and can be moved freely during the BA, it is not possible to measure the quality of a tracker directly w.r.t. this expected position (i.e. no absolute ground truth is possible). Therefore a 3D circle is fitted to the camera

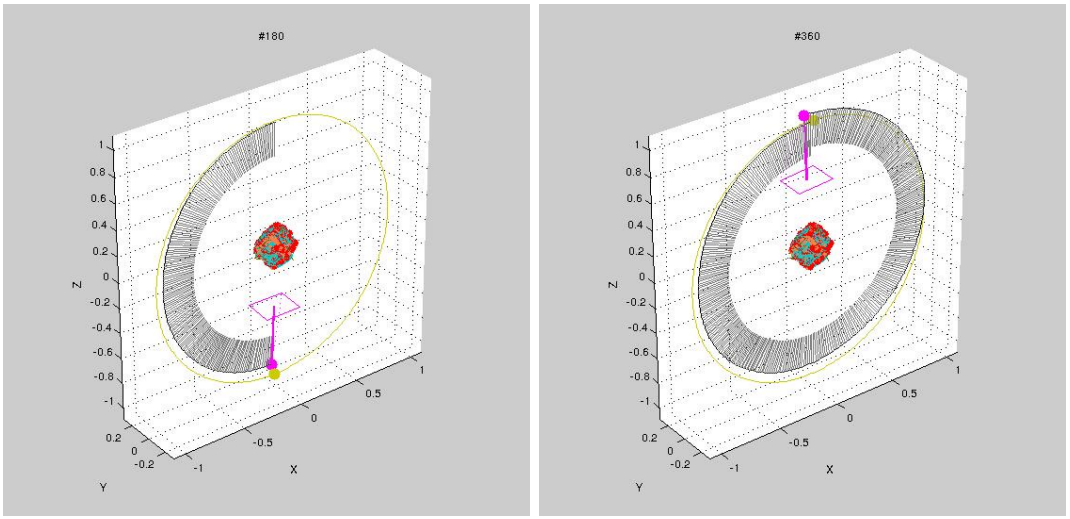


Figure 5.12: The 3D scene in $t = 180$ and 360. The camera and features are shown in the same way as in Figure 5.9, the camera trajectory (centres and principal directions for each previous frame) is shown in black. Ground-truth trajectory is in yellow. The details of the model can be seen in Figure 5.5.

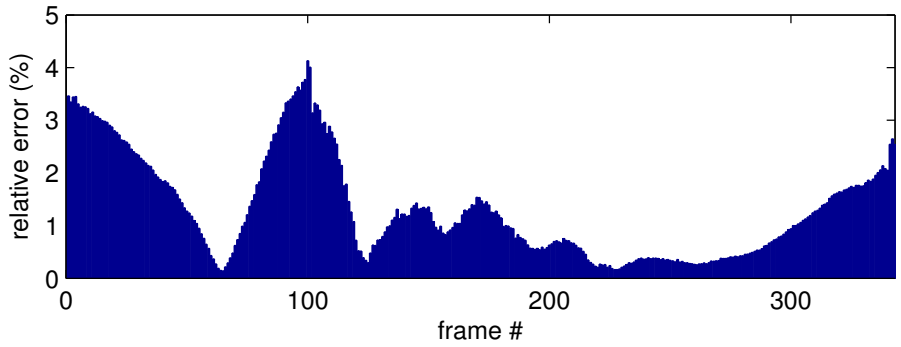


Figure 5.13: Deviation of the trajectory from a perfect circle (least-squares fitted). Errors are relative to the radius of the circle.

trajectory and the error measured as the distance from this circle. Figure 5.13 shows the results. If we assume the camera orbits at a distance of 1 m, the mean camera pose error is 1.3 cm. This indicates a close approximation to the circular trajectory. Despite being based on sparse data, the learned shape model represents the cubic shape (of side equal to approximately 17 cm) accurately, having a mean reconstruction error of 3.4 mm. The trajectories at the beginning and end of the sequence did not meet (due to accumulated error prior to loop closure), thus the deviation from the fitted

ground truth is distributed between the two. The peak around frame #100 is due to a temporary inaccuracy during the transition between sides of the cube, when the continuously visible side is lacking visual features. However, TMAGIC recovers once sufficient visual evidence has been accumulated. The input video and additional results can be found online [120].

To compare the method with currently used reconstruction approaches, this sequence (with no background to account for) was processed with VisualSfM [204, 205] and Bundler [178, 179]. While Bundler surprisingly failed, reconstructing 2 separate cubes, VisualSFM performs worse than TMAGIC with a comparable reconstruction error of 2.8 mm, but 72 % larger camera trajectory error of 2.3 cm. On real sequences including background the reconstruction techniques perform even worse, rarely modelling any reasonable part of the target object. This can be seen in the next chapter, where more reconstruction-oriented experiments are performed.

5.4.2 Real data

The performance of TMAGIC was further analysed on several sequences, used in previous 2D visual tracking publications. These sequences contain visible out-of-plane rotation in most cases. Additionally, several new sequences of drifting cars were used, which have significant out-of-plane rotation (in addition to strong motion blur). These new sequences are available online including human annotated Ground Truth (GT) [120]. Selected frames are shown in Figures 5.14 and 5.15. Figure 5.16 shows qualitative results of TMAGIC on selected sequences.

The TMAGIC tracker is compared with several state of the art tracking algorithms: Local-Global Tracker (LGT) [25], Tracking-Learning-Detection (TLD) [106] and Flock of Trackers (FoT) [137]. The FoT tracker is similar to the proposed approach, in that it employs a group of independently tracked features with a higher management layer, however it operates in 2D only. GT in 3D is not available for these sequences: neither shape not trajectory. Therefore the performance metric used was *localisation error*,

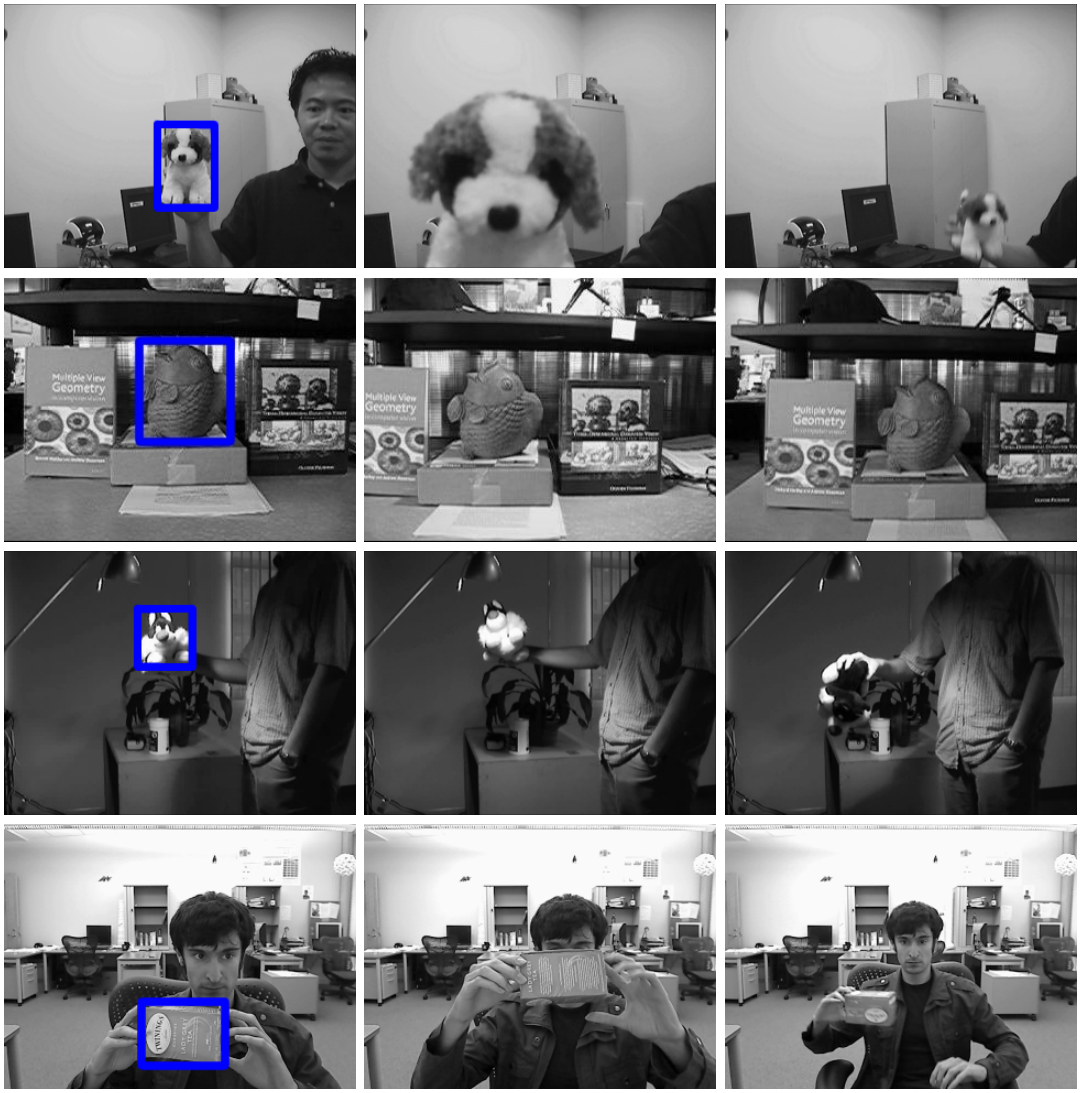


Figure 5.14: Selected frames from the test sequences (sequences from literature). From top: Dog, Fish, Sylvester and Twinings. The initial bounding boxes are overlaid.

i.e. the distance of the centre of the bounding box[2] to the ground-truth centre, and additionally *overlap* of the tracked and ground-truth bounding boxes. TLD can report an absence of the object. There are, however, no full occlusions in the tested scenes, thus in such cases the frames were assigned the maximal error found in the sequence. The results are visualised in Figure 5.17 and the mean values tabulated in Table 5.1.

[2]In the case of LGT the centre is returned directly.

Figure 5.15: Selected frames from the test sequences (newly created sequences). From top: Rally-Lancer, Rally-VW, TopGear1 and TopGear2. The initial bounding boxes are overlaid.

On the Dog sequence, the TMAGIC and TLD trackers perform similarly, and LGT slightly worse. All these trackers experience difficulties at about frame #1000, where the dog is partially occluded by the image border. This is however not a problem for FoT, which estimates the position accurately even under such strong occlusion. The Fish sequence is relatively simple, with all the trackers reaching low errors and TMAGIC being the best. On the Sylvester scene, TMAGIC as well as FoT track consistently well until the end. Both LGT and TLD have similar momentary failures (frames #450 and #1100, respectively) but both are able to recover. For LGT, the duration of the problematic part of the sequence is shorter and the error is smaller, rendering it the best tracker for this sequence. The Twinings sequence contains full rotation and was originally created to measure trackers' robustness to out-of-plane

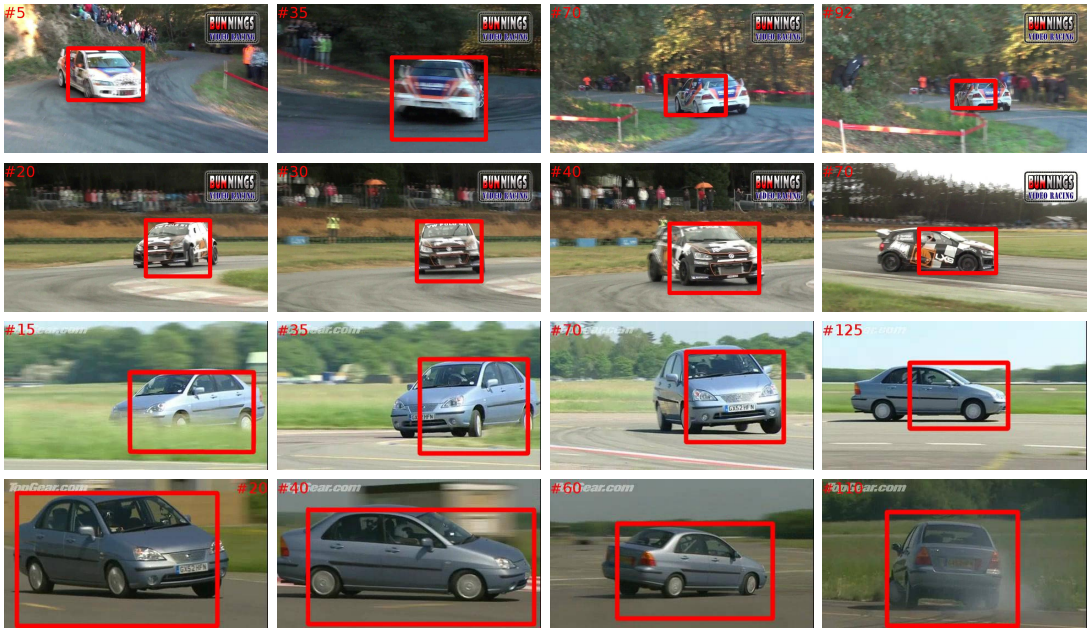


Figure 5.16: Qualitative results of the TMAGIC tracker. From top to bottom: Rally-Lancer, Rally-VW, TopGear1.

	LGT	TLD	FoT	T	TMIC	TMAGIC
Dog [27]	19.3/20	12.3/56	**4.7**/63	13.9/65	9.3/**71**	12.1/46
Fish [166]	15.6/20	8.8/72	9.2/**75**	8.0/68	10.4/65	**5.6**/69
Sylvester [166]	**13.1**/16	18.0/**58**	18.0/**58**	37.3/42	35.3/9	17.8/48
Twinings [10]	22.5/18	13.2/38	15.5/44	42.9/31	16.2/29	**9.1**/**53**
Rally-Lancer	145.8/19	333.5/13	734.5/13	127.8/43	121.3/42	**94.3**/**53**
Rally-VW	91.3/21	152.5/48	196.9/39	149.4/39	148.3/39	**47.6**/**62**
TopGear1	**16.3**/49	65.1/34	80.4/41	39.0/49	44.5/43	40.0/**56**
TopGear2	84.3/37	104.3/21	117.9/29	48.4/51	84.1/31	**34.7**/**59**
Average	48.4/25	88.5/43	147.1/45	58.3/49	58.7/41	**32.7**/**56**

Table 5.1: Tracking results: mean localisation error (in pixels) and mean overlap (in percents). Bold numbers indicate the best result, underlined numbers the second best.

rotation [10]. Unsurprisingly, TMAGIC significantly outperforms the state of the art on this sequence. The average localisation error reduction for all these scenes is 22 %.

The car sequences are chosen because they contain rigid 3D objects under strong out-of-plane rotation (around 180°) with significant camera motion. Therefore the TLD and FoT trackers, which are trying to track a plane only (one side of the car) instead of the 3D object, fail. As the cars rotate, the tracked parts are no longer usable and

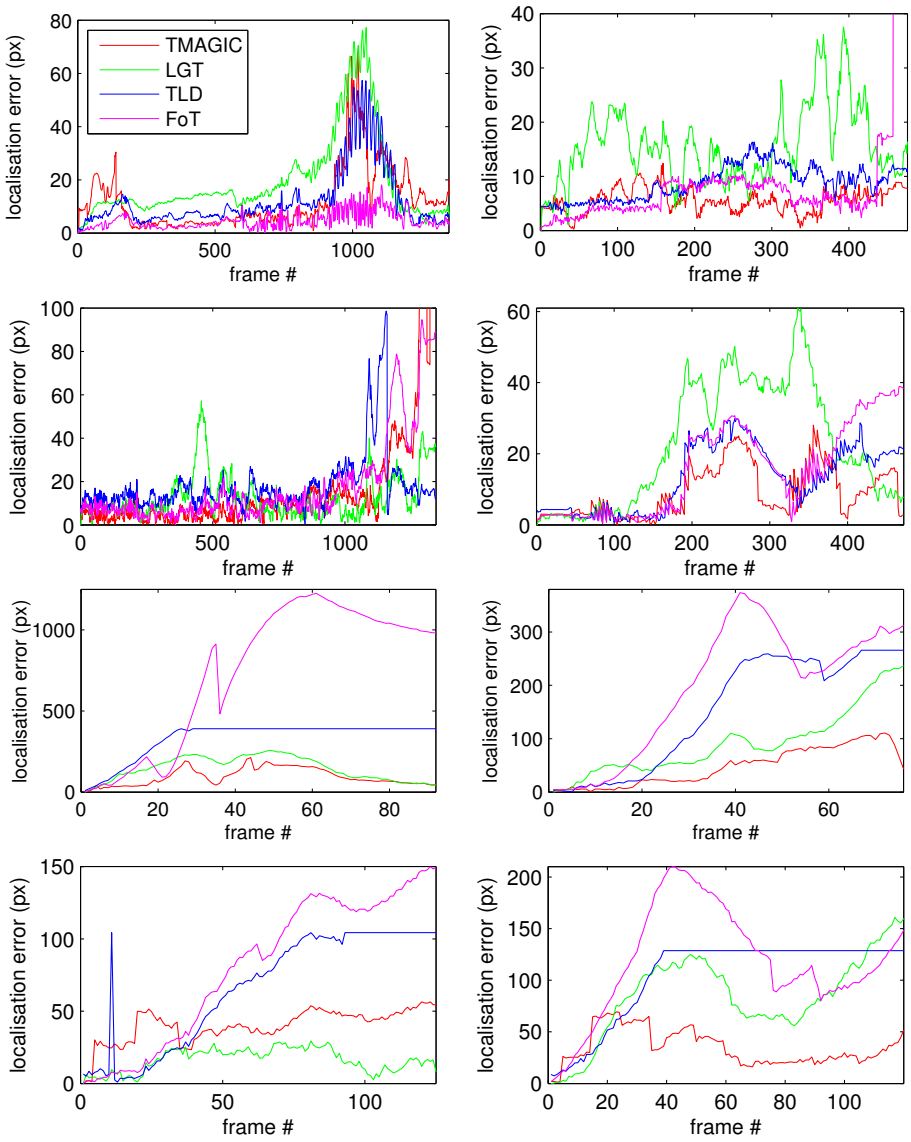


Figure 5.17: Visualisation of results of the quantitative performance analysis. From top and left: Dog, Fish, Sylvester, Twinings, Rally-Lancer, Rally-VW, TopGear1 and TopGear2.

