Semi-Dense Filter-Based Visual Odometry for Automotive Augmented Reality Applications

A thesis accepted by the Faculty of Aerospace Engineering and Geodesy of the University of Stuttgart in partial fulfilment of the requirements for the degree of Doctor of Engineering Sciences (Dr.-Ing.)

by

MSc. Stephan Schmid

born in Böblingen

Main referee:	Prof. Dr.-Ing. Dieter Fritsch
Co-referee:	Prof. Dr. Luc Van Gool
Date of defense:	29.3.2019

Institute for Photogrammetry
University of Stuttgart
2019

Contents

Abstract

In order to integrate virtual objects convincingly into a real scene, Augmented Reality (AR) systems typically need to solve two problems: Firstly, the movement and position of the AR system within the environment needs to be known to be able to compensate the motion of the AR system in order to make placement of the virtual objects stable relative to the real world and to provide overall correct placement of virtual objects. Secondly, an AR system needs to have a notion of the geometry of the real environment to be able to properly integrate virtual objects into the real scene via techniques such as the determination of the occlusion relation between real and virtual objects or context-aware positioning of virtual content. To solve the second problem, the following two approaches have emerged: A simple solution is to create a map of the real scene *a priori* by whatever means and to then use this map in real-time operation of the AR system. A more challenging, but also more flexible solution is to create a map of the environment dynamically from real time data of sensors of the AR-system.

Our target applications are Augmented Reality in-car infotainment systems in which a video of a forward facing camera is augmented. Using map data to determine the geometry of the environment of the vehicle is limited by the fact that currently available digital maps only provide a rather coarse and abstract picture of the world. Furthermore, map coverage and amount of detail vary greatly regionally and between different maps.

Hence, the objective of the presented thesis is to obtain the geometry of the environment in real time from vehicle sensors. More specifically, the aim is to obtain the scene geometry by triangulating it from the camera images at different camera positions (i.e. stereo computation) while the vehicle moves.

The problem of estimating geometry from camera images where the camera positions are not (exactly) known is investigated in the (overlapping) fields of visual odometry (VO) and structure from motion (SfM). Since Augmented Reality applications have tight latency requirements, it is necessary to obtain an estimate of the current scene geometry for each frame of the video stream without delay. Furthermore, Augmented Reality applications need detailed information about the scene geometry, which means dense (or semi-dense) depth estimation, that is one depth estimate per pixel. The capability of low-latency geometry estimation is currently only found in *filter based* VO methods, which model the depth estimates of the pixels as the state vector of a probabilistic filter (e.g. Kalman filter). However, such filters maintain a covariance matrix for the uncertainty of the pixel depth estimates whose complexity is quadratic in the number of estimated pixel depths, which causes infeasible complexity for dense depth estimation.

To resolve this conflict, the (full) covariance matrix will be replaced by a matrix requiring only linear complexity in processing and storage. This way, filter-based VO methods can

be combined with dense estimation techniques and efficiently scaled up to arbitrarily large image sizes while allowing easy parallelization.

For treating the covariance matrix of the filter state, two methods are introduced and discussed. These methods are implemented as modifications to the (existing) VO method LSD-SLAM, yielding the "continuous" variant C-LSD-SLAM. In the first method, a diagonal matrix is used as the covariance matrix. In particular, the correlation between different scene point estimates is neglected. For stabilizing the resulting VO method in forward motion, a reweighting scheme is introduced based on how far scene point estimates are moved when reprojecting them from one frame to the next frame. This way, erroneous scene point estimates are prevented from causing the VO method to diverge.

The second method for treating the covariance matrix models the correlation of the scene point estimates caused by camera pose uncertainty by approximating the combined influence of all camera pose estimates in a small subspace of the scene point estimates. This subspace has fixed dimension 15, which forces the complexity of the replacement of the covariance matrix to be linear in the number of scene point estimates.

