Scene and Camera Motion
Estimation From Video

A dissertation submitted to
ETH ZURICH

for the degree of
Doctor of Sciences (Dr. sc. ETH Zürich)

presented by
Till Kroeger
Master of Applied Computer Science
University of Heidelberg
born June 19, 1983
citizen of Bielefeld, Germany

accepted on the recommendation of
Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Thomas Brox, co-examiner
Prof. Dr. Andreas Geiger, co-examiner

2018
Abstract

Human visual perception is strongly dependent on recognition of object shape and motion. In particular, the algorithmic understanding of motion — observed scene motion, as well as the motion of the observer — is an important building block for most higher-level tasks in visual sensing and scene understanding from image sequences. The problem of motion estimation can be further divided into different sub-problems of varying difficulty, goals and underlying assumptions.

The most general case of scene motion estimation is finding the observed motion at every image location from one frame of an image sequence to the next frame. The task of estimating this two-dimensional vector field is usually named Optical Flow. It does not require any assumptions about the observed scene nor the observer. It is, however, a high-dimensional and computationally demanding task. Furthermore, it does not take into account that observed natural scenes can often be segmented into groups of objects with similar and smooth motion.

This observation can be exploited for object motion estimation by assuming that the spatial grouping and segmentation of image content at each time instance into objects is available through other means, for instance by using frame-wise object detections. The motion estimation task is then simplified to finding a coherent temporal grouping and motion estimation on object level. Finding and temporally grouping such sets of similar objects is often referred to as tracking-by-detection. If multiple synchronized image sequences, observing the same scene, are available, the temporal grouping can be extended to jointly find consistent groups over time and multiple views.

Observed motion is the combination of motion induced by the observer (i.e. moving camera) and object motion. Often, the observed scene can be segmented into a non-moving background and multiple moving objects. In those cases separating the tasks of estimating the motion of the observer and object motion is possible. It is, however, an ill-defined problem to identify and separate a static background from moving objects, without knowledge or priors on the observer motion. Therefore, it is often easier to focus on recovering the observer motion first, by assuming that the motion of the static background dominates, before proceeding to object motion. If a precise model of the static 3D geometry of the observed scene is available, the observer motion can be recovered by registering the image sequence to this model. Another cue to help estimating the orientation of the observer
and the geometry of the static environment in scenes depicting man-made structures are the convergence points of imaged scene-parallel structures, i.e. *vanishing points*.

This thesis is a contribution to the estimation and understanding of scene and observer motion from image sequences in these fields. We will describe our proposed method to efficiently compute generic two-dimensional optical flow, as published in [Kroeger et al., 2016], our method for multi-target tracking-by-detection across multiple sequences in the presence of camera localization noise, as published in [Kroeger et al., 2014], our methods for vanishing point detection and tracking, as published in [Kroeger et al., 2015a] and [Kroeger et al., 2015b], and our method for video registration to given structure-from-motion 3D-models, as published in [Kroeger and Van Gool, 2014].
# Contents

1 Introduction ................................................. 1
   1.1 Motivation ........................................... 2
   1.2 Contributions ........................................ 3
   1.3 Organization .......................................... 5

2 Fast Optical Flow using Dense Inverse Search ................. 6
   2.1 Introduction .......................................... 6
      2.1.1 Related work ........................................ 8
      2.1.2 Contributions ...................................... 9
   2.2 Proposed method ....................................... 10
      2.2.1 Fast inverse search for correspondences .......... 11
      2.2.2 Fast optical flow with multi-scale reasoning ...... 12
      2.2.3 Fast Variational refinement ......................... 13
      2.2.4 Extensions ......................................... 13
   2.3 Experiments ............................................ 16
      2.3.1 Implementation and Parameter Selection .......... 16
      2.3.2 Parallelization and memory consumption .......... 17
      2.3.3 Evaluation of Inverse Search ....................... 20
      2.3.4 MPI Sintel optical flow results ................... 20
      2.3.5 KITTI optical flow results ........................ 23
      2.3.6 High frame-rate optical flow ....................... 23
      2.3.7 Depth from Stereo with DIS ....................... 26
   2.4 Conclusions ............................................ 26

3 Multi-View Multi-Target Tracking .......................... 27
   3.1 Introduction .......................................... 27
      3.1.1 Related Work: ...................................... 28
   3.2 Tracking .............................................. 30
   3.3 Modeling of Probabilities ............................. 31
      3.3.1 Extraction of Reconstruction Candidates .......... 31
      3.3.2 Localization of Detections and Reconstructions .. 32
      3.3.3 Detection and Reconstruction Distances .......... 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.4 Non-Linear Least-Squares Optimization</td>
<td>82</td>
</tr>
<tr>
<td>6.2.5 Combinations and Variants</td>
<td>83</td>
</tr>
<tr>
<td>6.3 Experiments</td>
<td>83</td>
</tr>
<tr>
<td>6.3.1 Synthetic Video Sequence</td>
<td>84</td>
</tr>
<tr>
<td>6.3.2 Street-View Video Sequence - Dataset and PnP Baseline</td>
<td>85</td>
</tr>
<tr>
<td>6.3.3 Street-View Video Sequence - Video Registration</td>
<td>87</td>
</tr>
<tr>
<td>6.3.4 Gap Interpolation</td>
<td>89</td>
</tr>
<tr>
<td>6.4 Discussion and Conclusion</td>
<td>90</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>92</td>
</tr>
<tr>
<td>7.1 Summary</td>
<td>92</td>
</tr>
<tr>
<td>7.2 Future work</td>
<td>93</td>
</tr>
</tbody>
</table>
1

Introduction

We move in order to see and we see in order to move.

J.J. Gibson

One of the most important ways for environmental perception for nearly all living organisms is perception through visual signals. In various forms of sophistication all mammals and many other complex life forms have developed biological visual sensors, which capture two-dimensional projections of the surroundings world. In order for an organisms to interact with this environment complex sensor processing and scene understanding capabilities are required.

The complexity of processing visual information for the goal of understanding the observed scene and recognizing known and unknown objects is due to the large amount of data to be processed, the variability of visual information (photo-metric variability, deformations, occlusions, ambiguities), and required low latency of any visual reasoning.

The stack of all tasks in visual scene understanding is compositional in its nature. Object and scene-agnostic estimation of apparent motion is performed at an early stage in the visual cortex of the human brain, with very low latency. Higher-level recognition and localization of known objects is performed at a later stage in the visual cortex. Among other inputs it receives motion estimates from earlier processing stages of the visual cortex, and operates with a larger latency. But the estimation of apparent motion is not only an input for higher-level visual reasoning, it is also an important stream of information for low-latency body control and reaction: the hand catching the incoming ball, and the rabbit escaping the predator require visual servoing with millisecond-delay.

This importance of motion estimation for interaction with the environment has been recognized, not only by the neuro-science community, but also by the robotic and computer vision research community. Estimating the apparent visual motion in image sequence and tracking distinct objects over time were among the first problems to receive attention in the community.
However, despite significant scientific advances of the past decades, these tasks are not solved to satisfaction yet.

1.1 Motivation

This thesis is motivated by a vision of enabling more efficient and robust algorithmic understanding of apparent visual motion in image sequences. This larger vision can be further split into different groups of sub-problems. The most obvious split-lines follow common geometric and physical constraints to simplify this task. Similarly, prior assumptions can be used to exploit partial given knowledge about the observed scene.

In this thesis we will considered a variety of problems ranging from completely unconstrained motion estimation to employing strong assumptions about the static nature of the observed scene for rigid camera movement estimation.

In the most generic and unconstrained form, we will consider the task of estimating the observed, i.e. projected motion in image space. This task is usually referred to as Optical Flow estimation. Since neither spatial nor temporal grouping is assumed, and since the search space of allowed motion is unconstrained for all image locations, this is a complex task of very high dimensionality. Optical flow is used as an input to many higher-level reasoning pipelines, with applications in robotic control for obstacle avoidance, navigation planning, visual servoing, motion segmentation and activity recognition. Since it is used as a low-level input, and with most applications in autonomous robotics, real-time performance is critical.

One of the most prominent constraints to be applied in this domain is the separation of motion induced by the observer with respect to the scene, and the independent motion of objects within the scene. However, it is an ill-defined problem to distinguish between these two sources of motion purely based in visual input. Estimating the true scene-motion of an object based on its projected and observed motion is under-constrained: It can be cause by the observer as well as the object itself. Besides this ambiguity, estimating the scene motion based on visual input leads to the problem that no frame of reference exist, and that due to the projected sensing all world-scale information is lost. Fortunately, several easy solutions for simplifying the problem with prior assumptions exist.

Without using additional external sensors (acceleration and orientation sensors, absolute position sensors – GPS), which will not be considered in this thesis, we can chose the scale and frame of reference arbitrarily. A useful assumption about the natural world is that it is compositional and that it can be naturally split into distinct objects. These objects may freely move, deform, appear and disappear, but they will obey certain physical rules. Even stronger, for certain scenarios we will be able to further restrict some of those rules: For instance, static world object (buildings, ground plane) and some moving objects (cars)
can be assumed to be generally non-deformable within a short-time period, and we can usually assume that all observed objects will be temporally consistent, and not disappear suddenly. For many objects we can employ priors on the regularity: Object shapes follow rules, such as a preference for right angles and parallel structures in man-made objects. Additionally, we can exploit statistical priors: Object appearance, shape and movement statistics can be gather from training data.

In projected image space this implies that we can generally assume clear object boundaries, spatial and temporal smoothness and known statistical priors of the observed world. For object and scene motion estimation this can exploited in various ways. Object boundaries in the sequence can be provided by image-based object detectors to give coarse outlines (bounding boxes). Image-based pixel-wise object segmentation can give precise object contours. Object motion estimation in this scenario is equivalent to temporally associating object evidence given at each time instance. Recovering the motion of the observer, relative to a static world, can also be attempted by identifying the static world object, for instance, by fixing the observed temporally and spatially dominant object in the sequence, and recovering the observer motion with respect to it.

1.2 Contributions

The contributions of this thesis follow the above outline and motivation. We will start from the most general case of unconstrained motion estimation, and proceed to higher-level and long-term motion tracking, and ego-motion estimation employing various constraints:

1. We propose a solution for estimating dense optical flow with very low time complexity and competitive accuracy. It consists of three parts: 1) inverse search for patch correspondences; 2) dense displacement field creation through patch aggregation along multiple scales; 3) variational refinement. At the core of our Dense Inverse Search-based method (DIS) is the efficient search of correspondences inspired by the inverse compositional image alignment proposed by Baker & Matthews [2001, 2004]. Our proposed system is competitive on standard optical flow benchmarks. The core method (without pre-processing) runs at 300Hz up to 600Hz on a single CPU core, reaching the temporal resolution of human’s biological vision system.

2. We propose a novel tracking-by-detection algorithm for multiple targets from multiple moving cameras with noisy localization. In contrast to the estimation of dense optical flow, in multi-target tracking-by-detection, scene motion is estimated on object-level, and frame-wise object evidence is provided. Additionally, if multiple cameras observe the scene then detection evidence has to be associated across time
1.2. Contributions

as well as across cameras. This is complicated even more if the relative position of the cameras with respect to each other is not, or only approximately known. In the past tracking in this setting has either been done with multiple static cameras, or single and stereo moving cameras. Our proposed system extends this framework to multiple moving cameras with only approximately known motion. The camera motion and positional uncertainty can be efficiently incorporated into a flow-network formulation for tracking-by-detection.

3. For man-made objects regularity and preference of parallel structures can often be assumed. Exploiting this assumption, we present a novel vanishing point detection and tracking algorithm for calibrated monocular image sequences. Previous vanishing point detection and tracking methods usually assume known camera poses for all frames or detect and track separately. We advance the state-of-the-art by combining vanishing point extraction on a Gaussian sphere with recent advances in multi-target tracking on probabilistic occupancy fields. The solution is obtained by solving a Linear Program. This enables the joint detection and tracking of multiple vanishing points over sequences. Unlike existing works we do not need known camera poses, and at the same time avoid detecting and tracking in separate steps. We also propose an extension to enforce vanishing point orthogonality.

4. The proposed vanishing point tracking method can be further refined: Man-made environments are also dominated by a strong preference for right angles and vanishing point orthogonality. Hence, large parts of our man-made environments, outdoors as well as indoors, are either parallely or orthogonally aligned to their surroundings. In scenarios where this constraint is valid, groups of mutually orthogonal vanishing points can be found and exploited. We propose a system for detecting and tracking groups of mutually orthogonal vanishing points (MOVP), also known as Manhattan frames, jointly from monocular videos. The method is unique in that it is designed to enforce orthogonality in groups of VPs, temporal consistency of each individual MOVP, and orientation consistency of all putative MOVP. Experiments show that the method outperforms greedy MOVP tracking method considerably. In addition, we also test the method for camera orientation estimation and show that it obtains very promising results on a challenging street-view dataset.

5. It is possible to recover the motion of a moving camera (change in position and orientation) with respect to the observed sequence. Registering image data to structure-from-motion (SfM) point clouds is widely used to find precise camera location and orientation with respect to a world model. This process has successfully been used to extract the camera motion over multiple frames of a video sequence. However, without temporal smoothness the magnitude of the pose error in each frame of a video will often dominate the magnitude of frame-to-frame pose change. This hinders application of methods requiring stable poses estimates (e.g. tracking, augmented reality). We propose a system that incorporates temporal smoothness con-
1.3. Organization

This thesis is laid out as follows. In chapter 2. *Fast Optical Flow using Dense Inverse Search* we present our contribution on fast optical flow extraction as published in [Kroeger et al., 2016]. In chapter 3. *Multi-View Multi-Target Tracking* we present our method for multi-target tracking-by-detection across multiple sequences in the presence of camera localization noise, as published in [Kroeger et al., 2014]. In chapters 4. *Joint Vanishing Point Detection and Tracking* and 5. *Mutually Orthogonal Vanishing Points* we detail our systems for joint vanishing point detection and tracking, as well as our extension to sets of mutually orthogonal vanishing points, as published in [Kroeger et al., 2015a] and [Kroeger et al., 2015b], respectively. In chapter 6. *Video Registration* we present our method for video registration to given structure-from-motion 3D-models, as published in [Kroeger and Van Gool, 2014]. Chapter 7 concludes the presented work and gives an outlook for future research. Chapters 2, 3, 4, 5 and 6, describing each of the outlined contributions, can be read independently and in any order.
2

Fast Optical Flow using Dense Inverse Search

Most recent works in optical flow extraction focus on the accuracy and neglect the time complexity. However, in real-life visual applications, such as tracking, activity detection and recognition, the time complexity is critical. We propose a solution with very low time complexity and competitive accuracy for the computation of dense optical flow. It consists of three parts: 1) inverse search for patch correspondences; 2) dense displacement field creation through patch aggregation along multiple scales; 3) variational refinement. At the core of our Dense Inverse Search-based method (DIS) is the efficient search of correspondences inspired by the inverse compositional image alignment proposed by Baker & Matthews [2001, 2004]. DIS is competitive on standard optical flow benchmarks. DIS runs at 300Hz up to 600Hz on a single CPU core\(^1\), reaching the temporal resolution of human’s biological vision system. It is order(s) of magnitude faster than state-of-the-art methods in the same range of accuracy, making DIS ideal for real-time applications.

2.1 Introduction

Optical flow estimation is under constant pressure to increase both its quality and speed. Such progress allows for new applications. A higher speed enables its inclusion into larger systems with extensive subsequent processing (e.g. reliable features for motion segmentation, tracking or action/activity recognition) and its deployment in computationally constrained scenarios (e.g. embedded systems, autonomous robots, large-scale data processing).

A robust optical flow algorithm should cope with discontinuities (outliers, occlusions, motion discontinuities), appearance changes (illumination, chromaticity, blur, deformations), and large displacements. Decades after the pioneering research of Horn & Schunck

\(^1\)1024×436 resolution. 42Hz / 46Hz when including pre-processing: disk access, image re-scaling, gradient computation. More details in section 2.3.1.
Figure 2.1: Our DIS method runs at 10Hz up to 600Hz on a single core CPU for an average end-point pixel error smaller or similar to top optical flow methods at similar speed. This plot excludes pre-processing time for all methods. Details in sections 2.3.1 and 2.3.4.

[1981] and Lucas & Kanade [1981] we have solutions for the first two issues (Black & Anandan [1996]; Papenberg et al. [2006]). Recent endeavors lead to significant progress in handling large displacements: Steinbrucker et al. [2009]; Brox & Malik [2011]; Braux-Zin et al. [2013]; Leordeanu et al. [2013]; Timofte & Van Gool [2015]; Kennedy & Taylor [2015]; Menze et al. [2015]; Revaud et al. [2015]; Bail et al. [2015]; Weinzaepfel et al. [2013]; Wills et al. [2006]; Wulff & Black [2015]; Xu et al. [2012]; Fischer et al. [2015]. This came at the cost of high run-times usually not acceptable in computationally constrained scenarios such as real-time applications. Recently, only very few works aimed at balancing accuracy and run-time in favor of efficiency: Farnebäck [2003]; Tao et al. [2012]; Wulff & Black [2015], or employed massively parallelized dedicated hardware to achieve acceptable run-times: Bao et al. [2014]; Plyer et al. [2014]; Fischer et al. [2015]. In contrast to this, recently it has been noted by Handa et al. [2012]; Dai et al. [2015]; Srinivasan et al. [2013]; Benosman et al. [2014]; Barranco et al. [2014] for several computer vision tasks, that it is often desirable to trade-off powerful but complex algorithms for simple and efficient methods, and rely on high frame-rates and smaller search spaces for good accuracy. In this chapter we focus on improving the speed of optical flow in general, non-domain-specific scenarios, while remaining close to the state-of-the-art flow quality. We propose two novel components with low time complexity, one using inverse search for fast patch correspondences, and one based on multi-scale aggregation for fast dense flow estimation. Additionally, a fast variational refinement step further improves the accuracy of our dense inverse search-based method. Altogether, we obtain speed-ups of 1-2 orders of magnitude over state-of-the-art methods at similar flow quality operating
2.1. INTRODUCTION

The run-times are in the range of 10-600 Hz on 1024×436 resolution images, depending on the selected trade-off between run-time and accuracy, by using a single CPU core on a common desktop PC. The method reaches the temporal resolution of human’s biological vision system. To the best of our knowledge, this is the first time that optical flow at several hundred frames-per-second has been reached with such high flow quality on any hardware.

2.1.1 Related work

Providing an exhaustive overview as by Fortun et al. [2015] of optical flow estimation is beyond the scope of this section. Most of the work on improving the time complexity (without trading-off quality) combines some of the following ideas:

While, initially, the feature descriptors of choice were extracted sparsely, invariant under scaling or affine transformations, as analyzed by Mikolajczyk et al. [2005], the recent trend in optical flow estimation is to densely extract rigid (square) descriptors from local frames, as proposed by Tola et al. [2008]; Brox & Malik [2011]; Liu et al. [2011]. HOG, proposed by Dalal & Triggs [2005], SIFT, proposed by Lowe [2004], and SURF, proposed by Bay et al. [2008] are among the most popular square patch support descriptors. In the context of scene correspondence, the algorithms SIFT-flow, proposed by Liu et al. [2011], and PatchMatch, proposed by Barnes et al. [2010], use descriptors or small patches. The descriptors are invariant only to similarities which may be insufficient especially for large displacements and challenging deformations, as analyzed by Brox & Malik [2011]. Gadot & Wolf [2016] learn descriptors appropriate for optical flow using Siamese CNNs.

The feature matching usually employs a (reciprocal) nearest neighbor operation, used—among others— by Lowe [2004]; Barnes et al. [2010]; Brox & Malik [2011]; Gadot & Wolf [2016]. Important exceptions are the recent works of Weinzaepfel et al. [2013] (non-rigid matching inspired by deep convolutional nets), of Leordeanu et al. [2013] (enforcing affine constraints), and of Timofte & Van Gool [2015] (robust matching inspired by compressed sensing). They follow Brox & Malik [2011] and guide a variational optical flow estimation through (sparse) correspondences from the descriptor matcher and can thus handle arbitrarily large displacements. Xu et al. [2012] combine SIFT, proposed by Lowe [2004], and PatchMatch, proposed by Barnes et al. [2010], for refined flow level initialization at the expense of computational costs.

An optimization problem is often at the core of the flow extraction methods. The flow is estimated by minimizing an energy that sums up matching errors and smoothness constraints. While Horn & Schunck [1981] proposed a variational approach to globally optimize the flow, Lucas & Kanade [1981] solve the correspondence problem locally and independently for image patches. Local methods, as proposed by Lucas & Kanade [1981]; Tao et al. [2012]; Senst et al. [2012], are usually faster but less accurate than the global ones. Given location and smoothness priors over the image, MRF formulations
are used by Heitz & Bouthemy [1993]; Szeliski et al. [2008]. Recently full optimization over discrete grids has been successfully applied by Menze et al. [2015]; Chen & Koltun [2016].

Parallel computation is a natural way of improving the run-time of the optical flow methods by (re)designing them for parallelization. The industry historically favored specialized hardware such as FPGAs, proposed by Pauwels et al. [2012], while the recent years brought the advance of GPUs, used by Plyer et al. [2014]; Bao et al. [2014]; Fischer et al. [2015]; Zach et al. [2007]. Yet, multi-core design on the same machine is the most common parallelization. However, many complex flow methods are difficult to adapt for parallel processing.

Learning. Most of the optical flow methods exploit training images for parameter tuning. However, this is only a rough embedding of prior knowledge. Only recently methods were proposed that successfully learn specific models from such training material. Wulff & Black [2015] assume that any flow field can be approximated by a decomposition over a learned basis of flow fields. Fischer et al. [2015] construct Convolutional Neural Networks (CNNs) to solve the optical flow estimation. Gadot & Wolf [2016] learn patch similarities using Siamese CNNs. Fischer et al. [2015] directly regresses two-dimensional flow fields from two input images using a Siamese CNN.

Coarse-to-fine optimizations have been applied frequently to flow estimation: Enkelmann [1988]; Brox & Malik [2011]; Hu et al. [2016]. They are beneficial to avoid poor local minima, especially for large motions, and thus to improve the performance and to speed up the convergence.

Branch and bound and priority queues have been used to find smart strategies to first explore the flow in the most favorable image regions and gradually refine it for the more ambiguous regions. This often leads to a reduction in computational costs. The PatchMatch methods by Barnes et al. [2010]; Gadot & Wolf [2016]; Hu et al. [2016] follow a branch and bound strategy, gradually fixing the most promising correspondences. Bao et al. [2014] propose an edge-preserving extension (EPPM) based on PatchMatch.

Dynamic Vision Sensors, as proposed by Lichtsteiner et al. [2008], asynchronously capturing illumination changes at microsecond latency, have been used to compute optical flow. Benosman et al. [2014] and Barranco et al. [2014] note that realistic motion estimation, even with large displacements, becomes simple when capturing image evidence in the kilohertz-range.

2.1.2 Contributions

We present a novel optical flow method based on dense inverse search (DIS), which we demonstrate to provide high quality flow estimation at 10-600 Hz on a single CPU core. This method is 1-2 orders of magnitude times faster than previous results by Weinzaepfel
2.2. Proposed method

In the following, we introduce our dense inverse search-based method (DIS) by describing: how we extract single point correspondences between two images in section 2.2.1, how we merge a set of noisy point correspondences on each level \( s \) of a scale-pyramid into a dense flow field \( U_s \) in section 2.2.2, how we refine \( U_s \) using variational refinement in section 2.2.3, and possible extensions of DIS in section 2.2.4.
2.2. PROPOSED METHOD

2.2.1 Fast inverse search for correspondences

The core component in our method to achieve high performance is the efficient search for patch correspondences. In the following we will detail how we extract one single point correspondence between two frames.

For a given template patch $T$ in the reference image $I_t$, with a size of $\theta_{ps} \times \theta_{ps}$ pixels, centered on location $x = (x, y)^T$, we find the best-matching sub-window of $\theta_{ps} \times \theta_{ps}$ pixels in the query image $I_{t+1}$ using gradient descent. We are interested in finding a warping vector $u = (u, v)$ such that we minimize the sum of squared differences over the sub-window between template and query location:

$$u = \operatorname{argmin}_{u'} \sum_x [I_{t+1}(x + u') - T(x)]^2.$$  \hfill (2.1)

Minimizing this quantity is non-linear and is optimized iteratively using the inverse Lukas-Kanade algorithm as proposed in Baker & Matthews [2004]. For this method two steps are alternated for a number of iterations or until the quantity (2.1) converges. For the first step, the quantity (2.2) is minimized around the current estimate $u$ for an update vector $\Delta u$ such that

$$\Delta u = \operatorname{argmin}_{\Delta u'} \sum_x [I_{t+1}(x + u + \Delta u') - T(x)]^2.$$  \hfill (2.2)

The first step requires extraction and bilinear interpolation of a sub-window $I_{t+1}(x + u)$ for sub-pixel accurate warp updates. The second step updates the warping $u \leftarrow u + \Delta u$.

The original Lucas & Kanade [1981] algorithm required expensive re-evaluation of the Hessian of the image warp at every iteration. As proposed by Baker & Matthews [2004] the inverse objective function $\sum_x [T(x - \Delta u) - I_{t+1}(x + u)]^2$ can be optimized instead of (2.2), removing the need to extract the image gradients for $I_{t+1}(x + u)$ and to re-compute the Jacobian and Hessian at every iteration. Due to the large speed-up this inversion has been used for point tracking in SLAM by Klein & Murray [2007], camera pose estimation by Forster et al. [2014], and is covered in detail by Baker & Matthews [2004].

In order to gain some robustness against absolute illumination changes, we mean-normalize each patch. One challenge of finding sparse correspondences with this approach is that the true displacements cannot be larger than the patch size $\theta_{ps}$, since the gradient descent is dependent on similar image context in both patches. Often a coarse-to-fine approach with fixed window-size but changing image size is used (Klein & Murray [2007]; Forster et al. [2014]), firstly, to incorporate larger smoothed contexts at coarser scales and thereby lessen the problem of falling into local optima, secondly, to find larger displacements, and, thirdly, to ensure fast convergence.
2.2. Proposed method

Algorithm 1 Dense Inverse Search (DIS)

1: Set initial flow field $U_{\theta_{ss}+1} \leftarrow 0$
2: for $s = \theta_{ss}$ to $\theta_{sf}$ do
3: (1.) Create uniform grid of $N_s$ patches
4: (2.) Initialize displacements from $U_{s+1}$
5: for $i = 1$ to $N_s$ do
6: (3.) Inverse search for patch $i$
7: (4.) Densification: Compute dense flow field $U_s$
8: (5.) Variational refinement of $U_s$

2.2.2 Fast optical flow with multi-scale reasoning

We follow such a multi-scale approach, but, instead of optimizing patches independently, we compute an intermediate dense flow field and re-initialize patches at each level. We do this because of two reasons: 1) the intermediate dense flow field smooths displacements and provides robustness, effectively filtering outliers and 2) it reduces the number of patches on coarser scales, thereby providing a speed-up. We operate in a coarse-to-fine fashion from a first (coarsest) level $\theta_{ss}$ in a scale pyramid with a down-scaling quotient of $\theta_{sd}$ to the last (finest) level $\theta_{sf}$. On each level our method consists of five steps, summarized in algorithm 1, yielding a dense flow field $U_s$ in each iteration $s$.