TLD reports this (the horizontal sections in Figure 5.17). FoT is incapable of reporting object disappearance and it attempts to continue tracking, exacerbating the situation. The LGT tracker, which has a less rigid model of the object, is sometimes capable of tracking after the cars start to rotate, if the rotation is slow enough for the 2D shape model to adapt. The TMAGIC tracker is also able to adapt as the object rotates, and

explicitly modelling the car in 3D improves robustness by allowing it to intelligently detect new features. While 2D trackers attempt to mitigate the effects of out-of-plane rotation, TMAGIC actively exploits it. This gives it a significant edge, resulting in the localisation error being reduced by 58 % on average. Notice that the errors in the RALLY sequences are generally higher, due to the higher resolution. The resulting model for the RALLY-VW sequence is visualised in Figures 5.1 and 5.5. The car is modelled accurately, except for missing elements at the rear of the vehicle, which have not been observed during the sequence.

The last three columns of Tab. 5.1 show the effect of the different stages of TMAGIC on performance. The simplest version, **T**, assumes a fixed 3D model (sphere) and feature locations. **TMIC** additionally performs refinement of the 3D features and fits a naïve spherical model. Firstly, FoT is compared with **T**, which can be seen as 2D and 3D trackers based on the same principle. FoT performs better on sequences without rotation (higher accuracy of the solution) and the advantage of 3D tracking becomes apparent with stronger out-of-plane rotations (decreasing the error up to five-fold). The next step is **T**→**TMIC**. However, the effect of the more advanced procedure on the performance of tracking in the image plane is imperceptible, despite the improved plausibility of the feature cloud. The final stage is training a more complex GP shape model using the feature cloud (**TMIC**→**TMAGIC**, using the full system). This yields the most significant improvement (average error reduction of 50 %), rendering TMAGIC by far the best of the evaluated trackers in cases of out-of-plane rotations. In the case of TOPGEAR1, mostly the front part of the car is being modelled, shifting the centre of the bounding box forward and therefore adversely affecting the final results.

Figure 5.18 shows an example of a failure case, where TMAGIC failed to pick the foreground object. Although after the first frame (upper left) foreground features were successfully identified and used for tracking, but the situation has changed after the first BA (upper right). The updated model had slightly wider extent, causing more background features to be included in the segmented area, leading to correct point features being discarded as epipolar outliers (according to Section 5.2.1). This did not

Figure 5.18: Results on the TORUS sequence: a failure case.

cause TMAGIC to fail immediately, as correct line features were still present, but it led to drift and eventually to another increase in model size after the next retraining (bottom left). This exacerbated the situation, leading to a complete tracking failure (bottom right).

5.5 Closing Remarks on TMAGIC

The experiments show that the Tracking, Modelling And Gaussian-process Inference Combined (TMAGIC) tracker is able to track standard sequences, used in many previous publications, with a comparable performance to the state of the art. However, by explicitly modelling the 3D object, it handles out-of-plane rotations significantly better and can also track in cases of full rotation. TMAGIC consistently outperforms simpler variants (TMIC etc.), especially in scenarios when the object/background segmentation is vital. This shows the benefit of the shape model, used for filtering features and initialisation of their 3D positions.

TMAGIC works under the assumption that the object is rigid. It is robust to small shape variations (e.g. a face), but is not capable of tracking articulated objects, e.g. a walking person. Another limitation would be full occlusions in long-term tracking. However, the algorithm is robust to low textured objects through the use of line features (similarly to Chapter 3). TMAGIC assumes fixed camera calibration during the tracking. Cases of zooming in the sequences or cropped sequences usually do not cause tracking failures, but the resulting model is distorted. Online estimation of calibration parameters is addressed in the following chapter, as well as tracking and modelling of non-rigid targets.

Chapter 6

Dense 3D Tracking of Non-Rigid Objects

In all the previous chapters of this thesis, the assumption was made that the target object is rigid. While this is valid in many scenarios and approximately holds in many more, it still remains a limiting factor. In this chapter, this assumption is abandoned. Explicitly non-rigid modelling of the object of interest is explored; such modelling allows tracking in a wider range of scenarios. See Figure 6.1 for an example of a synthetic sequence, containing an object undergoing strong shape variation. The reconstructed non-rigid model shows how 3D tracking combined with Non-Rigid Structure from Motion (NRS*f*M) can lead to a success even in such a challenging scenario.

The second major contribution of this chapter is 3D tracking through shot cuts. Since this chapter is focused on tracking and reconstruction in unconstrained scenes, it is necessary to take into accounts scenarios such as archive, broadcast or online-sourced footage. These are often composed of sub-sequences separated by shot cuts. Therefore the problem of re-detecting the target after a shot cut and resuming tracking is examined. See Figure 6.2 for an example sequence and a resulting model proving the effectivness of combining tracking with Structure from Motion (S*f*M).

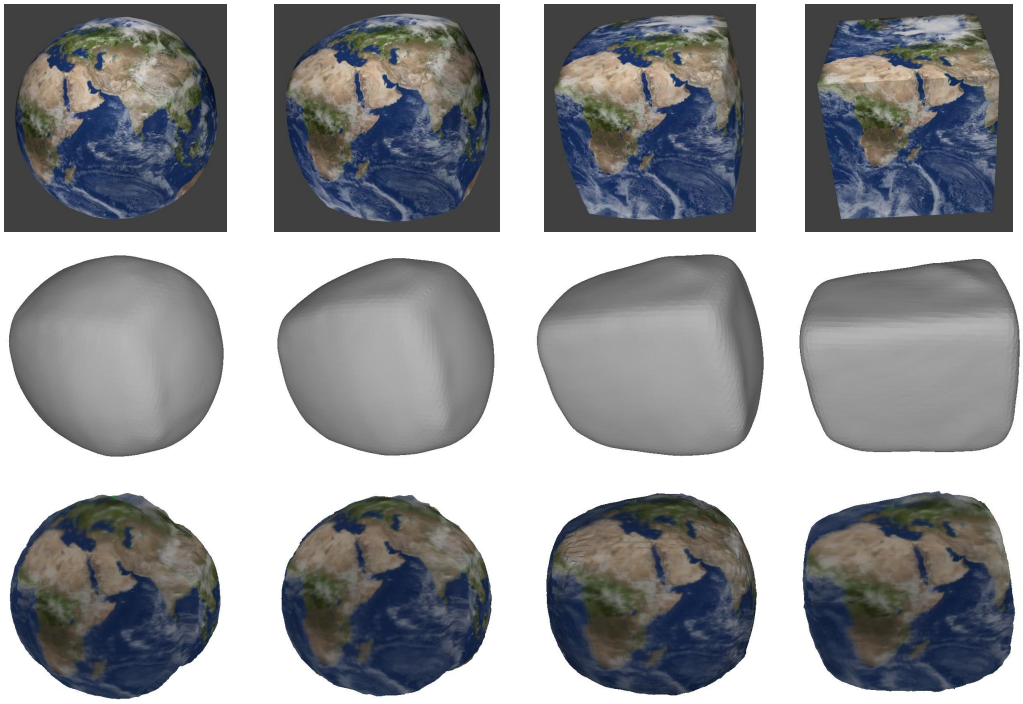


Figure 6.1: Example of input sequence and models output by the proposed method: frames #60, 70, 80 and 90 of the sequence CubicGlobe.

Figure 6.2: The Hillclimb sequence, a broadcast video divided into many sub-sequences by shot cuts. We provide an automatic modelling algorithm working through these sub-sequences.

To support both of these aims, the feature cloud is densified. Besides sparse features, as used in the previous chapter, dense features, based on optical flow, are employed. The additional information from dense features proves useful to provide more constrains to

the problem of non-rigid motion estimation. Furthermore, aspects of the motion such as per-frame focal length can be estimated using this additional data. Finally, with dense reconstruction, a better 3D model can be built, which (besides being one of the targets on its own) helps with camera alignment after a shot cut. Since optical-flow based tracking suffers from inherent drift, frame-to-frame optical flow is usually replaced by registration of every frame to a *reference frame* (*e.g.* in NRS*f*M literature [4, 6, 154, 185]). This, however, imposes a very strong limitation on out-of-plane rotation of the target, preventing a truly 3D solution. For instance, in neither of the examples in Figures 6.1 and 6.2 could a reference frame be used. In this chapter, the problem of optical flow drift is addressed explicitly.

6.1 Tracking under Non-Rigid Motion

The tracking and modelling approach explored in this chapter is in principle similar to Tracking, Modelling And Gaussian-process Inference Combined (TMAGIC) as presented in Chapter 5 and shown in Figure 5.2 (which for brevity will occasionally be referred to as *vanilla TMAGIC*). There are, however, several important differences that make it more generic and able to handle less constrained scenarios. Foremostly, the 3D feature clouds are considered varying in time (as detailed in Section 6.1.1), while they are fixed in TMAGIC. This makes it possible to model non-rigid deformations of the tracked targets. Secondly, dense points, an additional kind of feature, are employed (see Section 6.1.2). These provide additional constraints for more stable computation of the camera parameters and higher detail of the object model. In addition to the online-learned Gaussian Process (GP) model, similar to vanilla TMAGIC, an explicit polygonal mesh model is produced (Section 6.2.2). It is then used to tackle shot cuts as described in Section 6.2.1, in addition to being one of the motivations and outputs of the algorithm. In cases where the video consists of several discontinuous shots, the model is aligned with the first frame of the new sub-sequence and used to initialise processing of this sequence. This can be imagined as an outermost, re-initialisation

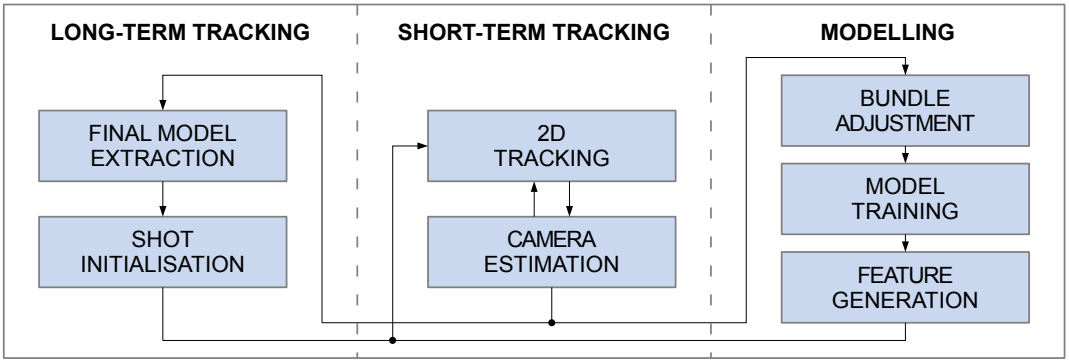


Figure 6.3: Overview of the proposed dense non-rigid tracking and reconstruction scheme.

loop around the TMAGIC processing loop from Figure 5.2 (which is executed on each sub-sequence). It is visualised in Figure 6.3.

Even though the proposed technique does not require any supervision (beyond the video sequence and a single target bounding box), it extends easily to supervised scenarios more traditional in NRS*f*M. Additional correspondences, such as tracks of SIFT features, regressed facial landmarks, *etc.*, can be exploited within the framework to further improve performance.

6.1.1 Representing Time-Varying Shapes

Along with the majority of state-of-the-art approaches, the instantaneous 3D shape (a cloud of features $\mathbf{F}_i^t$) is expressed as a linear combination of K basis shapes. The i-th feature position is combined as:

$$\mathbf{F}_i^t = \mathrm{F}_i \boldsymbol{\alpha}^t = \sum_{j=1}^{K} \mathbf{F}_{i\textcircled{j}} \; \alpha^t_{\textcircled{j}} \,, \tag{6.1}$$

where $\boldsymbol{\alpha}^t = (\alpha^t_{\textcircled{1}}, \alpha^t_{\textcircled{2}}, ..., \alpha^t_{\textcircled{K}})^\top$ is a vector of mixing coefficients, controlling the shape in the t-th frame. The encircled subscripts are used throughout this chapter to indicate indices of shapes within a basis or their coefficients. This instantaneous shape can be

projected to find the equivalent 2D observations:

$$\mathbf{f}_i^t = \Pi\left(\mathbf{F}_i^t | \mathrm{C}^t\right) , \tag{6.2}$$

i.e. every 3D feature $\mathbf{F}_i^t$ is projected by a camera with parameters C^t to create the corresponding 2D feature $\mathbf{f}_i^t$. The camera model used is full projective, however the approach generalises to any other camera model (*e.g.* orthographic, spherical, *etc.*) as long as it provides a unique back-projection (a 2D point to a 3D ray) for any 2D image location. This way the motion for every frame t is separated into the rigid camera motion (captured by C^t) and the non-rigid shape deformation (captured by $\boldsymbol{\alpha}^t$). The common $3K$-rank constraint used extensively throughout the NRS*f*M literature, is equivalent to fixing the number of basis shapes to K.

In this chapter a novel regularisation is introduced which forces the basis shapes to be meaningful modes, or "extremes", of the target's shape. While used extensively in the field of computer graphics [157] (known as blend shapes or morph target animation), this regularisation has not been widely used in the area of NRS*f*M. It becomes important during the optimisation process and is also useful for modelling and visualisation. It is done via the following constraints:

$$\mathbf{1}_K^\top \boldsymbol{\alpha}^t = 1 \quad \text{and} \quad \forall j : \alpha_{(j)}^t \in [0; 1] , \tag{6.3}$$

where $\mathbf{1}_K$ is a vector of K ones. This effectively limits the targets shape to a *convex* combination of the basis shapes: a finite $K-1$ dimensional manifold in the full shape space.