Zusammenfassung

Um virtuelle Objekte überzeugend in eine reale Szene einzufügen, müssen Augmented Reality-Systeme (AR-Systeme) typischerweise zwei Probleme lösen. Einerseits muss die Bewegung und die Position des AR-Systems in der Umgebung bekannt sein, um eine stabile Positionierung der virtuellen Inhalte relativ zur realen Szene zu gewährleisten und um insgesamt eine korrekte Positionierung zu ermöglichen. Andererseits muss ein AR-System Kenntnis der Geometrie der realen Umgebung besitzen, um virtuelle Inhalte mittels Techniken wie Verdeckungsbestimmung oder umgebungsadaptiver Positionierung sauber in die reale Szene einfügen zu können. Zur Lösung des letzteren Problems haben sich die folgenden beiden Ansätze herausgebildet: Eine einfache Lösungsmöglichkeit ist, *a priori* mit beliebiger Technik ein Modell der realen Szene zu erstellen und dieses Modell dann schlicht im Realbetrieb des AR-Systems zu verwenden. Eine anspruchsvollere, aber auch flexiblere Lösungsmöglichkeit ist, das benötigte Modell der Umgebung dynamisch aus den Sensordaten des AR-Systems zu erzeugen.

Unsere Zielanwendung ist ein AR-System als Teil eines Infotainmentsystems für die Passagiere eines Automobils. Dabei wird der Bildstrom einer vorwärts gerichteten Kamera mit Augmentierungen versehen und den Passagieren des Fahrzeugs angezeigt. Die Verwendung von Kartendaten zur Bestimmung der Geometrie der Umgebung wird durch die Tatsache eingeschränkt, dass aktuell verfügbare Karten die Welt nur in stark abstrahierter Form und sehr grobmaschig darstellen. Weiterhin variiert die Vollständigkeit und Detaillierungsgrad der Karten regional und zwischen unterschiedlichen Karten.

Somit ist die Zielsetzung der vorliegenden Dissertation, die Geometrie der Umgebung in Echtzeit aus Daten der Fahrzeugsensorik zu bestimmen. Im Detail wird der Ansatz verfolgt, dass die Szenengeometrie während der Fahrzeugbewegung durch Triangulation aus den Kamerabildern an verschiedenen Kameraposition (d.h. Stereorechnung) bestimmt wird.

Die Problemstellung, die Geometrie einer Szene aus Kamerabildern bei gleichzeitig nicht (genau) bekannten Kamerapositionen zu bestimmen wird in den (überlappenden) Gebieten der visuellen Odometrie (VO) und Structure from Motion (SfM) untersucht. Da Augmented Reality-Anwendungen enge Latenzanforderungen haben, ist es notwendig für jedes einzelne Kamerabild verzögerungsfrei eine Schätzung der jeweiligen Szenengeometrie bereitzustellen. Weiterhin benötigen Augmented Reality-Anwendungen detaillierte Information über die Szenengeometrie. Dies bedeutet typischerweise, dass die Geometrieschätzung in der Form einer dichten Tiefenkarte bereitgestellt wird, d.h. für jedes Bildpixel wird eine Entfernungsschätzung bereitgestellt. Die Fähigkeit zur latenzarmen Generierung von Geometrieschätzungen findet sich derzeit nur in *filterbasierten* VO-Verfahren. Bei diesen werden die Entfernungschätzungen der Pixel als Zustandsvektor eines probabilistischen

Filters (z.B. Kalman-Filter) aufgefasst. Bei diesen Filtern wird jedoch eine Kovarianzmatrix zur Beschreibung der Unsicherheit der Schätzungen der Szenenpunkte mitgeführt. Da deren Komplexität quadratisch in der Anzahl der Szenenpunkte ist, verursacht dies bei dichter Tiefenschätzung prohibitiv hohe Komplexität.

Um diesen Widerspruch aufzulösen werden wir die (voll besetzte) Kovarianzmatrix durch eine Matrix mit linearer Speicher- und Laufzeitkomplexität ersetzen. Dies ermöglicht die Kombination von filterbasierten VO-Verfahren mit dichter Tiefenschätzung und erlaubt es, solche Techniken effizient auf große Bildgrößen hochzusklalieren und einfach zu parallelisieren.