(1.) Creation of a grid: We initialize patches in a uniform grid over the image domain. The grid density and number of patches $N_s$ is implicitly determined by the parameter $\theta_{ov} \in [0, 1]$ which specifies the overlap of adjacent patches and is always floored to an integer overlap in pixels. A value of $\theta_{ov} = 0$ denotes a patch adjacency with no overlap and $\theta_{ov} = 1 - \epsilon$ results in a dense grid with one patch centered on each pixel in the reference image.

(2.) Initialization: For the first iteration ($s = \theta_{ss}$) we initialize all patches with the trivial zero flow. On each subsequent scale $s$ we initialize the displacement of each patch $i \in N_s$ at its location $x$ with the flow from the previous (coarser) scale: $u_{i, \text{init}} = U_{s+1}(x/\theta_{sd}) \cdot \theta_{sd}$.

(3.) Inverse search: Optimal displacements are computed independently for all patches, as detailed in section 2.2.1. The search time required for each patch lies in the range of 1-2 $\mu$s.

(4.) Densification: After step three we have updated displacement vectors $u_i$. For more robustness against outliers, we reset all patches to their initial flow $u_{i, \text{init}}$ for which the displacement update $\|u_{i, \text{init}} - u_i\|_2$ exceeds the patch size $\theta_{ps}$. We create a dense flow field $U_s$ in each pixel $x$ by applying weighted averaging to displacement estimates of all patches overlapping at $x$ in the reference image:

$$U_s(x) = \frac{1}{Z} \sum_{i}^{N_s} \frac{\lambda_{i,x}}{\max(1, \|d_i(x)\|_2)} \cdot u_i,$$  (2.3)
where the indicator \( \lambda_{i,x} = 1 \) if patch \( i \) overlaps with location \( x \) in the reference image, \( d_i(x) = I_{t+1}(x + u_i) - T(x) \) denotes the intensity difference between template patch and warped image at this pixel, \( u_i \) denotes the estimated displacement of patch \( i \), and normalization \( Z = \sum_i \lambda_{i,x} / \max(1, \|d_i(x)\|_2) \).

(5.) Variational energy minimization of flow \( U_s \), as detailed in section 2.2.3.

### 2.2.3 Fast Variational refinement

We use the variational refinement of Weinzaepfel et al. [2013] with three simplifications: (i) We use no feature matching term, (ii) intensity images only, and (iii) refine only on the current scale. The energy is a weighted sum of intensity and gradient data terms \( (E_I, E_G) \) and a smoothness term \( (E_S) \) over the image domain:

\[
E(U) = \int_{\Omega} \sigma \Psi(E_I) + \gamma \Psi(E_G) + \alpha \Psi(E_S) \, dx
\]  

We use a robust penalizer \( \Psi(a^2) = \sqrt{a^2 + \epsilon^2} \), with \( \epsilon = 0.001 \) for all terms as proposed by Sun et al. [2010]. We use a separate penalization of intensity and gradient constancy assumption, with normalization as proposed by Zimmer et al. [2011]: With the brightness constancy assumption \( (\nabla^T_3 I)u = 0 \), where \( \nabla_3 = (\partial_x, \partial_y, \partial_z)^T \) denotes the spatio-temporal gradient, we can model the intensity data term as \( E_I = u^T \bar{J}_0 u \). We use the normalized tensor \( \bar{J}_0 = \beta_0 (\nabla^T_3 I) (\nabla^T_3 I) \) to enforce brightness constancy, with normalization \( \beta_0 = (\|\nabla_3 I\|^2 + 0.01)^{-1} \) by the spatial derivatives and a term to avoid division by zero as in Zimmer et al. [2011].

Similarly, \( E_G \) penalizes the gradient constancy: \( E_G = u^T \bar{J}_{xy} u \) with

\[
\bar{J}_{xy} = \beta_x (\nabla_3^T I_{dx}) (\nabla_3^T I_{dx}) + \beta_y (\nabla_3^T I_{dy}) (\nabla_3^T I_{dy})
\]

and normalizations \( \beta_x = (\|\nabla_2 I_{dx}\|^2 + 0.01)^{-1} \) and \( \beta_y = (\|\nabla_2 I_{dy}\|^2 + 0.01)^{-1} \). The smoothness term is a penalization over the norm of the gradient of displacements:

\[
E_S = \|\nabla u\|^2 + \|\nabla v\|^2
\]

The non-convex energy \( E(U) \) is minimized iteratively with \( \theta_{vo} \) fixed point iterations and \( \theta_{vs} \) iterations of Successive-Over-Relaxation for the linear system, as in Brox et al. [2004].

### 2.2.4 Extensions

Our method lends itself to multiple extensions. We examined the benefit of, firstly, using RGB color images instead of intensity images, secondly, enforcing forward-backward consistency of optical flow, and, thirdly, using robust error norms.
(i) Using RGB color images. Instead of using a patch matching error over intensity images, we can use a multi-channel image for the dense inverse search and the variational refinement. The objective function for dense the inverse search, eq. (2.1), becomes

$$\sum_{c \in \{R,G,B\}} \sum_{x} \left[ I_{t+1}^{c}(x + u) - T^{c}(x) \right]^2.$$  \hspace{1cm} (2.7)

where $c$ iterates over all color channels. Similarly, we can extend the variational refinement to operate on all color channels jointly, as detailed by Weinzaepfel et al. [2013].

(ii) Enforcing forward-backward consistency. In order to enforce forward-backward consistency, we run our algorithm in parallel from both directions: $I_t \rightarrow I_{t+1}$ and $I_{t+1} \rightarrow I_t$. With the exception of the densification (step 4), all steps of the algorithm run independently for forward and backward flow computation. In step 4 we merge both directions and create a dense forward flow field $U_{fs}^f$ in each pixel $x$ in the reference image $I_t$:

$$U_{fs}^f(x) = \frac{1}{Z} \left[ \sum_{i}^{N_f} \frac{\lambda_{i,x}^f}{\max(1, \|d_i^f(x)\|_2)} \cdot u_{i}^f - \sum_{j}^{N_b} \frac{\lambda_{j,x}^b}{\max(1, \|d_j^b(x)\|_2)} \cdot u_{j}^b \right]  \hspace{1cm} (2.8)$$

where the indicator $\lambda_{i,x}^f = 1$ iff patch $i$ for the forward displacement estimate overlaps with location $x$ in the reference image, $\lambda_{j,x}^b = 1$ iff patch $j$ for the backward displacement estimate overlaps with location $x$ in the reference image after the displacement $u_{j}^b$ was applied to it, $d_i^f(x) = I_{t+1}(x + u_{i}^f) - T(x)$ and $d_j^b(x) = I_{t}(x) - T(x - u_{j}^b)$ denote the forward and backward warp intensity differences between template patches and warped images, $u_{i}^f$ and $u_{j}^b$ denote the estimated forward and backward displacements of patches, and normalization

$$Z = \sum_{i}^{N_f} \frac{\lambda_{i,x}^f}{\max(1, \|d_i^f(x)\|_2)} + \sum_{j}^{N_b} \frac{\lambda_{j,x}^b}{\max(1, \|d_j^b(x)\|_2)}.$$  \hspace{1cm} (2.9)

Since displacement estimate $u_{j}^b$ is generally not integer, we employ bilinear interpolation for the second term in equation (2.8).

The densification for the dense backward flow field $U_{fs}^b$ is computed analogously. After the densification step on each scale, the variational refinement is again performed independently for each direction.
2.2. Proposed method

Figure 2.2: Evaluation of extensions. (1) Baseline, (2) baseline with color images, (3) baseline enforcing forward-backward consistency, (4) baseline using the L1 norm as objective function, (5) baseline using the Huber norm as objective function, (6) baseline using color images, enforcing forward-backward consistency and using the Huber norm as objective function.

(iii) Robust error norms. Equation (2.1) minimizes an L2 norm which is strongly affected by outliers. Since the L1 and the Huber norm are known to be more robust towards outliers, we examined their effect on our algorithm. However, the objective function (2.1) cannot easily be changed to directly minimize a different error norm. But, in each iteration we can transform the error $\varepsilon = I_{t+1}(x+u) - T(x)$ for each pixel, such that, implicitly after squaring the transformed error minimizes a different norm. We transform the error $\varepsilon$ on each pixel at location $x$ as follows:

- L1-Norm:
  $$\varepsilon \leftarrow \text{sign}(\varepsilon) \cdot \sqrt{|\varepsilon|}$$

- Huber-Norm:
  $$\varepsilon \leftarrow \begin{cases} 
  \varepsilon, & \text{if } \varepsilon < b \\
  \text{sign}(\varepsilon) \cdot \sqrt{2b |\varepsilon| - b^2}, & \text{otherwise}
  \end{cases}$$

After the transformed error $\varepsilon$ is squared in the objection function the corresponding L1 or Huber norm is minimized in each iteration.

We plotted the result for all three extensions in Fig. 2.2, where we compare all extensions on the Sintel training benchmark against our method without these extensions. As baseline we start from operating point (3) and evaluate all extensions separately. The six numbered operating points correspond to: (1) Baseline, (2) baseline with color images, (3) baseline enforcing forward-backward consistency, (4) baseline using the L1 norm as objective function, (5) baseline using the Huber norm as objective function, (6) baseline using color images, enforcing forward-backward consistency and using the Huber norm as objective function.
2.3 Experiments

In order to evaluate the performance of our method, we present three sets of experiments. Firstly, we conduct an analysis of our parameter selection in section 2.3.1. Here, we also study the impact of variational refinement in our method. Secondly, we evaluate the inverse search (step 3 in algorithm 1) in section 2.3.3 without densification (step 4). The complete pipeline for optical flow is evaluated in section 2.3.4, and 2.3.5. Thirdly, since the problem of recovering large displacements can also be handled by higher frame-rates combined with lower run-time per frame-pair, we conduct an experiment in section 2.3.6 to analyze the benefit of higher frame-rates. In section 2.3.7 we conclude with an adaptation of our method to the task of computing stereo disparities. For a calibrated pair of aligned cameras extracting stereo disparities on jointly exposed images is equivalent to range imaging. We analyze how DIS compares to existing methods.

2.3.1 Implementation and Parameter Selection

We implemented\(^2\) our method in C++ and run all experiments and baselines on a Core i7 CPU using a single core, and a GTX780 GPU for the EPPM baseline, published by Bao et al. [2014]. For all experiments on the Sintel and KITTI training datasets we report timings from which we exclude all operations which, in a typical robotics vision application, would be unnecessary, performed only once, or shared between multiple tasks: Disk access, creation of an image pyramid including image gradients with a down-sampling

\(^2\)Source code available: http://www.vision.ee.ethz.ch/~kroegert/OFlow/
quotient of 2, all initializations of the flow algorithms. We do this for our method and all baselines within their provided code. For EPPM, where only an executable was available, we subtracted the average overhead time of our method for fair comparison. For experiments on the Sintel and KITTI test datasets (Tables 2.7, 2.8) we include this pre-processing time to be comparable with reported timings in the online benchmarks.

### 2.3.2 Parallelization and memory consumption

For our method no special data structures are needed. What enables the high speed is a combination of a very fast flow initialization of DIS with a slow variational refinement per scale. Initialization and variational refinement constitute 32% and 57%, respectively, of the total run-time on each scale. Care was taken to allocate all memory at once, use SSE instructions and inplace-operations whenever possible, and terminate iterations early for small residuals. Bottlenecks for further speed-ups are repeated (bilinearly interpolated) patch extraction (step 3 in algorithm 1) and pixel-wise refinements (step 5).

In table 2.1 we show a direct run-time comparison on Sintel- and VGA-resolution images. The run-time scales approximately linearly with the image area.

<table>
<thead>
<tr>
<th></th>
<th>DIS (1)</th>
<th>DIS (2)</th>
<th>DIS (3)</th>
<th>DIS (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sintel</td>
<td>1.65 / 606</td>
<td>3.32 / 301</td>
<td>97.8 / 10.2</td>
<td>1925 / 0.52</td>
</tr>
<tr>
<td>VGA</td>
<td>0.93 / 1075</td>
<td>2.35 / 426</td>
<td>70.3 / 14.2</td>
<td>1280 / 0.78</td>
</tr>
</tbody>
</table>

*Table 2.1: Sintel (1024×436), VGA (640×480) run-times in (ms/Hz).*

We will break down the run-time of 3.32 ms for DIS (2) on Sintel-resolution images, running over 3 scales, for all components in table 2.2.

<table>
<thead>
<tr>
<th>Total run-time (3.32 ms)</th>
<th>ms</th>
<th>% of total run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory allocation</td>
<td>0.17</td>
<td>5.27</td>
</tr>
<tr>
<td>Processing scale $\theta_s = 5$</td>
<td>0.23</td>
<td>6.91</td>
</tr>
<tr>
<td>Processing scale $\theta_s = 4$</td>
<td>0.65</td>
<td>19.6</td>
</tr>
<tr>
<td>Processing scale $\theta_s = 3$</td>
<td>2.26</td>
<td>68.1</td>
</tr>
</tbody>
</table>

*Table 2.2: Break down of DIS (2) total run-time on Sintel images*

Run-time grows approximately by a factor of 4 for each finer scale, and is spend similarly on each scale. Representatively for the last scale ($\theta_s = 3$), corresponding to a down-scaling factor of 8 and 448 uniformly distributed 8x8 patches, we break down the time of 2.26 ms spend on this layer following Alg. 1 in table 2.3.

Step (3.) and (5.) are most time-consuming. Step (3.) breaks down in 1.66 $\mu$s for each of 448 patches. This breaks down into 0.08 $\mu$s for initialization (gradient/intensity extraction
2.3. Experiments

<table>
<thead>
<tr>
<th>Run-time on scale $\theta = 3$ (2.26 ms)</th>
<th>ms</th>
<th>% of run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patch initialization, steps (1./2.)</td>
<td>0.09</td>
<td>4.1</td>
</tr>
<tr>
<td>Inverse Search, step (3.)</td>
<td>0.74</td>
<td>32.8</td>
</tr>
<tr>
<td>Densification, (4.)</td>
<td>0.14</td>
<td>5.9</td>
</tr>
<tr>
<td>Variational refinement, (5.)</td>
<td>1.29</td>
<td>57.1</td>
</tr>
</tbody>
</table>

Table 2.3: Break down of DIS (2) run-time on one scale

<table>
<thead>
<tr>
<th>DIS operating point</th>
<th>(1.)</th>
<th>(2.)</th>
<th>(3.)</th>
<th>(4.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed-up (x)</td>
<td>1.29</td>
<td>1.75</td>
<td>2.15</td>
<td>3.70</td>
</tr>
</tbody>
</table>

Table 2.4: Speed-up factor from parallelization (OpenMP, 4 cores) for all operating points

in reference image, computation of Hessian) and 0.13 $\mu$s for each of 12 optimization iterations (template intensity extraction, parameter update). Step (5.) breaks down for each of $\theta_{\nu_o} = 4$ iterations into 0.12 ms for computation of the data and smoothness terms and 0.2 ms for solving of the linear systems and updating the flow field per pixel.

Parallelization

We examined the potential speed-up by parallelization using OpenMP on 4 cores. We parallelized step 3 and 5 of our implementation, to operate independently on each patch and pixel, respectively. The run-times for all for operating points are tabulated in table 2.4. Since thread creation and management leads to an overhead of a few milliseconds, the speed-up of 4 cores only becomes significant for longer run-times. For longer sequences, where threads are created only once, this overhead would be negligible. Since each thread executes the same instructions and there is no need to communicate, a massive parallelization on a GPU will potentially yield an even larger speed-up.

Memory consumption

We also examined the peak memory consumption of our algorithm on images from the Sintel benchmark. We tabulated the result in table 2.5. Roughly 15 MB are used by

<table>
<thead>
<tr>
<th>Peak Mem. (MB)</th>
<th>DIS (1.)</th>
<th>DIS (2.)</th>
<th>DIS (3.)</th>
<th>DIS (4.)</th>
<th>Farneback</th>
<th>PCAFlow</th>
<th>DeepFlow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed (ms)</td>
<td>1.65</td>
<td>3.32</td>
<td>97.8</td>
<td>1925</td>
<td>43.31</td>
<td>1216</td>
<td>6851</td>
</tr>
<tr>
<td>Speed (Hz)</td>
<td>606</td>
<td>301</td>
<td>10.2</td>
<td>0.52</td>
<td>38.06</td>
<td>2.38</td>
<td>0.018</td>
</tr>
</tbody>
</table>

Table 2.5: Peak memory consumption (MB) on Sintel-resolution images for all four operating points and for the baselines: Farneback method, PCAFlow and DeepFlow at full resolutions.
2.3. Experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_{sf}$</td>
<td>finest scale in multi-scale pyramid</td>
</tr>
<tr>
<td>$\theta_{it}$</td>
<td>number of gradient descent iterations per patch</td>
</tr>
<tr>
<td>$\theta_{ps}$</td>
<td>rectangular patch size in (pixel)</td>
</tr>
<tr>
<td>$\theta_{ov}$</td>
<td>patch overlap on each scale (percent)</td>
</tr>
<tr>
<td>$\theta_{sd}$</td>
<td>down-scaling quotient in scale pyramid</td>
</tr>
<tr>
<td>$\theta_{ss}$</td>
<td>coarsest scale in multi-scale pyramid</td>
</tr>
<tr>
<td>$\theta_{vo}, \theta_{vi}$</td>
<td>number of outer and inner iterations for variational refinement</td>
</tr>
<tr>
<td>$\delta, \gamma, \alpha$</td>
<td>intensity, gradient and smoothness weights for variational refinement</td>
</tr>
</tbody>
</table>

| Table 2.6: Parameters of our method. Parameters in **bold** have a significant impact on performance and are cross-validated in section 2.3.1. |

image data, and the rest by all patches and associated data, which is pre-allocated during initialization. If parallelization is not used, and memory is a bottleneck, the memory footprint can be reduced by not pre-allocating all patch memory.

Parameter selection

Our method has four main parameters which affect speed and performance as explained in section 2.2: $\theta_{ps}$ size of each rectangular patch, $\theta_{ov}$ patch overlap, $\theta_{it}$ number of iterations for the inverse search, $\theta_{sf}$ finest and final scale on which to compute the flow. We plot the change in the average end-point error (EPE) versus run-time on the Sintel (training, final) dataset (Butler et al. [2012]) in Fig. 2.3. We draw three conclusions: Firstly, operating on finer scales (lower $\theta_{sf}$), more patch iterations (higher $\theta_{it}$), higher patch density (higher $\theta_{ov}$) generally lowers the error, but, depending on the time budget, may not be worth it. Secondly, the patch size $\theta_{ps}$ has a clear optimum at 8 and 12 pixels. This also did not change when varying $\theta_{ps}$ at lower $\theta_{sf}$ or higher $\theta_{it}$. Thirdly, using variational refinement always significantly reduced the error for a moderate increase in run-time.

In addition we have several parameters of lower importance, which are fixed for all experiments. We set $\theta_{sd} = 2$, i.e. we use an image pyramid, where the resolution is halved with each down-scaling. We set the coarsest image scale $\theta_{ss} = 5$ for section 2.3.4 and $\theta_{ss} = 6$ for section 2.3.5 due to higher image resolutions. For different patch sizes and image pyramids the coarsest scale can be selected as $\theta_{ss} = \log_{\theta_{sd}}(2 \cdot width)/(f \cdot \theta_{ps})$ and raised to the nearest integer, to capture motions of at least $1/f$ of the image width. For the variational refinement we fix intensity, gradient and smoothness weights as $\delta = 5, \gamma = 10, \alpha = 10$ and keep iteration numbers fixed at $\theta_{vo} = 1 \cdot (s + 1)$, where $s$ denotes the current scale and $\theta_{vi} = 5$. In contrast to our comparison baselines, as published by Weinzaepfel et al. [2013]; Timofte & Van Gool [2015]; Wulff & Black [2015], we do not fine-tune DIS for a specific dataset. We use a 20 percent subset of Sintel training to develop our method, and only the remaining training material is used for evaluation. All parameters are summarized in Table 2.6. If the flow is not computed up to finest scale ($\theta_{sf} = 0$), we scale-up the result (linearly interpolated) to full resolution for comparison for all methods.
2.3. Experiments

### 2.3.3 Evaluation of Inverse Search

In this section we evaluate the sparse point correspondences created by inverse search on the Sintel training dataset. For each frame pair we initialized a sparse grid (given by Deep Matching, proposed by Weinzaepfel et al. [2013]) in the first image and computed point correspondences in the second image. The correspondences are computed by i) exhaustive Nearest Neighbor search on normalized cross-correlation (\(NCC\)), ii) our method where we skip the densification step between each scale change (\(DIS\) w/o Densification), iii) our method including the densification step (\(DIS\)), and using iv) DeepMatching. The results are shown in Fig. 2.5 and Table 2.4.

We have four observations: i) Nearest Neighbor search has a low number of incorrect matches, but precise correspondences and is very prone to outliers. ii) DeepMatching has a high percentage of erroneous correspondences (with small errors), but is very good at large displacements. iii) In contrast to this, our method (\(DIS\) w/o Densification) generally performs well in the range of small displacements, but is strongly affected by outliers. This is due to the fact that the implicit SSD (sum of squared differences) error minimization is not invariant to changes in orientation, contrast, and deformations. iv) Averaging all patches in each scale (\(DIS\)), taking into account their photometric error as described in eq. (2.3), introduces robustness towards these outliers. It also decreases the error for approximately correct matches. Furthermore, it enables reducing the number of patches at coarser scales, leading to lower run-time.

### 2.3.4 MPI Sintel optical flow results

Following our parameter evaluation in section 2.3.1, we selected four operating points:

1. \(\theta_{sf} = 3, \theta_{it} = 016, \theta_{ps} = 08, \theta_{ov} = 0.30\), at 600/46\(^3\) Hz,
2. \(\theta_{sf} = 3, \theta_{it} = 012, \theta_{ps} = 08, \theta_{ov} = 0.40\), at 300/42 Hz,
3. \(\theta_{sf} = 1, \theta_{it} = 016, \theta_{ps} = 12, \theta_{ov} = 0.75\), at 10/8.3 Hz,

\(^3\)Without / with image pre-processing: disk access, image gradients and re-scaling.
2.3. Experiments

Figure 2.6: Sintel-training results: Average end-point error (EPE, in pixels) versus run-time (millisecond) on various displacement ranges.

Table 2.7: Sintel test errors in pixels (http://sintel.is.tue.mpg.de/results), retrieved on 25th of July 2016 for final subset. Run-times are measured by us, except: † self-reported, and ‡ on other datasets with same or smaller resolution.

(4) $\theta_{sf} = 0, \theta_{it} = 256, \theta_{pa} = 12, \theta_{ov} = 0.75$, at 0.5/0.5 Hz.

We compare our method against a set of recently published baselines running on a single CPU core: DeepFlow, proposed by Weinzaepfel et al. [2013], SparseFlow, proposed by Timofte & Van Gool [2015], PCA-Flow, proposed by Wulff & Black [2015]; two older established methods: Pyramidal Lukas-Kanade Flow, as presented in Bouguet [2001]; Lucas & Kanade [1981], Farneback’s method, proposed by Farnebäck [2003]; and one recent GPU-based method: EPPM, proposed by Bao et al. [2014]. Since run-times for optical flow methods are strongly linked to image resolution, we incrementally speed-up all baselines by down-scaling the input images by factor of $2^n$, where $n$ starting at $n = 0$ is increased in increments of 0.5. We chose this non-intrusive parameter of image resolution to analyze each method’s trade-off between run-time and flow error. We bilinearly interpolate the resulting flow field to the original resolution for evaluation. We also experiment with temporal instead of spatial down-sampling for the same purpose, as described in section 2.3.6.

We run all baselines and DIS for all operating points on the Sintel (Butler et al. [2012]) final training (Fig. 2.6) and testing (Table 2.7) benchmark. On the testing benchmark we report operating point (2) for DIS. As noted in section 2.3.1, run-times for all methods are reported without pre-processing for the training dataset to facilitate comparison.
2.3. Experiments

![Log. run−time (ms) vs Avg. EPE for different methods](image)

**Figure 2.7:** KITTI (training) result. Average end-point error (px) versus run-time (ms) for all (left) and small displacements (right, s0-10).

<table>
<thead>
<tr>
<th>Method</th>
<th>Out-Noc</th>
<th>Out-All</th>
<th>Avg-Noc</th>
<th>Avg-All</th>
<th>time (s)</th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>PH-Flow, Yang &amp; Li [2015]</td>
<td>5.76 %</td>
<td>10.57 %</td>
<td>1.3 px</td>
<td>2.9 px</td>
<td>800</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DeepFlow, Weinzaepf [2013]</td>
<td>7.22 %</td>
<td>17.79 %</td>
<td>1.5 px</td>
<td>5.8 px</td>
<td>17</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SparseFlow, Timofte &amp; Van Gool [2015]</td>
<td>9.09 %</td>
<td>19.32 %</td>
<td>2.6 px</td>
<td>7.6 px</td>
<td>10</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>EPPM, Bao et al. [2014]</td>
<td>12.75 %</td>
<td>23.55 %</td>
<td>2.5 px</td>
<td>9.2 px</td>
<td>0.25</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>PCA-Flow, Wulff &amp; Black [2015]</td>
<td>15.67 %</td>
<td>24.59 %</td>
<td>2.7 px</td>
<td>6.2 px</td>
<td>0.19</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>eFolki, Plyer et al. [2014]</td>
<td>19.31 %</td>
<td>28.79 %</td>
<td>5.2 px</td>
<td>10.9 px</td>
<td>0.026</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>LDOF, Brox &amp; Malik [2011]</td>
<td>21.93 %</td>
<td>31.39 %</td>
<td>5.6 px</td>
<td>12.4 px</td>
<td>0.60</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DIS-Fast</td>
<td>37.05 %</td>
<td>44.49 %</td>
<td>5.0 px</td>
<td>9.1 px</td>
<td>0.08</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FlowNetS-ft, Fischer et al. [2015]</td>
<td>38.58 %</td>
<td>46.21 %</td>
<td>7.8 px</td>
<td>14.4 px</td>
<td><strong>0.024</strong></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>RLOF, Senst et al. [2012]</td>
<td>38.60 %</td>
<td>46.13 %</td>
<td>8.7 px</td>
<td>16.5 px</td>
<td>0.488</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.8: KITTI test results (http://www.cvlibs.net/datasets/kitti/eval_flow.php), retrieved on 25th of July 2016, for all pixels, at 3px threshold.