In the first frame, sparse and dense 2D features ($\mathcal{X}^1 = \left\{\mathbf{x}_i^1\right\}$ and $\mathcal{D}^1 = \left\{\mathbf{d}_i^1\right\}$) are extracted as detailed in Section 6.1.2. Optionally, further supervision points $\mathcal{S}^1 = \left\{\mathbf{s}_i^1\right\}$ can be supplied from another source (such as regressed landmarks in the case of a face sequence). These 2D points are backprojected to the dense object model to create 3D feature clouds $\dot{\mathcal{X}}$, $\dot{\mathcal{D}}$ and $\dot{\mathcal{S}}$ (see Section 5.3.3 for details on the initialisation of point

depths). These are then first estimated (like in standard rigid 3D tracking or S*f*M) as single points $\mathbf{F}_i$ and then duplicated K times:

$$\mathtt{F}_i = (\mathbf{F}_{i①}, \mathbf{F}_{i②}, ..., \mathbf{F}_{iⓀ}) , \tag{6.4}$$

with added Gaussian noise (of magnitude negligible compared to the point coordinates) to form the initial basis shapes. The initial mixing coefficients are chosen as:

$$\boldsymbol{\alpha}^t = 1/K \cdot \mathbf{1}_K . \tag{6.5}$$

Similarly to TMAGIC, on every subsequent frame the existing 2D features are first tracked in the new image frame. Using these 2D tracks and their 3D correspondences ($\mathbf{f}_i^t \leftrightarrow \mathtt{F}_i$), the current camera parameters C^t are estimated. The projection equation provides a simple geometric error to be minimised during the rigid camera pose estimation:

$$\mathrm{C}^t = \underset{\underline{\mathrm{C}}}{\arg\min} \sum_{\dot{\mathcal{F}} \in \{\dot{\mathcal{X}}, \dot{\mathcal{D}}, \dot{\mathcal{S}}\}} \sum_{i=1}^{|\mathcal{F}^t|} w_i \rho_1 \left(\left|\left| \mathbf{f}_i^t - \Pi(\mathbf{F}_i^t | \underline{\mathrm{C}}) \right|\right| \right) \tag{6.6}$$

where w_i is a feature weight and ρ_1 is a robust cost function to provide outlier tolerance (similar to [190]). This is minimised using the conditional gradient method. Unless the camera has undergone significant motion, the algorithm then continues processing the next frame.

There are two ways in which the instantaneous 3D shape for each frame could be estimated. Firstly, the unknown set of coefficients $\boldsymbol{\alpha}^t$ could be included as parameters to Equation (6.6) and estimated each frame, jointly with the camera pose. The second approach is to postpone the estimation of the mixing coefficients ($\boldsymbol{\alpha}^t \leftarrow \boldsymbol{\alpha}^{t-1}$) until the next bundle adjustment. Empirically it was found that the latter approach is more stable as it allows more observations and additional regularisation to be used to constrain the non-rigid deformations.

As in the case of TMAGIC, the whole system of point clouds and camera trajectory is optimised by Bundle Adjustment (BA) at regular intervals. Since BA is the most time-consuming stage of the algorithm even with sparse execution (see Table 6.2), the rule of minimal baseline is imposed (Equation (5.4)). Besides the camera trajectory $\mathcal{C}^t$ and basis shapes $\dot{\mathcal{F}}$, the per-frame mixing coefficients up to the current time $\mathcal{A}^t = (\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2, ..., \boldsymbol{\alpha}^t)$ are optimised. Bundle Adjustment is preferred over filtering and other methods since it provides better performance given the same inputs [181]. Due to the novel regularisation, the obtained basis shapes are well constrained and stable. This helps to avoid difficulties with the *basis ambiguity issue* – without additional constraints such as in Equation (6.3), a linear transformation of a set of basis shapes is a new set of eligible bases [208].

The cost function optimised in BA is similar to (5.5) and (6.6), using the constraints of Equation (6.3):

$$\begin{aligned} \min_{\dot{\mathcal{X}},\dot{\mathcal{D}},\dot{\mathcal{S}},\mathcal{C}^t,\mathcal{A}^t} \sum_{u=1}^{t} \sum_{\dot{\mathcal{F}}\in\{\dot{\mathcal{X}},\dot{\mathcal{D}},\dot{\mathcal{S}}\}} \sum_{i=1}^{|\mathcal{F}^u|} w_i \rho_2 \left(||\mathbf{f}_i^u - \Pi(\mathbf{F}_i \boldsymbol{\alpha}^u | \mathrm{C}^u)|| \right) + \\ \Lambda_{\boldsymbol{\alpha}}(\mathcal{A}^t) + \Lambda_{\mathrm{C}}(\mathcal{C}^t) + \Lambda_{\dot{\mathcal{F}}}(\dot{\mathcal{X}}, \dot{\mathcal{D}}, \dot{\mathcal{S}}) \\ \text{s.t.} \quad \mathbf{1}_K^\top \boldsymbol{\alpha}^t = 1 \quad \text{and} \quad \alpha^t_{\textcircled{j}} \in [0;1] \quad \forall j, t\,, \end{aligned} \tag{6.7}$$

where $\mathcal{C}^t$ contains all cameras up to frame t. Since it is vital to update the mixing coefficients $\boldsymbol{\alpha}$ during BA, the combination of basis shapes is expressed explicitly. The projection errors are summed across all the frames seen thus far (a windowed version, limited to a recent history may be considered if speed is an issue). As in TMAGIC (Chapter 5), the robust cost function ρ_2 employed here is the Cauchy loss, as provided by the Ceres Solver [2]. Finally, there are additional priors and regularisations employed. Significant effort is given to these throughout the literature, and sometimes they constitute the major novelty of an article [12, 155].

Firstly the *temporal smoothness of shape* prior is employed. This means the shape cannot change suddenly between two consecutive frames. This is achieved by penalising

fast changes in the mixing coefficients:

$$\Lambda_{\boldsymbol{\alpha}}(\mathcal{A}^t) = w_{\boldsymbol{\alpha}} \sum_{u=2}^{t} ||\boldsymbol{\alpha}^{u-1} - \boldsymbol{\alpha}^u||^2 \tag{6.8}$$

where $w_{\boldsymbol{\alpha}}$ is an appropriate weighting.

To enforce the prior of *temporal smoothness of camera trajectory*, a different cost is chosen. It is desirable to penalise sudden changes in camera parameters without creating an energy inhibiting free camera motion in the world. Therefore the following is used:

$$\Lambda_{\mathrm{C}}(\mathcal{C}^t) = w_{\mathrm{C}} \sum_{u=2}^{t} \begin{cases} 1 & \text{if} \quad ||\mathbf{C}^{u-1} - \mathbf{C}^u|| \geq \theta_{\mathrm{C}} \\ 0 & \text{if} \quad ||\mathbf{C}^{u-1} - \mathbf{C}^u|| < \theta_{\mathrm{C}} \,, \end{cases} \tag{6.9}$$

where θ_{C} is a chosen threshold and w_{C} is a large (relative to the other costs) constant. Consistently with the previously chapter, $\mathbf{C}^t$ is the centre of the camera C^t (a 3D vector).

Finally, in the proposed method, the basis shapes are constrained to be extremes (rare, but feasible instances) of the shape variation. In other words, the instantaneous shapes are required to span a (convex) subspace, tightly bounded by the basis shapes. This renders the method very robust to overfitting (*e.g.* in the case when the chosen K is too large for the observed data, see [119] for a comparison of results with different K used). The first requirement, that the instantaneous shapes span a limited space, is achieved by limiting the $\boldsymbol{\alpha}$ coefficients (Equation (6.3)). The second requirement, that the bounding subspace is tight around the observed poses, stems from the need to decouple the rigid and non-rigid motions, to make the model *as rigid as possible*. Therefore the final regularisation term is introduced:

$$\Lambda_{\dot{\mathcal{F}}}(\dot{\mathcal{X}}, \dot{\mathcal{D}}, \dot{\mathcal{S}}) = w_{\dot{\mathcal{F}}} \sum_{\dot{\mathcal{F}} \in \{\dot{\mathcal{X}}, \dot{\mathcal{D}}, \dot{\mathcal{S}}\}} \sum_{i=1}^{|\dot{\mathcal{F}}|} \sum_{j=2}^{K} \sum_{k=1}^{j-1} \left|\left|\mathbf{F}_{i\textcircled{j}} - \mathbf{F}_{i\textcircled{k}}\right|\right|^2 \tag{6.10}$$

where $w_{\dot{\mathcal{F}}}$ is an appropriate weighting.

An online-learned Gaussian Process (GP) model is used analogously to the case of vanilla TMAGIC. For training, the *canonical* (average over all previous frames) feature clouds are used. It is possible to model the non-rigid deformations within the GP, however this would drastically increase computational costs and did not prove necessary in the experiments performed. It is, however, a very interesting direction for future research.

6.1.2 Dense Features and Drift in Frame-to-Frame Optical Flow

In (rigid) 3D reconstruction, the standard approach is to compute the scene geometry based on sparse feature correspondences and then triangulate a dense cloud as a post-processing step. Alternatively, sparse features can be densified by iterating *expand and filter* steps [55]. However, the proposed approach aims for online processing, returning to earlier frames only for the purpose of texturing the obtained model. Inspiration is thus taken from Non-Rigid Structure from Motion (NRS*f*M) approaches, which often use *dense trajectories* as a basic building block. Therefore, in addition to long-living sparse features (keypoints) as used in the previous chapter, dense (optical-flow based) trajectories are used. While sparse features are robust to drift and provide long-term global geometric consistency (including *loop closures*), dense features bring another benefits, including additional constraints and a higher-detail model. They, however, bring also several challenges. Besides inevitably higher computational demands (*e.g.* seconds per frame for optical flow estimation on its own), there is an inherent problem of drift in using optical flow. Without enforced long-term consistency, tracking via concatenating frame-to-frame optical flow accumulates error quickly. For this reason, concatenation of flow vectors has been in general avoided in literature.

Throughout this thesis, the problems of 2D point tracking and object modelling are addressed jointly. This includes this chapter, where tracking is combined with NRS*f*M. As pointed out in Chapter 2, all state-of-the-art template-free NRS*f*M techniques use precomputed 2D tracks (*i.e.* correspondences between images/video frames) of points

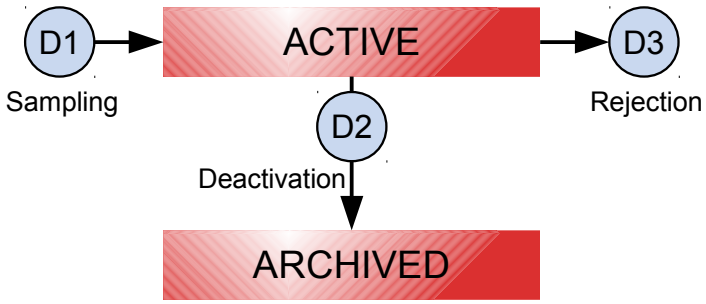


Figure 6.4: Life cycle of dense features.

as their input. These 2D tracks required are either precomputed or taken from known annotations. For this precomputation, it is common to work with a reference template or video frame, against which all other frames are registered. This is important as the concatenation of frame-to-frame correspondences inevitably leads to the mentioned accumulation of errors. This is not a problem if each correspondence is computed independently to a reference frame, however this limits the possible applications of the technique. In contrast, the problem of track drift is addressed here explicitly. Multiple overlapping (both spatially and temporally) sets of dense trajectories are used, in addition to the easily localised sparse trajectories, for long-term consistency. This obviates the need for a reference frame, and makes it possible to process a wider range of scenarios. These include strong rotations and self-occlusions, where there may be zero overlap between the first frame and some frames later in the video.

Figure 6.4 illustrates the life cycle of dense features, with transitions D1–D3. They are sampled from the image on a regular grid (D1) within the initial bounding box (in the first frame), or within the area of the estimated object boundary found by projecting the GP model into the current frame (using $\Pi(\mathrm{M}|\mathrm{C}^t)$, the segmentation is identical to the one of vanilla TMAGIC). After each BA, new dense features are created, spanning the entire area of the projected model, to ensure overlap between the subsets of dense trajectories within $\dot{\mathcal{D}}$. The grid size is based on the image resolution and required density of the model. There is a trade-off between resulting quality and processing time, so the sampling density can be used as a user-defined parameter. The

dense trajectories are estimated on a frame-to-frame basis from a Convolutional Neural Network (CNN) based dense optical flow (FastDeepFlow [201]). This is prone to drift, thus the dense trajectories are kept short. For this reason there is no *invisible* state. Dense features are either successfully tracked during their life span, after which they are transferred (D2) into the *archived* state, or discarded as outliers (using epipolar constraint like in Section 5.2.1, D3). Features in the *archived* state are used for 3D modelling, but are not tracked any more.

There is a significant imbalance between the numbers and importance of sparse and dense features. For this reason, one might want to give them different weights w_i during the process – in camera tracking, bundle adjustment, model creation, *etc*. In all these cases, the dense features are downweighed by a factor 0.01 in the experiments. This reflects their numbers (usually 1–2 orders of magnitude more than that of sparse features) and the notion of dense features being omnipresent but less reliable.

6.2 3D Tracking through Shot Cuts

Tracking thus far was limited to a single sequence where small motions can be tracked at the feature level via Lucas-Kanade (LK). However, to track and build a compound model from multiple sequences or over shot cuts in broadcast video it is necessary to combine information from discontinuous shots.

6.2.1 Model-based Reinitialisation

There are two basic approaches to this problem: 1) processing the sub-sequences independently and merging the models post-hoc (batch processing) and 2) redetecting the object in each new sub-sequence (sequential processing).

In the first case, the sub-sequences (shots) can be each processed independently and the final trajectories and models merged as a post-processing step. While this has the advantage of not needing the re-alignment of the partial model with every new

sub-sequence, there is the challenge of aligning the partial models, which may have only a small overlap (as the parts of the object visible in one shot may be mostly unseen in others). Furthermore, the evidence in a single shot may not be sufficient for even a partial 3D reconstruction (*e.g.* due to insufficient camera motion); this may also adversely affect tracking results. Finally, this approach needs a user-given initialisation for every sub-sequence.

The other possible approach is to re-detect the object at the beginning of a new sub-sequence (analogously to re-detection in long-term tracking, such as in Chapter 3 or [106]) and continue processing from that point, using the information gathered from the previous parts of the footage. This alleviates problems with insufficient information present in any particular shot (except the very first one) and no matching and alignment of partial models is needed. However, the problem of aligning the model gathered thus far with the new sub-sequence needs to be resolved.

Since the aim is to provide an automated approach with minimal user input, the latter approach is employed. To achieve this, a rough alignment of the model with the first frame of the next sub-sequence is first obtained (*i.e.* the camera pose in that frame is estimated assuming a fixed model), using the sparse feature cloud and keypoints independently extracted in the frame. The 3D features from the cloud $\dot{\mathcal{X}}$ were originally detected as keypoints using the same technique. Feature descriptors (*e.g.* SIFT) can therefore be extracted for both and matches between the sets found. On these 2D-to-3D matches, P3P-RANSAC is executed. The inliers (consistent matches) to the camera pose are taken as *visible* sparse features after the reinitialisation.

Given an explicit, textured model (see Section 6.2.2 for details how this model is obtained), the pose can be refined using every model point and not just a sparse subset. This is done (using a conditional gradient method) as follows. The *brightness constancy* constraint is defined as:

$$\mathtt{T}(\mathbf{M}) = \mathtt{I}^t\left(\Pi(\mathbf{M}|\mathrm{C}^t)\right) \qquad \forall \mathbf{M} \in \dot{\mathcal{M}} \text{ where } v(\mathbf{M}|\mathrm{C}^t)\,, \tag{6.11}$$

where $\mathtt{T}$ is the object texture, $v(\mathbf{M}|\mathrm{C})$ is a function indicating the visibility of the point $\mathbf{M}$ by the camera C and $\dot{\mathcal{M}}$ is the set of all the points on the surface of the object model. In other words, for each visible 3D point on the model $\mathbf{M}$, its colour in the texture $\mathtt{T}$ must be the same as the colour in the image $\mathtt{I}^t$ where it is projected using the true (unknown) camera C^t. The brightness constancy is approximated by a first-order Taylor expansion. This gives us a simple equation:

$$\mathtt{T}(\mathbf{M}) \approx \mathtt{I}^t\left(\Pi(\mathbf{M}|\mathrm{C})\right) + \nabla\mathtt{I}^t\mathtt{J}_\Pi\Delta\mathrm{C}\,, \tag{6.12}$$

which is similar to the optical flow constraint [93], including the projection function Π and its Jacobian $\mathtt{J}_\Pi$. Linear least squares are used to solve for an unknown step in the camera parameters $\Delta\mathrm{C}$[1]. Since the formulation is in terms of intensity and the input video consistes typically of colour (RGB) frames, the total number of equations is 3 times the number of sampled pixels. This optimisation is initialised with the solution of the P3P-RANSAC and iterated until convergence, optimising the alignment of the textured model with the first frame of the new sub-sequence. A multi-scale approach is taken, solving on a blurred image in the early steps, to increase robustness against local minima.

The new sequence is processed using this initialisation, without the need for human intervention. The first step is local redetection of sparse features (S3 in Figure 5.3), to maximise the number of visible existing-model features used and therefore the accuracy of camera pose estimation in the first frames. After this sequence has been processed, the new feature cloud is integrated into the original one and a new, more detailed and complete model is created. There is no need for any additional alignment since both feature sets are in the same coordinate frame.

[1]The camera C is used here instead of the camera parameter vector $\mathbf{c}$ for clarity.