Es werden zwei Methoden zur Ersetzung der Kovarianzmatrix des Filterzustands eingeführt und diskutiert. Diese Methoden werden aufbauend auf dem (bestehenden) VO-Verfahrens LSD-SLAM implementiert. Diese so resultierende, "kontinuierliche" Variante von LSD-SLAM wird als C-LSD-SLAM bezeichnet. In der ersten Methode wird die Kovarianzmatrix durch eine Diagonalmatrix ersetzt. Insbesondere wird dadurch die Korrelation der Positionsschätzungen der Szenenpunkte vernachlässigt. Um C-LSD-SLAM bei Vorwärtsbewegung zu stabilisieren, wird eine Umgewichtungsregel für die Szenenpunktschätzungen eingeführt. Diese führt die Umgewichtung auf Basis der Bewegung der Szenenpunktschätzungen bei Reprojektion von einer Kameraposition zur nächsten durch. Damit werden fehlerhafte Szenenpunktschätzungen daran gehindert, im VO-Verfahren Divergenz zu verursachen.

Die zweite Methode zur Ersetzung der Kovarianzmatrix modelliert die durch die Unsicherheit der Kameraposen verursachte Korrelation der Szenenpunktschätzungen, in dem der Einfluss aller Kameraposenschätzungen in einem kleinen Unterraum des Vektorraums der Szenenpunktschätzungen approximiert wird. Dieser Unterraum hat die feste Dimension 15, wodurch erzwungen wird, dass der Ersatz der Kovarianzmatrix lineare Komplexität in der Anzahl der Szenenpunkte hat.

1 Introduction

1.1 Motivation

The aim of this research is to provide intuitive means of visualizing information for the passengers of a car by integrating suitable virtual objects into the real world, e.g. by drawing the virtual objects on a live camera feed of the vehicle surrounding and showing this *augmented* camera feed to the passengers. Natural use-cases are e.g. providing navigation directions by drawing navigation arrows onto the street or providing information about nearby locations by displaying points of interest (POIs), cf. fig. 1.

Figure 1: Concept art and prototype of vehicular AR-System. Navigation directions, street names and house numbers as well as POIs are shown

Conventionally, camera images are augmented by just painting the virtual objects over the camera images. I.e. it is assumed that the virtual objects are always in front of the real objects, which means that virtual objects are assumed to never be occluded by real objects. This is an assumption made frequently e.g. when augmenting objects onto a flat surface such as a tabletop. It quickly fails though in complex, cluttered environments such as traffic scenes. Here, drawing virtual content in an occlusion-correct fashion is an important factor for integration virtual content into the real world. Indeed, occlusion is an important component in the depth perception of humans. E.g. navigation directions benefit greatly from the increased intuitiveness provided by navigation arrows drawn in an occlusion-correct fashion.

Hence, a key motivation of this thesis is to find methods to determine which parts of the virtual scene are occluded by or occlude parts of the real scene. In particular, we want to know for each pixel of the virtual scene whether that pixel is in front or behind the corresponding pixel of the camera image.

The virtual scene is generated algorithmically and is thus perfectly known to the Augmented Reality system. Thus, information about the geometry of the real scene needs to be obtained for occlusion.

There are several sources for information on the scene geometry available to vehicular

AR systems. One source is to obtain information from digital maps, that is from data sources outside the vehicle. Digital maps such as OpenStreetMap and Google maps currently feature geometry information such as building footprints or 3D building models. A technical obstacle in this approach is that for accurate usage, the vehicle pose needs to be determined precisely relative to the map, which is difficult with current absolute localization technologies such as GNSS. If implemented successfully, a major advantage of this method is that its range is essentially infinite. On the other hand, this approach is bound to the fidelity and accuracy of the map data: Current digital maps model the real world typically in a highly abstract fashion. For example, most maps are designed for navigation so roads are typically represented as edges in a road graph. That means the two-/three-dimensional road is reduced to an one-dimensional object. It is often impossible to recover the geometry of even the most important structures such as roads and buildings from this abstract representation.

The second source for information on the geometry of the real scene are the vehicle sensors. Due to the current development effort towards autonomous driving, there is a wide variety of sensors and systems available for vehicles for environment perception. In particular, a major portion of these systems aim at perceiving traffic participants, that is *moving objects* in the scene. Due to being designed as safety-critical components and constraints posed by the communication networks within the vehicle, these systems typically provide a schematic representation of the environment. Thus, they cannot be used directly for occlusion determination. In this venue, note that in the AR-system a live camera feed is augmented. This camera can also be used for environmental perception. In particular, it can be used via stereo computation to obtain a *dense distance maps* for the images of the camera, which can then be used for occlusion computation. By reusing the camera images, this approach is highly self-contained, which offers several advantages. In particular, dependency on other sensors is avoided, which would introduce issues such as cross-sensor calibration and communication between different electronic control units (ECUs) in the vehicle, which are complex in a multi-stakeholder mass-production environment.