Of algorithms running in the same environment at high speed, and with pre-processing for the online testing benchmark to allow comparison with self-reported times. From the experiments on the testing and training dataset, we draw several conclusions: Operating point (2) points to the best trade-off between run-time and flow error. For the average EPE of around 6 pixels, DIS is approximately two orders of magnitude faster than the fastest CPU baseline (PCA-Flow, Wulff & Black [2015]) and also more than one order of magnitude faster than the fastest GPU baseline (EPPM, Bao et al. [2014]). DIS can be further sped-up by removing the variational refinement as in operating point (1) while maintaining reasonable flow quality (see Fig. 2.8). We also tested using only the variational refinement without sparse initialization ($\theta^1 = 0$), and found experimentally that the result is close to the trivial zero-flow solution. Finer resolution changes over scales and more iterations for the refinement will yield better results at significantly increased cost. Operating point (3) is comparable with the performance of EPPM, but slightly better for small displacements and worse for large displacements. If we use all available scales, and increase the number of iterations, we obtain operating point (4). At the run-time of several seconds per frame pair, more complex methods, such as DeepFlow, perform better, in particular for large displacements.
2.3. Experiments

2.3.5 KITTI optical flow results

Complementary to the experiment on the synthetic Sintel dataset, we ran our method on the KITTI Optical Flow benchmark (Geiger et al. [2013]) for realistic driving scenarios. We use the same experimental setup and operating points as in section 2.3.4. The result is presented in Fig. 2.7, 2.9 (training) and Table 2.8 (testing). Our conclusions from the Sintel dataset in section 2.3.4 also apply for this dataset, suggesting a stable performance of our method, since we did not optimize any parameters for this dataset. On the online test benchmark, for which we include our pre-processing time, we are on par with RLOF, as proposed by Senst et al. [2012], and FlowNet, as proposed by Fischer et al. [2015]. Even though both take advantage of a GPU, we are still significantly faster at comparable performance.

2.3.6 High frame-rate optical flow

Often, a simpler and faster algorithm, combined with a higher temporal resolution in the data, can yield better accuracy than a more powerful algorithm, on lower temporal res-
2.3. Experiments

Figure 2.10: Flow result on Sintel with low temporal resolution. Accuracy of DeepFlow on large displacements versus DIS on small displacements, tracked through all intermediate frames. As baseline we included the accuracy of DeepFlow for tracking small displacements. Note: While we use the same frame pairs to compute each vertical set of points, frame pairs differ over step-sizes.

Figure 2.11: Optical flow on Sintel with lower temporal resolution. In each block of 3x4: Rows, top to bottom, correspond to step sizes 1 (original frame-rate), 6, 10 frames. Columns, left to right, correspond to new ground truth, DeepFlow result, DIS result (through all intermediate frames), original images. Large displacements are significantly better preserved by DIS through higher frame-rates.

olutions. This has been analyzed in detail by Handa et al. [2012] for the task of visual odometry. As noted by Benosman et al. [2014]; Barranco et al. [2014] this is also the case for optical flow, where large displacements, due to low-frame rate or strong motions are significantly more difficult to estimate than small displacements. In contrast to the recent focus on handling ever larger displacements (Brox & Malik [2011]; Xu et al. [2012]; Weinzaepfel et al. [2013]; Timofte & Van Gool [2015]; Menze et al. [2015]; Hu et al. [2016]), we want to analyze how decreasing the run-time while increasing the frame-rate
affects our algorithm. For this experiment we selected a random subset of the Sintel training dataset, and synthesized new ground truth flow for lower frame-rates from the one provided in the dataset. We create new ground truth for 1/2 to 1/10 of the source frame-rate from the original ground truth and the additionally provided segmentation masks to invalidate occluded regions. We compare DeepFlow at a speed of 0.5Hz on this lower temporal resolution against DIS (operating point (3), 10 Hz), running through all intermediate frames at the original, higher frame-rate. Thus, while DeepFlow has to handle larger displacements in one frame pair, DIS has to handle smaller displacements, tracked through multiple frames and accumulates error drift. We observe (Fig. 2.10) that DIS starts to outperform DeepFlow when running at 2× the original frame-rate, notably for large displacements, while still being 10× faster. Fig. 2.11 shows examples of the new ground truth, results of DeepFlow and DIS. We conclude, that it is advantageous to choose DIS over DeepFlow, aimed at recovering large displacements, when the combination of frame-rate and run-time per frame can be chosen freely.
2.4 Conclusions

2.3.7 Depth from Stereo with DIS

We can also apply our algorithm to the problem of computing depth from a stereo pair of images. If the image planes of both cameras are parallel and aligned in depth, the epipoles are at infinity, and the depth computation task becomes the problem of pixel-wise estimation of horizontal displacements. We remove the vertical degree of freedom from our method, and evaluate on the Middlebury dataset for depth from stereo, as proposed by Scharstein et al. [2014]. We evaluate against four methods: Semi-Global Matching (SGM), proposed by Hirschmüller [2008], Block Matching (BM), as proposed by Konolige [1998], Efficient Large-scale Stereo Matching (ELAS), as proposed by Geiger et al. [2011] and Slanted-Plane Stereo (SPS), as proposed by Yamaguchi et al. [2014]. We use the same 4 operating points, but change: iteration numbers are halved, \( \theta_{it} \leftarrow \theta_{it}/2 \).

The result is displayed in Fig. 2.12. We have two observations. Firstly, while operating point (1) and (2) are still much faster than all baseline methods for the same error, the speed-benefit is smaller than in the optical flow experiments. Secondly, for all baseline methods the optimal performance is achieved with a down-scaled input image pair instead of the finest resolution. This suggests that these methods were fine-tuned for images with smaller resolutions than those provided in the recently published Middlebury benchmark. In consequence, for these methods several tuning parameters have to be adapted to deal with large input resolutions. We observe that our method is more robust to those changes.

2.4 Conclusions

In this chapter we presented a novel and simple way of computing dense optical flow. The presented approach trades off a lower flow estimation error for large decreases in runtime: For the same level of error, the presented method is two orders of magnitude faster than current state-of-the-art approaches, as shown in experiments on synthetic (Sintel) and realistic (KITTI) optical flow benchmarks. Our study leaves several open challenges: Due to the coarse-to-fine approach small and fast motions can sometimes get lost beyond recovery. A sampling-based approach to recover over-smoothed object motions at finer scales may alleviate this problem. The implicit minimization of the L2 matching error in our method is not invariant to many modes of change, such as in contrast, deformations, and occlusions. More robust error metrics may be helpful here. Furthermore, a GPU implementation may yield another significant speed-up.
We propose a new tracking-by-detection algorithm for multiple targets from multiple dynamic, unlocalized and unconstrained cameras. In the past tracking has either been done with multiple static cameras, or single and stereo dynamic cameras. We register several moving cameras using a given 3D model from Structure from Motion (SfM), and initialize the tracking given the registration. The camera uncertainty estimate can be efficiently incorporated into a flow-network formulation for tracking. As this is a novel task in the tracking domain, we evaluate our method on a new challenging dataset for tracking with multiple moving cameras and show that our tracking method can effectively deal with independently moving cameras and camera registration noise.

3.1 Introduction

Simultaneous object tracking across multiple views is commonly solved with strong restrictive assumptions of known static cameras and planar movement constraints. It is easy to see that both constraints generally do not hold for many tracking tasks, such as simultaneous tracking using cameras on unmanned aerial vehicles, or tracking in synchronized dynamic videos from hand-held cameras. The knowledge about camera configurations will be unreliable or nonexistent. The movement on ground planes (GP) is an important special case, but even in standard tracking scenarios, e.g. pedestrians within cities, often too restrictive. We exploit connections between tracking and the Structure from Motion (SfM) domain, which can help making generic tracking scenarios solvable. Knowledge about camera localization and 3D structure can help with this task. Structure models built by today’s SfM methods generally show good quality, are easy to create and widely available. Because of the ubiquity of available 3D models we propose to merge established methods for multi-view tracking-by-detection and localization methods developed for SfM to enable more generic tracking tasks.

Our contribution is a method for tracking-by-detection for multiple dynamic cameras, with known but noisy 6-DoF camera motion. We propose a novel method for linking data across time and views incorporating the motion uncertainty of the cameras.
3.1. Introduction

We are the first to present a generalization of the strongly constrained 2D multi-view tracking scenario to unconstrained 3D scenarios. We do not make common restrictive assumptions, such as planar motion, known number of objects, constrained camera configurations, background modeling and explicit introduction of knowledge about the tracked objects movement characteristics in motion models.

We describe the tracking framework in section 3.2. In section 3.3 we explain unary and pairwise cost terms for the flow-network framework. In section 3.4 an experimental analysis is given. In section 3.5 we conclude and discuss future work. We use view and camera to refer to image data, orientation and position of one camera at a given time. We use frame to refer to image data from all views at a given time. A dynamic camera means a moving camera.

3.1.1 Related Work:

Multi-object tracking has been studied extensively. The most successful methods define tracking as a global optimization over a complete sequence, with given frame-wise object hypotheses. This is usually called tracking-by-detection.

Optimal single view data association for many detections per frame has been formulated by Zhang et al. [2008]; Jiang et al. [2007]. Shitrit et al. [2011] tracks without explicit detections on probabilistic occupancy grids. Andriyenko et al. [2012] extends this with iterations of discrete associations and continuous refinement.

Data is usually collected as independent detections from a given detector. This setup has been used by Zhang et al. [2008]; Morefield [1977]; Andriyenko et al. [2012]; Andriyenko & Schindler [2010]; Leibe et al. [2007b]; Pellegrini et al. [2010]; Hofmann et al. [2013]; Pirsiavash et al. [2011]. Background subtraction for collecting detection of object evidence has been used by Leal-Taixé et al. [2012]; Jiang et al. [2007]; Berclaz et al. [2011]; Fleuret et al. [2008]; Shitrit et al. [2011]; Andriyenko & Schindler [2010]; Possegger et al. [2013] work directly on a discretized grid or volume representing the space of all possible locations.

Several common constraints are used to facilitate tracking. Motion is often limited to a known GP, as in Berclaz et al. [2011]; Shitrit et al. [2011]; Andriyenko et al. [2012]; Fleuret et al. [2008]; Berclaz et al. [2011]; Leibe et al. [2007b]; Pellegrini et al. [2010]; Hofmann et al. [2013]; Khan & Shah [2006]; Possegger et al. [2013]. Leibe et al. [2007a]; Ess et al. [2009] assume planar motion, but infer the GP automatically. Shitrit et al. [2011] use global appearance constraints. Pellegrini et al. [2010] use social grouping cues. Leal-Taixé et al. [2012]; Berclaz et al. [2011]; Jiang et al. [2007]; Fleuret et al. [2008] need static background and cameras. Specific camera configurations are needed: head-level cameras by Fleuret et al. [2008], top-down views by Hofmann et al. [2013], visible feet locations by Khan & Shah [2006].

Solutions for association networks are found by Linear Programming (LP), as by Leal-Taixé et al. [2012]; Jiang et al. [2007]; Shitrit et al. [2011]; Andriyenko & Schindler [2010]; Morefield [1977]; Hofmann et al. [2013]. Dynamic Programming (DP) as by Berclaz et al. [2011]; Fleuret et al. [2008]; Zhang et al. [2008], Energy minimization in MRF/CRFs as by Andriyenko et al. [2012]; Pellegrini et al. [2010]. To reduce computational demands greedy approximations are available, as proposed by Pirsiavash et al. [2011].

Camera registration/localization has a variety of methods and distinct applications. In many SLAM methods, such as proposed by Davison et al. [2007], precise 6-DoF poses to 3D models are estimated, given location priors and feature matches. Lim et al. [2012] solve robot self-localization given large-scale 3D models. Sattler et al. [2011]; Irschara et al. [2009] propose methods for registering image collections for large scale models. Kroeger and Van Gool [2014] achieve high-precision registration for Videos to SfM models.

Our work is related to the proposed method of Hofmann et al. [2013], with important differences: We perform the data association in 3D, while Hofmann et al. [2013] uses a given GP. Hofmann et al. [2013] employ static entry/exit regions and hence the number of views to be used is known a priori. We incorporate camera and detector uncertainties into linking (3.16) and reconstruction probabilities (3.12), while Hofmann et al. [2013] only use these for error bounds while pruning reconstructions. Hofmann et al. [2013] need a form of enumeration for all possible sets of reconstructions, which grows factorially with number of cameras. Our work uses LP to optimally select the top reconstructions without
3.2 Tracking

We adopt a similar notation to Hofmann et al. [2013]. We extract person detections using the Deformable Part Model detector, proposed by Felzenszwalb et al. [2010]. Each 2D detection in camera $j$ at time $t$ is described by $x^{(j,t)}_i = (x, s)$, with position $x$ and scale $s$. All detections at time $t$ from all cameras are denoted $X^{(t)}$. One set of coupled detections from multiple cameras at time $t$ is denoted as reconstruction $R^{(t)}_k \in R^{(t)} \subset R$. From each camera at most one detection can be included in each $R^{(t)}_k$:

$$R^{(t)}_k \subset \{ X^{(t)} \mid \forall x^{(j,t)}_i, x^{(j',t)}_{i'} \in R^{(t)}_k : j \neq j' \lor i = i' \} . \quad (3.1)$$

Ideally one reconstruction $R^{(t)}_k$ corresponds to one real world object which caused the detections in each view. We denote the number of detections in one reconstruction as its cardinality. The set $R^{(t)}$ denotes all reconstructions at time $t$. A trajectory hypothesis is defined as

$$T_u^* = \arg\max_T P(T|R) = \arg\max_T P(R|T) \prod_{k \in T} P(T_k). \quad (3.3)$$

This assumes conditional independence of likelihoods given $T$ and independent object motion in (3.3). We further assume non-overlap of tracks, i.e. $T_k \cap T_j = \emptyset, \forall k \neq j$. The terms in (3.3) are defined as follows:

$$P(T_k) = P_{en}(R^{(t)}_{u_1}) \, P_{t_1}(R^{(t+1)}_{u_2} \mid R^{(t)}_{u_1}) \ldots \, P_{ex}(R^{(t+n)}_{u_n}), \quad (3.4)$$

$$P(R_i \mid T) = \begin{cases} P_{rec}, & \text{if } \exists T_k \in T, R_i \in T_k \\ 1 - P_{rec}, & \text{otherwise} \end{cases}. \quad (3.5)$$

This problem can be turned into a cost-flow network and solved exactly in an LP as proposed in Hofmann et al. [2013]; Andriyenko & Schindler [2010]; Shitz et al. [2011]; Jiang et al. [2007]; Leal-Taixé et al. [2012], or using DP, as by Zhang et al. [2008]; Fleuret
3.3. Modeling of Probabilities

The flow-network method links reconstructions across views and time given unary and binary costs. Similar to the work by Zhang et al. [2008]; Hofmann et al. [2013] we model these costs as probabilities $P_{\text{rec}}$ and $P_{\text{li}}$.

3.3.1 Extraction of Reconstruction Candidates

To start the tracking we need a set $\mathcal{R}^{(t)}$ of object hypotheses for all times $t$ containing the true solution. Because of this we focus on extracting the sets $\mathcal{R}^{(t)}$ large enough to ensure maximum recall. However, the combinatorial explosion prohibits inclusion of all feasible reconstructions in $\mathcal{R}^{(t)}$. We extract a set of $L$ top-ranking reconstructions for each cardinality $m$ as follows: We apply a distance function $D_1(x_{j,t}^i, x_{j',t'}^i)$ in 3D world coordinates between all pairs of two detections originating from different views, which we define in (3.9). Using this distance function we define the compactness of a reconstruction $\mathcal{R}_k^{(t)}$ as the sum of all pairwise detection distances $\sum_{x,x' \in \mathcal{R}_k^{(t)}, x \neq x'} D_1(x, x')$. This allows solving for the best (i.e. most compact) reconstruction of a given cardinality $m$ at time $t$ as an LP. We extract the $L$ top ranking solution using CPLEX for each cardinality and insert all extracted reconstructions at time $t$ into $\mathcal{R}^{(t)}$. 

**Figure 3.2:** Detection $x_{j,t}^i$ from camera $C_j^{(t)}$ is transformed into 3D with center $M_{x_{j,t}^i}$ and uncertainty $\Sigma_{x_{j,t}^i}$. We propagate both into view $j'$ at time $t'$ and compute $D_3(x_{\text{proj}}, x_{j',t'}^i)$ (3.11) as the Mahalanobis distance in the image (colored in red).
3.3. Modeling of Probabilities

Figure 3.3: A reconstruction $R_u^{(t)}$ consisting of 3 detections (red, blue, green), seen from 3 cameras at time $t$ is localized in 3D. Detection uncertainties in 2D become depth uncertainties in 3D, which increase with distance to the camera. Eq. (3.8) gives center and uncertainty of $R_u^{(t)}$ (black, dashed).

3.3.2 Localization of Detections and Reconstructions

Most previous multi-view multi-target tracking methods rely on the assumption of a given or automatically inferred ground plane (GP). However, GPs may not exist in some scenes, may not be visible (i.e. occluded in head-level cameras), or non-planar tracking is required. Localization of 2D object detections in 3D from head-level cameras using a GP assumption will be unreliable, even if a perfect GP is available, due to the small viewing angle. We side-step these problems by using the height of a detection as depth cue. Even if a bounding box is partly occluded, given the width and expected aspect ratio of a walking person, approximation of the height is possible. We use a distribution of heights with large uncertainty: mean $\lambda = 1.74 m$ and standard deviation $\sigma = \lambda / 4$.

We assume a calibrated camera $j$ at time $t$ is given with approximately known pose $C_j^{(t)} = (R, tr)$, where $tr$ and $R$ denote the position and orientation given from some arbitrary noisy image-based registration method, and known internal calibration matrix $K_j$. Let $\Sigma_{C_j^{(t)}}$ be the positional covariance of the camera. Given a detection $x_i^{(j,t)} = (x, s)$ with center $x$ and scale $s$ in view $j$ at time $t$ we compute a 3D location and uncertainty in world coordinates for the detection. On undistorted images, the detection direction $\hat{x}$ is given by $\hat{x} = K_j^{-1} \tilde{x}$, where $\tilde{x}$ is homogeneous $x$ and $\|\hat{x}\| = 1$.

We compute the detection distance from the camera center as $d = \lambda \cdot f_j / s$, where $s$ is the detection’s pixel height, and $f_j$ the focal length. This is a reasonable distance approximation if the object is close to the optical axis. Knowing $d$, we model the location in world coordinates as a normal distribution with mean $M_{x_i^{(j,t)}}$ and covariance $\Sigma_{x_i^{(j,t)}}$:

$$M_{x_i^{(j,t)}} = R^T \cdot [(\hat{x} \cdot d) - tr], \quad (3.6)$$

$$\Sigma_{x_i^{(j,t)}} = R^T \cdot \text{diag} (\lambda / 2, \lambda / 2, \lambda \cdot f_j / (4s) ) \cdot R + \Sigma_{C_j^{(t)}}. \quad (3.7)$$

The covariance is composed of two terms: Orthogonal to the optical axis, the detection 3D uncertainty is only dependent on the object scale $\lambda$. The detection height uncertainty in 2D becomes detection depth uncertainty in 3D as shown in figures 3.2 and 3.3. The
second term is the isotropic camera localization uncertainty $\Sigma_{C_j^t}$. A normal distribution over height in 2D will generally not translate into a normal distribution over depth in 3D. However, modeling (and thereby approximating) the true depth uncertainty as a normal distribution in 3D allows us to easily propagate the resulting uncertainty back into other 2D views and obtain a normal distribution again. See Fig. 3.2 and (3.10) for an explanation of the back-propagation.

To model the 3D reconstruction $R_k^t$ consisting of many detections, we fit a normally distributed uncertainty with mean $M_{R_k^t}$ and covariance $\Sigma_{R_k^t}$ such that:

$$x \sim N(M_{R_k^t}, \Sigma_{R_k^t}) \sim \frac{1}{Z} \sum_{x \in R_k^t} N(M_x, \Sigma_x).$$

(3.8)

Fig. 3.3 shows an example of 3 detections combined into one reconstruction. We use only one normal distribution for $R_k^t$ and not the mixture of all distribution from included detections to allow quick propagation of a single normal distribution to different views.

### 3.3.3 Detection and Reconstruction Distances

In order to model $P_{rec}$ and $P_{li}$, or the reconstruction unary and pairwise costs, respectively, we need a set of geometric and appearance-based comparison measures between detections from different views and reconstructions. Given the localization and uncertainties of detections and reconstructions in 3D we will define four geometric and one appearance-based distance measure to be used for $P_{rec}$ and $P_{li}$, in section 3.3.5 and 3.3.6.

- **D1**: Mahalanobis distance between detections in 3D:

  $$D_1(x, y) = D_{mah}(M_y, M_x, \Sigma_x) + D_{mah}(M_y, M_y, \Sigma_y).$$

  (3.9)

- **D2**: Similar to $D_1$, Mahalanobis distance between two reconstructions $R_k^t, R_k^{t'}$ using mean $M_{R_k^t}, M_{R_k^{t'}}$ and covariance $\Sigma_{R_k^t}, \Sigma_{R_k^{t'}}$.

- **D3**: Defined between a 2D detection $x_j^{(j,t)}$, which has been projected into 3D as $M_j^{(j,t)}$ with uncertainty $\Sigma_j^{(j,t)}$, and a 2D detection $x_j^{(j',t')}$.

  (See Fig. 3.2). We project $M_j^{(j,t)}$ into view $j'$ at time $t'$ with camera $C_j^{(t')} = (R, tr)$ and produce $x_{proj} = K_{j'}[R tr] \cdot M_j^{(j,t)}$. To propagate the uncertainty, the 3D covariance $\Sigma_j^{(j,t)}$ is projected into view $j'$ at time $t'$ using the projection’s Jacobian matrix $J_C(x)$ of $C_j^{(t')}$. We also include $C_j^{(t')}$’s localization uncertainty to handle camera errors:

  $$\Sigma_{proj} = J_C(x)^T \cdot (\Sigma_j^{(j,t)} + \Sigma_{C_j^{(t')}}) \cdot J_C(x).$$

  (3.10)
Using projected mean $x_{proj}$ and covariance $\Sigma_{proj}$, we define $D_3$ as the Mahalanobis distance
\[ D_3(x_{proj}, x_{proj}^{(j',t')}) = D_{mah}(x_{proj}^{(j',t')}, x_{proj}, \Sigma_{proj}) . \] (3.11)

- $D_4$: Defined between a reconstruction and a 2D detection equivalently to $D_3$ using the reconstruction’s mean and covariance for reprojection.
- $D_5$: Between two 2D detections we define $D_5$ as the Earth mover’s distance over RGB color histograms. We experimented with adding a HoG-based distance measure, but found it not to be helpful, due to large baselines between different views.

### 3.3.4 Entry and Exit Probability

Another often exploited advantage of static cameras is the possibility of defining explicit entry and exit zones in the images. Only in those areas are tracks allowed to start and end. This is easily modeled by setting $P_{en}$ and $P_{ex}$ to zero for all detections occurring outside these zones. For dynamic cameras this option does not exist. We estimate $P_{en} = P_{ex}$ for all reconstructions uniformly as the average missdetection probability of the detector over all sequences and cameras. These probabilities are transformed into costs for the flow-network as $W_{en} = W_{ex} = -\log(P_{ex}) = -\log(P_{en})$.

### 3.3.5 Reconstruction Probability

The probability $P_{rec}$ measures the pairwise similarity in appearance and spatial proximity of all detections in a reconstruction. For a reconstruction with detections of the same object, all detections should be similar in appearance and spatially close together. We set
\[ P_{rec} = P(d_{rep}) \cdot P(d_{col}) . \] (3.12)

$P(d_{rep})$ is computed as the probability that each detection reprojects well to all detections in other views. We average the distance $D_3$ for all pairs of included detections:
\[ d_{rep} = \frac{1}{|R_k(t)|^2} \sum_{x_i, x_j \in R_k(t)} D_3(x_i, x_j) . \] (3.13)

Average bounding box color dissimilarities are computed:
\[ d_{col} = \frac{1}{|R_k(t)|^2} \sum_{x_i, x_j \in R_k(t)} D_5(x_i, x_j) . \] (3.14)

The distributions of $d_{rep}$, $d_{col}$ are trained from ground truth and turned into matching probabilities $P(d_{rep}), P(d_{col})$. We transform the probability $P_{rec}$ into the node’s flow cost $W_{rec}$ and add the detector confidence for every detection.
3.4 Experiments

\[ W_{\text{rec}} = \log \left( \frac{1 - P_{\text{rec}}}{P_{\text{rec}}} \right) + \sum_{x_i \in \mathcal{R}_k^{(t)}} \log \left( \frac{\beta(x_i)}{1 - \beta(x_i)} \right) , \]  

(3.15)

where \( \beta(x_i) \) describes the detection’s false positive probability based on the detector score. Note that this cost neither penalizes nor prefers reconstructions with large or small cardinality. In contrast to the work of Hofmann et al. [2013] we cannot assume to know the number of cameras in which an object is visible, and prefer to keep this score invariant to the cardinality.

### 3.3.6 Linking Probability

For two given reconstruction candidates \( \mathcal{R}_k^{(t)}, \mathcal{R}_{k'}^{(t+1)} \) the transition probability \( P_{li} \) indicates the probability of \( \mathcal{R}_{k'}^{(t+1)} \) following \( \mathcal{R}_k^{(t)} \) in a track. We set

\[ P_{li} = P(d_{\text{pro}}) \cdot P(d_{\text{col}}) \cdot P(d_{\text{cen}}) . \]  

(3.16)

To establish these probabilities we will compute projection distances \( d_{\text{pro}} \), pairwise color dissimilarities \( d_{\text{col}} \), and the distance between the reprojections’ center points \( d_{\text{cen}} \). Let \( m, n \) be the cardinalities of \( \mathcal{R}_k^{(t)}, \mathcal{R}_{k'}^{(t+1)} \), respectively. We calculate how close the reconstruction \( \mathcal{R}_k^{(t)} \)'s 3D center projects to all detections \( x_j \in \mathcal{R}_{k'}^{(t+1)} \) and vice versa:

\[ d_{\text{pro}} = \frac{1}{n} \sum_{j=1}^{n} D_4(M_{\mathcal{R}_k^{(t)}}, x_j) + \frac{1}{m} \sum_{i=1}^{m} D_4(M_{\mathcal{R}_{k'}^{(t+1)}}, x_i) . \]  

(3.17)

\( d_{\text{col}} \) is computed as the average color dissimilarity between all detection pairs \( x_i \in \mathcal{R}_k^{(t)}, x_j \in \mathcal{R}_{k'}^{(t+1)} \), similar to (3.14). We compute the distance between reconstructions centers \( d_{\text{cen}} = D_2(M_{\mathcal{R}_k^{(t)}}, M_{\mathcal{R}_{k'}^{(t+1)}}) \). This implicitly assumes maximum probability for stationary objects, which is generally not correct. However, this is an acceptable approximation because the frame-to-frame motion (1.5 meters/sec avg. walking speed) is 1-2 magnitudes smaller than the camera localization error. The distributions of \( d_{\text{pro}}, d_{\text{col}} \) and \( d_{\text{cen}} \) are trained from ground truth and turned into matching probabilities for (3.16). The probability is transformed into a cost as \( W_{li} = -\log(P_{li}) \).