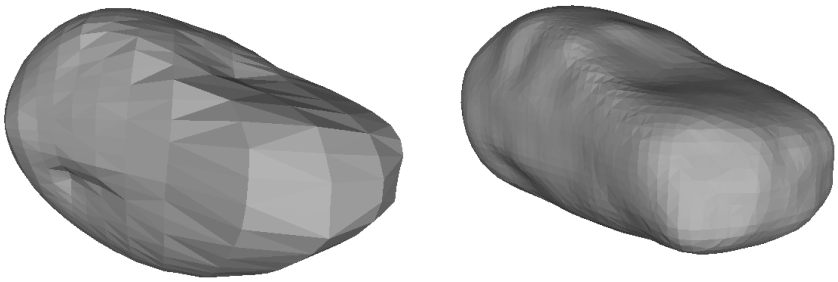


Figure 6.5: Sparse GP model (made explicit) and dense polygonal model.

6.2.2 Final model extraction

Due to its implicit nature, the online-learned GP model is not suitable for the re-alignment of the first frame of a new sub-sequence. While it could be made explicit by sampling and triangulating a set of vertices on the surface, and then textured, the GP model has several disadvantages. Firstly, it is limited to star domains, as mentioned in Section 5.3.2. Secondly, the GP tends to oversmooth the surface both in areas with sufficient data and in extrapolated regions. Finally, the canonical GP model cannot be warped according to the mixing coefficients $\boldsymbol{\alpha}$. For these reasons, an explicit polygonal mesh model is created (directly from the feature clouds), which is then used for the task of shot alignment. Additionally, this model can be, as one of the outputs of the algorithm, used in numerous applications, as with a standard NRS*f*M technique. This model is created at the end of each (sub-)sequence. As for the coarse model, sparse features in both the *active* and *invisible* states and dense features in the *archived* state are used for modelling. See Figure 6.5 for an example of an extracted model and the difference to the sparse GP model.

There are numerous approaches to reconstruct a surface model from a set of scattered points, such as marching cubes [132], marching triangles [86], ball pivoting [17] or methods based on Moving Least Squares [167]. For an extensive study a recent survey [202] is recommended, evaluating a broad range of techniques and summarising their properties. In this work, the Poisson reconstruction [108, 109] is used, in its screened variant. Input surface normals are provided by the GP model at locations of

the features from $\dot{\mathcal{F}}$ (by sampling points in a very close neighbourhood and fitting a tangent plane). These are interpreted as samples of a vector function $\vec{\boldsymbol{\beta}}$. The implicit *indicator function* χ (resolving the inside/outside problem) is found as a solution of the Poisson equation, such that its gradient $\nabla\chi$ approximates the sampled normals $\vec{\boldsymbol{\beta}}$:

$$\Delta\chi \;=\; \nabla\cdot\nabla\chi \;\approx\; \nabla\cdot\vec{\boldsymbol{\beta}}\,. \tag{6.13}$$

Poisson reconstruction provides a global solution to this approximation and hence to the reconstruction problem and therefore smoothly fills even large gaps in the surface. This is the main reason why it is used in this work, as a watertight surface is required. The set of all the points on the model (χ_0-isosurface) is referred to as

$$\dot{\mathcal{M}} = \{\mathbf{M}|\chi(\mathbf{M}) = \chi_0\}\ , \tag{6.14}$$

where χ_0 is chosen as the mean χ of the training points:

$$\chi_0 = \frac{1}{|\dot{\mathcal{F}}|}\sum_{\mathbf{F}_i\in\dot{\mathcal{F}}}\chi(\mathbf{F}_i^*)\,. \tag{6.15}$$

A collection of smoothed model vertices $\mathbf{V}$ may then be selected from the χ_0 isosurface.

Since the task given is tracking and reconstruction of *time-varying* shapes, the model needs to be non-rigid, as are the feature clouds. The transfer of the deformation is achieved as follows. Firstly, the model is created using the canonical shape (averaged over all observed poses), analogously to the GP model training. Every canonical vertex $\mathbf{V}^*$ of the polygonal mesh model is assigned a fixed set of features $\dot{\mathcal{N}}(\mathbf{V}^*)$ in the cloud, determined as its k nearest neighbours (k is a small constant, set to 3 in the experiments in this chapter). Since for each 3D feature the offset of basis poses (from the canonical position $\mathbf{F}_i^*$) is known (*i.e.* $\mathbf{F}_{i\textcircled{j}} - \mathbf{F}_i^*$), the offset of basis poses of each vertex can be computed as a mean of offsets of its k nearest neighbours ($\dot{\mathcal{N}}$). Since the topology of the model does not change when performing the warp, its basis shapes differ only by the vertex coordinates: these are computed by applying the offsets to the canonical

model:

$$\mathbf{V}_{\textcircled{j}} = \mathbf{V}^* + \frac{1}{k} \sum_{\mathbf{F}_i \in \dot{\mathcal{N}}(\mathbf{V}^*)} \mathbf{F}_{i\textcircled{j}} - \mathbf{F}_i^* \,. \tag{6.16}$$

As mentioned, the texture is a vital part of the model for shot cut reinitialisation, visualisation and further applications. As the model is only provided at the end, the sequence is processed again in the second pass. The model is warped into the appropriate shape for each frame using previously estimated mixing coefficients, and the texture of visible mesh faces is updated. For each point of the texture the colour is computed as the median of the observations from all frames (cameras $\mathcal{V}(\mathbf{M})$) where the particular point was visible:

$$\mathtt{T}(\mathbf{M}) = \underset{\mathrm{C}^t \in \mathcal{V}(\mathbf{M})}{\mathrm{median}} \left(\mathtt{I}^t \big(\Pi(\mathbf{M}|\mathrm{C}^t) \big) \right), \tag{6.17}$$

$$\mathcal{V}(\mathbf{M}) = \{ \mathrm{C} \in \mathcal{C} \,|v(\mathbf{M}|\mathrm{C}) \} \,. \tag{6.18}$$

To avoid extracting unreliable information, *e.g.* near the edges of the object, the angle between the surface normal and the ray from the camera centre is required to be below a given threshold (85° in the tested implementation). The texture is extracted at a resolution specified by the user. With precise camera tracking and a high-detail shape model, it is possible to sample points at resolutions exceeding the original video, leading ultimately to a 3D super-resolution model.

6.3 Experimental Evaluation of Dense Modelling

In this section, the proposed framework is evaluated experimentally. Firstly, the benefits of using dense features are demonstrated in the scenario of rigid simultaneous tracking and reconstruction. This includes experiments on long sequences containing shot cuts. Secondly, it is shown how the improved object model allows tracking in scenarios containing non-rigid objects in both synthetic and real video sequences.

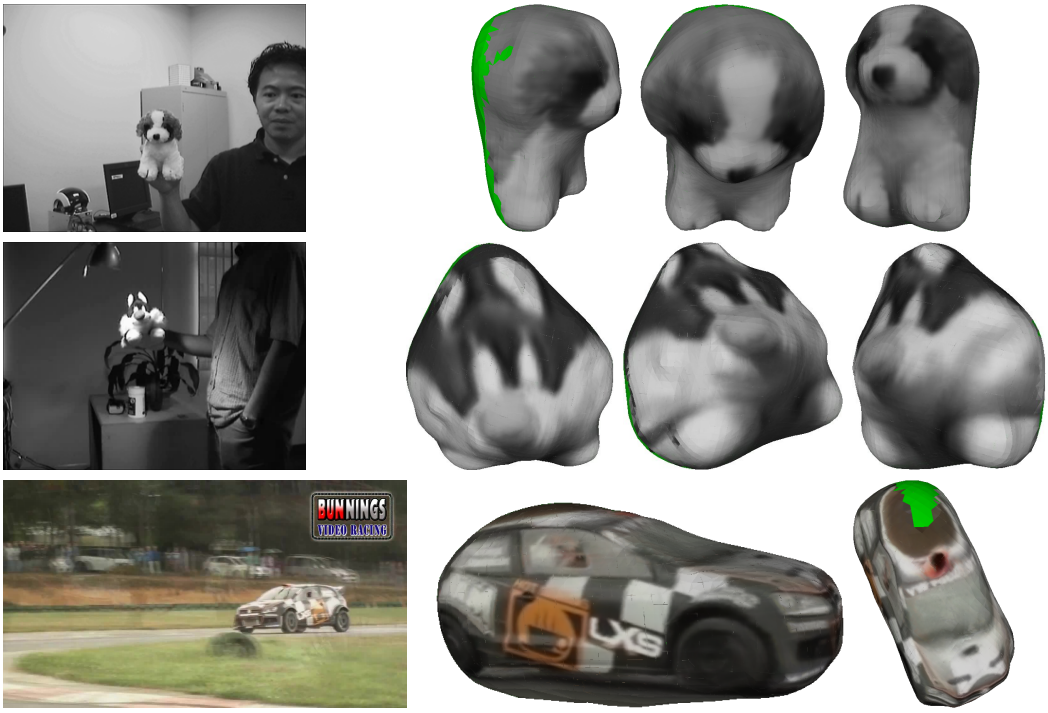


Figure 6.6: Resulting models on sequences from literature with varying resolution. From top to bottom: DOG1 (90 px [27]), SYLVESTER (50 px [166]) and RALLY-VW (410 px, Chapter 5).

6.3.1 Rigid Object Experiments

To evaluate performance of the proposed algorithm, it is first tested in the short-term scenario, *i.e.* one continuous video where the target is fully visible in all the frames. See Figure 6.6 for example frames and results on several video-sequences used in recent tracking publications. Model regions, which were unseen in the original sequence and are therefore untextured, are marked by bright green colour. Since this subsection tests rigid reconstruction, the number of basis shapes is fixed to one.

For comparison, reconstructions made by the *CMP SfM WebService* [81] is shown in Figure 6.7. In the first row, the reconstruction from the raw video can be seen, with only a fraction of the car partially reconstructed. When supplied with Ground Truth (GT) bounding boxes at every frame to segment out the background and focal length estimates (provided in $\mathcal{C}$), it yields the model shown in the second row. The

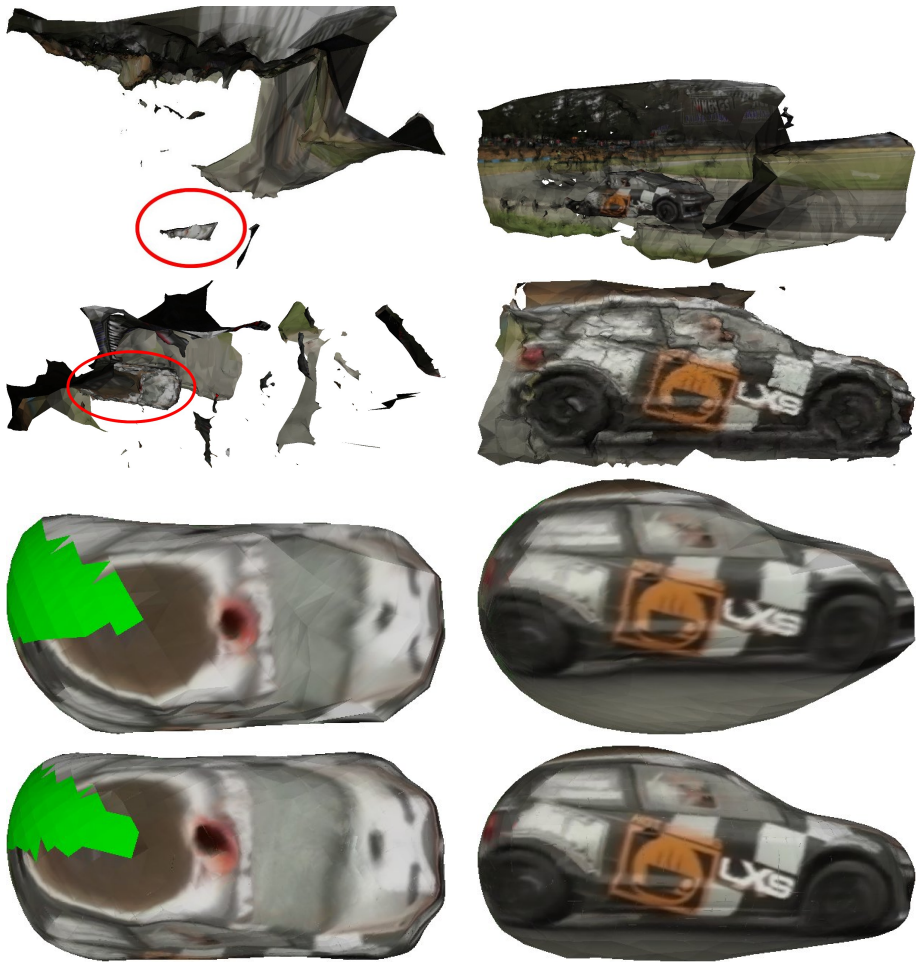


Figure 6.7: Results on the RALLY-VW sequence. Top and bottom rows show the top and side views of models from the same method (the right column is manually cropped for CMP results). Top to bottom: CMP SfM WebService [81] (raw video input); CMP SfM WebService [81] with added information; textured sparse GP model; full final model.

proposed approach (last row) returns significantly cleaner results automatically without such extensive user interaction. The textured sparse GP model (roughly equivalent to vanilla TMAGIC) is shown in the third row. An example of an untextured resulting model is shown in Figure 6.5.

Additionally, in Figure 6.8 several models are shown, obtained from four different sub-sequences of the rally sequence HILLCLIMB. In all of these sequences, there is

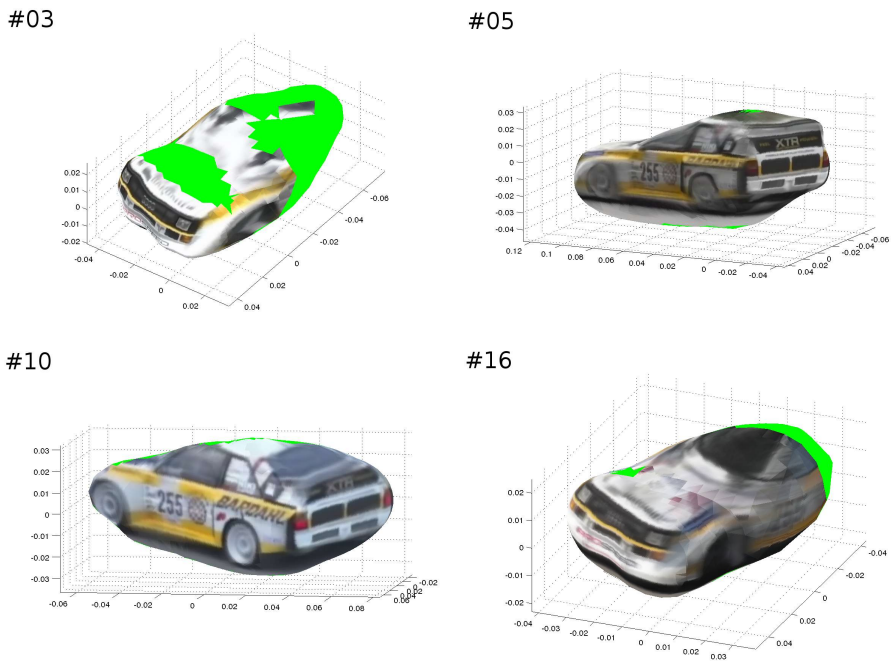


Figure 6.8: Resulting models on several sub-sequences from the HILLCLIMB sequence.

strong background clutter (distractors), which makes conventional S*f*M approaches fail. Notice, however, how inaccurate and incomplete the models are, compared to the model combined from multiple shots in Figure 6.10. The original data of the HILLCLIMB sequence, together with additional results, are available online [119].

To test the ability of the proposed algorithm to model the geometry of long sequences including shot cuts, the modelling results are shown on the HILLCLIMB sequence. As new shots are added to the reconstruction, the model becomes slowly more detailed and complete. See Figure 6.9 for a visualisation of the shot cut handling and Figure 6.10 for the results. The first sequences track the car mostly from the front, *i.e.* the rear parts are missing information. However, the later addition of sub-sequences covering the rear of the car incorporates these missing regions. One region which remains unmodelled after all the sequences are processed is the bottom of the model, which is completely unseen. This is, however, an inherent property of this dataset and cannot be addressed without human intervention.

Figure 6.9: Shot initialisation after the first three cuts in the HILLCLIMB sequence. Left: the last frame of a previous shot with the model overlaid; middle: the first frame of a new shot; right: the model is registered with the new frame.

(a) After 1 sequence. (b) After 2 sequences. (c) Final model.