Hence, the aim of this thesis is to obtain depth information from a monocular camera image stream and to use this depth information for occlusion computation in a vehicular Augmented Reality system. We assume that the image stream being augmented is the same as the image stream used for depth determination, so no additional alignment step is needed. Figure 2 shows a possible processing pipeline for such an Augmented Reality system.

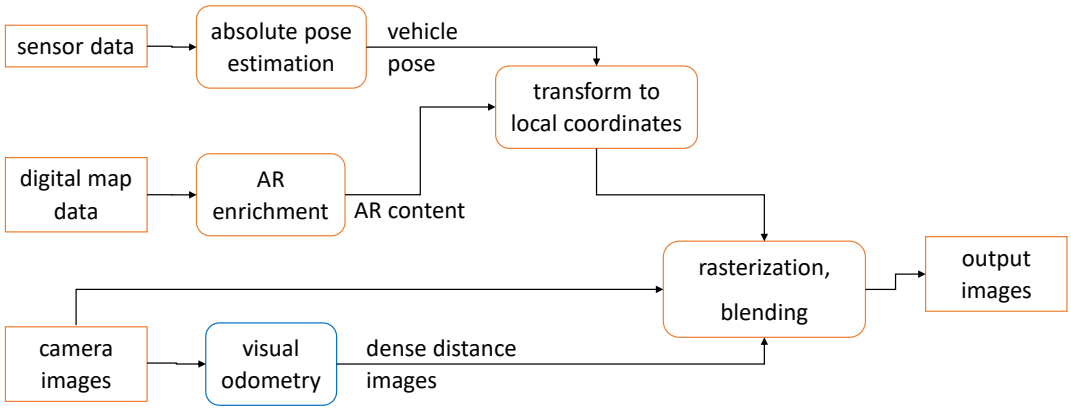


Figure 2: Processing pipeline for a video-based Augmented Reality system incorporating visual odometry for depth estimation and occlusion determination

1.2 Objectives

The objective of this thesis is to investigate and develop systems for the determination of the occlusion relation between real and virtual objects in a vehicular AR system. Such a system generally consists of a component for the depth estimation of the real world pixels and of a component which composes the real image and the virtual scene in an occlusion-correct fashion using data provided by the first component.

In more detail, we have the following objectives in the development of these components.

- Long range: View ranges of several hundreds of meters are common in traffic scenes. Vehicle speeds in urban areas typically range from 10m/s (residential area) to about 20m/s. Hence in order to integrate virtual objects into the scene several seconds in advance, the depth estimation component should be able to estimate object distances up to a range of about 100m. This contrasts the short ranges encountered in the majority of Augmented Reality applications which are designed for providing augmentations in a very small region such as on and above a table or inside a room or a building or generally the immediate surrounding of a mostly stationary user.
- Capability of high speed forward movement: This is the most challenging situation for visual odometry and visual SLAM systems. Our system is always in this situation since the AR system looks in the direction of vehicle movement, so the depth estimation component must be well capable of functioning in this situation.

- Parallelization and efficiency of algorithms: The algorithms used are computationally fairly expensive since they typically involve performing some operation for each pixel of each camera frame. In particular, stereo matching involves performing many iterations for each pixel of each camera frame. The only processors available on vehicle ECUs capable of performing this amount of number crunching and data crunching are automotive-grade GPUs or other devices with limited instruction control flow flexibility such as FPGAs or DSPs. Hence, the algorithms should feature a high level of data independence for parallel processing and the control flow should be largely independent of the data to facilitate efficient usage of vectorized (SIMD) processing units.
- Optimization for determination of occlusion relation: The depth estimation component should be optimized in such a way that errors affect occlusion relations as little as possible. In particular, it should avoid outliers since these introduce highly visible artifacts in occlusion calculation. Furthermore, the composition component should compose the virtual scene and the real image in such a way that the effects of misocclusion are de-emphasized.