### 3.4 Experiments

**Dataset for Evaluation:**

We used three sets of videos for tracking, each set is called a *sequence*. Each sequence consists of 301 frames, seen from 7 synchronized cameras. Two of the 7 cameras are
3.4. Experiments

**Figure 3.4:** Tracking result in 4 dynamic cameras of seq. 3. Static cameras 1+2 show current (red) and past (blue) estimated camera positions. Anonymized for publication purposes.

**Table 3.1:** Tracking and video registration result for three sequences. Explanation in section 3.4.

<table>
<thead>
<tr>
<th></th>
<th>MOTA(%)</th>
<th>MOTP io(%)</th>
<th>MOTP 3d(m)</th>
<th>MT[%]</th>
<th>ML[%]</th>
<th>PT[%]</th>
<th>ids</th>
<th>Frag</th>
<th>C.Err1 (m)</th>
<th>C.Err2 (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.1</td>
<td>1.01</td>
<td>0.88 / 0.93</td>
<td>2.65 / 1.36</td>
<td>7.14</td>
<td>28.57</td>
<td>64.29</td>
<td>21</td>
<td>91</td>
<td>0.73 / 0.65</td>
<td>2.16 / 1.79</td>
</tr>
<tr>
<td>S.2</td>
<td>0.64</td>
<td>0.91 / 0.95</td>
<td>1.25 / 0.96</td>
<td>9.09</td>
<td>54.55</td>
<td>36.36</td>
<td>3</td>
<td>104</td>
<td>0.78 / 0.67</td>
<td>3.55 / 1.71</td>
</tr>
<tr>
<td>S.3</td>
<td>1.44</td>
<td>0.89 / 0.94</td>
<td>1.32 / 0.88</td>
<td>0</td>
<td>54.55</td>
<td>45.45</td>
<td>4</td>
<td>119</td>
<td>0.62 / 0.48</td>
<td>2.82 / 1.21</td>
</tr>
</tbody>
</table>

static, show the scene from a top-down view, are manually registered to the SfM model, and only used to create ground truth 3D locations for pedestrians and the 5 videographers. The remaining 5 cameras are dynamic, always seen from the static cameras, and used for tracking. In Fig. 3.4 and 3.5, 4 out of 5 dynamic and 2 static cameras are shown. Due to space constraints the 5th camera was omitted. Pedestrian tracks are annotated in all 7 views and identities across views are given. A manually determined GP is calculated for visualization and evaluation, but is not used in the algorithm. We manually collect 3D locations on this GP for all person tracks, including the videographers. The scene is not perfectly planar as can be seen in Fig. 3.4 and 3.5. Nevertheless, we expect the 3D localization error for all tracks to be below 30 cm.

**Video Registration:** For each sequence, we independently register all 5 dynamic videos to the SfM model using the PnP method developed by Moreno-Noguer et al. [2007]. The SfM model covering the scene was manually created using high-resolution DSLR images. Because we have ground truth 3D position on a GP for each videographer, we can evaluate the localization of each camera. The camera registration results in a mean and median error of 0.71 and 0.62 meters, respectively. Column C.Err1 in table 3.1 show this positional error for all sequences separately. Column C.Err2 shows the mean and median error between GT camera location and the estimated position of the corresponding videographer.
Figure 3.5: Tracking result in 4 dynamic cameras of seq. 1. Static cameras 1+2 show current (red) and past (blue) estimated camera positions. Anonymized for publication purposes.

Tracking: We use an independently trained Deformable Part model detector, proposed by Felzenszwalb et al. [2010], for person detection. Given registered videos we start the tracking by computing 3D positions and uncertainties for all detections, in all views and frames. We only have tracking annotation for persons which are seen from the two static cameras in at least one frame. We cannot evaluate the tracking for persons outside of this area. Therefore, we manually eliminate all detections, based on their 3D projection, which cannot be seen from the static cameras. This avoids creating tracks outside of the valid tracking region, which would result in a misleading FP number. Following section 3.3.1 we extract reconstruction candidates for all frames. We set $L = 100$ as sufficiently large set of hypotheses. We create the flow graph exactly as proposed by Hofmann et al. [2013], using our defined probabilities, and solve it using CPLEX. As evaluation criterion we use the metric presented by Li et al. [2009]: Identity switches (ids), Number of interruptions of a ground truth track (Frag), Mostly tracked (MT), Mostly lost (ML), Partly tracked (PT). Additionally, we use the CLEAR metric proposed by Bernardin et al. [2008]; Kasturi et al. [2009]: Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP). The hit/miss decision is computed in image space. A hit has an intersection over union score of over 0.4. We compute MOTP in image space as intersection over union (MOTP io, in percent, mean/median) and as the distance between estimated and true 3D track location (MOTP 3D, in meters, mean/median).

Discussion: We achieve consistently good precision in image space (MOTP io). In addition, we achieve good precision in 3D localization (MOTP 3d) considering that detection height was used as initial depth estimate and no GP was available. Additionally, we only have low confusion, as indicated by low identity switches (ids). The low ids score is partly explained by frequent interruptions of the ground truth track, resulting in high fragmentation (frag). Tracks break, rather than incorrectly switching targets. Often detections are incorrectly left out from a track, resulting in many false negatives and a low MOTA. This
is due to the fact that the scene setup does not allow to specify a known cardinality for an object detection. Furthermore, several difficult tracks (severely occluded persons, far away persons) are missed altogether, resulting in a low MT, and high ML. Fig. 3.4 and 3.5 show the tracking result from 4 of 5 dynamic cameras in two sequences. Identical colors and numbers in different views indicates the same track. The two static cameras show current estimated camera positions (red) and past videographer position (blue) on the GP. Typical failure cases include track 143 and 153 in Fig. 3.4: The object is correctly tracked but the tracks are not merged over views. Track 114 and 131 in Fig. 3.5 show a confusions between tracked objects within each track. In Fig. 3.5 some far-away persons are missed, due to unreliable detections.

### 3.5 Conclusion

With this work we contribute to the solution of multi-target tracking in multiple moving, approximately localized cameras. We extend an established tracking-by-detection framework using flow-networks, and show that even without many common constraints, such as availability of GP, static cameras or background subtraction, a satisfying tracking solution can be found nevertheless. This work presents the first generalization of multi-view multi-target tracking of objects on a GP to moving objects and cameras in 3D. Our tracking method is not limited to person detections. All objects with approximately constant size can be tracked given an appropriate detector. We consider several directions for future work. The MOTA scores we obtain are significantly lower than scores usually obtained on established datasets with static and known cameras, such as PETS2009. Primarily, this is due to the additional unknown and varying camera locations and the lack of available track entry and exit regions. We aim to improve this by using the 3D model visibility information to infer likely entry and exit regions. Another reason is the strong influence of incorrect and noisy camera poses. We aim to improve this by including pairwise essential matrix constraints and estimation of the relative poses directly within the tracking framework. We also explore possibilities of creating reconstruction candidates build from short 2D object tracklets, instead of single-frame detections to reduce computational demands.
Joint Vanishing Point Detection and Tracking

We present a novel vanishing point (VP) detection and tracking algorithm for calibrated monocular image sequences. Previous VP detection and tracking methods usually assume known camera poses for all frames or detect and track separately. We advance the state-of-the-art by combining VP extraction on a Gaussian sphere with recent advances in multi-target tracking on probabilistic occupancy fields. The solution is obtained by solving a Linear Program (LP). This enables the joint detection and tracking of multiple VPs over sequences. Unlike existing works we do not need known camera poses, and at the same time avoid detecting and tracking in separate steps. We also propose an extension to enforce VP orthogonality. We augment an existing video dataset consisting of 48 monocular videos with multiple annotated VPs in 14448 frames for evaluation. Although the method is designed for unknown camera poses, it is also helpful in scenarios with known poses, since a multi-frame approach in VP detection helps to regularize in frames with weak VP line support.

4.1 Introduction

A vanishing point (VP) is the point of convergence of a set of parallel lines in the imaged scene under a projective transformation. Man-made structures often consist of geometric primitives, such as multiple sets of parallel or orthogonal planes and lines in the scene. Because of this, the detection of projected VPs in images provides strong cues for the extraction of knowledge about the unknown 3D world structure. VPs can often be further constrained to mutual orthogonality, due to the preference of right angles in man-made structures. Detected VPs have been used as a low-level input to many higher-level computer vision tasks, such as 3D reconstruction (Crandall et al. [2012]; Kim & Manduchi [2014]), autonomous navigation (Moghadam & Dong [2012]), camera calibration (Grammatikopoulos et al. [2007]; Wildenauer & Hanbury [2012]) and pose estimation (Košecká & Zhang [2002]; Micusik & Wildenauer [2013]).
4.1. Introduction

Many applications, which take video sequences or unordered image sets as input, require VP estimates in every frame and VP identities across views or frames. Usually, when this is needed, the camera pose is assumed to be known for every frame, as in the method proposed by Antone & Teller [2000]; Hornáček & Maierhofer [2011], thereby rendering the VP association across images simple, or separate steps for VP detection and tracking (particle filters by Moghadam & Dong [2012]; Rasmussen [2008], greedy assignment by Elloumi et al. [2012]) are used. Since pose knowledge can only be obtained through expensive odometry or external motion measurements, it will often not be available. Separate VP detection and tracking often results in missed detections or loss and re-initialization of VP tracks due to weak line support in some frames. Even in the case of known poses, joint detection over multiple frames benefits from integrating image evidence of many frames. Joint reasoning over sequences is particularly useful if long-term VP identities are required, and re-initializations are expensive.

To the best of our knowledge, no method exists that jointly extracts multiple VPs in all frames of a video with unknown camera motion. Our contributions are twofold:

1. Method. We propose the first algorithm for the joint VP extraction over all images of a sequence with unknown camera poses. We borrow from recent advances in multi-target tracking, published by Berclaz et al. [2011]; Zhang et al. [2008], and model the problem as a variant of a network-flow tracking problem. We compute line segments in each frame and discretize the set of possible VPs on a probabilistic spherical occupancy grid. Line-VP association probabilities and VP transition probabilities are converted into an acyclic graph for joint VP extraction and tracking. VPs are extracted by Linear Programming (LP).

2. Dataset. As the field lacks a dataset for the evaluation of VP extraction in videos, we augmented the Street-View video dataset published by Kroeger and Van Gool [2014] with

**Figure 4.1:** Tracked vanishing directions are shown together with associated imaged line segments in three frames of a sequence. This and the subsequent figures are best viewed in color.
annotated VPs, which will be publicly available. We evaluate our approach on this dataset using established multi-object tracking metrics proposed by Bernardin et al. [2008]; Li et al. [2009] for unknown camera poses. We chose this dataset because camera poses are available for all frames, which enables one additional experiment: We evaluate the improvement of our algorithm when camera pose information is incorporated. Since our method also works for single frames and orthogonal VPs, we compare to a recent method by Tardif [2009] for VP detection in Manhattan Scenes on the York Urban Dataset, presented by Denis et al. [2008].

This chapter is organized as follows: section 4.2 introduces our VP parameterization. section 4.3 describes the method, with LP formulation in section 4.3.1, and score modeling in section 4.3.2. We evaluate in section 4.4 and conclude in section 4.5 with a discussion of future work.

### 4.1.1 Related Work

VP extraction has been studied extensively in Computer Vision. The most relevant recent works can be categorized according to several algorithmic design choices:

**Input:** While some approaches start directly from continuous image gradients or texture (Moghadam & Dong [2012]; Rasmussen [2008]; Schindler & Dellaert [2004]) and thresholded edges images (Tretyak et al. [2012]), most works rely on short line segments (Antunes & Barreto [2013]; Barnard [1982]; Grammatikopoulos et al. [2007]; Hornáček & Maierhofer [2011]; Magee & Aggarwal [1984]; Rother [2002]; Tardif [2009]; Wilde-nauer & Hanbury [2012]), or full lines (Caprile & Torre [1990]; Elloumi et al. [2012]). If the 3D geometry is known, surface normals can be used, as presented by Straub et al. [2014].

**Accumulator Space:** Intersections of imaged lines can be computed in the original (unbounded) image space (Antunes & Barreto [2013]; Elloumi et al. [2012]; Rother [2002]; Schindler & Dellaert [2004]; Tardif [2009]; Xu et al. [2013]) or on a (bounded) Gaussian unit sphere, first proposed by Barnard [1982] and used by Antone & Teller [2000]; Bazin & Pollefeys [2012]; Grammatikopoulos et al. [2007]; Hornáček & Maierhofer [2011]; Košcká & Zhang [2002]; Lutton et al. [1994]; Magee & Aggarwal [1984]; Micusik & Wildenauer [2013]; Quan & Mohr [1989]; Straub et al. [2014]. We use the latter approach, explained in section 4.2. It allows for easy discretization, as explained by Barnard [1982]; Lutton et al. [1994]; Magee & Aggarwal [1984]. Lezama et al. [2014] propose a line parameterization in parallel coordinates to extract VPs.

**Line-VP Consistency and VP Refinement:** Consistency between an estimated VP and image lines can be computed directly by measuring line endpoint distances in the image (Antunes & Barreto [2013]; Bazin & Pollefeys [2012]; Hornáček & Maierhofer [2011]; Lezama et al. [2014]; Tardif [2009]), angular differences in the image (Denis et al. [2008];
Rother [2002]), with explicit probabilistic modeling of the line endpoint errors (Xu et al. [2013]), or with angles between normals of interpretation planes in the Gaussian sphere, used by us and by Lutton et al. [1994]; Magee & Aggarwal [1984]; Quan & Mohr [1989]. VP computation or refinement with given associated lines is done via Hough voting and non-maximum suppression (Lutton et al. [1994]; Magee & Aggarwal [1984]; Moghadam & Dong [2012]; Tuytelaars et al. [1997]; Quan & Mohr [1989]), solving a quadratic program (Antunes & Barreto [2013]), implicitly in an EM setting (Antone & Teller [2000]; Schindler & Dellaert [2004]; Tardif [2009]; Xu et al. [2013]), or by linear least-squares, as in this work and by Košcká & Zhang [2002].

Solution: Several different methods exist for combining input, accumulator space and line-VP consistency measures into a final extraction solution. If no discretization of the accumulator space is attempted, solutions are found with efficient search (Denis et al. [2008]; Rother [2002]), direct clustering (Lezama et al. [2014]; Tardif [2009]), multi-line RANSAC (Bazin & Pollefeys [2012]; Wildenauer & Hanbury [2012]), EM procedures (Antone & Teller [2000]; Hornáček & Maierhofer [2011]; Košcká & Zhang [2002]; Schindler & Dellaert [2004]; Xu et al. [2013]), or MCMC inference (Straub et al. [2014]). With a discretized accumulator space solutions are found by voting schemes (Lutton et al. [1994]; Magee & Aggarwal [1984]; Moghadam & Dong [2012]; Quan & Mohr [1989]) or inference in graphical models, as proposed by Antunes & Barreto [2013]; Tretyak et al. [2012].

Camera Calibration and VP Orthogonality: Some VP extraction methods assume known internal camera calibration (Elloumi et al. [2012]; Hornáček & Maierhofer [2011]; Lutton et al. [1994]; Moghadam & Dong [2012]; Quan & Mohr [1989]; Rasmussen [2008]). Others do not need calibration (Antunes & Barreto [2013]; Košcká & Zhang [2002]; Magee & Aggarwal [1984]; Rother [2002]; Xu et al. [2013]). Internal parameters can be estimated from extracted VPs (Caprile & Torre [1990]; Grammatikopoulos et al. [2007]; Wildenauer & Hanbury [2012]). VPs have also been used for estimation of external camera parameters: orientation of camera to scene (Bazin et al. [2012]; Košcká & Zhang [2002]), 3D shape to camera (Barnard [1982]), and as additional constraints for full camera poses (Micusik & Wildenauer [2013]). Often, further scene-dependent VP constraints are included: mutual VP orthogonality (Manhattan World) (Denis et al. [2008]; Elloumi et al. [2012]; Hornáček & Maierhofer [2011]; Wildenauer & Hanbury [2012]), sets of mutually orthogonal VPs (Straub et al. [2014]; Kroeger et al. [2015a]), with a shared vertical VP (Atlanta World) (Antunes & Barreto [2013]; Schindler & Dellaert [2004]).

4.2 VP Representation

A 2D VP is the intersection of (the unbounded continuation of) two 2D line segments, imaged from two 3D scene-parallel lines. In Fig. 4.2 imaged line segments $l_1, l_2$ of scene parallel lines $\hat{l}_1, \hat{l}_2$ form a VP $V_{2D}$ on the image plane $\pi$. The 2D VP may lie inside or outside the image frustum of $\pi$, or at infinity, in cases when imaged lines remain parallel.

An alternative parameterization, proposed by Barnard [1982], models VP locations on the unit sphere. A point $\hat{x}$ in homogeneous image coordinates is normalized by $x = K^{-1}\hat{x}$, with $K$ the camera calibration matrix. A plane $P$ is spanned by the center of projection at $[0, 0, 0]^T$, and the endpoints $x_1, x_2$ in normalized homogeneous image coordinates of an imaged line segment $l$. The plane is computed as $P = x_1 \times x_2/(\|x_1\|\|x_2\|)$, and is called interpretation plane. For line segments $l_1, l_2$, interpretation planes $P_1, P_2$ are shown as intersection circles of the planes and the unit sphere in Fig. 4.2. The VP is given as their intersection on the sphere: $V_{3D} = \pm P_1 \times P_2$. In the following we use vanishing direction
4.2.1 VP Discretization

In order to use the proposed parameterization for tracking we need to discretize the solution space of possible VPs. Fig. 4.3 illustrates the chosen discretization: a simple triangular tessellation of the sphere by iterative subdivisions of the faces of an icosahedron. Three frames of a sequence of a rotating cube are shown (top row). For visualization purposes four line segments belonging to one horizontal VP are drawn in red. With known internal camera calibration $K$, four interpretation planes are computed and plotted together with the discretized unit sphere (bottom row). The red color strength illustrates the likelihood of a VP in each cell of the spherical grid. This is determined by summing up all line-VP consistency scores as described later in Eq. (4.14). Since the cube is rotating around a
4.3 Proposed Algorithm

The proposed VP representation and discretization can now be used in a directed acyclic graph, similar to flow networks in multi-target tracking-by-detection. Such graphs for probabilistic multi-target tracking were first proposed by Zhang et al. [2008], where object detections are linked across time through pairwise object transition arcs. Transition probabilities indicate which detections at different times capture the same object. This technique has been very successful in multi-target tracking, since it allows association of detection evidence jointly in time and space. Berclaz et al. [2011] extend this by lifting the need for object hypotheses, and operates on a discretized ground plane with occupancy vertical axis, the horizontal VP rotates accordingly, which can be seen by the change in VP likelihoods on the spherical grid. Note that the spherical grid itself does not rotate, only the planes and their intersections on the sphere follow the cube’s rotation. While in this example correct VP-line associations have been selected manually, our proposed method will have to extract multiple VP tracks and VP-line associations jointly.

Figure 4.4: Illustration why aggregated occupancy probabilities are insufficient. Same example as in Fig. 4.1. Top: contribution to VP bins from all interpretation planes regardless of VP association. Color strength indicates increasing VP likelihood. Note the wide (but incorrect) peak within the image frustum outlined in blue. Bottom: after optimization we know the correct VPs, and color all plane contributions following their association to the red, green and blue VP. The misleading peak in the image frustum (top row) originates from overlapping interpretation planes of the true VPs.
probabilities in each grid cell. Instead of object transitions, grid transitions are encoded in the graph. The task is solved by $k$-shortest path search through the graph with Dynamic or Linear Programming.

In order to avoid a separate VP detection step we follow the second approach, and consider the discretized sphere as an occupancy sphere, where probabilistic evidence in each bin indicates the likelihood for a VP. However, in contrast to the work of Berclaz et al. [2011], we cannot simply aggregate all occupancy evidence for a VP bin, since one interpretation plane could give its evidence to several, mutually contradictory bins along a great circle on the sphere. In general, there will usually be a greater number of line intersections within the image frustum than outside of it. Because of this VPs can be hallucinated in the image frustum if each line is allowed to vote for all bins along the intersection of interpretation plane and sphere. This is illustrated in Fig. 4.4 (top row), where evidence (Eq. 4.14) from all interpretation planes for multiple VPs is aggregated, similar to Fig. 4.3 for one VP. Line segments to horizontal (red, blue) and vertical (green) VPs join in a wide, but incorrect, peak in the occupancy probability within the image frustum. Weaker, but sharper and temporally more stable peaks, corresponding to true VPs, are often lost in this noise. Fig. 4.4 (bottom row) visualizes this: we colored the interpretation plane contributions according to their (correct) VP association. It can be seen that the peak within the image frustum for aggregated occupancy probabilities does not correspond to any real VP.

Therefore, we do not want to aggregate occupancy probabilities, but need to enforce that each line segment and interpretation plane is assigned to maximally one VP bin. To achieve this, we keep the association between VP bins and interpretation planes as free variable in the joint VP extraction and tracking framework. This approach is related to the Uncapacitated Facility Location problem for VP extraction, as described by Antunes & Barreto [2013]. However, Antunes & Barreto [2013] focus on single-frame extraction of multiple sets of orthogonal VPs, while we apply this idea to joint detection and tracking over time, and add optional constraints to enforce orthogonality in VP locations.

### 4.3.1 Linear Program Formulation

**Overview:** Solving this tasks requires deciding, firstly, which VP bins are active, secondly, which line segments are uniquely assigned to each VP, and, thirdly, where active VP bins are continued over time. Since these decisions are interdependent, we map this problem into a graph, a variant of flow-cost networks (Zhang et al. [2008]), as visualized in Fig. 4.3, to enable a joint solution. In this graph arcs between nodes define binary decision variables, which are activated or deactivated depending on whether associations between VP bins over time and VPs to line segments are made or not. The graph is structured such that each VP track starts at the starting node, traverses smoothly connected VP bins over time, collects uniquely assigned line evidence in each traversed frame, and
ends at the terminal node. Tracks are constrained not to overlap, and to enclose a constant angle to each other active VP, with optional perpendicularity.

**LP Formulation:** For an image sequence of $T$ frames, with $I_t$ line segments at time $t$, we denote the interpretation plane for line segment $i$ at time $t$ as $P_{i,t}$. $V_{j,t}$ denotes the unit normal of VP bin $j$ out of $J$ possible bins at time $t$.

The graph is constructed as follows (See Fig. 4.3): for each bin $V_{j,t}$ we have an arc from the starting node $S$ and an arc to the terminal node $T$ with associated cost $C_s(j,t)$, $C_e(j,t)$, respectively. For each bin $V_{j,t}$ we have transition arcs to all bins $V_{j',t+1}$, $\forall j' \in J$ in the following frame, with associated cost $C_t(j,t,j')$. For each bin $V_{j,t}$ we define a unary cost $C_u(j,t)$. For each bin $V_{j,t}$ we have line association arcs to every interpretation plane $P_{i,t}$, with associated score $S_l(j,i,t)$. The solution to our problem is then given by the set of shortest paths (i.e. with lowest aggregate scores plus costs) from nodes $S$ to $T$.

Inspired by the work of Zhang et al. [2008], we model the problem such that only image evidence (i.e. line segments) encourages active VP tracks with scores $S_l$ while all other terms $C_s$, $C_t$, $C_u$, $C_e$ inhibit them as costs.

We sum all scores and costs for the objective function $f$:

$$f(\lambda) = \sum_{t}^{T} \sum_{j}^{J} \left[ \left( \sum_{i}^{I_t} \lambda_l(j,i,t) \cdot S_l(j,i,t) + \sum_{j'}^{J} \lambda_t(j,t,j') \cdot C_t(j,t,j') \right) + \lambda_b(j,t) \cdot C_b(j,t) + \lambda_s(j,t) \cdot C_s(j,t) + \lambda_e(j,t) \cdot C_e(j,t) \right] ,$$

where $\lambda = [\lambda_l, \lambda_t, \lambda_b, \lambda_s, \lambda_e]$ are binary variables, indicating active ([l]ine, [t]ransition, [b]in activation, [s]tart, [e]xit) arcs. Since a VP bin can only be active with an active outgoing transition or exit arc we replace $\lambda_b(j,t) = \lambda_e(j,t) + \sum_{j'} \lambda_t(j,t,j')$ in the optimization.

The LP solution is given by:

$$\lambda^* = \arg\min_{\lambda} f(\lambda) ,$$

subject to constraints enforcing the graph structure:

**C1. Flow conservation.** Every VP bin can maximally be traversed by one track:

$$\forall j, t : \lambda_s(j,t) + \sum_{j' \in J} \lambda_t(j',t-1,j) = \lambda_e(j,t) + \sum_{j' \in J} \lambda_t(j,t,j') \leq 1 .$$
4.3. Proposed Algorithm

**C2. Line-VP association.** Only active VP bins can support line-VP arcs. A line can at most be linked to one VP:

\[
\forall j, t : \quad \lambda_b(j, t) \cdot I_t - \sum_{i}^{\lambda_l} \lambda_l(j, i, t) \geq 0 , \\
\forall i, t : \quad \sum_{j' \in \mathcal{J}} \lambda_l(j, i, t) \leq 1 .
\]  

(4.5)

(4.6)

Additionally, we discovered that it is helpful to directly constrain which VP bins can be active together.