Figure 6.10: Progressive growth of the model after processing particular shots from HILLCLIMB. The first two models and the final model are shown.

A natural way of processing the sequence is in temporal order, *i.e.* the first shot first and then the rest as they follow. However, changing the ordering can be beneficial. One such heuristic is used for the ordering of the sub-sequences. After breaking the original

sequence into shots, the longest one is first used, as it can be expected to yield the most complete model. Subsequent sequences are then chosen in the order of decreasing quality of alignment, *i.e.* the P3P-RANSAC is executed (as detailed in Section 6.2.1) on all the sequences and the one with the highest number of inliers (successfully matched features) is chosen.

6.3.2 Synthetic Non-rigid Experiments

An initial quantitative evaluation on the synthetic CubicGlobe dataset is now performed. This sequence contains a rotating globe which repeatedly warps into cubic shape and then back to sphere. The performance of the proposed algorithm is reported by a number of quantitative measures, comparing against several state-of-the-art template-free NRS*f*M techniques which have source code available online. These include Bilinear modeling via Augmented Lagrange Multipliers (BALM) [37], using Augmented Lagrange multipliers to solve for the bilinear factorisation problem in the presence of missing data, Local Isometric and Infinitesimally Planar NRS*f*M (LIIP) [28], using isometric deformation instead of basis shape combination and SoftInex [197], which employs the material inextensibility prior as a soft constraint in its energy function. These tests measure three important properties. Firstly the accuracy of modelling: the fit of the estimated basis shapes to a perfect cube/sphere (the deviation of each reconstructed feature is measured, relative to the model size). The second property is the accuracy of the 3D tracking. For this the camera rotation error is measured in the angle-axis representation (for each camera pose, the axis error is measured as the angle between the reported and GT axis and the angle error as the absolute deviation from the GT). Since the global coordinate frame is not fixed, the rotation is measured as relative to the first frame. Finally, the depth error of the instantaneous point locations is measured, *i.e.* accuracy of deformations. This is measured as Spearman correlation between the measured and ground-truth depth, to overcome the inherent scale ambiguity of the 3D reconstruction (the reconstructed and GT point cloud depths are used as datasets with 1-to-1 correspondences to obtain every per-frame correlation). The

sequence is available online including all ground truth information, such as shape, trajectory, depth, *etc*. along with additional results [119]. In this experiment, the number of basis shapes K was set to two, to capture the two extremes of the shape.

It is important to note that all three state-of-the-art comparison methods use the orthographic camera model to simplify computation. This makes it more challenging to evaluate the camera trajectory and depth correlations against the ground truth. To resolve this issue a state of the art Perspective-n-Points (PnP) algorithm [51] with outlier rejection was used to find the optimal projective camera pose, corresponding to the reconstructed 3D point clouds.

Since there are no ground-truth point tracks for this sequence, the state-of-the-art techniques were provided with tracks obtained by the proposed technique. LIIP and SoftInex do not handle occlusions; therefore they were only provided a limited portion of the sequence (the first 50 frames), with only those tracks, which were visible in all the 50 frames. Furthermore, neither of the techniques directly provide meaningful basis shapes. Therefore for the shape comparison the instantaneous shape was used from frames 30, 90 and 150 for cube, and 1, 60, 120 and 180 for sphere (where the GT shape is pure). The table contains the best observed performance for each of these.

See Table 6.1 for results. It is clearly visible that BALM failed completely on this sequence, producing large reconstruction and camera rotation errors. Similarly, the depth reported by BALM is not correlated to the GT depth. The results of LIIP are significantly better, with much lower reconstruction errors and rotation error reduced by an order of magnitude, compared to BALM. The average depth correlation is 0.5. SoftInex produces better 3D reconstructions, with error comparable to the proposed method (although of only one side of the object since it does not handle occlusions). The camera pose is less accurate than that of LIIP, the reported depth is nevertheless strongly correlated with the ground truth.

The results of SoftInex demonstrate an interesting phenomenon. While the per-frame point depth, returned by the algorithm (and used to infer the non-rigid shape)

	Cube (%)	**Sphere** (%)	**Axis** (°)	**Angle** (°)	**Depth** (%)
BALM	51±54	14±12	52.6±28.0	56.9±39.7	5±26
LIIP	29±20	5± 5	12.9±20.4	8.1± 4.4	50±29
SoftInex	4± **3**	**3± 2**	22.3± 8.5	11.8± 8.1	79±13
Proposed	**2± 3**	**3**± 3	**0.4± 0.8**	**3.5± 1.4**	**95± 3**

Table 6.1: Quantitative tracking and reconstruction results on the CubicGlobe sequence. See the text for discussion.

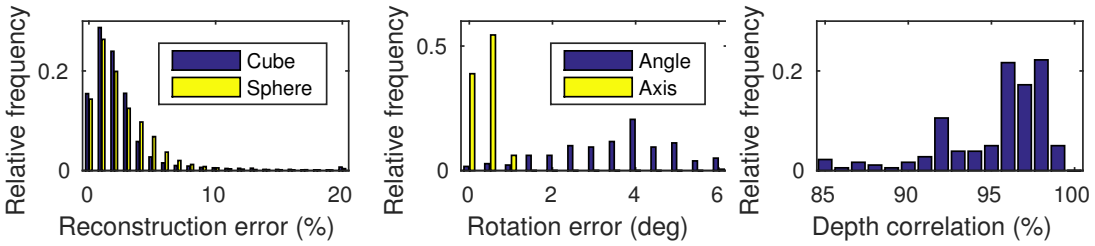


Figure 6.11: Distribution of errors on the CubicGlobe sequence.

is realistic, it is "flipped" in the z-direction (in the camera coordinate system) for some frames, *i.e.* the object side is turned inside out. This is probably due to the lack of a temporal smoothness constraint. For a fair comparison, it was necessary to detect and correct this during the experiments. Without this, the results of SoftInex are significantly worse, *e.g.* the mean depth correlation drops to 16 %. When using the proposed method, the reconstructed models cover the whole object (as visualised in Figure 6.1) with very low errors. The camera rotation demonstrates even better performance, with error reduced by an order of magnitude due to its inherent ability to perform tracking and modelling simultaneously. The depth estimated by the proposed method is nearly perfect, reaching 95 % correlation with the observed depth. The distribution of the errors can be seen in Figure 6.11.

See Figure 6.12 for visualisation of the obtained mixing coefficients $\boldsymbol{\alpha}^t$ in the first 180 frames of the CubicGlobe sequence. The shape is changing from spherical to cubic linearly, which was closely captured by the coefficient change. Notice the "cropped" peaks, a typical artefact of the proposed method. This is caused by the compactness prior, forcing the basis shapes (spherical and cubic in this case) to lie close to each

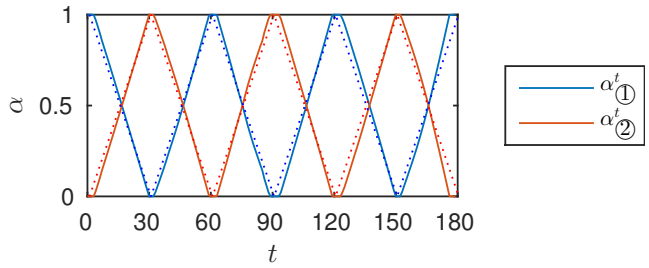


Figure 6.12: Mixing coefficients $\boldsymbol{\alpha}^t$ in the CubicGlobe sequence. The dotted signals show the GT.

	Tracking	**Reconstruction**	**Modelling**	**Total**
BALM (s)	1 357	248	372	1 977
LIIP (s)	1 357	18 768	372	20 497
SoftInex (s)	1 357	5 416	372	8 130
Proposed (s)	1 357	3 234	372	6 230
	22 %	52 %	6 %	100 %

Table 6.2: Times of processing the first 180 frames of the CubicGlobe sequence. The last row does not sum up to 100 % due to various overhead computations, visualisation, I/O wait, *etc*.

other and hence being unable to truly capture the very extremes of this sequence. It, however, does not significantly affect the overall performance, as can be seen in both the qualitative (Figure 6.1) and quantitative (Table 6.1) results.

Table 6.2 shows a breakdown of the execution speed for the different algorithms. It should be reiterated that the competing state-of-the-art techniques use point tracks provided by the proposed method. Therefore the times for tracking and GP model training (necessary for tracking) should be included in their timings for a fair comparison. These are marked in blue. It is also worth noting, that the time for LIIP and SoftInex was consumed in computing reconstruction from only 260 tracks in 50 frames, while the others from nearly 20 000 tracks in 180 frames. BALM also has scaling issues in terms of memory usage. Operating on the same point tracks used in the proposed approach, BALM consumed more than 200 GB of RAM, two orders of magnitude more than the proposed algorithm.

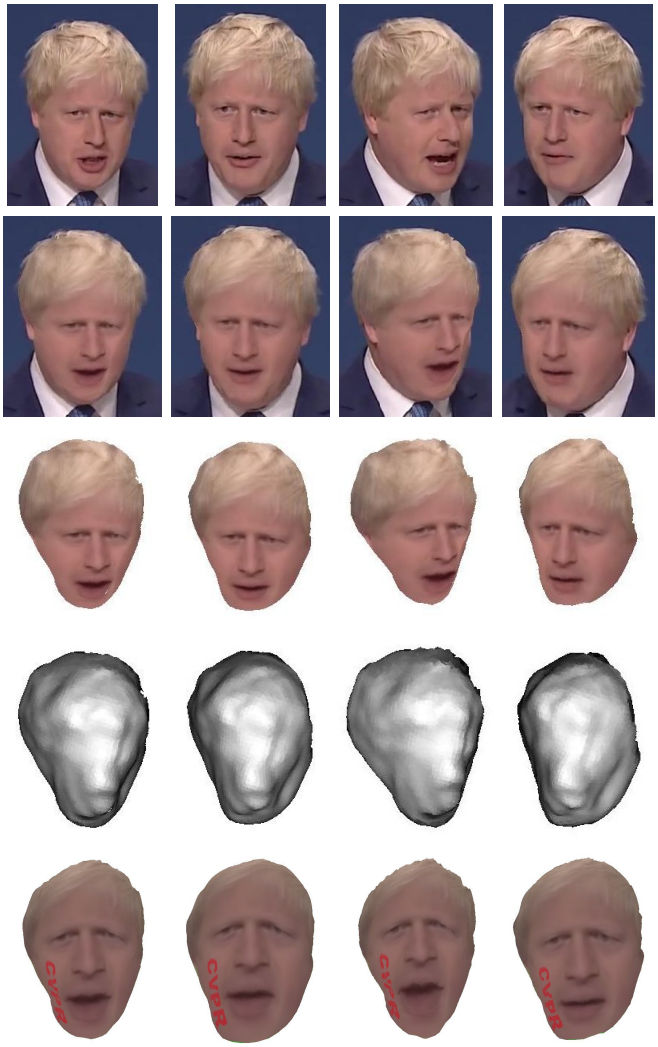


Figure 6.13: Example of modelling results on the 300VW:002 sequence. From top to bottom: original video frames; video frames overlaid with the instantaneous models; the instantaneous model in its estimated pose (textured and untextured); the instantaneous model facing forward with modified texture.

6.3.3 Real-Data Non-rigid Experiments

To show the performance of the proposed algorithm on real data, the recently published 300VW dataset [29, 100, 192] is used. See Figures 6.13 and 6.14 for example of modelling results (obtained with $K = 3$). The models are similar to the results generated by state-of-the-art NRSfM techniques. Although they show slightly lower

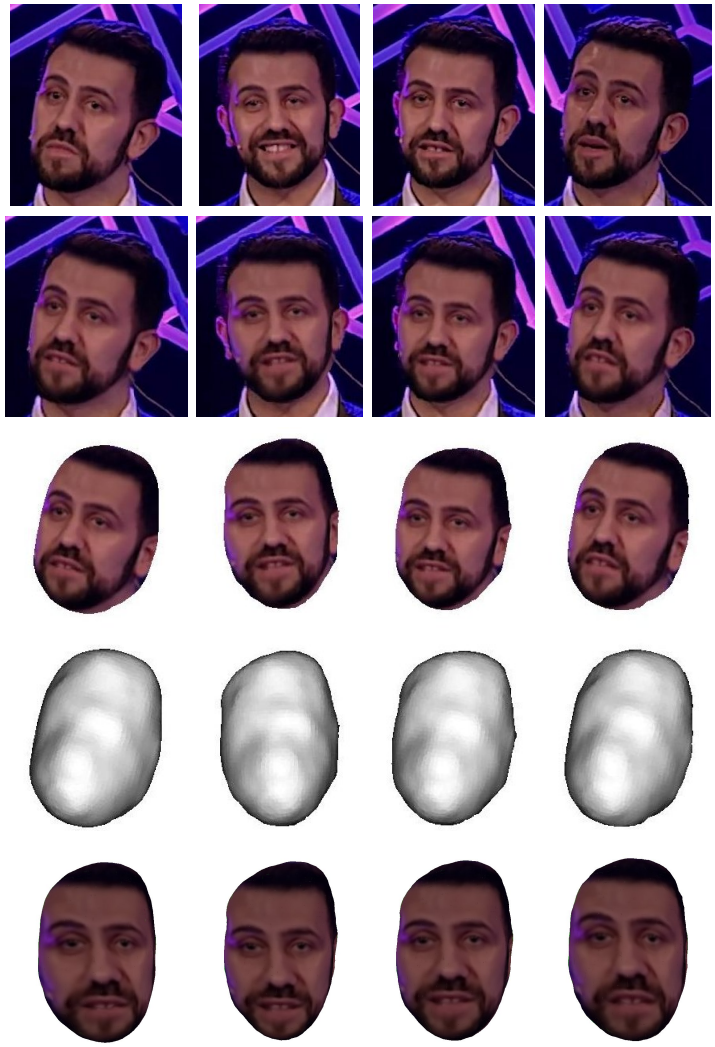


Figure 6.14: Example of modelling results on the 300VW:004 sequence. From top to bottom: original video frames; video frames overlaid with the instantaneous models; the instantaneous model in its estimated pose (textured and untextured); the instantaneous model facing forward.

levels of detail, it should be re-emphasised that the problem addressed here is much more challenging: the fully unsupervised simultaneous tracking and modelling. As can be seen, the proposed method captures the rough shape of the target with variations in time. In Figure 6.15 the basis shapes are shown for one of the sequences. Although exaggerated, they are all valid face models which could be observed during the sequence. Finally, Figure 6.16 shows the resulting reconstruction of the proposed technique and

Figure 6.15: Basis shapes in the non-rigid model produced by the proposed algorithm.

the trajectory of the model in the shape space. It is worth re-iterating that the shown texture is fixed to particular polygons in the produced mesh model, and the variations visible in the textured results are caused by warping of the model and not re-texturing from a particular frame.

In Figure 6.17, the performance of the proposed technique is compared against BALM on the 300VW:002 sequence. When given only the sparse facial landmarks, BALM performs similarly to the proposed technique. However, it has difficulties integrating noisier observations; when BALM is provided with the denser internal trajectories generated by the proposed method, it fails to produce a reasonable reconstruction. In contrast the proposed technique is able to fuse these, to produce a far more detailed reconstruction than from the landmarks alone. A similar situation was also observed with LIIP. Once again, this demonstrates the value of performing the tracking and modelling simultaneously.

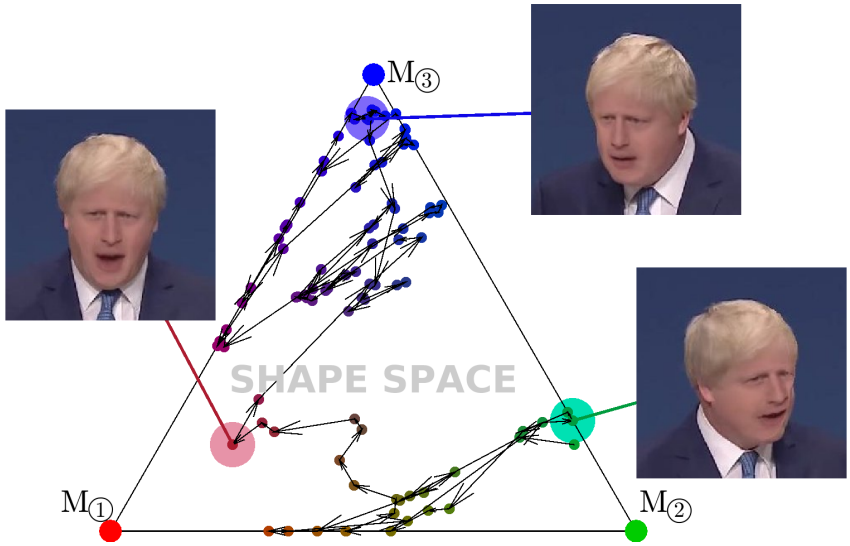


Figure 6.16: Reconstructed model overlaid over frames from the 300VW:002 sequence. The shape space visualises the weighted combination of the basis shapes.