Initially, a main focus was on how to achieve occlusion with given dense depth information (depth maps) and how to enhance these depth maps in such a way that occlusion can be determined well. However, the key problem with this approach is the fact that trying to enhance depth maps is essentially just reprocessing of existing data. By feeding additional data (apart from the bare depth information) such as the color information from camera images into the enhancement process, results can be improved with regard to regularity issues such as smoothness and edge fidelity. Ultimately, however there is a limit on how much existing depth maps can be enhanced by post-processing, and this limit is given by the quality of the input depth maps. In particular, range accuracy may be improved e.g. by averaging only up to the limit given by the noise level of the input depth maps and the number of samples that can be aggregated into an output value. As accuracy at long range is particularly important for urban scenes, this limit needs to be circumvented.

Hence, the main focus of this work is on how to extract as much depth information as possible from the camera images. In particular, this means triangulating the object positions directly from the camera images to obtain sufficiently accurate depth maps even in long-range situation.

These high-quality, raw depth maps may then be used in succeeding processing steps such as densification and enhancement, with the ultimate step being usage in occlusion computation.

1.3 Main contributions

The first main contribution is the revival of the filter-based visual odometry (VO) methods and modification of this concept in such a fashion that our methods can scale well up to large numbers of scene points: Filter-based VO methods traditionally treat uncertainty in the position estimates of the scene points by maintaining a covariance matrix in the style of a Kalman filter. The number of entries of this covariance matrix is quadratic in the number of scene points. Thus filter-based VO methods do not scale well to large numbers of scene points, so it is highly impractical to do depth estimation for large numbers of pixels. In particular, real-time dense depth estimation is currently virtually impossible for anything larger than tiny image resolutions. The method developed here, C-LSD-SLAM, is derived from the existing method LSD-SLAM: Large-Scale Direct Monocular SLAM[Engel et al., 2014] (LSD-SLAM). Note that LSD-SLAM is a keyframe-based SLAM method. For scene point triangulation and camera tracking, it employs direct image matching and alignment on the images, which are restricted to the semi-dense image regions with highest intensity gradient. The desire to obtain a depth map for each frame instead for each keyframe in LSD-SLAM (i.e. **c**ontinuous operation and output) motivated evolving LSD-SLAM into C-LSD-SLAM. This essentially meant making each frame into a keyframe and transforming the method into a filter-based method.

Note that this method treats the uncertainty of each point depth estimate individually without any treatment for the correlation in the scene point depth estimates caused by misestimation of the camera movement. That is, instead of a densely populated covariance matrix for the depth estimates, the covariance estimate is a diagonal matrix, which can be considered as an approximation of the "true" densely populated covariance matrix. The main advantage of this approximation is that the diagonal matrix has *linear* complexity in the number of estimated points, which is much less complex than the *quadratic* complexity of a densely populated matrix. This way, the computational complexity is essentially linear in the number of scene point depth estimates, which is the key mechanism allowing us to efficiently scale filter based visual odometry methods up to large image resolutions with large number of tracked scene points, which in turn improves robustness and accuracy. A summary of the problem structures of the different strategies for visual odometry is given in figures 3 and 4.

In a subsequent step, we investigate how to approximate the densely populated covariance matrix in such a way that the most significant effects of pose uncertainty are captured by the approximation while still featuring linear complexity in the number of estimated scene points.

As a second main contribution, the introduction of C-LSD-SLAM shows that the concept of a filter-based direct method for visual odometry is feasible and effective. A majority