**C3. Non-Maximum Suppression.** For an active VP bin, we suppress other active VP bins in neighborhood \( \mathcal{N}_s \):

\[
\forall j, t : \quad \lambda_b(j, t) \cdot |\mathcal{N}_s| + \sum_{j' \in \mathcal{N}_s(j)} \lambda_b(j', t) \leq |\mathcal{N}_s| .
\]

(4.7)

**C4. Angle preservation.** For two active VP bins \( \mathcal{V}_{j,t}, \mathcal{V}_{j',t} \) linked to active \( \mathcal{V}_{k,t+1}, \mathcal{V}_{k',t+1} \), respectively, we require constancy of the enclosed angle. This follows from the fact that vanishing directions are constant over time.

\[
|\arccos(\langle \mathcal{V}_{j,t}, \mathcal{V}_{j',t} \rangle) - |\arccos(\langle \mathcal{V}_{k,t+1}, \mathcal{V}_{k',t+1} \rangle) \| < \epsilon_a
\]

(4.8)

**C5. Orthogonality (optional).** Similar to the work of Elloumi et al. [2012]; Hornáček & Maierhofer [2011]; Wildenauer & Hanbury [2012] we can optionally enforce that all tracked VPs have to be mutually orthogonal at all times:

\[
\forall j, j', t | j \neq j' : \lambda_b(j, t) \cdot \lambda_b(j', t) \| (\mathcal{V}_{j,t}, \mathcal{V}_{j',t}) \| < \epsilon_a .
\]

(4.9)

C1 and C2 are essential to enforce the graph structure as visualized in Fig. 4.3. We found experimentally that C3 is only needed for strong noise in line endpoints, and C4 for horizontal VPs near infinity. If a Manhattan world is assumed, C5 can be used. Because of a lack of an appropriate dataset, we only evaluate inclusion of C5 for single frames.

Integer Linear Programming is NP-hard in general. However, using branch-and-bound, implemented in many solvers such as CPLEX\(^1\), our problem is optimally\(^2\) solved.

The solution \( \lambda^* \) gives a set of \( n \) VP tracks \( \mathcal{V} = \{ \mathcal{V}_1, \ldots, \mathcal{V}_n \} \), where a VP track \( \mathcal{V}_i = \{ V_{k,t_a}, \ldots, V_{k',t_b} \} \) consists of list of VP bins from time \( t_a \) to \( t_b \). Furthermore, for each active VP bin \( V_{k,t} \) in track \( \mathcal{V}_i \) at time \( t \) we obtain all interpretation planes assigned to that VP at that time.

\(^1\)www.ibm.com/software/commerce/optimization/cplex-optimizer/

\(^2\)In practice we terminate the search early for a small optimality gap for efficiency reasons with no measurable performance loss.
4.3. Proposed Algorithm

4.3.2 Score and Cost Modeling

We derive the scores and costs probabilistically and convert them into values for the objective function $f$ in (4.2).

We set all unary, start and exit probabilities for all bins $j$ at times $t$ uniformly to $P_u = P_s = P_e = 1/J$. This leads to decreased sensitivity of the method for finer discretization (higher $J$), and offsets the effect of stronger influence of line segment noise. It is easy to add further domain knowledge at this point: non-uniform $P_u$ over all spherical bins can encode e.g. higher probabilities for horizontal VPs, if the gravity direction is approximately known. Non-uniform $P_s, P_e$ over time can give a bias for known start and end times of VP tracks, e.g.: from initial labeling.

Between bins $j$ and $j'$ we assign a transition probability based on the enclosed angle $\alpha = |\arccos(\langle V_{j,t}, V_{j',t+1} \rangle)|$:

$$P_t(j, j') = (1 + e^{\gamma_1(\alpha - \gamma_2)})^{-1}. \quad (4.10)$$

This sigmoid function yields a smooth fall-off at $\alpha = \gamma_2$, with decay rate controlled by $\gamma_1$. Since bin locations are fixed, $P_t(j, j')$ is independent of time $t$.

Out of the many line-VP consistency measures used in related works, we select a simple angular distance between plane normal and VP bin. The plane $P_{i,t}$ exactly intersects the sphere on a greater circle through $V_{j,t}$ iff the angle $\beta = |\arcsin(\langle V_{j,t}, P_{i,t} \rangle)| = 0$, i.e. iff plane normal and VP are exactly orthogonal. We set the linking probability:

$$P_l(j, i, t) = (1 + e^{\gamma_3(\beta - \gamma_4)})^{-1}. \quad (4.11)$$

This sigmoid function yields a smooth fall-off at $\beta = \gamma_4$, with decay rate controlled by $\gamma_3$.

Probabilities $P_u, P_s, P_e, P_t, P_l$ are converted into costs and scores as proposed by Zhang et al. [2008]. Let costs for bins $j, j'$ at time $t$ be

$$C_u = C_s = C_e = -\log P_u = -\log P_s = -\log P_e,$$

$$C_t(j, t, j') = -\log P_t(j, j'). \quad (4.12)$$

Scores for bin $j$ at time $t$ and line segment $i$ are given as:

$$S_l(j, i, t) = \log \frac{1 - P_l(j, i, t)}{P_l(j, i, t)}. \quad (4.14)$$

Scores can be negative and encourage active tracks, while costs are strictly positive and penalize active tracks.
4.4 Experiments

For all experiments we use the same implementation and parameters, detailed in section 4.4.1. We will evaluate our approach for three scenarios: joint VP detection and tracking on our new dataset in section 4.4.3, VP detection and tracking when camera poses are known section 4.4.4, and single-frame orthogonal VP detection on the York Urban Dataset (YUD) (Denis et al. [2008]) in section 4.4.6.

4.4.1 Implementation Details

The neighborhood $N_s$ for a VP bin $j$ is selected such that no other VP is expected within $N_s$. For our experiments we conservatively assume this to be the case for $\sigma = 5$ degrees: $N_s(j) = \{V_{j,t} : |\arccos(V_{j',t}, V_{j,t})| < \sigma, j' \neq j \}$.

We set $\epsilon_a = \cos(1)$. Values $\gamma_1 = 10, \gamma_3 = 5, \gamma_2 = \gamma_4 = 2$ are set empirically after visual inspection of the dataset. Optimal $\gamma_{1,2}$ depend on the rate of change of orientation, $\gamma_{3,4}$ on line endpoint noise. While Fig. 4.3, Fig. 4.4 show a coarse 80-bin discretization, we chose a fine discretization of 5120 spherical bins for evaluation. This discretization is an empirically chosen trade-off between high VP accuracy (finer discretization) and short run-times (coarser discretization). Since solving large LPs may become computationally expensive, we propose five LP pruning strategies:

1. **Grouping of Line Segments.** We start by extracting LSD line segments, as propose by von Gioi et al. [2012], but reduce the number of lines, using the Hough transform, into maximally $I_t = 100$ lines.

2. **Limiting Line-VP Association.** We threshold and cut all line-VP bin associations with $P_l(j, i, t) < 0.5$.

3. **Removal of VP Bins.** We threshold and cut from the LP all VP bins $j$ at time $t$ for which $\sum_i^{I_t} S_l(j, i, t) > 0$. Furthermore, since all VP evidence is antipodal on the Gaussian sphere, we only use VP bins on one hemisphere.

4. **Pruning of Transitions.** We cut all transition arcs between $V_{j,t}$ and $V_{j',t+1}$, if $P_t(j, j') < 0.2$.

5. **Batch Processing.** We solve the LP in batches of maximally 30 frames using branch-and-bound in CPLEX. After the solution $\lambda^*$ is found, we refine each VP in each frame via least-squares optimization using Eq. (4.1). Batches are greedily merged, as explained for our baselines in section 4.4.3.

Experimentally we found that these pruning steps do not change the solution of the LP, but are helpful (esp. steps 1 and 3) to keep the computation to a few seconds per frame.
4.4.2 Annotation of the VP detection and tracking dataset

**Figure 4.5:** Left: SfM point cloud, with van path over 301 frames and selected camera poses. Middle: Rigid camera set-up on the van. The red/blue coloring is irrelevant for our purposes. Right: Example images from 4 out of 12 cameras at the same time point.

**Dataset:** As the field lacks a dataset for the evaluation of joint VP detection and tracking we augmented a dataset for video registration to SfM models ([Kroeger and Van Gool [2014]](https://example.com)) with our own VP annotations. Multiple vanishing directions and identities across frames were annotated semi-automatically in all frames, by using the known global camera pose and a supervised interpretation plane clustering. The dataset consists of 48 sequences of 301 frames (at 10 fps) of street-view video from van-mounted cameras, yielding a total of 14448 annotated frames with between zero and three VPs. Due to the non-orthogonal street-layout, the Manhattan world assumption is generally not valid for this dataset. The videos are of varying difficulty for VP extraction, and contain easy city scenes, as well as challenging scenes dominated by vegetation and street furniture.

We use a dataset which was published by [Kroeger and Van Gool [2014]](https://example.com) for the task of registering videos to Structure from Motion (SfM) point clouds. The datasets consists of 4 sets of street-view videos, with 12 videos in each set, and a total of 48 videos. The 12 videos in each set were captured simultaneously from 12 cameras, which were rigidly mounted onto a van, in different parts of the town of Antwerp at a framerate of 10 fps. For each of the 4 sets a SfM point cloud, corresponding 2D feature locations, and precise 6-DoF camera poses for all views and frames are provided. Fig. 4.5 shows for one set (S01) the SfM point cloud (left), the camera set-up on the van (middle), and example images from the cameras (right).

Since there exists no dataset for VP detection and tracking over time, we adapted this dataset by adding VP annotation. We chose this dataset because ground truth camera poses are available. This enables, firstly, semi-automatic VP annotation, as described in the following, and, secondly, the experiment for the inclusion of pose knowledge in VP extraction and tracking (section 4.4.4).

For our evaluation we need annotation of (possibly multiple) VPs in each frame, and VP identities across time. Since the dataset consists of a total of 14448 ($= 4 \times 12 \times 301$) video frames, manual VP annotation and linking over time seems infeasible.
Figure 4.6: All 12 views from video set 1 (top) and 2 (bottom) at a random frame with detected line segments. Line color indicates VP ground truth identity. Cameras 9:12 are rotated because they are mounted on the van in portrait mode.

We adopted a semi-automatic procedure by using the known camera orientations, loosely based on the work of Tardif [2009], and proceed as follows for each of the four sets:
4.4. Experiments

1. In 12 videos with each 301 frames, we detect line segments using the LSD line detector (von Gioi et al. [2012]) in each frame, convert them to homogeneous normalized image coordinates (using the known internal camera calibration), and compute the corresponding interpretation planes as described in section 4.2.

2. Since the precise 6-DoF camera pose for every frame is known we rotate all interpretation planes into the common world frame.

3. For (on average) $\sim 1100$ image line segments in each frame ($3612$ frames $= 12 \times 301$), we compute $\sim 4,000,000$ interpretation planes. We keep the $50,000$ interpretation planes corresponding to the longest line segments. To ensure good coverage of all frames, we keep at least 10 planes (corresponding to the longest line segments) from each frame.

4. In order to create VP exemplars we randomly sample (with replacement) 1000 pairs of intersection planes and compute the resulting vanishing direction for every pair (See section 4.2).

5. For every interpretation plane we compute the consistency to all VP exemplars as: $|\alpha - 90|$, where $\alpha$ is the angle between plane normal and exemplar vanishing direction.

6. Having such VP exemplars and interpretation plane consistency scores, we perform J-Linkage clustering, as used by Tardif [2009] for VP detection. This process results in a small set of dominant scene directions in the world coordinate system of one set of 12 videos. Small VP clusters (with less than 1 percent of all interpretation planes associated to it) are removed.

7. We decide which of the dominant scene directions is visible in a given frame based on the number of LSD line segments, extracted from this frame, which are consistent with it. We chose a high threshold to trigger the start of VP track and a lower threshold to check for continuation in following frames. Selecting two thresholds (one for starting, one for continuation) ensures that a VP track is only started when the VP detection is reliable, but enables continuation through noisier frames with weaker VP line support. This process is very similar to the high and low hysteresis thresholds used for canny edge detection. We randomly assign numeric ground truth IDs to VP tracks created in this way.

8. For each frame in each video, we rotate all visible vanishing directions back to the camera coordinate system, using the known camera pose, and store the direction.

9. The resulting VP tracks are manually inspected and occasional small errors (start/ending times and continuation of VP tracks) are corrected.
The whole process is inspired by single-frame VP detection proposed by Tardif [2009], but differs in two significant ways from it: Firstly, we jointly perform VP detection in multiple views and over time using known poses. Secondly, we reason with a Gaussian sphere parameterization, while Tardif [2009] parameterizes the VPs in image space.

Example images of all 48 sequences can be seen in Fig. 4.6. Extracted line segments are overlaid. The line color indicates ground truth VP identity.

**Possible Errors.** Since we annotate semi-automatically using another VP detection algorithm we have to discuss possible errors introduced into the annotation. We assume error-free ground truth poses, and error-free internal camera calibration given in the dataset. We also assume a negligible (orientation) error in the LSD line detection. Thus, interpretation planes from all lines of all frames will be transferred with negligible error into the global world reference frame.

Our approach, based on Tardif [2009], generally has a very good recall (VPs are rarely missed, and accurately localized), but may suffer from slightly worse precision (false positive VPs are possible). This is due to the fact that some line segments will be consistent with multiple VP hypotheses on the horizon line. After visual inspection we found that filtering by line support strength (step 6) removes all false positive VPs. However, the question of false positive is also partly a question of which level of detail in annotation is required. Since every set of two parallel lines (e.g. outline of roof of a house, bricks in a wall) defines a VP, we have to select a target VP sensitivity. We designed the sensitivity such that dominant VPs in the scene over multiple frames are captured, but short-lived VPs with weak support (e.g. from slanted roofs) are not captured anymore. This trade-off is selected such that the returned level of detail of VP tracks is best suited for most potential tasks (3D Reconstruction, Object Tracking, Autonomous Navigation) in this street-view scenario.

### 4.4.3 Evaluation on the Street-View Dataset

**Baselines:** Since this is the first work for joint VP extraction and tracking, we construct our own baseline based on Tardif [2009]. We compare our method to two types of baselines.

The first baseline, *LPSF* (*Linear Program, Single-Frame*), corresponds to VP detection with our proposed method, where every frame is treated separately. Following the frame-wise VP extraction, we greedily grow VP tracks. Initially, the set of VP tracks is empty. For a new frame we merge VPs to existing VP tracks if the angular difference is smaller than 5 degrees. The remaining VPs of this new frame start new tracks. Finally, we remove VP tracks shorter than 3 frames. The second baseline, in several variants, called *TNO*, for Tardif, Non-Orthogonal, corresponding to a recent single-frame VP extraction method by Tardif [2009], where we ignore the extension for orthogonal VPs. Since the
4.4. Experiments

Figure 4.7: Cumulative MOTA and ID switches over 48 sequences for the Street-View evaluation in section 4.4.3. The result for the proposed method (LPMF) is colored in red. For MOTA our approach outperforms all baselines for 58 percent of the dataset, and is on par with the best baselines for the rest. For ID switches our method outperforms all baselines significantly over the whole dataset.

VP detection sensitivity of Tardif [2009] is strongly linked to the image resolution, we downscale the image by factors \{0.3, 0.4, 0.5, 0.6\} to obtain better scores for this baseline. VP tracks are grown greedily as for LPSF. VP tracks shorter than 3 frames are removed for the variants marked with \(f\).

Results: We evaluate with two metrics commonly used in tracking: multi-object tracking accuracy (MOTA, higher=better), proposed by Bernardin et al. [2008], and ID switches (IDS,lower=better), proposed by Li et al. [2009]. The result is visualized in Fig. 4.7. The VP matching threshold was set to an angle of 2 degrees. Our method is denoted as LPMF (Linear Program, Multi-Frame). For MOTA the best baseline results are generally obtained with LPSF and TNO 0.4 \(f\). The MOTA scores of TNO 0.4 \(f\) and TNO 0.5 \(f\) are very similar, but TNO 0.4 \(f\) has significantly fewer ID switches. Stronger down-scaling (\(\leq 0.3\)) leads to an increase in missed detections, and weaker down-scaling (\(\geq 0.6\)) to many false positive detections. Removing short (< 3 frames) VP tracks always increases MOTA and decreases ID switches. Stronger filtering leads to a loss in MOTA.

Greedy VP track linking in all baselines often fails, because of the lack of temporal smoothness constraints in the VP detection. This leads to frequent loss and re-initialization of VP tracks, yielding many ID switches. Our method significantly outperforms all baselines in ID switches: in 75 percent of the dataset no ground truth track is split. The best baseline on IDS, TNO 0.3 \(f\), achieves this only for 41 percent of the dataset, with significantly worse MOTA. In MOTA our approach outperforms all baselines for 58 percent of the dataset, and is on par with the best baselines for MOTA > 0. For a MOTA threshold of 0.5, our method, and the two best baselines LPSF and TNO 0.4 \(f\), have 42, 29 and 13 percent, respectively, of all sequences above this score. MOTA in all methods drops significantly for the most difficult 20 percent of all sequences. For all methods the strongest
negative influence on MOTA comes from missed VPs due to weak line support. In MOTP (multi-object tracking precision) all methods offer very similar performance.

Our unoptimized single-core Matlab implementation (using CPLEX) requires 2.8 seconds per frame on average on a Intel Core i7 CPU. One second is needed for pre-processing (line extraction, Hough grouping) and the rest for solving the LP. The best baseline TNO 0.4 f runs for 0.7 seconds per frame including line extraction. Recent related works report runtimes from ’a few seconds’ (Xu et al. [2013]) to half a minute (Tretyak et al. [2012]; Lezama et al. [2014]) per frame for similar image resolutions, but without the need for finding temporal correspondences. The extra time required for our approach in comparison to the TNO 0.4 f baseline is spent well on joint temporal data association, since our method leads to significantly fewer failure cases. This is demonstrated by two important experimental results: Firstly, our approach creates significantly fewer false positive tracks, as reflected in the MOTA score. Secondly, our approach rarely splits a ground truth track: in 75 percent of the dataset our approach does not have any ID switch, while this is only the case for 21 percent of all sequences for TNO 0.4 f. Low IDS is especially crucial for long-term operations, in which continuous VP identity information is needed and re-initializations are costly.

Some qualitative results are shown in Fig. 4.8. Lines which were associated to the same VP track are drawn in the same color. Examples for challenging scenes with failure cases are shown as well. In these examples only short and noisy line segments are available, due to high-frequency texture on the ground (cobble stone) and dominance of vegetation.

4.4.4 Inclusion of Known Camera Orientation

Introduction. If the camera orientation is known for every frame (e.g.: from odometry or 3D reconstruction) the problem of finding VPs over time can be simplified: 3D vanishing directions in the world reference frame are constant in time. Because of this, knowing the camera orientation in the world reference frame is equivalent to having the tracking component of our joint problem partially solved. We want to evaluate how much our method and the baselines improve when taking advantage of this knowledge.

Benchmark. In practice we can easily incorporate the known camera orientation for our method by rotating all VP evidence (i.e. 3D interpretation planes for all line segments) into the common world coordinate system for every frame. For our baselines we incorporate the known orientation after VP extraction, and apply the rotation to the extracted VPs. All other components of the methods remain unchanged. Since the used Street-View dataset provides precise camera poses for all frames, we can evaluate the behavior of our method when including pose knowledge on this dataset. The dataset generally does not have strong orientation changes. Because of this we sub-sampled the sequences to 1 fps, i.e. every 10th frame, for evaluation with strong frame-wise orientation changes. In
4.4. Experiments

**Figure 4.8**: VP detection and tracking examples. Line segments are colored according to their association to a VP. **Top 2 rows**: in each example three VPs are visible, correctly extracted and tracked. **Bottom 2 rows**: two examples for challenging scenarios. Short line segments are a problem in both cases. 3rd row: The vertical VP (red) is only weakly supported by noisy vertical line segments on vegetation. Some line segments for a horizontal VP (blue) are incorrectly associated. 4th row: The vertical VP (green, red, cyan, magenta) is not reliably tracked and has multiple ID switches.

Fig. 4.9 we show the cumulative MOTA score for our original method and the best performing baseline without \( \text{LPMF, TNO 0.4 f} \) and with \( \text{LPMF kp, TNO 0.4 f kp} \) inclusion of known poses.

**Results.** We observe that the separate VP extraction and greedy tracking in \( \text{TNO 0.4 f kp} \) improves strongly with known camera orientations. While our approach, \( \text{LPMF kp} \), still outperforms both baselines the improvement is smaller than for \( \text{TNO 0.4 f kp} \). This is explained by the smooth transition probability Eq. (4.10) which recovers the VP motion quite well. However, using the pose we could achieve a speed-up by increasing the sensitivity of Eq. (4.10) (i.e. setting \( \gamma_2 \) small and \( \gamma_1 \) large), which reduces the search space for the LP solution. The remaining gap in MOTA between our method and \( \text{TNO 0.4 f kp} \) is explained by the fact that our method supports VP tracks through multiple frames where line support is weak, while frame-wise extraction and greedy tracking in \( \text{TNO 0.4 f kp} \) will often lose tracks in those cases.

4.4.5 Vanishing Point Detection, Precision and Recall

We formulated the problem as tracking problem, and therefore choose to evaluate with MOTA, which incorporates an equally weighted missed detection and false positive count.
4.4. Experiments

Figure 4.9: Cumulative MOTA for the evaluation in section 4.4.4. If pose knowledge is incorporated our method (LPMF kp) still outperforms the best baseline (TNO 0.4 f kp) for most sequences.

But if the problem is regarded as a detection task, and temporal continuity ignored, the VP tracks can be split into frame-wise VP detections. This enables easier evaluation in terms of detection precision and recall.

We extracted VP tracks as described previously and split the tracks into independent VP detections in each frame. The proposed method is on par with the best performing baselines as shown by the precision and recall curve in Fig. 4.10

4.4.6 Evaluation on the York Urban Dataset (YUD)

Because the proposed method can also be applied to single-frame VP extraction, we evaluated our approach on the established York Urban Dataset (YUD) Denis et al. [2008] for Manhattan world VP extraction. We compare again to Tardif [2009], labeled in the figure as TO, for Tardif, Orthogonal, where we include an EM step to refine the set of orthogonal VPs. We run our method on single frames, using constraint C5 (4.9) to enforce orthogonality. As can be seen in Fig. 4.11, our approach is on par with Tardif [2009] for VP estimation accuracy.
4.5 Conclusion

Vanishing points encode low-level information of the scene structure and are used in many applications from scene understanding to 3D reconstruction. Many of these applications
operate on video input and can benefit from knowledge about VP continuation over time. In this work we presented an approach for jointly extracting and tracking VPs from video with known internal camera calibration.

We are the first to propose a method for this problem and provide a new dataset for evaluation. We showed that our method significantly outperforms various iterative detection and tracking approaches. We showed that even in cases where poses are known, a multi-frame approach is helpful as a temporal regularizer.

We focused on scenarios where no prior knowledge about the scene is available, and tested with simple cost and score models. As extensions, the method can be easily combined with many more powerful subsystems, as mentioned in section 4.1.1: line-VP consistency in image space, priors from partially or approximately known orientations, non-uniform spherical discretization, and VP refinement steps. Since the extracted VPs are constant in the world reference frame, our method can also be used to compute the change in camera orientation over time. Our method can also be adapted to other VP representations for which a non-parametric probability density over VP locations is available.
Mutually Orthogonal Vanishing Points

While vanishing point (VP) estimation has received extensive attention, most approaches focus on static images or perform detection and tracking separately. In this work, we focus on man-made environments and propose a novel method for detecting and tracking groups of mutually orthogonal vanishing points (MOVP), also known as Manhattan frames, jointly from monocular videos. The method is unique in that it is designed to enforce orthogonality in groups of VPs, temporal consistency of each individual MOVP, and orientation consistency of all putative MOVP. To this end, the method consists of three steps: 1) proposal of MOVP candidates by directly incorporating mutual orthogonality; 2) extracting consistent tracks of MOVPs by minimizing the flow cost over a network where nodes are putative MOVPs and edges are putative links across time; and 3) refinement of all MOVPs by enforcing consistency between lines, their identified vanishing directions and consistency of global camera orientation. The method is evaluated on six newly collected and annotated videos of urban scenes. Extensive experiments show that the method outperforms greedy MOVP tracking method considerably. In addition, we also test the method for camera orientation estimation and show that it obtains very promising results on a challenging street-view dataset.

5.1 Introduction

Often a number of simplifying assumptions are made in order to facilitate the reasoning about complex man-made environments. Most man-made structures can be described in terms of geometric primitives, such as parallel or orthogonal planes and lines. Under a projective transformation, sets of parallel lines often converge to an intersection point in the imaged scene. This point is known as a vanishing point (VP). The vanishing points provide strong cues for the 3D geometry of the scene. Since for scenes like urban environments the orthogonal planes are the dominant geometric primitives, one can constrain the detection to mutually orthogonal vanishing points (MOVP, also know as Manhattan frames, Straub et al. [2014]). One MOVP is depicted in Fig. 4.1 and Fig. 5.1. Generally
5.1. Introduction

Figure 5.1: One mutually orthogonal vanishing point (MOVP) discovered from a video sequence and visualized using 5 frames. The discovered MOVP allows extraction of the global camera orientation for each frame.

in man-made environments, there can be multiple MOVPs present, which may or may not share one common VP. Often a clearly dominant MOVP is not present, and a set of MOVPs have to be estimated, as visualized in Fig. 5.6.

Camera calibration (Grammatikopoulos et al. [2007]), pose estimation (Micusik & Wilde-nauer [2013]), 3D reconstruction (Crandall et al. [2012]; Kim & Manduchi [2014]), and autonomous navigation (Moghadam & Dong [2012]), are areas in the field of computer vision, where the VPs are used as low-level input. Many such applications, working on video sequences or image sets, require VP estimates in every frame and links across views or frames. If this is needed, the camera pose for each frame is usually assumed to be known (Antone & Teller [2000]; Hornáček & Maierhofer [2011]), facilitating the VP association across images. Alternatively, the VP detection and tracking tasks are separated, and association is done by greedy assignment (Elloumi et al. [2012]) or particle filters (Moghadam & Dong [2012]; Rasmussen [2008]). However, the pose knowledge is often not available, requires odometry or external motion measurements.

We propose the discovery of sets of MOVPs from videos where only the internal camera calibration is known. For this purpose, the method is designed to leverage mainly three sources of information: orthogonality in groups of VPs, temporal consistency of each individual MOVP, and orientation consistency of all putative MOVPs. We extract MOVP proposals in each video frame by directly incorporating mutual orthogonality, then enforce temporal consistency by using a multi-target tracking formulation, and finally refine the MOVP tracks by enforcing consistency between lines and their identified MOVP and consistency of global orientation of all MOVP. Our main contributions are:
1. We are the first to consider the problem of discovery of multiple MOVPs from videos with unknown camera pose. We provide a new evaluation dataset for this task.

2. We adapt the established Multi-Target Tracking formulation using min-cost network flows to the problem of MOVP discovery.

3. We propose a Non-Linear Least-Squares refinement step to jointly refine all discovered MOVPs and to reliably extract the global camera orientation.

The method is tested on six newly collected videos of real urban scenes, in which all vanishing points are manually labeled. Extensive experiments validate the effectiveness of the method, especially for challenging scenes where multiple MOVPs, with equally strong line support, appear. Furthermore, we apply the method to the task of global camera orientation estimation and show promising results on the large, challenging Antwerp street-view dataset, presented by Kroeger and Van Gool [2014].