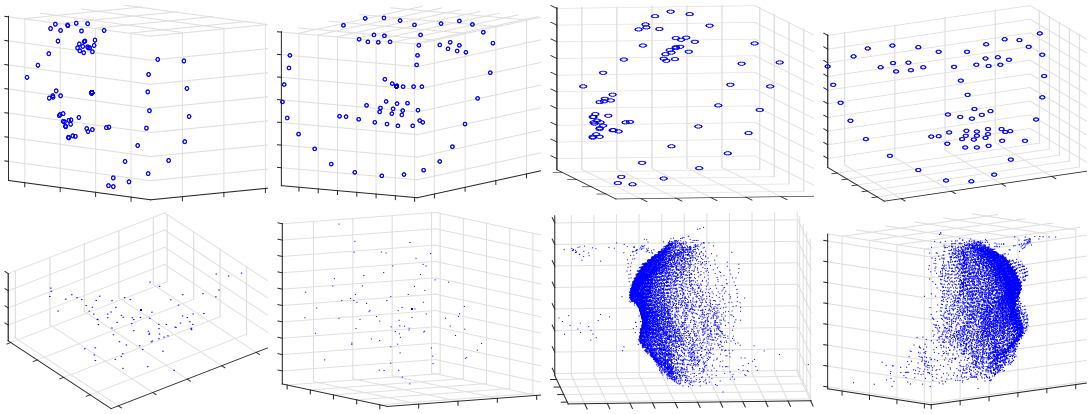


Figure 6.17: Comparison of BALM (left 2 columns) against the proposed technique (right 2 columns) on the 300VW:002 sequence. Results are shown using only the sparse supervision (top row), and using the sparse supervision with additional densely estimated trajectories (bottom row).

It is not only the reconstruction which benefits from the proposed joint approach. Using a non-rigid model can significantly improve tracking results as well. This is demonstrated in Figure 6.18, where results compared between rigid and non-rigid tracking. For non-rigid objects, a "centre" is ill defined. Therefore, a face-tracking scenario is used and the accuracy of landmark tracking is measured. The error is defined as the

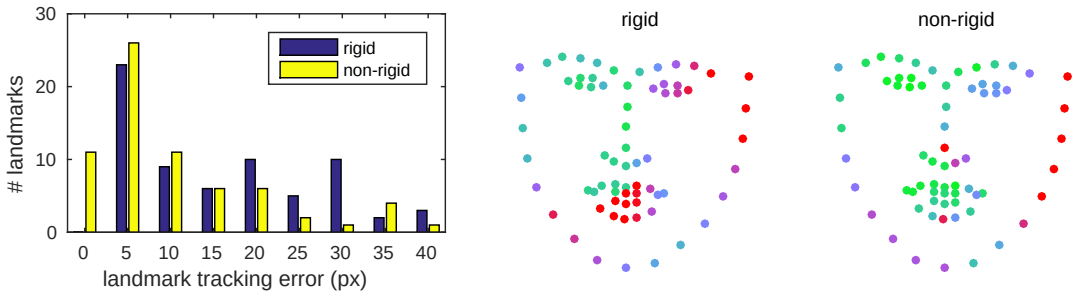


Figure 6.18: Landmark tracking error on the 300VW:002 sequence, when using rigid (*i.e.* vanilla TMAGIC) and non-rigid tracking and reconstruction. Left: error histogram, right: landmark error from low (green) to high (red).

distance between the GT and the landmarks tracked using the non-rigid 3D model. For each landmark, the error is averaged over all frames.

Firstly, 3D landmarks were obtained by back-projecting the manually annotated landmarks from the first frame into the 3D model space (Section 5.3.3) and in the non-rigid variant warped as described in Section 6.2.2. These were then projected back into each consecutive frame using the obtained camera parameters and mixing coefficients. It can be seen that the proposed method has a fraction of landmarks tracked with near-zero error, while the rigid case has no "perfectly tracked" landmarks. Additionally, the rigid variant has a significant portion of landmarks tracked with errors around 20–30 px (mostly near the mouth where the non-rigid deformation is the most pronounced). On average, the tracking error is reduced from 17.4±14.1 to 10.8±10.5 px by using a non-rigid model.

In Figures 6.19 and 6.20, the performance of the proposed technique in the fully unsupervised scenario is explored, on the FACE [58] and T-SHIRT [194] sequences (with $K = 4$). As can be seen, all estimated target poses are feasible despite the lack of supervision. It is also obvious from the second rows that estimates of the rigid motion (*i.e.* the camera pose) are accurate. Table 6.3 brings quantitative comparison on the T-SHIRT sequence. The results indicate the proposed approach is competitive with state of the art, even though it does not use a template or another kind of prior knowledge and operates directly on the raw RGB images.

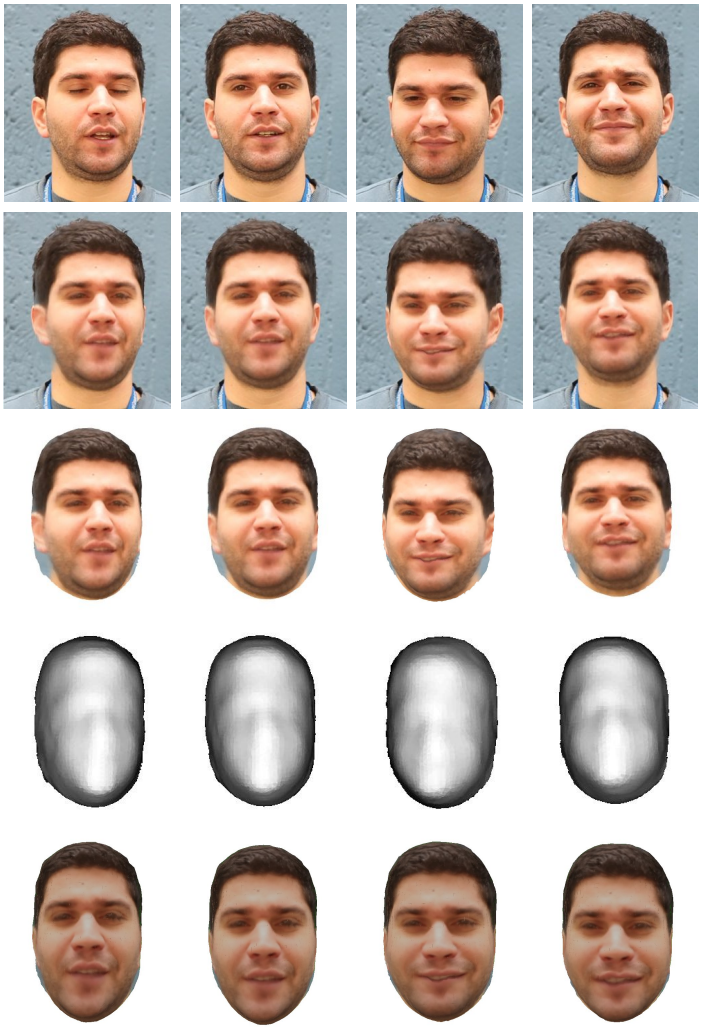


Figure 6.19: Example of modelling results on the FACE sequence. From top to bottom: original video frames; video frames overlaid with the instantaneous models; the instantaneous model in its estimated pose (textured and untextured); the instantaneous model facing forward.

	PCA [194]	Uncon. LVM [194]	CLVM [194]	DDD [213]	Proposed
Error (mm)	18.44	15.50±1.78	14.79±0.90	7.05	17.82±4.72

Table 6.3: Quantitative results on the T-SHIRT sequence.

In Figure 6.21, the canonical T-SHIRT model (before cropping to contain only the region of interest) is shown in detail. Notice the creases near the top of the model,

Figure 6.20: Example of modelling results on the T-SHIRT sequence. From top to bottom: original video frames; video frames overlaid with the instantaneous models; the instantaneous model in its estimated pose (textured and untextured); the instantaneous model facing forward.

caused by the way the T-shirt is held. Finally, Figure 6.22 shows the basis shapes for the FACE sequence, automatically identified by the proposed method (with a wireframe mesh overlaid to help visualise the 3D shape).

6.4 Closing Remarks on Dense Tracking and Modelling

In this chapter, an algorithm for automated tracking and reconstruction of non-rigid targets in unconstrained, unstructured, discontinuous videos was presented. Besides

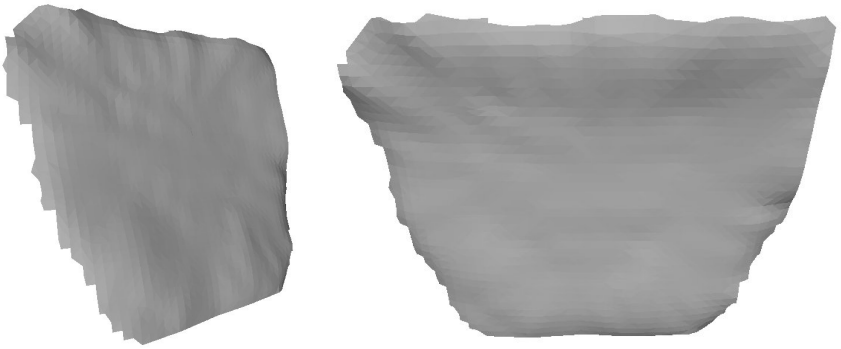


Figure 6.21: Details of the model obtained (directly) from the T-SHIRT sequence.

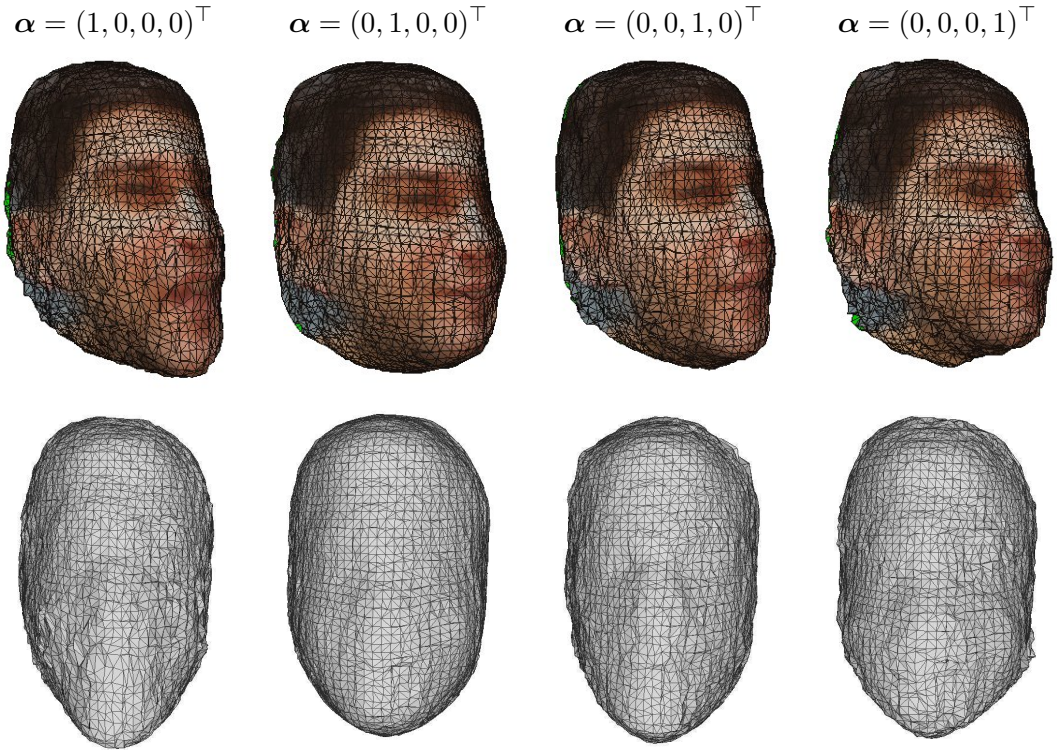


Figure 6.22: Basis shapes obtained from the FACE sequence.

the 3D camera trajectory, it provides a textured model of an a priori unknown object with the only user input being a single bounding-box initialisation of the object to be reconstructed – even from videos consisting of several shots. It actively avoids modelling the scene background, removing the assumption that the object of interest covers most

of the frame. Use of tracking techniques also removes several important limitations of conventional NRS*f*M methods, most importantly it provides robustness against strong target rotation and self-occlusion. The presented algortithm can be extended trivially to different camera models and additional priors, constraints and regularisations.

The modelling approach, as presented, has several limitations. Resolution of the video-sequence is one limiting factor. While it is possible to obtain a 3D camera trajectory and a coarse model of objects at resolutions as low as 320×240 px (*e.g.* Sylvester in the eponymous sequence is approximately 50 px in size – see Figure 6.6), the resulting models are blob-like with low levels of detail; therefore higher resolution is recommended. Most of the examples shown in this chapter come from videos with the resolution 1280×720 px (with target size in hundreds of pixels). Additionally, the reconstruction is thus far limited to star domain shapes (technically only in the case of the sparse model, however the final model is influenced by this as well). This is however a limitation of virtually all current model-free NRS*f*M approaches. This limitation may be addressed in the future work, for instance using an intermediate reparameterisation layer.

Chapter 7

Conclusions and Future Work

7.1 Summary and Conclusions

This thesis examined numerous failure cases endemic to visual object tracking. These stem from several sources. They include challenges caused by the properties of the tracked target, such as shape variation, low texture or transparency. Another set of failures is induced by the scene within which the target exists; these include background clutter, distractors and occlusions. Finally, there is a class of issues related to the capture circumstances. These include 1) technical issues, such as video resolution, temporal resolution (causing motion blur and large inter-frame displacement) and incorrect camera focus or exposure, and 2) issues caused by variation in the capture settings, such as viewpoint, illumination or target distance. These challenges need to be tackled by any tracker used in realistic scenarios, otherwise it will be limited to applications constrained to laboratory conditions. Some of these challenges were explored in higher detail in this thesis and solutions were presented.

The issues, caused by the nature of the object appearance (such as low texture or transparency), often lead to failure of conventional trackers. In Chapter 3, it was shown how edge-based features can be used to tackle this kind of challenge. Since edge points often suffer from the aperture problem, the idea of virtual corners was introduced and

examined. This leads to reliable tracking, independent of the illumination, texture or appearance of the target object.

Edge-based tracking gives a reliable estimate of frame-to-frame tracking, however does not provide long-term stability. Similarly, numerous state-of-the-art trackers focus on short-term tracking, where a fully visible object is tracked for a limited amount of time. However, many applications require tracking in a long-term scenario. Two challenges arise in such a case. Firstly, the long duration of the tracking imposes strong emphasis on the robustness of any tracker to error accumulation, *i.e.* drift. This creates the need for a long-term tracker to be conservative about its model updates. Secondly, the target may disappear from the video either due to (full) occlusion by another part of the scene, leaving the scene (*i.e.* occlusion by the frame boundary) or a shot cut. After this happens, a tracker needs to detect this, and then resume tracking when the target object reappears. These challenges were addressed within Chapter 3. A tracking approach based on the previously mentioned edge-based correspondences was combined with an effective redetection scheme. It was demonstrated as successful, being placed near the top of several benchmarks.

There are issues related to the scene properties, such as background clutter and distractors. For this reason, trackers dedicate significant effort to object segmentation and actively ignoring the background. The explored edge-based approach is not an exception in this. However, the rest of the scene can often contain useful information. This information can be exploited to increase the robustness and accuracy of a tracker in difficult settings, such as full occlusion or rapid motion. This was explored within Chapter 4, with the focus on causal relationships. Two types of causal relationships were examined, between the camera motion and the object motion, and between different elements of the scene. A method was introduced, which can be employed with any tracker, supplying it prior information to improve its performance.

As indicated by recent benchmarks, many failures stem from problems with viewpoint variation. Conventional trackers attempt to model the viewpoint-induced ap-

pearance change as a change in the model. However, this is often not possible in cases of rapid out-of-plane rotation, since the ability to quickly adapt the model necessarily decreases resistance to drift. One major breakthrough of this thesis is in opposition to this standard (but inadequate) solution. Instead, the 3D motion is perceived as a 3D motion, and the object is modelled as a 3D object in a 3D world. Following this, the second half of this thesis investigated the possibilities for 3D tracking. In Chapter 5, it was demonstrated that 3D tracking, while competitive in the standard 2D-tracking scenario, is much more general and extends into cases where 2D algorithms fail.