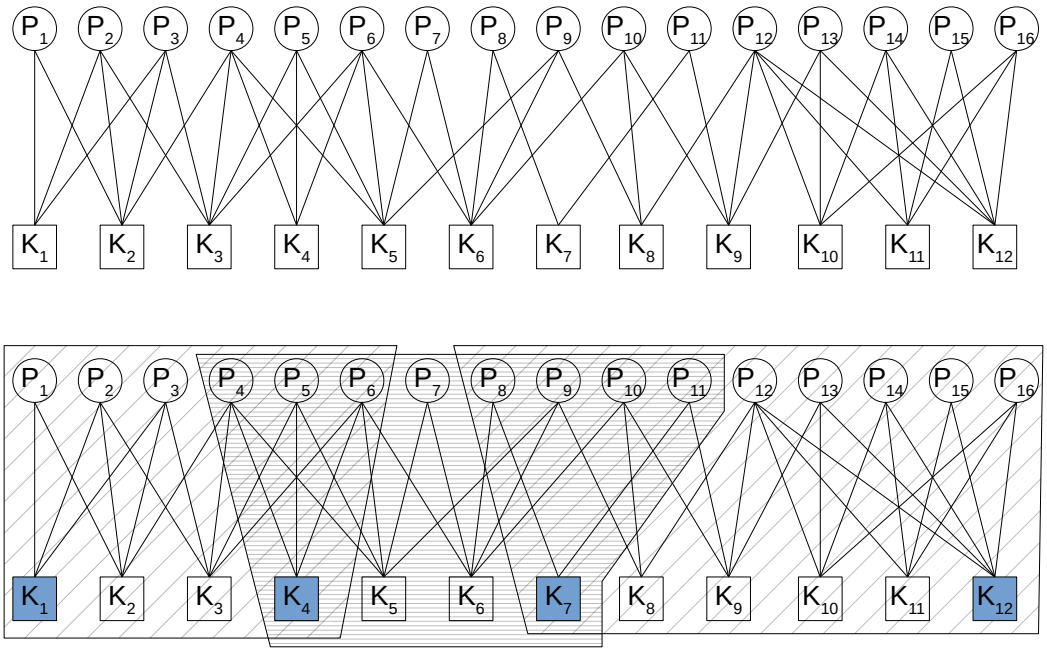


Figure 3: Sparsity structure for bundle adjustment (top) and keyframe-based visual odometry (bottom). The P_i represent the (unknown) scene point positions. The K_i represent the (not exactly known) camera poses. A line is drawn for each scene point observed in a camera image. Each line gives a summand in the log-likelihood function used for optimization. Thus when solving the nonlinear optimization problem via linearization (e.g. in the Gauss-Newton algorithm), the lines give the sparsity structure of the system matrix. In traditional bundle adjustment, the whole optimization problem is solved via offline global optimization. Keyframe-based visual odometry methods divide the sequence of images into smaller sections which overlap at *keyframes* (marked in blue). Optimization is done for each section separately, which allows obtaining intermediate results after each keyframe in online processing.