5.1.1 Related Work

VP extraction is a popular topic in computer vision. We categorize according to algorithmic design choices the most relevant recent literature:

**Input:** Most works start from lines (Caprile & Torre [1990]; Elloumi et al. [2012]), or line segments (Barnard [1982]; Magee & Aggarwal [1984]; Rother [2002]; Grammatikopoulos et al. [2007]; Tardif [2009]; Hornáček & Maierhofer [2011]; Wildenauer & Hanbury [2012]; Antunes & Barreto [2013]). Some approaches employ continuous image gradients or texture (Schindler & Dellaert [2004]; Rasmussen [2008]; Moghadam & Dong [2012]) and thresholded edges images (Tretyak et al. [2012]). When the 3D geometry is known, the surface normals can be directly used, as shown in Straub et al. [2014].

**Accumulator space:** The intersections of imaged lines are computed in the (unbounded) image space (Rother [2002]; Schindler & Dellaert [2004]; Tardif [2009]; Elloumi et al. [2012]; Antunes & Barreto [2013]; Xu et al. [2013]) or on a (bounded) Gaussian unit sphere, as introduced by Barnard [1982] and used by Magee & Aggarwal [1984]; Quan & Mohr [1989]; Lutton et al. [1994]; Antone & Teller [2000]; Košlecká & Zhang [2002]; Grammatikopoulos et al. [2007]; Hornáček & Maierhofer [2011]; Bazin & Pollefeys [2012]; Micusik & Wildenauer [2013]; Straub et al. [2014].

**Line-VP consistency and VP refinement:** The consistency between an estimated VP and the image lines is usually measured using line endpoint distances in the image, used by us and Tardif [2009]; Hornáček & Maierhofer [2011]; Bazin & Pollefeys [2012]; Antunes & Barreto [2013]; Lezama et al. [2014]. The angular differences in the image is used by Rother [2002]; Denis et al. [2008], with explicit probabilistic modeling of the line end
5.1. Introduction

point errors by Xu et al. [2013], or with angles between normals of interpretation planes in the Gaussian sphere by Magee & Aggarwal [1984]; Quan & Mohr [1989]; Lutton et al. [1994]. We sample MOVP candidates on the Gaussian sphere because testing for orthogonality directly translates to vector cross products, but revert to image-space fitting errors for refinement to avoid distorted errors and to attenuate dependence on (potentially) noisy internal camera calibration.

The VP computation or refinement with given associated lines is commonly done by Hough voting and non-maximum suppression (Magee & Aggarwal [1984]; Quan & Mohr [1989]; Lutton et al. [1994]; Tuytelaars et al. [1997]; Moghadam & Dong [2012]), by solving a quadratic program (Antunes & Barreto [2013]), implicitly in an Expectation-Maximization (EM) setting (Antone & Teller [2000]; Schindler & Dellaert [2004]; Tardif [2009]; Xu et al. [2013]), or by linear least-squares, as presented by Košcká & Zhang [2002].

Solution: For a final solution, different methods combine the input, the accumulator space and the line-VP consistency measures and refinement. Efficient search (Rother [2002]; Denis et al. [2008]), direct clustering (Tardif [2009]; Lezama et al. [2014]), multiline RANSAC (Bazin & Pollefeys [2012]; Wildenauer & Hanbury [2012]), EM procedures (Antone & Teller [2000]; Košcká & Zhang [2002]; Schindler & Dellaert [2004]; Hornáček & Maierhofer [2011]; Xu et al. [2013]), or MCMC inference (Straub et al. [2014]) are among the methods employed directly on the accumulator space. If a discretization is enforced on the accumulator space, the solutions are found by voting schemes (Magee & Aggarwal [1984]; Quan & Mohr [1989]; Lutton et al. [1994]; Moghadam & Dong [2012]) or inference over graphical models (Tretyak et al. [2012]; Antunes & Barreto [2013]).

Camera calibration and VP orthogonality: The internal camera calibration is assumed known in the works of Quan & Mohr [1989]; Lutton et al. [1994]; Rasmussen [2008]; Hornáček & Maierhofer [2011]; Elloumi et al. [2012]; Moghadam & Dong [2012]; Kroeger et al. [2015b]. Other works do not require known camera calibration: Magee & Aggarwal [1984]; Košcká & Zhang [2002]; Rother [2002]; Xu et al. [2013]; Antunes & Barreto [2013]. From the extracted VPs the internal parameters can be estimated, as shown by Caprile & Torre [1990]; Grammatikopoulos et al. [2007]; Wildenauer & Hanbury [2012]. Also VPs have been used for the estimation of external camera parameters such as the orientation of camera to scene (Košcká & Zhang [2002]) and the orientation of 3D shape to camera (Barnard [1982]), and as additional constraints for full camera pose (Micusik & Wildenauer [2013]). Often, the VPs are extracted by imposing further scene-dependent constraints. In the works of Bazin & Pollefeys [2012]; Denis et al. [2008]; Hornáček & Maierhofer [2011]; Elloumi et al. [2012]; Wildenauer & Hanbury [2012] mutual VP orthogonality is assumed (Manhattan World). A shared vertical VP is assumed (Atlanta World) in the methods of Schindler & Dellaert [2004]; Antunes & Barreto [2013]. In this work and by Straub et al. [2014] sets of mutually orthogonal VPs (MOVPs) as assumed to be present.
Multi-view extraction and VP Tracking: Antone & Teller [2000] use known camera poses to solve multi-view VP extraction by Hough voting with EM refinement. Hornáček & Maierhofer [2011] use Structure-from-Motion (SfM) camera pose estimates to extract orthogonal VPs independently in multiple views and enforce consistency. Elloumi et al. [2012] extract orthogonal VPs separately in each video frame. VP sets are then greedily linked across frames. Rasmussen [2008]; Moghadam & Dong [2012] aim at road direction finding based on tracked VPs. A single finite VP is extracted and tracked using particle filters. It corresponds to the heading direction.

5.2 Our Approach

We aim at the discovery of multiple sets of MOVPs. For this, we start from line segments as image primitives, The lines in the image space are the observations we make over the scene world and support the presence of MOVPs. Therefore, the sets of MOVPs compete on the set of observations. Since we work over a video sequence, the temporal consistency of the MOVPs is another key information we use. We expect that over a whole sequence a reduced set of MOVPs is capable to explain all the observations and to be temporally consistent. Since all VPs are constant in space, and only the camera can move freely, temporal constancy of MOVPs directly translates to finding the global camera orientation in all frames, such that all locally extracted MOVPs are constant when transferred to the global reference frame.

In the following we derive the algorithmic formulation of our method. We will describe MOV candidate generation in section 5.2.1, temporal linking in section 5.2.2 and refinement in section 5.2.3.

5.2.1 MOV candidate extraction

For reasoning over VPs we first extract line segments (von Gioi et al. [2012]) as image primitives. Since exhaustively searching for all line convergence points is intractable due to the large amount of line segments, we employ a 3-line RANSAC sampling to extract MOV candidate, similar to the work of Bazin & Pollefeys [2012]. In highly textured scenes the number of line segments is large, and in consequence we will obtain many duplicate MOVPs. We reduce all samples to a set of representative candidates in a subsequent non-maximum suppression step.

Additionally, we need an approximate orientation change $D_{n,n+1}$ between all pairs of frames $n$ to $n+1$ in order to compute the linking cost between two MOVPs in section 5.2.2. This can be done using image descriptors, such as SIFT (Lowe [2004]), feature matching, Essential matrix computation, and decomposition. SIFT features can be
expensive to compute and match, may not be discriminative for repetitive structures, and
the orientation estimate may be noisy, as shown in the experiments in section 5.3.2. Be-
cause of this we chose to estimate the orientation change differently using a RANSAC
process again: We randomly sample one MOVP candidate from frame $n$ as well as $n+1$,
compute the necessary camera orientation change for a perfect overlap, and compute the
inliers, i.e. how many MOVP candidates from frame $n$ find a close fit in frame $n+1$.
For each frame $n$ we keep the best orientation change $D_{n,n+1}$, which produces the most
MOVP candidate inliers in the next frame.

5.2.2 Multi-MOVPs Tracking

The data association of the MOVPs extracted in each frame to global identities is formu-
lated as a Maximum A Posteriori (MAP) problem. We follow (including the notations)
the traditional approach of Zhang et al. [2008] as used for multi-object tracking. We use a
cost-flow network to model the problem and a min-cost flow algorithm to solve it. The in-
tuition is that finding non-overlapping MOVPs tracks is analogous to finding edge-disjoint
paths in a graph, which admits a solution by efficient network flow algorithms.

Let $X = \{x_i\}$ be the set of MOVP observations, each defined by a $3 \times 3$ orthonormal
matrix $x_i \in SO(3)$, and time step (frame index), $x_i = (x_i, t_i)$. A time ordered list of
MOVPs observations represents a single track hypothesis, i.e. $T_k = \{x_{k_1}, x_{k_2}, \cdots, x_{k_l}\}$
where $x_{k_i} \in X$ and $l_k$ is the length. A set of such track hypotheses defines an association
hypothesis, $T = \{T_k\}$.

The objective is to maximize the posteriori probability of $T$ given the observation set $X$:

$$
T^* = \arg \max_T P(T|X) = \arg \max_T P(X|T)P(T)
\tag{5.1}
$$

under the assumption of conditional independence of the likelihood probabilities given
the hypothesis $T$.

Optimizing directly over the space of $T$ is infeasible. The search space can be reduced
if we use the fact that an observation can not belong to more than one track, therefore
$T_k \in T$ can not overlap with each other:

$$
T_k \cap T_l = \emptyset, \forall k \neq l \tag{5.2}
$$

Generally, MOVP tracks may not be independent. But if we assume the camera orien-
tation to be given or computed (in our case by RANSAC in section 5.2.1), then we can
assume independence of MOVP tracks. Thus, Eq. (5.1) becomes:

$$
T^* = \arg \max_T \prod_i P(x_i|T) \prod_{T_k \in T} P(T_k)
\text{s.t. } T_k \cap T_l = \emptyset, \forall k \neq l \tag{5.3}
$$
where

\[ P(x_i|\mathcal{T}) = \begin{cases} 
1 - \beta_i & \exists T_k \in \mathcal{T}, x_i \in T_k \\
\beta_i & \text{otherwise}
\end{cases} \tag{5.4} \]

\[ P(T_k) = P(\{x_{k_0}, x_{k_1}, \ldots, x_{k_{l_k}}\}) = P_{\text{entr}}(x_{k_0})P_{\text{link}}(x_{k_1}|x_{k_0}), P_{\text{link}}(x_{k_2}|x_{k_1}) \cdots P_{\text{link}}(x_{k_{l_k}}|x_{k_{l_k}-1})P_{\text{exit}}(x_{k_{l_k}}) \tag{5.5} \]

\( P(x_i|\mathcal{T}) \) is the likelihood for an observation \( x_i \), \( \beta_i \) being the false alarm probability of \( x_i \). \( P(T_k) \) is the likelihood for a track \( T_k \) and is modeled through a Markov chain of transition probabilities \( P_{\text{link}}(x_{k_{i+1}}|x_{k_i}) \), initialization \( P_{\text{entr}} \) and termination \( P_{\text{exit}} \) probabilities. Since \( P(x_i|\mathcal{T}) \) models not only \( \mathcal{T} \) associated observations (true MOVPs) but also those without association (false alarms), the method is able to prune the observations by selecting the most consistent observations, thus forming strong tracks.

**Min-cost flow solution**

We use the following 0-1 indicators:

\[ f_{en,i} = \begin{cases} 
1 & \exists T_k \in \mathcal{T}, T_k \text{ starts from } x_i \\
0 & \text{otherwise}
\end{cases} \tag{5.6} \]

\[ f_{ex,i} = \begin{cases} 
1 & \exists T_k \in \mathcal{T}, T_k \text{ ends at } x_i \\
0 & \text{otherwise}
\end{cases} \tag{5.7} \]
\[ f_{i,j} = \begin{cases} 
1 & \text{if } \exists T_k \in \mathcal{T}, \text{ } x_j \text{ is right after } x_i \text{ in } T_k \\
0 & \text{otherwise} 
\end{cases} \quad (5.8) \]

\[ f_i = \begin{cases} 
1 & \text{if } \exists T_k \in \mathcal{T}, x_i \in T_k \\
0 & \text{otherwise} 
\end{cases} \quad (5.9) \]

and the notations:

\[ C_{en,i} = -\log P_{\text{entr}}(x_i) \quad C_{ex,i} = -\log P_{\text{exit}}(x_i) \]

\[ C_{i,j} = -\log P_{\text{link}}(x_j | x_i) \quad C_i = \log \frac{\beta_i}{1-\beta_i} \quad (5.10) \]

Given the above notations, the objective function \((5.1)\) in logarithmic form is as follows:

\[
\mathcal{T}^* = \arg \max_{\mathcal{T}} \sum_{T_k \in \mathcal{T}} -\log P(T_k) + \sum_i -\log P(x_i | T) \\
= \arg \max_{\mathcal{T}} \sum_{T_k \in \mathcal{T}} (C_{en,k_0} f_{en,k_0} \\
+ \sum_j C_{i,j,k_{j+1}} f_{j,k_{j+1}} + C_{ex,k_1} f_{ex,k_1}) \\
+ \sum_i (-\log(1-\beta_i) f_i - \log \beta_i (1- f_i)) \quad (5.11) \]

subject to that hypotheses in \(\mathcal{T}\) do not overlap, equivalent to

\[ f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j}, \forall i \quad (5.12) \]

This still allows for MOVPs to share one vanishing direction, such as the gravity direction, but prohibits tracks in which all vanishing directions are shared.

The MAP formulation, in logarithmic form \((5.11)\), can now be expressed in terms of a cost-flow network \(G(\mathcal{X})\) with source \(s\) and sink \(t\), as by Zhang et al. [2008]. A cost-flow network is depicted in Fig. 5.2. To each MOVP observation \(x_i \in \mathcal{X}\) correspond two nodes \(u_i, v_i\), an edge \((u_i, v_i)\) of cost \(c(u_i, v_i) = C_i\) and flow \(f(u_i, v_i) = f_i\), an edge \((v_i, t)\) of cost \(c(v_i, t) = C_{ex,i}\) and flow \(f(v_i, t) = f_{ex,i}\), and an edge \((s, u_i)\) of cost \(c(s, u_i) = C_{en,i}\) and flow \(f(s, u_i) = f_{en,i}\). For each \(P_{\text{link}}(x_j | x_i) \neq 0\) will correspond a transition edge \((v_i, u_j)\) of cost \(c(v_i, u_j) = C_{i,j}\) and flow \(f(v_i, u_j) = f_{i,j}\). The eqs. \((5.11)\) and \((5.12)\) are equivalent
to the flow conservation constraint and the cost of flow in network \(G\). Optimizing over

the data association hypothesis \(\mathcal{T}^*\) is equivalent to sending the flow from source \(s\) to sink \(t\), thus achieving the min-cost flow. To solve for the min-cost flow we use the efficient

push-relabel algorithm proposed by Goldberg [1997].

Costs

In the following we will define the terms \(P_{\text{entr}}, P_{\text{exit}}, P_{\text{link}},\) and \(\beta_i\) as needed in \((5.10)\). Entry and exit probabilities are a constant penalty for each started track, similar to a fixed

model cost, and can be used to fine-tune the overall sensitivity of the MAP solution. We
Figure 5.3: Fitting error for line segment to associated VP: The segment endpoints are projected onto a perfect line from the segment centroid to the VP. The projection error is denoted as $err_L$.

estimated the best sensitivity to be $P_{entr} = P_{exit} = .01$ on hold-out sequences. The probability $P_{link}$ describes the linking probability for two MOVPs in subsequent frames. Between MOVP $x_i$ and $x_j$ we assign a linking probability based on their angular difference $\alpha$ after applying the the camera orientation change $D_{n,n+1}$ computed in section 5.2.1.

$$P_{link}(x_j|x_i) = (1 + e^{-\gamma_1(\alpha - \gamma_2)})^{-1},$$

(5.13)

where $\alpha$ denotes the angular difference between $D_{n,n+1} \cdot x_i$ in frame $n$ and $x_j$ in frame $n + 1$. This sigmoid function yields a smooth fall-off at an angular difference of $\alpha - \gamma_2$, with decay rate controlled by $\gamma_1$. We learn these parameters on hold out sequences as $\gamma_1 = 4$, and $\gamma_2 = 1$. The remaining probability $\beta_i$ is the probability of MOVP being a false positive. We set $\beta_i$ to 1 minus the probability of sampling this MOVP in RANSAC given all detected line segments. To achieve this, we set $\beta_i$ to 1 minus the percentage of all MOVP samples created in the RANSAC candidate generation step (section 5.2.1) which agree with the MOVP candidate $i$. To be robust against missed line detection we set the limit $C_i = \min(C_i, 0)$.

### 5.2.3 MOVP refinement

From the data association in section 5.2.2 we obtain tracks $T_k$ which contain linked MOVP observations $x_i$. Each $x_i$ defines one MOVP as $3 \times 3$ orthonormal matrix $\in \text{SO}(3)$ within the local reference frame of the camera. With respect to the global reference frame, all MOVP observations $x_i$ in each track have to be constant. Using the hypotheses for camera orientation change $D_{n,n+1}, \forall n \in [1, N - 1]$ between all frames, we can transform all MOVPs to the global camera reference frame.

We initialize the global camera orientation as $R_1 = \text{diag}([1 1 1])$ for the first frame and $R_n = D_{n-1} \cdot R_{n-1}$ for subsequent frames. We set $\mathcal{R} = \{R_1, \ldots, R_N\}$. For each track $T_k$, starting at frame $S_k$, and all observations $x_i$ we initialize a global MOVP $M_k$

---

$^1$For all angular differences between MOVPs we follow Huynh [2009], but take care to consider that axes may be ordered differently between MOVPs.
by transforming all observations to the global reference frame and averaging the SO(3)
matrices as unit quaternions:

\[ M_k = |T_k|^{-1} \sum_i Q \left( R^T_{i+S_k-1} \cdot x_i \right), \]  \hspace{1cm} (5.14)

where \( Q \) computes the quaternions for a SO(3) matrix. We normalize \( M_k \) to unit norm,
and convert the quaternions to an SO(3) matrix. We set \( \mathcal{M} = \{M_1, \ldots, M_K\} \)

Because of the accumulation of errors in \( \mathcal{R} \), and noise in frame-wise extracted \( x_i \) the
solution for global orientation \( \mathcal{R} \) and the discovered MOVPs \( \mathcal{M} \) will generally not fit the
line segments in each frame perfectly. We refine the initial solution by jointly optimizing
\( \mathcal{R} \) and \( \mathcal{M} \) for the fitting errors of all line segments to all associated MOVPs in all frames
in a Non-Linear Least-Squares framework:

\[ \text{RSS}(\mathcal{R}, \mathcal{M}) = \sum_k \sum_i \text{err}_L(R_{i+S_k-1} \cdot M_k, L_{k,i}) \]

The error function \( \text{err}_L \) accepts a MOVP defined in a camera-centric reference frame
and line segments \( L_{k,i} \) associated to each vanishing direction. The line segment consistency error is computed for each line segment by projecting the segment endpoints onto a hypothesized perfect line through the line segment centroid and the associated VP. Fig. 5.3 illustrates this. Using this projection error has the advantage of using the undistorted image-space MOVP fitting error, treating finite and infinite VPs uniformly, and explicitly giving more weights to longer segments, as suggested by Rother [2002]. Since the problem is very sparse the optimization is tractable even for long sequences using a Trust-Region minimization, as often used in similar Bundle Adjustment problems, as detailed by Triggs et al. [2000]. After jointly minimizing the squared endpoint errors for all MOVPs in all frames we obtain optimal camera orientation estimates \( \mathcal{R} \), and MOVPs \( \mathcal{M} \).

5.3 Experiments

We conducted two experiments. First, we evaluated our approach on a new dataset of
6 inner-city sequences, each 100 frames long, using established Multi-Object tracking
metrics. Second, we evaluated how reliably we can extract the global camera pose over a
large dataset of street-view videos provided by a recent video registration work of Kroeger
and Van Gool [2014].

5.3.1 MOVPs discovery

Benchmark. We evaluate with three metrics commonly used in tracking: multi-object
tracking accuracy (MOTA, higher=better), multi-object tracking precision, the angular
5.3. Experiments

Figure 5.4: Example frames of the new dataset used in section 5.3.1 for sequences 1 to 6 (top left to bottom right)

matching error, (MOTP, lower=better), as proposed by Bernardin et al. [2008], and ID Switches (ids, lower=better), as proposed by Li et al. [2009]. The VP matching threshold was set to an angle of 5 degrees. We collected 6 sequences, each with 100 frames, and manually annotated sets of MOVPs in every 10th frame. Example frames for all videos are shown in Fig. 5.4. We included MOVP identity information over time. In each sequence between 1 and 4 MOVPs are jointly visible.

Methods. We evaluated our approach on these videos and visualize qualitative results for several frames of one sequence in Fig. 5.6. In the quantitative evaluation we compare four methods: 1) Our method including optimal tracking of section 5.2.2 and refinement of section 5.2.3, 2) our method without refinement, against 3) greedy MOVP association with refinement, and 4) greedy association without refinement.

For the greedy association we start from the same MOVP candidates as described in section 5.2.1. Instead of optimal data association using min-cost flow algorithm we greedily grow MOVP tracks. Initially, the set of MOVP tracks is empty. For a new frame we merge MOVP observations to existing MOVP tracks if the angular difference is smaller then $\alpha$ degrees. Since the performance of the greedy tracking is strongly dependent on $\alpha$, we evaluated with multiple values for $\alpha$ and compared to the best result with $\alpha = 6$. The
5.3. Experiments

Figure 5.5: ID Switches, MOTP, MOTA for MOVP discovery in 6 sequences. We compare: 1) Greedy tracking without refinement (only mean is displayed), 2) Greedy tracking with refinement, 3) Optimal MOVP tracking without refinement, 4) optimal MOVP tracking with refinement. Adding refinement generally improves greedy and optimal tracking. Our method outperforms the greedy tracking in all metrics.

remaining MOVPs of this new frame start new tracks. We remove MOVP tracks shorter than 5 frames. In Fig. 5.5 we provide a qualitative evaluation.

Results. Adding a refinement step generally improves the greedy as well as the optimal tracking. The benefit of Least-Squares refinement of line endpoint errors is most visible in sequences in which many MOVPs are visible simultaneously. This is because in those cases, each MOVP may not be very strong or reliable, and the camera orientation change hypotheses $D_{n,n+1}$ may be noisy as well. Especially in these cases enforcing a joint agreement on a global orientation and static MOVPs in the global frame improves results. We also observe, that our optimal tracking outperforms the refined greedy tracking even when no refinement is employed. Errors influencing the MOTA scores for the greedy and optimal tracking are largely dominated by false positive tracks, which may share strong line support with other MOVPs, such as on the gravity direction. Since they have partial strong line support and may move consistently with other MOVPs sometimes they are incorrectly included in the tracks. The greedy and optimal tracking both suffer equally from this problem.

It is important to emphasize that MOTA, MOTP and ID Switches are strongly interdependent, and that no singular focus on a single metric should be placed. It is possible to achieve good MOTP, i.e. low angular error, as for the greedy tracking without refinement, by simply accepting the closest tracks in each frame regardless of temporal consistency, which results in many ID Switches. Conversely, similar MOTA scores over all methods, as for sequence 3, are only a good discriminative metric, if information about the ID Switches on ground truth tracks is considered as well.

Runnites. The greedy and optimal tracking, both with refinement, run for 9.6 and 9.8 seconds per frame, respectively. The runtimes are largely dominated by our unoptimized
5.3. EXPERIMENTS

Figure 5.6: Qualitative results two discovered MOVPs in one sequence using our method. Top: Four frames of the sequence with line segments, colored according to MOVP assignment. Bottom two rows: All line segments are extended to the point of convergence. The two tracks share a common vertical VP corresponding to the gravity direction.

MATLAB implementation of MOVP candidate generation, which runs for 9.1 seconds per frame on average.

5.3.2 Camera orientation estimation

**Benchmark.** In the first experiment we evaluated how accurate we can discover all MOVPs in the scene. For many tasks the identification of Manhattan Frames (MOVPs) is just the first step in discovering a more fine-grained scene structure. Manhattan frame discovery can help in this, since we get the camera orientation change for free when at least two VPs are identified in two different views (Caprile & Torre [1990]). After the refinement, proposed in section 5.2.3, we obtain a global camera orientation estimate, which we will evaluate in this section.

The Antwerp Street-View Dataset, introduced by Kroeger and Van Gool [2014] and used for Video Registration, provides 48 sequences of 301 frames with precisely known camera pose at all times. Several example frames are shown in Fig. 5.1. In order to make the orientation estimation more challenging we uniformly sub-sampled the sequences to 101 frames.

**Methods.** We track multiple MOVPs in all 48 sequences using the greedy and optimal MOVP discovery, including refinement for both methods. We also compared to the hypothesized global orientation before refinement, as mentioned in section 5.2.3. Additionally, we extracted SIFT features, computed Essential matrices between successive frames, extracted the frame-to-frame change in camera orientation, and transformed it
Figure 5.7: Error accumulation of global camera orientation estimation on the Antwerp Street-View dataset (Kroeger and Van Gool [2014]) for our optimal MOVP tracking, greedy MOVP tracking, hypothesized global orientations before refinement, and frame-to-frame SIFT features matching with essential matrix decomposition. The dotted lines in each color denote the 75% and 25% quantiles for each method.

Results. The comparison of all four methods is plotted in Fig. 5.7. We notice that already the hypothesized camera orientation, even before refinement, has half the accumulated orientation error of SIFT features. We again half this drift error by adding greedy or optimal MOVP tracking and refinement. The optimal tracking gains over the greedy tracking mostly in the worst case scenarios, where fewer mistakes result in fewer local optima in which the refinement can get trapped. Overall optimal MOVP tracking and refinement results in less than 10 degree orientation drift (on median) against 40 degrees for SIFT features.

5.4 Conclusion

In this work we presented a novel method for discovery of sets of mutually orthogonal vanishing points from monocular video sequences with unknown camera pose. We contribute an optimal way of extracting MOVPs over time using a hypothesized global
orientation from all MOVP candidates, and a method to jointly refine MOVPs and global camera poses. This refinement, similar in spirit to Bundle Adjustment for Structure-from-Motion problems, greatly improves both greedy and optimal MOVP tracking results.