Finally, a further step was taken to increasing generality and removing limitations, when this idea was extended to the task of non-rigid tracking and modelling. An improved model was introduced, allowing 3D tracking of non-rigid objects through shot cuts. The use of dense, optical-flow based features increases the detail of the model and stabilises the estimation. Chapter 6 combined 3D tracking and non-rigid reconstruction, and was shown as competitive to online Non-Rigid Structure from Motion (NRS*f*M) techniques, providing a time-varying 3D model directly from video, without template, reference frame or any other kind of prior knowledge often used in NRS*f*M. Such a combination of simultaneous tracking and reconstruction benefits both: reconstruction by effectively segmenting the target and thus relaxing requirements on the input data; and tracking by providing an improved target model to better span the space of possible variations.

7.2 Possible Future Directions

This section presents several suggestions of possible directions of future research within the topics of this thesis. These include technical improvements of the proposed tracking approaches, with the potential to boost performance. Furthermore, several directions of a more principled nature are discussed, possibly reaching beyond the sub-area of visual tracking.

There are several possible directions where this work could be extended. The majority of this thesis is concerned with feature-based tracking, where trackers maintain clouds of features to estimate the object trajectory. These features are validated based on different measures, but in general they are either retained or discarded. The work on both 2D and 3D tracking might benefit from a soft feature management, where the feature quality could be judged based on its history.

In both 2D and 3D, tracking of a single object was considered. A naïve extension to allow multi-object tracking would be trivial. Since neither requires any pre-learning and the object properties are modelled online, it is possible to simply run multiple instances in parallel. A more interesting direction of further research would be advanced reasoning about correlation (or causal links) or occlusion between these multiple tracked objects.

In TMAGIC, the internal object model is thus far constrained to star domain shapes. Technically, this is true only in the case of the sparse GP model, however the final model can be influenced by this as well and could be addressed in future work, for instance using an intermediate reparameterisation layer. A possible solution would be to fit a non-radial model (such as 2D B-spline surface) to the 3D feature cloud and then the GP model can be applied in the spline control coordinates.

Furthermore, it would be interesting to see the influence of the causal model on the trackers presented, when applied inside the trackers (as mentioned in Section 4.8). Preliminary experiments were carried out with variable results, therefore further investigation of integration with the trackers is needed. In the case of LT-FLOTrack, this would mean predicting the object position in 2D, as discussed in Chapter 4. In the case of TMAGIC, the situation is slightly more complicated. The prediction could be either in 2D, to help with the underlying feature tracking (which is one of the limiting factors) in a rather standard way, or alternatively to predict the camera parameters directly in 3D.

The concept of virtual corners assumes that edges are rigidly attached in 2D. In Chapter 5, this condition is relaxed to rigid attachment in 3D. However, further relax-

ation of this condition is possible, allowing the use of virtual corners in the non-rigid scenario. Researching the possibilities of such a direction is another interesting topic for further research. Another challenge of non-rigid targets is their online 3D modelling. There is a potential for great improvements in the area of 3D tracking of deformable objects using better models. An example of this may be to employ a Latent Variable GP, with the observed variables modelling the 3D shape and the latent variables the deformation.

While modelling the 3D shape, the feature clouds are used as positive samples, indicating where the object is. However, it might be a viable idea to use *negative* data as well, indicating where the object is *not*. For this objective, the relationship between tracking and segmentation could be explored in closer detail. The segmented object outline (in 2D) can provide 3D constraints on the extent of the object. Finally, the research on 3D tracking brought a number of interesting insights into the camera tracking problem in general as well as into general real world reconstruction. It would be interesting to take these insights into the field of visual SLAM where similar challenges are encountered.

References

[1] GPy library. https://github.com/SheffieldML/GPy. 122

[2] S. Agarwal, K. Mierle, et al. Ceres solver. http://code.google.com/p/ceres-solver/. 122, 149

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *Proceedings of the International Conference on Computer Vision*, pages 72–79, 2009. 20

[4] A. Agudo, L. Agapito, B. Calvo, and J. M. M. Montiel. Good vibrations: A modal analysis approach for sequential non-rigid structure from motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1558–1565, 2014. 22, 145

[5] A. Agudo, B. Calvo, and J. Montiel. Finite element based sequential bayesian non-rigid structure from motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1418–1425, 2012. 22

[6] A. Agudo, J. Montiel, L. Agapito, and B. Calvo. Online dense non-rigid 3D shape and camera motion recovery. In *Proceedings of the British Machine Vision Conference*, pages 103.1–103.12, 2014. 145

[7] S. H. Ahn, J. Choi, N. L. Doh, and W. K. Chung. A practical approach for EKF-SLAM in an indoor environment: fusing ultrasonic sensors and stereo camera. *Autonomous Robots*, 24(3):315–335, 2008. 24

[8] Aristotle. *Physics II.* 27

[9] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó. The SLAM problem: A survey. In *Proceedings of the International Conference of the Catalan Association for Artificial Intelligence*, pages 363–371, 2008. 24

[10] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 14, 137

[11] Y. Bar-Shalom. *Tracking and Data Association.* Academic Press Professional, Inc., 1987. 24

[12] A. Bartoli, V. Gay-Bellile, U. Castellani, J. Peyras, S. Olsen, and P. Sayd. Coarse-to-fine low-rank structure-from-motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 22, 149

[13] M. Bauer, J. Cox, M. Caveness, J. J. Downs, and N. Thornhill. Finding the direction of disturbance propagation in a chemical process using transfer entropy. *Control Systems Technology*, 15(1):12–21, 2007. 27

[14] S. Bazeille and D. Filliat. Incremental topo-metric SLAM using vision and robot odometry. In *Proceedings of the International Conference on Robotics and Automation*, pages 4067–4073, 2011. 24

[15] N. Bellotto, K. Burn, E. Fletcher, and S. Wermter. Appearance-based localization for mobile robots using digital zoom and visual compass. *Robotics and Autonomous Systems*, 56(2):143–156, 2008. 24

[16] J. Bennett, R. Grout, P. Pebay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Proceedings of the International Conference on Cluster Computing*, pages 1–8, 2009. 51

[17] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999. 156

[18] P. Borges, N. Conci, and A. Cavallaro. Video-based human behavior understanding: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(11):1993–2008, 2013. 84

[19] M. Brand. Physics-based visual understanding. *Computer Vision and Image Understanding*, 65(2):192–205, 1996. 28

[20] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 2, pages 690–696, 2000. 21

[21] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. volume 4387, pages 95–102, 2001. 13

[22] Z. Cai, L. Wen, J. Yang, Z. Lei, and S. Li. Structured visual tracking with dynamic graph. In *Proceedings of the Asian Conference on Computer Vision*, volume 7726 of *Lecture Notes in Computer Science*, pages 86–97, 2012. ix

[23] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. 39

[24] L. Cehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *Proceedings of the International Conference on Computer Vision*, pages 1363–1370, 2011. xi

[25] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):941–953, 2013. xi, 16, 17, 59, 113, 134

[26] K. Chen, Y.-K. Lai, and S.-M. Hu. 3D indoor scene modeling from RGB-D data: a survey. *Computational Visual Media*, 1(4):267–278, 2015. 26

[27] M. Chen, S. K. Pang, T. J. Cham, and A. Goh. Visual tracking with generative template model based on Riemannian manifold of covariances. In *Proceedings of the International Conference on Information Fusion*, pages 1–8, 2011. 60, 61, 137, 159

[28] A. Chhatkuli, D. Pizarro, and A. Bartoli. Non-rigid shape-from-motion for isometric surfaces using infinitesimal planarity. In *Proceedings of the British Machine Vision Conference*, pages 41.1–41.12, 2014. 163

[29] G. Chrysos, E. Antonakos, S. Zafeiriou, and P. Snape. Offline deformable face tracking in arbitrary videos. In *Proceedings of the ICCV workshop on 300 Videos in the Wild: Facial Landmark Tracking in-the-Wild*, pages 954–962, 2015. 167

[30] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *Proceedings of the DAGM Symposium*, volume 2781 of *Lecture Notes in Computer Science*, pages 236–243, 2003. xi, 118

[31] M. Csorba. *Simultaneous Localisation and Map Building*. PhD thesis, University of Oxford, 1997. 24

[32] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. 107(2):101–122, 2014. 21

[33] A. Dame, V. Prisacariu, C. Ren, and I. Reid. Dense reconstruction using 3D object shape priors. In *In Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1288–1295, 2013. 18

[34] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *Proceedings of the British Machine Vision Conference*, pages 38.1–38.11, 2014. ix

[35] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the International Conference on Computer Vision*, pages 4310–4318, 2015. xii

[36] A. J. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. 25

[37] A. Del Bue, J. Xavier, L. Agapito, and M. Paladini. Bilinear modeling via augmented lagrange multipliers (BALM). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1496–1508, 2012. 163

[38] T. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1177–1184, 2011. 29

[39] N. Dowson and R. Bowden. N-tier simultaneous modelling and tracking for arbitrary warps. In *Proceedings of the British Machine Vision Conference*, pages 569–578, 2006. 16

[40] N. Dowson and R. Bowden. Mutual information for Lucas-Kanade tracking (MILK): An inverse compositional formulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):180–185, 2008. 12, 77

[41] N. Dowson, T. Kadir, and R. Bowden. Estimating the joint statistics of images using nonparametric windows with application to registration using mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1841–1857, 2008. 77

[42] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002. 18, 19

[43] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006. 23

[44] H. Durrant-Whyte, D. Rye, and E. Nebot. Localization of autonomous guided vehicles. In *Proceedings of the International Symposium on Robotics Research*, pages 613–625, 1995. 24

[45] G. A. Einicke and L. B. White. Robust extended Kalman filtering. *Signal Processing*, 47(9):2596–2599, 1999. x, 101

[46] S. P. Engelson and D. V. McDermott. Error correction in mobile robot map-learning. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 2555–2560, 1992. 25

[47] A. Eriksson and A. van den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L1 norm. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 771–778, 2010. 22

[48] Y. Fan, H. Yang, S. Zheng, H. Su, and S. Wu. Video sensor-based complex scene analysis with Granger causality. *Sensors*, 13(10):13685–13707, 2013. 28

[49] M. Felsberg. Enhanced distribution field tracking using channel representations. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, pages 121–128, 2013. x

[50] Y. Feng, Y. Wu, and L. Fan. On-line object reconstruction and tracking for 3D interaction. In *International Conference on Multimedia and Expo*, pages 711–716, 2012. 19

[51] L. Ferraz, X. Binefa, and F. Moreno-Noguer. Very fast solution to the PnP problem with algebraic outlier rejection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2014. 164

[52] A. Fire and S.-C. Zhu. Using causal induction in humans to learn and infer causality from video. In *Annual Conference of the Cognitive Science Society*, 2013. 28

[53] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. xi

[54] K. Friston, R. Moran, and A. K. Seth. Analysing connectivity with Granger causality and dynamic causal modelling. *Current Opinion in Neurobiology*, 23(2):172–178, 2013. 27

[55] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 151

[56] M. A. Garcia and A. Solanas. 3D simultaneous localization and modeling from stereo vision. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 847–853, 2004. 25

[57] R. Garg, A. Roussos, and L. Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1272–1279, 2013. 22

[58] R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *International Journal of Computer Vision*, 104(3):286–314, 2013. 171

[59] A. Gelb. *Applied Optimal Estimation*. M.I.T. Press, 1984. x, 23

[60] R. Gherardi, M. Farenzena, and A. Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1594–1600, 2010. 20

[61] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. 15

[62] H. Grabner, M. Grabner, and H. Bischof. Real-Time Tracking via On-line Boosting. In *Proceedings of the British Machine Vision Conference*, pages 6.1–6.10, 2006. 14

[63] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of the European Conference on Computer Vision*, pages 234–247, 2008. 14

[64] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1285–1292, 2010. 29, 108, 109, 110, 111

[65] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969. 27, 93

[66] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots – learning a visually grounded storyline model from annotated videos. In *In Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2012–2019, 2009. 28

[67] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003. 83

[68] R. S. Hacker and A. Hatemi-J. Tests for causality between integrated variables using asymptotic and bootstrap distributions: theory and application. *Applied Economics*, 38(13):1489–1500, 2006. 27

[69] S. Hadfield, K. Lebeda, and R. Bowden. Natural action recognition using invariant 3D motion encoding. In *Proceedings of the European Conference on Computer Vision*, volume 8690 of *Lecture Notes in Computer Science*, pages 758–771, 2014. vii

[70] S. Hadfield, K. Lebeda, and R. Bowden. Hollywood 3D: What are the best 3D features for action recognition? *International Journal of Computer Vision*, 2016. viii

[71] S. Hadfield, K. Lebeda, and R. Bowden. PnP-HARD: PnP optimization using a hybrid approximate representation. In *Under review for the European Conference on Computer Vision*, 2016. viii, 120

[72] S. Hadfield, K. Lebeda, and R. Bowden. Stereo reconstruction using top-down cues from urban environment. *Computer Vision and Image Understanding*, 2016. viii

[73] C. Ham, S. Lucey, and S. Singh. Hand waving away scale. In *Proceedings of the European Conference on Computer Vision*, volume 8692 of *Lecture Notes in Computer Science*, pages 279–293, 2014. 24

[74] J. P. Hamilton, G. Chen, M. E. Thomason, M. E. Schwartz, and I. H. Gotlib. Investigating neural primacy in Major Depressive Disorder: multivariate Granger causality analysis of resting-state fMRI time-series data. *Molecular Psychiatry*, 16(7):763–772, 2011. 27

[75] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. M. Cheng, S. Hicks, and P. Torr. Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. xii, 14

[76] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *Proceedings of the International Conference on Computer Vision*, pages 263–270, 2011. xii, 76, 104, 107, 113

[77] C. Harris and C. Stennett. RAPID – a video rate object tracker. In *Proceedings of the British Machine Vision Conference*, pages 73–78, 1990. 18, 19, 36, 114

[78] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Alvey Vision Conference*, pages 147–151, 1988. 18

[79] R. I. Hartley. Projective reconstruction from line correspondences. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 903–907, 1994. 21, 42

[80] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 118, 119, 120

[81] J. Heller, M. Havlena, M. Jancosek, A. Torii, and T. Pajdla. 3D reconstruction from photographs by CMP SfM web service. *Machine Vision and Applications*, pages 30–34, 2015. http://ptak.felk.cvut.cz/sfmservice/. 159, 160

[82] C. K. Heng, S. Y. Y. Lim, Z. H. Niu, and B. Li. Single scale pixel based LUT tracker. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge, appendix A.21 of [115]*, page 109, 2013. xi

[83] J. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2014. x, 13

[84] C. Hiemstra and J. D. Jones. Testing for linear and nonlinear Granger causality in the stock price-volume relation. *The Journal of Finance*, 49(5):1639–1664, 1994. 27

[85] E. C. Hildreth. *The measurement of visual motion*. MIT Press, 1984. 33

[86] A. Hilton and J. Illingworth. Marching triangles: Delaunay implicit surface triangulation. Technical report, University of Surrey, 1997. 156

[87] K. Hirose and H. Saito. Fast line description for line-based slam. In *Proceedings of the British Machine Vision Conference*, pages 83.1–83.11, 2012. 25

[88] K. Hlaváčková-Schindler, M. Palus, M. Vejmelka, and J. Bhattacharya. Causality detection based on information-theoretic approaches in time series analysis. *Physics Reports*, 441(1):1–46, 2007. 28, 78, 79

[89] J. Hoey. Tracking using flocks of features, with application to assisted handwashing. In *Proceedings of the British Machine Vision Conference*, pages 167.1–167.10, 2006. x

[90] W. Hoff, K. Nguyen, and T. Lyon. Computer vision-based registration techniques for augmented reality. In *Proceedings of Intelligent Robots and Control Systems XV, Intelligent Control Systems and Advanced Manufacturing*, pages 538–548, 1996. 19, 114

[91] S. A. Holmes and D. W. Murray. Monocular SLAM with conditionally independent split mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1451–1463, 2013. 25

[92] C. Hoppe, M. Klopschitz, M. Rumpler, A. Wendel, S. Kluckner, H. Bischof, and G. Reitmayr. Online feedback for structure-from-motion image acquisition. In *Proceedings of the British Machine Vision Conference*, pages 70.1–70.12, 2012. 21, 115