In future work we plan to tackle current limitations of the method: 1) false positives due to strong shared VPs and line association ambiguity for VPs on the horizon line. 2) Our method is generic and does not favor specific VPs. However, when considering city scenes, detecting zenith and horizon lines could provide powerful additional constraints.
6

Video Registration

Registering image data to Structure from Motion (SfM) point clouds is widely used to find precise camera location and orientation with respect to a world model. In case of videos one constraint has previously been unexploited: temporal smoothness. Without temporal smoothness the magnitude of the pose error in each frame of a video will often dominate the magnitude of frame-to-frame pose change. This hinders application of methods requiring stable poses estimates (e.g. tracking, augmented reality). We incorporate temporal constraints into the image-based registration setting and solve the problem by pose regularization with model fitting and smoothing methods. This leads to accurate, gap-free and smooth poses for all frames. We evaluate different methods on challenging synthetic and real street-view SfM data for varying scenarios of motion speed, outlier contamination, pose estimation failures and 2D-3D correspondence noise. For all test cases a 2 to 60-fold reduction in root mean squared (RMS) positional error is observed, depending on pose estimation difficulty. For varying scenarios, different methods perform best. We give guidance which methods should be preferred depending on circumstances and requirements.

6.1 Introduction

Due to recent advances in 3D range imaging highly accurate and large 3D models for real-world environments can easily be obtained, as shown by Agarwal et al. [2009]; Klingner et al. [2013]; Tanskanen et al. [2013], and are already available for many city areas. Given structural information about the world, many new opportunities for computer vision (CV) applications in scene understanding arise. Videos are a rich source for capturing and analyzing social activities, human/vehicular traffic and events. This allows for CV applications such as multi-view object tracking, vehicle and pedestrian trajectory analysis, video cutting, multi-video event and scene summarization. Registration of video data to a 3D world model using visual information is an essential requirement for many of these applications. They benefit from accuracy and robustness of pose estimations (6-DoF, position and orientation) for one or several videos at all frames, rather than live performance
6.1. Introduction

The standard approach of image-based registration to SfM models (Sattler et al. [2011, 2012]; Irschara et al. [2009]; Li et al. [2010, 2012b]) involves the following steps for each image/video frame: computing 2D image features, matching them to features associated with 3D points, finding the pose using a standard perspective-n-point (PnP) algorithm in a RANSAC loop with 2D-3D correspondences. Direct application of this technique to video data results in very noisy pose estimates, as illustrated in Fig. 6.1 (left & top right). The path of the camera is drawn in red and exhibits strong positional noise: the average positional difference between poses of successive frames is one order of magnitude larger than the ground truth motion. The true motion is completely dominated by uncorrelated positional errors. Using frame-wise PnP estimates we also have to deal with estimation failures (i.e. gaps) and pose outliers in addition to noisy poses. This is unsatisfactory since many tasks, such as multi-view tracking or augmented reality, require accurate, gap-free and smooth poses as input. This is why we explore several regression methods to exploit temporal smoothness for refining PnP camera poses, which were independently estimated for every video frame. Our aim is to bridge the gap between unreliable, noisy, incomplete, frame-wise pose estimates in SfM models to accurate and smooth pose trajectories to be used for higher-level CV applications.

Our main contribution is the reduction of pose errors for all frames of a video, for which approximate and possibly incomplete frame-wise estimated poses are available. In order to achieve this, we adapt several model fitting techniques (Splines Smoothing, Kernel

Figure 6.1: Left & Top right: frame-wise registered hand-held video with 300 frames, The camera’s path (red) is noisy due to PnP estimation errors: The average frame-to-frame position difference is 57cm while the ground truth camera moves with approximately 5cm/frame. Bottom right: refined camera path (with Kernel Regression).
Regression, Non-Linear Least-Squares optimization) to the problem. We propose a new pose parameterization to be able to use spline and kernel smoothing methods for camera poses. In Non-Linear Least-Squares optimization we introduce a novel bending energy minimization extension for camera pose smoothing. We discuss several combinations of the three methods. All methods are evaluated on real and synthetic data for various difficult scenarios. We give guidance on which method works best under which circumstances. Until now, no such comprehensive description and evaluation for global pose trajectory refinement exists. We are the first to contribute a carefully designed benchmark on synthetic and real data for this.

**Paper Overview:** Section 6.1.1, lists related work. The video pose registration methods (Spline Smoothing, SP, Kernel Regression KR, non-linear least-squares optimization LS) and variants are proposed in section 6.2, and evaluated in section 6.3. Section 6.4 concludes this chapter.

### 6.1.1 Related Work

**Landmark recognition, localization of images** are active fields of research: An image is positioned with respect to reference images with known localization (e.g. GPS) by Zamir & Shah [2010]; Kalantidis et al. [2011], at city scale, with efficient feature representation by Schindler et al. [2007]; Knopp et al. [2010], using databases of building facades by Robertson & Cipolla [2004], and even larger world-wide approximate localization by Hays & Efros [2008]. Other methods rely on localization by recognition of landmarks, e.g. by Bergamo & Torresani [2013]. These methods do not rely on 3D structure, but on localized reference images and 2D features. Generally, most of these techniques employ image retrieval techniques and 2D similarity measures based on feature matching. In contrast to this Hao et al. [2012] relies on 3D features to recognize places.

**Image-based registration** to 3D SfM models is concerned with complete (6-DoF) pose estimation. Poses are computed with 2D-3D correspondences based on feature matching (Se et al. [2001]). Poses retrieved in this way are more accurate and dependency on visually similar reference images is reduced. Scaling these techniques is difficult, due to the large amounts of features in the matching step. Perspective-n-Point (PnP) algorithms find the pose given a set of 2D-3D correspondences (Zheng et al. [2013b,a]; Lepetit et al. [2009]; Li et al. [2012a]). Li et al. [2012b] improve the feature matching step by RANSAC co-occurrence-based sampling. Focus has been put on efficient feature storage and matching (Boix et al. [2013]), with vocabulary trees (Irschara et al. [2009]), prioritized matching (Li et al. [2010]), efficient correspondence search (Sattler et al. [2012, 2011]), match pre-filtering using image retrieval (Cao & Snavely [2013]), and discriminative visual element mining in challenging scenarios, such as registering paintings by Aubry et al. [2013]. Hao et al. [2013]; Gordon & Lowe [2006] address
the problem of finding the (6-DoF) pose of observed objects. Ramalingam et al. [2011] estimate the pose using lines instead of point features.

**Localization of image sequences and videos** has received attention as well. Hakeem et al. [2006] register video frames by employing fundamental matrix constraints between a video frame and the two closest GPS-annotated reference images. GPS coordinates are extracted for all frames by Spline Smoothing. Similarly, Vaca-Castano et al. [2012] retrieve geo-localized images and uses Bayesian tracking for refinement. Kalogerakis et al. [2009] coarsely localize image sequences with large time-gaps, such as series of photographs of entire tourist trips on a world-wide grid. Visual odometry in car-mounted cameras is used for localization in a known road network by Brubaker et al. [2013]. Agrawal [2013] optimize poses when no model but frame-to-frame pose changes and measurement uncertainties are available from essential matrices or inertial sensors.

**Video registration to 3D SfM models** received less attention than image-based registration. However, it is an integral part of SLAM (Davison et al. [2007]; Klein & Murray [2007]; Newcombe et al. [2011]; Newcombe & Davison [2010]). There, the focus lies on jointly tracking features, and improving their and the camera’s localization. In contrast to this the scene structure is predetermined in our task. We do not have a prior on the camera’s location. Additionally, the reconstructed environments in SLAM are usually small controlled indoor environments. Imaging conditions for SfM and localization are the same in SLAM, which is generally not the case when a query video has to be registered to separately reconstructed 3D models. Se et al. [2001] localize video sequences by matching and tracking SIFT features. Similarly, Lim et al. [2012] estimate poses by matching and tracking DAISY features. Registration to high-quality CAD models has been worked on as well: ego-motion is tracked by Koch & Teller [2007] by edge matching in omni-directional videos, by Hsu et al. [2000] by feature tracking and coarse-to-fine refinement of edge alignment. Irschara et al. [2009] find poses for every frame of videos separately, simplifying the matching by computing virtual views. Zhao et al. [2004]; Rodriguez & Aggarwal [1990] rely on computing SfM from a query video first, and retrieve poses by alignment of the world model and the SfM model from the query video.

SLAM and feature-tracking based techniques work for small datasets or when features can be matched reliably. If matching is difficult (larger city scenes, strongly varying imaging conditions), tracking features will easily result in propagation of matching errors. Techniques that reconstruct the sequence first and match later suffer from typical SfM problems: model deformation and fragmentation, matching problems and the need for manual subsequent alignment with a world model. Because of these principled problems we want to match as many frames as possible directly to the world model and rely on global pose refinement.
6.2 Registration of Videos to SfM Models

Frame-wise registered videos can exhibit strong noise in individual poses, estimation gaps and pose outliers as illustrated in Fig. 6.1. Noise, outliers and estimation gaps can be dealt with when incorporating temporal smoothness. On approximately and incompletely registered poses for each frame, described in section 6.2.1, we build the refinement methods proposed in section 6.2.2, 6.2.3, and 6.2.4. The goal is to improve every frame’s pose estimate while being robust towards outliers. We chose Spline Smoothing as a well-known representatives of regularization and basis expansion techniques, Kernel Regression as a representative for probabilistic kernel methods and Non-Linear Least-Squares optimization as representative for direct optimization of re-projection errors as also used in bundle adjustment.

6.2.1 Image-Based Registration

A SfM model is represented by 3D points and associated SIFT (Lowe [2004]) feature descriptors from the views in which the 3D points were observed. We match a new query image by extracting SIFT features, matching them to all features associated with the 3D points, and thereby retrieve a putative set of 2D to 3D correspondences. For known internal parameters many recent pose estimation algorithms (EPnP by Lepetit et al. [2009], ASPnP by Zheng et al. [2013b], OPnP by Zheng et al. [2013a], RPnP by Li et al. [2012a]) can be used directly in a RANSAC-loop to retrieve (6-DoF) camera position and orientation as detailed by Sattler et al. [2011, 2012]; Irschara et al. [2009]; Li et al. [2010, 2012b]. In the remainder of this chapter we assume given internal camera parameters (focal length, projection center, no radial distortion).

6.2.2 Spline Smoothing

In Spline Smoothing (SP) piece-wise polynomial functions \( f(x_i) \) are fitted to \( N \) sites \( x_i \) with observations \( y_i \) by minimizing the residual sum of squares (RSS):

\[
RSS(f, \lambda) = \lambda \sum_{i=1}^{N} w_i (y_i - f(x_i))^2 + (1 - \lambda) \int (f''(x))^2 dx.
\]  

(6.1)

The camera pose estimate at time \( x_i \) is denoted with \( y_i \) (observed), and \( f(x_i) \) (smoothed). The \( N \) data sites correspond to the number of video frames. A camera pose is represented as a position \( t \) and rotation matrix \( R \). We parameterize the pose as a 9-dimensional vector \( y = [t^T \ r_1^T \ r_2^T] \) with unit vectors \( r_1 \) and \( r_2 \) as viewing direction and up-vector of the camera. The RSS is regularized by \( f \)'s second derivative, i.e. to minimizing the bending...
6.2. Registration of Videos to SfM Models

energy. The data fidelity term is weighted by $w_i$, the inlier count after RANSAC, down-weighting poses with few 2D-3D correspondences. The regularization parameter $\lambda \in [0, 1]$ is found via leave-one-out cross-validation.

We propose a variant of a smoothing spline including the camera parameters’ covariance $\Sigma$ and the Mahalanobis distance in the data fidelity term ($\text{SP+C}$). Deviation from estimated poses are penalized stronger in the data fidelity term if the cameras’ pose estimates are with low variance:

$$RSS(f, \lambda) = \lambda \sum_{i=1}^{N} w_i (y_i - f(x_i))^T \Sigma^{-1} (y_i - f(x_i)) + (1 - \lambda) \int (f''(x))^2 dx.$$  \hfill (6.2)

The solution for both spline variants is a weighted linear combination of the observations. The chosen pose parameterization is an approximation to rigid Euclidean motion which is part of the Special Euclidean Lie Group $\text{SE}(3)$. Some constraints on orientation cannot be enforced by the spline formulation: orthogonality ($r_1 \cdot r_2 = 0$) and unit norm ($\|r_1\| = \|r_2\| = 1$). However, if the change in $R$ is small, we can assume $\|r_1\| \approx 1 \approx \|r_2\|$ and $r_1 \cdot r_2 \approx 0$. This allows using this under-constrained approximation in smoothing. We can enforce constraints afterwards: For each $f(x_i)$ we re-normalize $r_1$, $r_2$ to unit norm and recover $R' = [r_3 \ r_1 \ r_2]^T$ with $r_3 = r_1 \times r_2$. To get a valid rotation matrix we enforce orthogonality by singular value decomposition $[U, S, V] = \text{svd}(R')$ and set $R' = U \cdot V^T$. This approximation is valid as long as the between-frame change in $R$ is slow. See experiments in section 6.3.1, 6.3.4 for an analysis of the limits of this parameterization. As alternative parameterization, we experimented with quaternions and an angle-axis representation, with less stable results.

6.2.3 Kernel Regression

A smoothing spline works well in cases of outlier-free data, perturbed by Gaussian noise. However, even after RANSAC a few pose outliers can remain. In order to avoid a hard inlier-outlier decision for poses, we can still use a RANSAC-inspired pose estimation approach. But instead of keeping only the best result (i.e. sample with highest inlier count) we keep the $M$ best pose samples. This leads to $M$ pose estimates for all $N$ frames. Using the best RANSAC samples requires randomly distributed outliers. If outliers are systematic, $M$ random samples have to be used to avoid biased estimates. A Nadaraya-Watson model, or Kernel Regression ($\text{KR}$), can represent poses over $N$ data sites probabilistically:

$$p(y, x) = \frac{1}{W} \sum_{i=1}^{N} \sum_{j=1}^{M} w_{i,j} \ k(x - x_{i,j}, y - y_{i,j})$$ \hfill (6.3)
where $k$ is the density function, $w_{i,j}$ sample inlier count, $W = \sum_{i=1}^{N} \sum_{j=1}^{M} w_{i,j}$. The pose sample $j$ at time $x_i$ is denoted $y_{i,j}$. We use a Gaussian kernel

$$k(x, y) = \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{1}{2} \left( \frac{x^2}{h_x} + \frac{y^2}{h_y} \right) \right)$$  \hspace{1cm} (6.4)$$

with bandwidths $h_y$ in parametric space for the camera pose and $h_x$ in time. Both parameters are found in leave-one-out cross-validation. The regression function $f(x_i)$ corresponds to conditional averages of target $y_i$ conditioned on time $x_i$:

$$f(x_i) = \mathbb{E}(y|x_i) = \int y \, p(y, x_i) \, dy .$$  \hspace{1cm} (6.5)$$

This representation allows for a non-parametric probabilistic interpretation of the camera's pose at all times. Outliers can be filtered out effectively. However, it depends on integration over all kernel functions, which can be time consuming.

### 6.2.4 Non-Linear Least-Squares Optimization

SP and KR (section 6.2.2, 6.2.3) operate directly on estimated poses, and are therefore dependent on initial PnP pose quality. We propose a similar objective function as eq. (6.1) with data fidelity and smoothing term, but instead of using estimated poses in the data fidelity term, we can use the 2D-3D correspondences directly by measuring the 3D point re-projection error. The objective function can be minimized as a Non-Linear Least-Squares (LS) problem:

$$RSS(P_1, \ldots, P_n) = \sum_{i=1}^{N} \sum_{j=1}^{J_i} (z_{i,j} - P_i Z_{i,j})^2 + \lambda^T K T$$  \hspace{1cm} (6.6)$$

where $P_i = C \cdot [R_i, -R_i \cdot t_i]$, with $C$ known camera calibration, $J_i$ the number of 2D-3D correspondences after RANSAC for pose $P_i$, and $z_{i,j}, Z_{i,j}$ the known 2D and 3D locations of correspondence $j$ in pose $P_i$. $K$ is the bending penalty matrix as in the Reinsch–form for SP [Hastie et al., 2009, p.154]. $T = [t_1, \ldots, t_n]$ is a matrix of camera locations. PnP poses are only required as initialization, reducing dependency on PnP methods. The additional advantage of this approach is that other constraints, such as planarity of camera movement (LS+CP), can be integrated:

$$RSS(P_1, \ldots, P_n, CP) = \sum_{i=1}^{N} \sum_{j=1}^{J_i} (z_{i,j} - P_i Z_{i,j})^2 + \lambda^T K T + \theta \sum_{i=1}^{N} D^2(CP, t_n)$$  \hspace{1cm} (6.7)$$

where $D$ (3rd term) returns the distance of camera position $t_n$ to camera plane $CP$. The camera plane is a free variable in the optimization. Because the regularization parameters $\lambda$ and $\theta$ cannot easily be found automatically, we set them manually to balance the
influence of all residuals. Starting the optimization with PnP poses and associated 2D-3D correspondences, we cannot easily remove the influence of incorrect 2D-3D correspondences. However, we can use a Cauchy loss for the re-projection error $\Delta(x) = \log(1 + x)$ to mitigate the influence from outlying correspondences. The Ceres-Solver\footnote{http://code.google.com/p/ceres-solver/} is used to minimize eq. (6.6, 6.7).

6.2.5 Combinations and Variants

Besides the discussed methods SP, KR, LS and variants SP+C, LS+CP we include several combinations in our experiments when suitable. The estimated poses from PnP algorithms can be further refined by using Non-Linear Least-Squares optimization of the re-projection error, i.e. eq. (6.6) without smoothing ($\lambda = 0$) (LSWS). Based on LSWS we can again start Spline Smoothing (LSWS+SP) or Kernel Regression (LSWS+KR). When LS is started from PnP pose estimates, outliers are corrected due to the smoothing term. This correction is improved when LS is initialized from Spline Smoothing solutions (SP+LS), and may also be combined with the assumption of planarity of camera movement (SP+LS+CP). For some frames no pose estimates exist. This is due to RANSAC failures because of too many outliers or noisy correspondences. Gaps in PnP pose estimates can also be deliberate: Feature matching and RANSAC pose estimation is the main bottleneck for large SfM scenes. It may be necessary to consider only every $n$th camera pose and interpolate in between. Such gaps can be closed after pose refinement by using standard cubic interpolating splines with knots given by the output of our proposed methods.

6.3 Experiments

In three experiments we evaluate the performance of the proposed methods and variants, while assuming that unreliable PnP poses from an arbitrary source are available as input. We evaluate with exemplary state-of-the-art PnP methods as mentioned in section 6.2.1. The first experiment (section 6.3.1) on synthetic data shows the stability of the smoothing methods with respect to different degrees of pose changes, 2D observation noise, number of 2D-3D correspondences and outlier contamination. The second experiment (Sec 6.3.2,6.3.3) shows the performance of the methods on real SfM data of city environments. In the first two experiments we ignore any occurring gap in the PnP pose estimates. In the third experiment (section 6.3.4) we compare the methods when interpolating over gaps following the smoothing. We focus on the positional (root mean squared, RMS) error to ground truth camera locations. The reasons are 1) space constraints, 2) the camera position is more sensitive to typical problems in feature-based pose estimation.
(noisy/incorrect 2D-3D correspondences) than orientation and 3) position and orientation errors are strongly correlated. We consider frame-wise PnP pose estimates, computed with ASPnP Zheng et al. [2013b], as baseline. We chose ASPnP as best performing PnP method (See table 6.3 in section 6.3.2). Results for further refined poses (LSWS) without smoothing, and simple Kalman Filtering (KF) are also included. The experimental setup remains the same in all experiments: regularization parameters for SP and KR are automatically determined via cross-validation. Regularization parameters for LS are set manually and are the same for all experiments: \( \lambda = \theta = 10^5 \). We scale the 3D model to real-world scale. All variants of LS run for 10 iterations. State and observation covariances for KF are computed in an EM-style algorithm. To limit memory-complexity in KR, we set \( M = 20 \) pose samples per frame.

### 6.3.1 Synthetic Video Sequence

The synthetic data consists of a camera (focal length 1000 px, 1280x720 resolution), viewing a simple 3D structure (2 walls at a 135 angle) with 800 3D-points at a distance of 10 meters. We create sequences of 300 frames by rotating the camera around the visible structure (Fig. 6.2, top left). For every frame we randomly sample 25 2D-3D correspondences and compute the pose. In different sequences with increasing speed of rotation (degrees/frame) we test the effect on smoothing and stability of the pose parameterization. All experiments are repeated 25 times and results averaged. Fig. 6.2 shows the positional error of our proposed methods against the ASPnP baseline. Increasing speed of rotation around the structure, shown in Fig. 6.2 (a), enlarges pose differences between successive frames. This shows how each method is affected by increasing pose differences and the sensitivity of the pose parameterization for SP and KR, outlined in section 6.2.2. In Fig. 6.2 (b) we examine the performance if the number of 2D-3D correspondences before computing the PnP pose in each frame is decreased, (c) Gaussian noise is added to the 2D feature locations, and (d) the percentage of (uniformly distributed) pose outliers is increased. These plots show the reliability of each method with respect to typical challenges in feature-based pose estimates. We observe:

- KR and SP perform well for slow pose changes. The under-constrained pose parameterization leads to a rising performance loss for fast pose changes.
- LSWS and LS are unreliable (out of scope in plots) due to strong 2D, 3D noise, leading to local optima in optimization. SP+C is unreliable as well.
- The overall best performing and stable method is LS optimization initialized with the result of Spline Smoothing (SP+LS): The local optima problem of LS and LSWS are avoided by initializing the poses near the real optima. SP+LS is hardly affected by low feature count, noise, outliers and fast pose changes.
6.3. Experiments

Figure 6.2: Refinement results for synthetic video. Top Left: Synthetic sequence (300 frames) of camera (red) rotating around structure (green). Refinement result for (a) varying speed of movement around the structure (degrees/frame), (b) number of 2D-3D correspondences, (c) 2D Gaussian noise, (d) contamination with PnP pose outliers. The legend of (a) also applies to (b,c,d). In (a) LS,LSWS,SP+C,KF are partly out of scope.

Figure 6.3: Left: Rigid camera setup for street-view image capture. Blue camera: used in SfM, red: used only for evaluation. Right: Exemplary SfM Model from 300 frames. Path of van and mounted cameras in scene every 40 frames.

6.3.2 Street-View Video Sequence - Dataset and PnP Baseline

Since no public dataset for video registration is available we created our own: Street-view image data was captured with 8 cameras, rigidly mounted onto a van, in 1628x1236 resolution, at 10 fps for 30 seconds. The visible street scene was reconstructed with (on average) 400K 3D points. Additionally to the 8 cameras used for SfM, 2 or 4 additional cameras were mounted on the van for evaluation. SfM reconstruction with a rigid multi-camera installation on a moving van returns poses for all cameras and the van at all times. Using known rigid camera setup and van pose, precise pose ground truth can be inferred for the additional cameras as well. The rigid camera configuration and an example SfM
Table 6.1: Time for parameter estimation via Cross Validation (SP, SP+C, KR), Expectation Maximization (KF), and solution (sec). Parameters in LS are set manually.

<table>
<thead>
<tr>
<th>Method</th>
<th>KF</th>
<th>SP</th>
<th>SP+C</th>
<th>KR</th>
<th>LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.Estim.</td>
<td>&lt; 1</td>
<td>7</td>
<td>81</td>
<td>637</td>
<td>NaN</td>
</tr>
<tr>
<td>Solution</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>2</td>
<td>164</td>
<td>52</td>
</tr>
</tbody>
</table>

Table 6.2: PnP failure rates (percent of all frames). A: pose estim. failure, no pose returned. B: failure to find good pose (pos.err < 1m), C: pose estim. failure when at least one other method found a good pose, D: failure to find good pose when all other methods returned good poses.

<table>
<thead>
<tr>
<th>Method</th>
<th>OPnP</th>
<th>ASPnP</th>
<th>EPnP</th>
<th>RPnP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.17</td>
<td>.03</td>
<td>0.64</td>
<td>2.19</td>
</tr>
<tr>
<td>B</td>
<td>13.46</td>
<td>11.82</td>
<td>11.85</td>
<td>14.98</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0.14</td>
<td>0.69</td>
</tr>
<tr>
<td>D</td>
<td>1.27</td>
<td>.30</td>
<td>0.69</td>
<td>2.82</td>
</tr>
</tbody>
</table>

Model can be seen in Fig. 6.3. Data was gathered in 4 locations, one of which is displayed in Fig. 6.3, resulting in 12 videos with each 300 frames. Typical problems for feature matching in this case are over/underexposure of the images, uneven distribution of feature locations, motion blur, lack of sufficient view overlap (the SfM cameras are looking down, the additional cameras are looking up). The additional cameras are (independently) registered to the SfM Model in our evaluation. PnP poses are obtained from OPnP, ASPnP, EPnP, RPnP. Table 6.2 compares (A) all methods in terms of percentages of failed pose estimates, (B) failure to find good (pos. err < 1m) poses, (C) failed pose estimates when at least one other method found a good pose, and (D) failure to find a good pose when all other methods found one. ASPnP offers the overall best performance.

Because strong pose outliers are still present for all PnP methods, we proceed to identify and remove outliers, and provide all refinement results for varying levels of removal. Ideally, outliers are identified automatically without the help of ground truth. This can be achieved by using positional differences between PnP poses of successive frames if outliers are randomly distributed and not systematic. For evaluation purposes we simplify the task and use the ground truth error for outlier removal: We define five positional error thresholds: \( \rho_{1,...,5} = \{.54, .65, 8.7, 47, \infty \} \) representing meters of allowed error in camera position to ground truth position. They correspond to \{80, 85, 90, 95, 100\} percent of the data as inliers. Poses with an error above a chosen threshold are removed from the PnP baseline and considered as gaps. For \( \rho_5 \) we do not remove any pose.

In table 6.3 all PnP methods are listed with RMS positional error to ground truth in meters (left) and error of viewing direction in degrees (right) for \( \rho_{1,...,5} \). We note that the positional error increases significantly in all PnP methods once fewer outliers are removed. The same order of magnitude for positional errors for image-based registration in typical
6.3. Experiments

<table>
<thead>
<tr>
<th>ρ = ρ₁</th>
<th>OPnP-t</th>
<th>ASPnP-t</th>
<th>EPnP-t</th>
<th>RPnP-t</th>
<th>OPnP-R</th>
<th>ASPnP-R</th>
<th>EPnP-R</th>
<th>RPnP-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.44</td>
<td>0.14</td>
<td>0.16</td>
<td>0.25</td>
<td>0.29</td>
<td>0.27</td>
<td>0.28</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>ρ = ρ₂</td>
<td>0.44</td>
<td>0.15</td>
<td>0.67</td>
<td>1.69</td>
<td>0.29</td>
<td>0.27</td>
<td>0.38</td>
<td>0.44</td>
</tr>
<tr>
<td>6.58</td>
<td>4.76</td>
<td>2.97</td>
<td>3.48</td>
<td>2.61</td>
<td>1.97</td>
<td>1.72</td>
<td>1.85</td>
<td></td>
</tr>
<tr>
<td>ρ = ρ₃</td>
<td>12.23</td>
<td>10.49</td>
<td>8.26</td>
<td>9.26</td>
<td>6.62</td>
<td>5.06</td>
<td>5.59</td>
<td>5.09</td>
</tr>
<tr>
<td>26.09</td>
<td>23.77</td>
<td>17.96</td>
<td>27.48</td>
<td>10.49</td>
<td>8.85</td>
<td>8.44</td>
<td>9.42</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: PnP pose estimation errors. Left 4 col.: positional RMS error in meters, Right 4 col.: viewing direction errors in degrees (ignoring roll). Rows: Estimation error for outlier varying outlier threshold ρ. Thresholds are chosen such that {80, 85, 90, 95, 100} percent of the data are inliers. See also table 6.4 (left) for median positional PnP errors.

city scenarios is reported independently by Sattler et al. [2011, 2012]; Li et al. [2010]; Hsu et al. [2000]. Confirming the conclusion of Zheng et al. [2013b], ASPnP offers the best results. EPnP is not as precise in easy pose estimation scenarios (ρ₁,₂) but gains if outliers are present (ρ₃,₄,₅). Note how the orientational and positional errors compare: for pose errors around 5-6 meters, the error in orientation is still < 2 degrees. Average PnP runtimes (seconds) are: OPnP 1.31, ASPnP 0.22, EPnP, 0.25, RPnP: 0.13.

In the remainder of the experiments ASPnP will be used as the preferred PnP baseline. We will report the results of all refinement methods based on ASPnP poses (Table 6.5). Additionally, we will provide the results of only the best refinement method based on all PnP baselines (Table 6.4 (right) and Fig. 6.4).

6.3.3 Street-View Video Sequence - Video Registration

We adopt the following shorthand notation. A: PnP baseline error, B: LSWS, C: KF, D: SP, E: SP+C, F: LSWS+SP, G: KR, H: LSWS+KR, I: LS, J: LS+CP, K: SP+LS, L: SP+LS+CP. Fig. 6.4 lists the best performing refinement method over all PnP methods with absolute positional RMS error. In table 6.5 the relative scores for all refinement methods in relation to the ASPnP baseline RMS positional error (Col. A) are listed (Col. B:L). Table 6.4 shows the best refinement results using median errors (right), and median PnP baseline error (left), to illustrate the performance when disregarding outliers. Table 6.1 lists runtimes for parameter estimation and solutions for the proposed methods. Comparing the best smoothing results for all PnP methods (Table 6.4 and Fig. 6.4) we observe:

- Refinement after ASPnP offers significantly better results with few outliers (ρ₁,₂,₃) than other PnP methods: the best method reduces the error of the worst method by 83 percent. For (ρ₄,₅) PnP dependency decreases: best method reduces the error of the worst method by only 19 percent.
<table>
<thead>
<tr>
<th>$\rho$</th>
<th>OPnP-t</th>
<th>ASPnP-t</th>
<th>EPnP-t</th>
<th>RnPnP-t</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_1$</td>
<td>0.381</td>
<td>0.065</td>
<td>0.066</td>
<td>0.133</td>
</tr>
<tr>
<td>$\rho_2$</td>
<td>0.375</td>
<td>0.064</td>
<td>0.290</td>
<td>0.576</td>
</tr>
<tr>
<td>$\rho_3$</td>
<td>1.042</td>
<td>0.599</td>
<td>1.300</td>
<td>0.809</td>
</tr>
<tr>
<td>$\rho_4$</td>
<td>1.479</td>
<td>1.697</td>
<td>2.281</td>
<td>1.531</td>
</tr>
<tr>
<td>$\rho_5$</td>
<td>2.609</td>
<td>2.684</td>
<td>2.812</td>
<td>2.267</td>
</tr>
</tbody>
</table>

Table 6.4: Median PnP baseline positional error (left) and best median refinement error (right). This table shows the gain in positional accuracy from PnP baseline to best performing refinement method with respect to the median ground truth positional error. The letter (same as in table 6.5) indicates the used method.

- In general, least-squares techniques, LS (I), LS+CP (J), SP+LS (K), SP+LS+CP (L) offer the best performance for all outlier levels.
- For $\rho_4, \rho_5$, the introduction of the camera plane assumption and the initialization using splines SP+LS (K), SP+LS+CP (L) offer a small gain.
- For median errors (table 6.4) variants of LS (I,J,K,L) also perform best. CP inclusion gives no improvement. The results do not depend on the PnP method.

Comparing relative method accuracy for ASPnP as baseline (table 6.5) we observe:

- SP (D) is increasingly helpful with growing outlier contamination ($\rho_3, \rho_4, \rho_5$). Similar to results on synthetic data, SP+C (E) does not help much.
- In general, all LS variants (I,J,K,L) offer significantly better results than any other method. In contrast to our experiments on synthetic data, for real data LSWS (B) and, as a consequence LS (I) / LS+CP (J) have similar scores in positional accuracy.
6.3. Experiments

| \( \rho = \rho_1 \) | 0.14 | 1.30 | 0.90 | 1.19 | 1.01 | 1.56 | 0.51 | 0.52 | 2.18 | **2.26** | 2.16 | 2.22 |
| \( \rho = \rho_2 \) | 0.15 | 1.24 | 0.96 | 1.19 | 1.01 | 1.53 | 0.46 | 0.44 | 2.23 | **2.26** | 2.19 | 2.24 |
| \( \rho = \rho_3 \) | 4.76 | 1.03 | 1.15 | 1.82 | 1.01 | 1.91 | 1.23 | 1.27 | 20.21 | **23.56** | 19.34 | 21.92 |
| \( \rho = \rho_4 \) | 10.49 | 1 | 1.11 | 2.23 | 1.24 | 2.20 | 2.01 | 2.03 | 18.40 | **21.99** | 19.65 | 20.76 |
| \( \rho = \rho_5 \) | 23.77 | 1 | 1.23 | 3.51 | 1.60 | 3.56 | 5.50 | 5.62 | **59.68** | 58.36 | 37.03 | **38.85** |

| \( \rho = \rho_1 \) | 0.27 | 1.10 | 0.94 | 1.10 | 1.01 | 1.20 | 0.79 | 0.80 | 1.34 | 1.35 | 1.34 | **1.35** |
| \( \rho = \rho_2 \) | 0.27 | 1.08 | 1.02 | 1.10 | 1.01 | 1.18 | 0.72 | 0.73 | 1.34 | 1.33 | **1.35** | 1.33 |
| \( \rho = \rho_3 \) | 1.97 | 1.04 | 0.99 | 1.36 | 1 | 1.39 | 1.06 | 1.13 | 1.18 | 1.18 | **1.43** | 1.42 |
| \( \rho = \rho_4 \) | 5.06 | 1.02 | 0.98 | 1.65 | 0.97 | **1.70** | 1.31 | 1.30 | 1.04 | 1.06 | 1.29 | 1.29 |
| \( \rho = \rho_5 \) | 8.85 | 1.01 | 0.97 | 2.54 | 0.95 | **2.58** | 1.76 | 1.76 | 0.99 | 1 | 1.45 | 1.44 |


- Initialization using splines slightly improves orientation estimation. Inclusion of a CP in the optimization marginally improves the result, but leads to a slower convergence.
- KF (C) and KR (G,H) help primarily in case of many outliers (\( \rho_{4,5} \)), LSWS (B) helps for (\( \rho_{1,2,3} \)). Initializing SP (D) or KR (G) with LSWS (B) in LSWS+SP (F), LSWS+KR (H) leads to a marginal improvement.

As in our experiment on synthetic data, variants of LS perform best on real data as well. The influence of initial PnP poses is weak if few outliers are present.

**Comparison with registration after reconstruction:** An alternative way of video registration is SfM reconstruction of a query video, and alignment of the new model to the ground truth. We reconstructed every video with standard SfM tools. The camera poses were rigidly aligned to the ground truth by minimizing the RMS positional error. The resulting positional error of **9.051 meters** is significantly worse than our best refinement result with an error of **2.267 meters** (See fig. 6.4, \( \rho_5 \) for no outlier removal). This is mainly due to SfM model deformation and fragmentation.

### 6.3.4 Gap Interpolation

There are three scenarios where gaps, i.e. missing pose estimates for a consecutive number of frames, can occur: 1) Failure of the PnP algorithm to converge, 2) Removal of
6.4 Discussion and Conclusion

The three experiments show that in all test cases in synthetic and real data most proposed methods improve pose accuracy over frame-wise registered poses by including temporal

identified pose outliers, 3) deliberate speed-up by matching every \( n \)th frame to the SfM model. In our third experiment gaps are created deliberately in the synthetic dataset (section 6.3.1, fast non-linear camera motion) and street-view dataset (Sec 6.3.2, mostly linear, slow camera motion). We keep every \( n \)th pose, leaving the remaining frames as gaps, and refine the camera path. We interpolate over the gaps with cubic interpolation splines by using the refined poses as knots, and evaluate on all frames including gaps. Fig. 6.5 shows the positional RMS error for increasing gap sizes on real (right) and synthetic data (left). We observe:

- KR and KF are rarely helpful: For large gaps KF’s linear dynamics assumption is violated, conditional averaging in KR is unstable. LS,LSWS,SP+C have the same problem as in section 6.3.1.

- SP shows reliable refinement over gaps. The gain over the ASPnP baseline depends on the degree of non-linearity of camera motion: The camera motion eliminates any gain after 7,10 (synthetic,real) skipped frames.

- As in our previous experiments, SP+LS offers the overall best performance on both datasets. The inclusion of a CP as constraint offers an additional boost in real data. SP+LS (left) and LS+CP (right) lose their gain over the baseline only after 20 and 30 skipped frames, respectively.

Figure 6.5: Positional RMS pose error for refined poses after gap interpolation. Every \( n \)th PnP pose is kept, remaining frames are gaps, poses are smoothed, and result interpolated with cubic splines. Left: Synthetic dataset, Right: Street-view videos. Note: due to high, very volatile error LSWS, LS (left) and SP+C (right) were not plotted. KF and KR (right) exhibit high error and are out of scope.

identifying pose outliers, deliberate speed-up by matching every \( n \)th frame to the SfM model. In our third experiment gaps are created deliberately in the synthetic dataset (section 6.3.1, fast non-linear camera motion) and street-view dataset (Sec 6.3.2, mostly linear, slow camera motion). We keep every \( n \)th pose, leaving the remaining frames as gaps, and refine the camera path. We interpolate over the gaps with cubic interpolation splines by using the refined poses as knots, and evaluate on all frames including gaps. Fig. 6.5 shows the positional RMS error for increasing gap sizes on real (right) and synthetic data (left). We observe:

- KR and KF are rarely helpful: For large gaps KF’s linear dynamics assumption is violated, conditional averaging in KR is unstable. LS,LSWS,SP+C have the same problem as in section 6.3.1.

- SP shows reliable refinement over gaps. The gain over the ASPnP baseline depends on the degree of non-linearity of camera motion: The camera motion eliminates any gain after 7,10 (synthetic,real) skipped frames.

- As in our previous experiments, SP+LS offers the overall best performance on both datasets. The inclusion of a CP as constraint offers an additional boost in real data. SP+LS (left) and LS+CP (right) lose their gain over the baseline only after 20 and 30 skipped frames, respectively.

6.4 Discussion and Conclusion

The three experiments show that in all test cases in synthetic and real data most proposed methods improve pose accuracy over frame-wise registered poses by including temporal
smoothness. The best achieved improvement ranges from 2 to 60-fold reduction in RMS positional error depending on the outlier contamination and magnitude of pose changes between frames.

Generally, variants of LS provide the best results for positional, but not for orientational accuracy. The positional accuracy can be improved further by adding additional constraints, such as a planar motion assumption (LS+CP). Robust initialization (SP+LS) can help with convergence when strong noise in 2D and 3D is present. SP provides the fastest method with good results. Inclusion of camera parameter covariances (SP+C) did not improve accuracy due to many spurious feature matches. KR was able to handle outlying poses efficiently, but conditional averaging decreases accuracy when poses are already good. If speed is a constraint SP and LS scale linearly and are close to real-time performance. In case orientation is more important than position and the data is strongly contaminated with outliers, SP offers the best performance. Even for medium sized SfM models (∼ 10^5 3D points) frame-wise feature matching and pose estimation is likely to be slower than our proposed pose refinements. This can be mitigated by matching only every nth frame, smoothing and interpolating. In case of interpolation, LS performs best, followed by SP. For real SfM data we note that refinement results strongly depend on the initially used PnP algorithm in case of few outliers (ρ₁) but not so for many outliers (ρ₅): ratio of best to worst result: 0.17 for ρ₁, but 0.81 for ρ₅. (See Fig. 6.4). The resulting refinement methods are applicable in many domains where video poses are needed: Besides 2D-3D correspondences no further knowledge is required.

The present work opens three main branches of future work. First, from the large body of works on regularization, basis expansion, and probabilistic kernel methods, we adapted several techniques (SP, KR) to the problem of video registration. Different parameterizations and techniques, such as random regression forests can be examined. Second, combinations of this method with pose estimation through feature tracking, as used by Lim et al. [2012]; Se et al. [2001], should be explored. Third, the LS refinement can naturally be combined with previous works on video pose estimation where SLAM is applied to a video first, and matched to a 3D world afterwards, similarly to the works of Zhao et al. [2004]; Rodriguez & Aggarwal [1990].
Conclusion

In this chapter, we summarize the presented contributions and give an outline for possible future work. We believe that our contributions are a significant step towards making motion estimation from video sequences more efficient, robust and accurate.

7.1 Summary

Our presented system for optical flow extraction addresses the need to have efficient systems to capture fast motions of unknown and arbitrarily deformable objects with unknown camera motion. This fast and unconstrained motion estimation enables systems which are required interact with an environment at a low-latency. Since the computation of optical flow is a low-level building block for many higher-level computer vision applications, the speed-up we achieved will propagate, and hopefully will improve the efficiency of larger computer vision pipelines. Furthermore, we showed that it is beneficial to adapt the frame-rate and time-budget per frame to find the optimal balance for a specific task. Often it is easier to deal with higher frame-rates, higher image noise, and lower time-budget per frame, than to solve large and complex image deformations and movements at a more generous time-budget.

Our presented system for multi-target tracking-by-detection in multiple cameras solves the problem of object evidence association across time as well as across different cameras in the presence of strong camera localization noise. While jointly associating and tracking across cameras was already solved for multiple static cameras, our contribution enables the same technique to be applied for sets of moving cameras with noisy and approximate localization. This extends the range of applications to all forms of hand-held cameras and robotic vision sensors.

Our works on vanishing point detection and tracking combine and advance techniques that were developed and tested for single-frame vanishing point detection to the domain of video data. Many of the applications which require vanishing point information, or which
exploit it as helpful cue, such as autonomous road navigation and camera calibration, require temporal associations as well. Replacing the established solution of frame-wise extraction and greedy temporal association, we provide a already temporally integrated solution. Additionally, by leveraging the information from multiple frames we gain robustness, due to the temporal regularization. Estimating the motion of vanishing points and lines over time also enables the exploitation of this information for camera orientation estimation.

Our presented system and evaluation of techniques for registering video sequence data to given structure-from-motion models addresses the need to precisely register videos to given 3d models. This augments the established method of registering the video either frame-by-frame, resulting in strong frame-to-frame noise, or by explicit feature tracking, resulting in tracking drift. High prevision is required for reprojection of augmented 3D models (i.e. for architectural planing) into the given sequence for visualization purposes. Precision is also required if multiple videos are considered for joint object identification and tracking.

7.2 Future work

The directions of research covered in this thesis are open-ended, and many additional experiments and extensions are worth exploring. While we propose short-term research projects in each chapter that would be immediately beneficial to explore, here, we outline more general, long-term research directions for the covered problems.

In the field of optical flow estimation, fast as well as highly accurate solution exist. However, there is still a significant accuracy-gap between making efficient (real-time) systems, and bringing state-of-the-art performance to mobile systems. Very recently method based on multi-layer neural networks have been proven to give excellent results on high-performance GPUs (Ilg et al. [2016]; Choy et al. [2016]). It will be worth investigating whether fully integrated (end-to-end) learned systems will be sufficient for better performance and efficiency, or whether the learned compositional representation can be used to improve already efficient methods, such as ours. We still that believe high-accuracy method for optical flow computation will only start to be more widely adopted once the computational requirements are eased. Furthermore, we see it as desirable to explore the extensions of existing optical flow methods for the task of scene flow computation. While this task requires only few additional constraints and input images, it enables full reconstruction of motion and depth in 3D.

The recent years have shown that tracking by temporal data association can be improved by powerful graph optimization. However, the biggest impact on tracking performance came, unsurprisingly, from better frame-wise object evidence, provided by impressive advances in object detection and neural networks. The same trend of learning better data...
representations may be applicable to learn better temporal association measures. Regarding camera localization with respect to a knowledge-base has also seen strong advances. We believe it is desirable to focus on research from both directions, camera localization and object detection and matching, to improve multi-camera multi-target tracking methods even further.

Similarly, vanishing point estimation is dependent on an intermediate feature representation of extracted line segments and associated feature descriptors. Learned representations are likely to improve the extraction and description of these intermediate representations. However, it may also be worthwhile to explore directly learning such detectors on data without going through intermediate representations.

Due to significant advances in the field of simultaneous localization and mapping (SLAM) we believe that merging approaches based on feature tracking followed by local reconstruction and approaches aiming at global registration is desirable. Such hybrid systems may provide the robustness due to local reconstructions, while avoiding the long-term drift due to global registration. Additionally, directly localizing images and video data with respect to given models has also proven to work with neural networks (Weyand et al. [2016]; Kendall et al. [2015]). However, while such learned systems have been shown to improve existing localization pipelines, for full positional and orientation registration they are still out-performed by feature and retrieval-based systems.
Bibliography


Baker, S., & Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *IJCV*. 1, 2, 2.1.2, 2.2.1, 2.2.1

Bao, L., Yang, Q., & Jin, H. (2014). Fast edge-preserving patchmatch for large displacement optical flow. *Image Processing, IEEE Transactions on*, 23(12), 4996–5006. 2.1, 2.1.1, 2.1.2, 2.3.1, 2.3.4, 2.3.4, 2.3.5


Brox, T., & Malik, J. (2011). Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. PAMI*. 2.1.1, 2.3.4, 2.3.5, 2.3.6


Cao, S., & Snavely, N. (2013). Graph-Based Discriminative Learning for Location Recognition. *CVPR*. 6.1.1


Elloumi, W., Treuillet, S., & Leconge, R. (2012). Tracking Orthogonal Vanishing Points in Video Sequences for a Reliable Camera Orientation in Manhattan World. In *CISP*. 4.1, 4.1.1, 4.3.1, 5.1, 5.1.1


URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6906584 2.2.1


URL http://www.sciencedirect.com/science/article/pii/S1077314215000429 2.1.1


URL http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=flow 2.1.2, 2.3.5


Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. Springer, 2nd, 2009 ed. 6.2.4


Hofmann, M., Wolf, D., & Rigoll, G. (2013). Hypergraphs for Joint Multi-View Reconstruction and Multi-Object Tracking. 3.1.1, 3.2, 3.2, 3.3, 3.3.5, 3.4


Hu, Y., Song, R., & Li, Y. (2016). Efficient coarse-to-fine patchmatch for large displacement optical flow. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.1.1, 2.3.6


Jiang, H., Fels, S., & Little, J. J. (2007). A Linear Programming Approach for Multiple Object Tracking. 3.1.1, 3.2


Li, S., Xu, C., & Xie, M. (2012a). A Robust O(n) Solution to the Perspective-n-Point Problem. *PAMI*. 6.1.1, 6.2.1

Li, Y., Snavely, N., Huttenlocher, D., & Fua, P. (2012b). Worldwide pose estimation using 3d point clouds. 6.1, 6.1.1, 6.2.1

Li, Y., Snavely, N., & Huttenlocher, D. P. (2010). Location Recognition using Prioritized Feature Matching. 6.1, 6.1.1, 6.2.1, 6.3.2


Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. IJCV. 2.1.1, 5.2.1, 6.2.1

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In IJCAI, vol. 81, (pp. 674–679). 2.1, 2.1.1, 2.1.2, 2.2.1, 2.3.4


Menze, M., Heipke, C., & Geiger, A. (2015). Discrete optimization for optical flow. In GCPR. 2.1, 2.1.1, 2.3.6

Micusik, B., & Wildenauer, H. (2013). Minimal solution for uncalibrated absolute pose problem with a known vanishing point. In 3DVV. 4.1, 4.1.1, 5.1, 5.1.1


Moreno-Noguer, F., Lepetit, V., & Fua, P. (2007). Accurate Non-Iterative O(n) Solution to the PnP Problem. IEEE. 3.4


Pellegrini, S., Ess, A., & Van Gool, L. (2010). Improving data association by Joint Modeling of Pedestrian Trajectories and Groupings. 3.1.1


URL http://dx.doi.org/10.1007/s11554-014-0423-0 2.1, 2.1.1, 2.1.2, 2.3.5

Possegger, H., Sternig, S., Mauthner, T., Roth, P. M., & Bischof, H. (2013). Robust Real-Time Tracking of Multiple Objects by Volumetric Mass Densities. 3.1.1


Sattler, T., Leibe, B., & Kobbelt, L. (2011). Fast image-based localization using direct 2D-to-3D matching. *Ieee*. 3.1.1, 6.1, 6.1.1, 6.2.1, 6.3.2

Sattler, T., Leibe, B., & Kobbelt, L. (2012). Improving Image-Based Localization by Active Correspondence Search. 6.1, 6.1.1, 6.2.1, 6.3.2


Senst, T., Eiselein, V., & Sikora, T. (2012). Robust local optical flow for feature tracking. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(9), 1377–1387. 2.1.1, 2.3.5, 2.3.5


Sun, D., Roth, S., & Black, M. J. (2010). Secrets of optical flow estimation and their principles. In *CVPR*. 2.2.3, 2.3.4


Tardif, J.-P. (2009). Non-Iterative Approach for Fast and Accurate Vanishing Point Detection. In *ICCV*. 4.1, 4.1.1, 4.4.2, 6, 4.4.2, 4.4.3, 4.4.6, 5.1.1

Timofte, R., & Van Gool, L. (2015). Sparseflow: Sparse matching for small to large displacement optical flow. In *WACV*, (pp. 1100–1106). 2.1, 2.1.1, 2.1.2, 2.3.2, 2.3.4, 2.3.5, 2.3.6


Weinzaepfel, P., Revaud, J., Harchaoui, Z., & Schmid, C. (2013). Deepflow: Large displacement optical flow with deep matching. In *ICCV*. 2.1, 2.1.1, 2.1.2, 2.2.3, 2.2.4, 2.3.2, 2.3.3, 2.3.4, 2.3.5, 2.3.6


Wulff, J., & Black, M. J. (2015). Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *CVPR*, (pp. 120–130). 2.1, 2.1.1, 2.1.2, 2.3.2, 2.3.4, 2.3.4, 2.3.5

Xu, L., Jia, J., & Matsushita, Y. (2012). Motion detail preserving optical flow estimation. *IEEE Trans. PAMI*. 2.1, 2.1.1, 2.3.6


Zamir, A. R., & Shah, M. (2010). Accurate image localization based on google maps street view. 6.1.1


List of Publications

Refereed Conference Proceedings


Others

Zusammenfassung


Die beobachtete Bewegung ist die Kombination von Bewegung des Beobachters und Umgebungsbewegung. Oft lässt sich die Bewegung der Umgebung aufteilen in einen statischen Hintergrund und ein oder mehrere bewegliche Objekte. In diesem Fall lassen sich die Aufgaben der Bewegungsschätzung für Objekte und Beobachter separieren in
unabhängige Aufgaben. Es ist jedoch ein unzureichend beschränktes Problem den sta-
tischen Hintergrund von beweglichen Objekten im Vordergrund zu unterscheiden, ohne
dabei die Bewegung des Beobachters zu kennen. Deshalb ist es oft einfacher zuerst die
Bewegung des Beobachters zu bestimmen. Hierfür kann angenommen werden, dass die
Bewegung des statischen Hintergrundes die gesamte beobachtete Bewegung in der Bild-
sequenz dominiert. Sobald die Beobachterbewegung im Verhältnis zum Hintergrund be-
stimmt ist, können bewegliche Objekte des Vordergrundes einfach segmentiert werden.
Falls ein präzises Modell der statischen Geometrie der beobachteten Szene zur Verfügung
steht, kann die Bewegung des Beobachters bestimmt werden, indem die projizierte Geo-
metrie in der Bildsequenz zu dem 3D Modell registriert wird. In Umgebungen, in denen
parallele Strukturen dominieren, zum Beispiel in Städten, lässt sich eine weitere Informa-
tionsquelle ausnutzen: Fluchtpunkte von parallelen Strukturen in Bildsequenzen lassen
sich mit der 3D Geometrie ins Verhältnis setzen und erlauben die Bestimmung der Orient-
tierung des Beobachters im Raum.

Die vorliegende Arbeit ist ein Beitrag zu der Schätzung und des Verständnisses von Bewe-
gung von Beobachter und beobachteter Szene aus Bildsequenzen in den oben genannnten
Themengebieten. Im Folgenden beschreiben wir die effiziente Berechnung von zweidi-
ensionalen Bewegungsfeldern, publiziert in [Kroeger et al., 2016], die Methode für das
multi-target tracking-by-detection über mehrere synchrone Bildsequenzen, publiziert in
[Kroeger et al., 2014], die Methode für das Bestimmen und zeitliche Assozieren von
Fluchtpunkten in Bildsequenzen, publiziert in [Kroeger et al., 2015a] und [Kroeger et al.,
2015b], und die Methode zur Registrierung von Bildsequenzen zu einem gegebenen 3D
Modell, publiziert in [Kroeger and Van Gool, 2014].
Curriculum Vitae

Personal Data

Name   Till Kroeger  
Date of birth 19th June 1983  
Place of birth Bielefeld, Germany  
Citizenship German

Education

2012 – 2018 ETH Zurich, Computer Vision Laboratory, Switzerland  
   Doctoral studies  
2002 – 2012 University of Heidelberg, Germany  
   Undergraduate and Graduate Studies of Applied Computer Science  
   Undergraduate and Graduate Studies of Medieval European History  
1999 – 2002 Carl-Severing Berufskolleg, Germany  
   Abitur

Work Experience

2017 – 2018 Zoox Inc., Research Engineer, Menlo Park, USA  
   Calibration, localization and mapping for autonomous driving  
2016 – 2017 Undisclosed Software Engineering Position, Switzerland  
   Algorithm research for Computer Vision in realtime robotic applications  
2012 – 2016 ETH Zürich, Computer Vision Laboratory, Switzerland  
   Teaching and research assistant  
2000 – 2004 Sternberg Software-Technik GmbH, Germany  
   Internship, part-time software engineer