[93] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185–203, 1981. 155

[94] D. Hume. *A Treatise of Human Nature*. 1738. 27

[95] M. Isard and A. Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. ix, 13

[96] E. Ito, T. Okatani, and K. Deguchi. Accurate and robust planar tracking based on a model of image sampling and reconstruction process. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 1–8, 2011. 19, 114

[97] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, 2003. 60, 61

[98] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1822–1829, 2012. ix, 107

[99] E. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: a scalable real-time causal approach. *International Journal of Robotic Research*, 30(4):407–430, 2011. 24

[100] J.Shen, S.Zafeiriou, G. Chrysos, J.Kossaifi, G.Tzimiropoulos, and M. Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *Proceedings of the ICCV workshop on 300 Videos in the Wild: Facial Landmark Tracking in-the-Wild*, pages 1003–1011, 2015. 167

[101] S. Julier and J. Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion and Target Recognition*, volume 3068, pages 182–193, 1997. x, 101

[102] A. Kaiser and T. Schreiber. Information transfer in continuous processes. *Physica D: Nonlinear Phenomena*, 166(1–2):43–62, 2002. 27, 28

[103] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In *Proceedings of the ICCV workshop on On-line Learning for Computer Vision*, pages 1417–1424, 2009. xii

[104] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 49–56, 2010. xii

[105] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proceedings of the International Conference on Pattern Recognition*, pages 2756–2759, 2010. 16

[106] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. xii, 16, 59, 65, 66, 67, 113, 134, 154

[107] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(Series D):35–45, 1960. x, 101

[108] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the Symposium on Geometry Processing*, pages 61–70, 2006. 156

[109] M. Kazhdan and H. Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29:1–29:13, 2013. 156

[110] K. Kim, V. Lepetit, and W. Woo. Keyframe-based modeling and tracking of multiple 3D objects. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 193–198, 2010. 18

[111] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. of ISMAR*, pages 1–10, 2007. 25, 115

[112] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based camera tracking for augmented reality applications. In *Proceedings of the Symposium on Virtual Reality Software and Technology*, pages 87–94, 1997. 19, 114

[113] M. Kölsch and M. Turk. Fast 2D hand tracking with flocks of features and multi-cue integration. In *Proceedings of the Computer Vision and Pattern Recognition Workshop*, pages 158–158, 2004. x, 15, 113

[114] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Čehovin, G. Fernández, T. Vojíř, G. Häger, G. Nebehay, R. Pflugfelder, et al. The visual object tracking VOT2015 challenge results. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, pages 564–586, 2015. viii, 14, 15

[115] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, et al. The visual object tracking VOT2013 challenge results. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, pages 98–111, 2013. vii, xii, 17, 52, 53, 54, 73, 74, 92, 93, 105, 187

[116] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, G. Fernández, et al. The visual object tracking VOT2014 challenge results. In *Proceedings of the ECCV workshop on Visual Object Tracking Challenge*, pages 191–217, 2014. vii, xii, 54, 56, 57, 190

[117] A. Kundu, K. Krishna, and C. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *Proceedings of the International Conference on Computer Vision*, pages 2080–2087, 2011. 19

[118] K. Lebeda. 2D tracking datasets: Feature-Less Objects, Youtube Long-Term Tracking and others. http://cvssp.org/Personal/KarelLebeda/data2D/. 60, 65

[119] K. Lebeda. Dense tracking and modelling input data and additional results. http://cvssp.org/Personal/KarelLebeda/TMAGIC/dense. 150, 161, 164

[120] K. Lebeda. TMAGIC input data and additional results. http://cvssp.org/Personal/KarelLebeda/TMAGIC/. 134

[121] K. Lebeda, S. Hadfield, and R. Bowden. 2D or not 2D: Bridging the gap between tracking and structure from motion. In *Proceedings of the Asian Conference on Computer Vision*, pages 642–658, 2014. vii, xii, 8

[122] K. Lebeda, S. Hadfield, and R. Bowden. Dense rigid reconstruction from unstructured discontinuous video. In *Proceedings of the ICCV workshop on 3D Representation and Recognition*, pages 814–822, 2015. viii, 9

[123] K. Lebeda, S. Hadfield, and R. Bowden. Exploring causal relationships in visual object tracking. In *Proceedings of the International Conference on Computer Vision*, pages 3065–3073, 2015. vii, 8

[124] K. Lebeda, S. Hadfield, and R. Bowden. Causal relationships in visual tracking. *Under review for IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. viii, 8

[125] K. Lebeda, S. Hadfield, and R. Bowden. Direct-from-video: Unsupervised NRSfM. In *Under review for the European Conference on Computer Vision*, 2016. viii, 9

[126] K. Lebeda, S. Hadfield, and R. Bowden. TMAGIC: A model-free 3D tracker. *Under review for IEEE Transactions on Image Processing*, 2016. viii, 8

[127] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Long-term tracking through failure cases. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, pages 153–160, 2013. vii, xi, 7

[128] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Texture-independent long-term tracking using virtual corners. *IEEE Transactions on Image Processing*, 25(1):359–371, 2016. viii, xi, 7, 34

[129] K. Lebeda, J. Matas, and R. Bowden. Tracking the untrackable: How to track when your object is featureless. In *Proceedings of the ACCV workshop on Detection and Tracking in Challenging Environments*, pages 343–355, 2012. vii, x, 7, 117

[130] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized RANSAC. In *Proceedings of the British Machine Vision Conference*, pages 1013–1023, 2012. xi, 39, 118

[131] Y. Li and J. Zhu. A kernel correlation filter tracker with Scale Adaptive and Feature Integration. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge, appendix A.9 of [116]*, page 207, 2014. xii

[132] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, volume 21, pages 163–169, 1987. 156

[133] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. xii

[134] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981. xi, 11

[135] R. Mann, A. Jepson, and J. M. Siskind. The computational perception of scene dynamics. *Computer Vision and Image Understanding*, 65(2):113–128, 1997. 28

[136] J. Matas. Visual tracking in the 21st century. In *Proceedings of the British Machine Vision Conference*, 2012. 74, 75

[137] J. Matas and T. Vojíř. Robustifying the flock of trackers. In *Proceedings of the Computer Vision Winter Workshop*, pages 91–97, 2011. x, 16, 59, 134

[138] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004. 15

[139] W. W. Mayol and D. W. Murray. Tracking with general regression. *Machine Vision and Applications*, 19(1):65–72, 2007. 14

[140] P. McLauchlan, X. Shen, A. Manessis, P. Palmer, and A. Hilton. Surface-based structure-from-motion using feature groupings. In *In Proceedings of the Asian Conference on Computer Vision*, pages 699–705, 2000. 21, 117

[141] I. F. Mondragón, P. Campoy, C. Martínez, and M. A. Olivares-Méndez. 3D pose estimation based on planar object tracking for UAVs control. In *Proceedings of the International Conference on Robotics and Automation*, pages 35–41, 2010. 19

[142] A. Mulloni, M. Ramachandran, G. Reitmayr, D. Wagner, R. Grasset, and S. Diaz. User friendly SLAM initialization. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 153–162, 2013. 25, 114

[143] R. Mur-Artal and J. Tardós. Fast relocalisation and loop closing in keyframe-based SLAM. In *Proceedings of the International Conference on Robotics and Automation*, pages 846–853, 2014. 26

[144] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. *Computing Research Repository*, abs/1510.07945, 2015. 14

[145] S. Narayan and K. Ramakrishnan. A cause and effect analysis of motion trajectories for modeling actions. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2633–2640, 2014. 28

[146] R. Newcombe, D. Fox, and S. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 343–352, 2015. 22, 26

[147] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. 26

[148] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *In Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1498–1505, 2010. 26

[149] R. A. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision*, pages 2320–2327, 2011. 26

[150] P. Newman and K. Ho. SLAM – Loop closing with visually salient features. In *Proceedings of the International Conference on Robotics and Automation*, pages 44–651, 2005. 24

[151] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing*, 6(1):103–113, 1997. 39

[152] E.-J. Ong, A. S. Micilotta, R. Bowden, and A. Hilton. Viewpoint invariant exemplar-based 3D human tracking. *Computer Vision and Image Understanding*, 104(23):178 – 189, 2006. 18

[153] S. Oron, A. Bar-Hillel, and S. Avidan. Extended Lucas-Kanade tracking. In *Proceedings of the European Conference on Computer Vision*, pages 142–156, 2014. 12

[154] M. Paladini, A. Bartoli, and L. Agapito. Sequential non-rigid structure-from-motion with the 3D-implicit low-rank shape model. In *Proceedings of the European Conference on Computer Vision*, volume 6312 of *Lecture Notes in Computer Science*, pages 15–28, 2010. 21, 22, 145

[155] M. Paladini, A. Del Bue, M. Stosic, M. Dodig, J. Xavier, and L. Agapito. Factorization for non-rigid and articulated structure using metric projections. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2898–2905, 2009. 149

[156] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In *Proceedings of the British Machine Vision Conference*, pages 112.1–112.11, 2009. 20, 115, 121

[157] F. I. Parke. Computer generated animation of faces. In *Proceedings of the ACM Annual Conference*, pages 451–457, 1972. 147

[158] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000. 27

[159] M. Perriollat, R. Hartley, and A. Bartoli. Monocular template-based reconstruction of inextensible surfaces. *International Journal of Computer Vision*, 95(2):124–137, 2011. 22

[160] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 20, 120

[161] K. Prabhakar, S. Oh, P. Wang, G. Abowd, and J. Rehg. Temporal causality for the analysis of visual events. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1967–1974, 2010. 27, 28

[162] V. A. Prisacariu, O. Kahler, D. W. Murray, and I. D. Reid. Simultaneous 3D tracking and reconstruction on a mobile phone. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 89–98, 2013. 20

[163] V. A. Prisacariu, A. V. Segal, and I. Reid. Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction. In *Proceedings of the Asian Conference on Computer Vision*, volume 7724 of *Lecture Notes in Computer Science*, pages 593–606, 2012. 18

[164] C. E. Rasmussen. Gaussian processes in machine learning. In *Lecture Notes in Computer Science*, 3176, pages 63–71. 2004. 128

[165] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2004. 90, 122, 124

[166] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008. 13, 32, 113, 137, 159

[167] C. Scheidegger, S. Fleishman, and C. Silva. Triangulating point set surfaces with bounded error. In *Proceedings of the Eurographics Symposium on Geometry Processing*, pages 63:1–63:11, 2005. 156

[168] T. Schreiber. Measuring information transfer. *Physical Review Letters*, 85(2):461–464, 2000. 27

[169] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994. 12, 25

[170] Y. Shi, S. Ji, Z. Shi, Y. Duan, and R. Shibasaki. GPS-supported visual SLAM with a rigorous sensor model for a panoramic camera in outdoor environments. *Sensors*, 13(1):119–136, 2013. 24

[171] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270–1281, 2008. 39

[172] L. Sigal, M. Isard, H. Haussecker, and M. J. Black. Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation. *International Journal of Computer Vision*, 98(1):15–48, 2012. 18

[173] G. Simon and M.-O. Berger. Pose estimation from planar structures. *Computer Graphics and Applications*, 22:46–53, 2002. 19

[174] G. Simon, A. W. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *Proceedings of the International Symposium on Augmented Reality*, pages 120–128, 2000. 19, 114

[175] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *International Journal of Computer Vision*, 67(2):189–210, 2006. 108

[176] P. Smith, I. Reid, and A. J. Davison. Real-time monocular SLAM with straight lines. In *Proceedings of the British Machine Vision Conference*, pages 3.1–3.10, 2006. 25, 117, 120

[177] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous Robot Vehicles*, pages 167–193, 1990. 23

[178] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006. 134

[179] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2007. 20, 134

[180] S. Song and J. Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *Proceedings of the International Conference on Computer Vision*, pages 233–240, 2013. 26

[181] H. Strasdat, J. M. M. Montiel, and A. Davison. Real-time monocular SLAM: Why filter? In *Proceedings of the International Conference on Robotics and Automation*, pages 2657–2664, 2010. 25, 149

[182] H. Sumioka, Y. Yoshikawa, and M. Asada. Learning of joint attention from detecting causality based on transfer entropy. *Journal of Robotics and Mechatronics*, 20(3):378–385, 2008. 28

[183] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. Robust monocular SLAM in dynamic environments. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 209–2018, 2013. 26

[184] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3D reconstruction on mobile phones. In *Proceedings of the International Conference on Computer Vision*, pages 65–72, 2013. 20

[185] L. Tao and B. J. Matuszewski. Non-rigid structure from motion with diffusion maps prior. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1530–1537, 2013. 21, 145

[186] D. L. Thornton and D. S. Batten. Lag-length selection and tests of Granger causality between money and income. *Journal of Money, Credit and Banking*, 17(2):164–178, 1985. 27

[187] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, 1991. CMU-CS-91-132. x, 12

[188] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992. 21

[189] M. Tomono. Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm. In *Proceedings of the International Conference on Robotics and Automation*, pages 4306–4311, 2009. 25

[190] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. 120, 148

[191] Y. Tsin, Y. Genc, Y. Zhu, and V. Ramesh. Learn to track edges. In *Proceedings of the International Conference on Computer Vision*, pages 1–8, 2007. 18

[192] G. Tzimiropoulos. Project-out cascaded regression with an application to face alignment. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3659–3667, 2015. 167

[193] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3D tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1385–1391, 2004. 19, 114

[194] A. Varol, P. Salzmann, and R. Urtasun. A constrained latent variable model. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2248–2255, 2012. 171, 172

[195] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008. 118

[196] R. Vicente, M. Wibral, M. Lindner, and G. Pipa. Transfer entropy – a model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, 30(1):45–67, 2011. 27

[197] S. Vicente and L. Agapito. Soft inextensibility constraints for template-free non-rigid reconstruction. In *Proceedings of the European Conference on Computer Vision*, pages 426–440, 2012. 22, 163

[198] S. Vidas and S. Sridharan. Hand-held monocular slam in thermal-infrared. In *Proceedings of the International Conference on Control Automation Robotics Vision*, pages 859–864, 2012. 24

[199] T. Vojíř and J. Matas. The enhanced flock of trackers. In *Registration and Recognition in Images and Videos*, volume 532 of *Studies in Computational Intelligence*, pages 113–136. 2014. x, 16, 76, 88, 104, 108

[200] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010. 118

[201] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *Proceedings of the International Conference on Computer Vision*, pages 1385–1392, 2013. 153

[202] T. Wiemann, H. Annuth, K. Lingemann, and J. Hertzberg. An extended evaluation of open source surface reconstruction software for robotic applications. *Journal of Intelligent and Robotic Systems*, 77(1):149–170, 2015. 156

[203] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):882–897, 2013. 18

[204] C. Wu. Towards linear-time incremental structure from motion. In *Proceedings of the International Conference on 3D Vision*, pages 127–134, 2013. 134

[205] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 3057–3064, 2011. 134

[206] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proceeding of the Conference on Computer Vision and Pattern Recognition*, pages 2411–2418, 2013. xii, 32, 56, 58

[207] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. xii, 105

[208] J. Xiao, J. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. *International Journal of Computer Vision*, 67(2):233–246, 2006. 149

[209] J. Xiao, R. Stolkin, and A. Leonardis. An enhanced adaptive coupled-layer LGTracker++. In *Proceedings of the ICCV workshop on Visual Object Tracking Challenge*, pages 137–144, 2013. xi, 17

[210] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan. Robust object tracking with online multi-lifespan dictionary learning. In *Proceedings of the International Conference on Computer Vision*, pages 665–672, 2013. 14

[211] S. Yi and V. Pavlovic. Sparse Granger causality graphs for human action classification. In *Proceedings of the International Conference on Pattern Recognition*, pages 3374–33477, 2012. 27, 28

[212] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006. 101

[213] R. Yu, C. Russell, N. D. F. Campbell, and L. Agapito. Direct, dense, and deformable: Template-based non-rigid 3D reconstruction from RGB video. In *Proceedings of the International Conference on Computer Vision*, pages 918–926, 2015. 19, 22, 172

[214] L. Zhang. *Line Primitives and Their Applications in Geometric Computer Vision*. PhD thesis, Department of Computer Science, Kiel University, 2013. 21, 117, 120

[215] F. Zheng, L. Shao, J. Brownjohn, and V. Racic. Learn++ for robust object tracking. In *Proceedings of the British Machine Vision Conference*, pages 43.1–43.12, 2014. 14

[216] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1838–1845, 2012. xii, 107

[217] M. Zollhofer, M. Niessner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics*, 33(4):156:1–156:12, 2014. 22

List of Figures

List of Tables