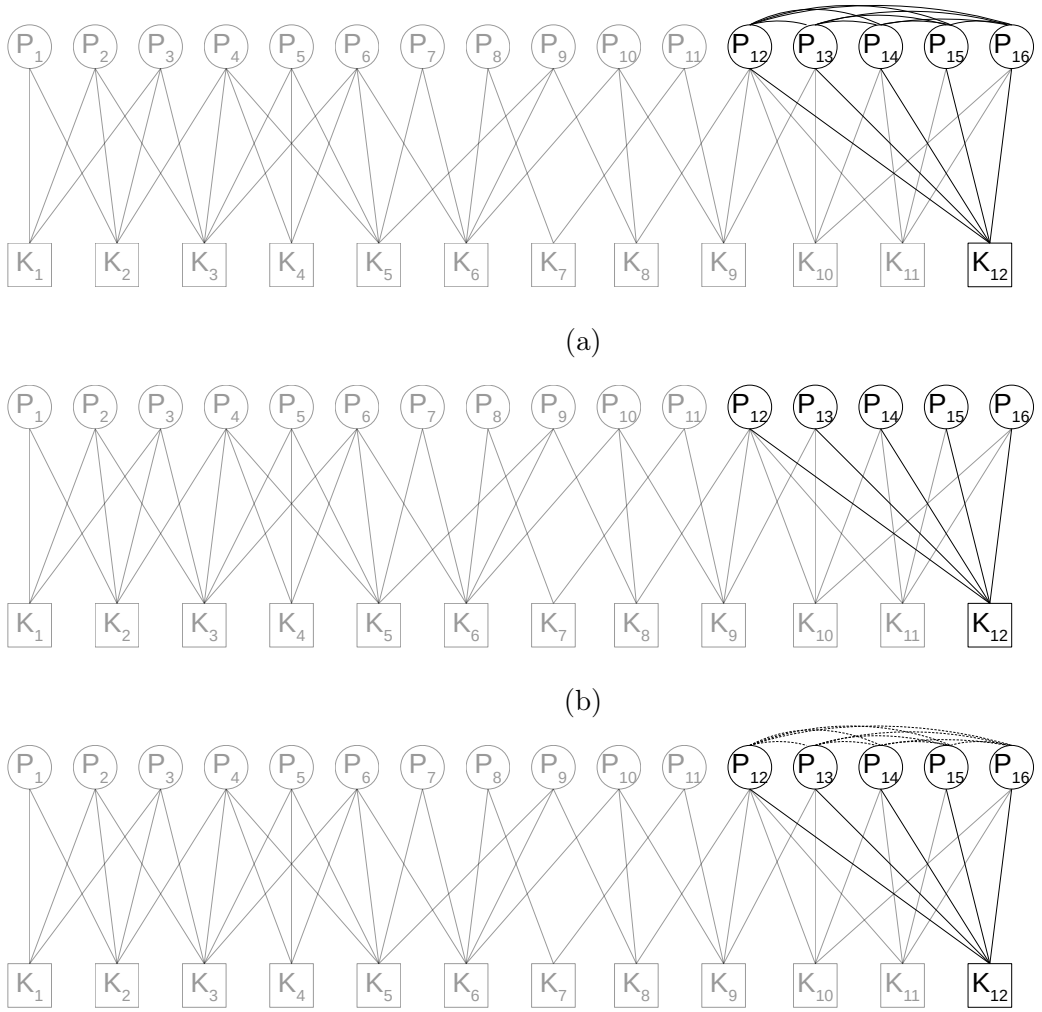


Figure 4: Sparsity structure of filter-based visual odometry methods (notation used as in fig. 3): (a) Orthodox probabilistic filter (e.g. [Davison et al., 2007]) (b) diagonal C-LSD-SLAM (c) extended C-LSD-SLAM. Eliminating the previous camera poses and all scene points not visible in the current image (drawn in grey) destroys the sparsity structure from fig. 3 since in bundle adjustment, relations between different scene points are only possible via an intermediate camera pose whereas here, these relations must connect scene points directly and they usually connect virtually all pairs of scene points. Hence the computational complexity per frame is approximately quadratic in the number of scene points visible in that frame. Diagonal C-LSD-SLAM approximates the relations between different scene points by zero, thus giving an approximately linear complexity. Extended C-LSD-SLAM approximates the relations between different scene points with a structure of linear complexity in the number of number scene points in the current frame.

of VO methods are feature based, meaning that an essential step of image preprocessing is the extraction of (usually point-like) features from the image. This serves as a way of creating an abstract representation of the images. The central routines of the VO method (association of different images via matching, triangulation, inference of the camera position, etc.) then all operate on features. In particular, this means that the reconstructed geometry of the scene visible to the camera is represented as a collection of features with estimated position, that is as a sparse point cloud. In contrast, direct methods skip the (often computationally expensive) step of feature extraction and operate directly on the image intensities. While this exposes most of the VO pipeline to image defects such as e.g. image noise, it provides a natural path to a *dense* representation of the scene in front of the camera, which is a key motivation for us provided by our intended application.

To our knowledge, filter-based methods have exclusively been feature-based until now. The reason for this is the fact that filter-based methods traditionally do not scale well to large number of estimated scene points as outlined above, which motivates restricting the estimation of scene points to a sparse set of pre-selected, "valuable" points, which are the feature points. The efficiency improvements given above to the concept of filter-based VO allow us to completely bypass this constraint on the number of estimated scene points and to obtain the filter-based direct method C-LSD-SLAM.

The third main contribution is to devise a reweighting strategy for the confidence of scene point estimates in filter-based VO which gives C-LSD-SLAM the capability of working well in situations of fast forward motion. Such situations are challenging to VO methods (in particular to monocular VO methods) since they involve strong nonlinearities near the epipolar direction, which are directly in the image center in these situations.

The final main contribution is investigating the application of visual odometry to the problem of drawing augmentations into the video stream of a forward facing vehicle mounted camera in an occlusion correct fashion. Here, we analyse whether the range accuracy of distance estimates obtained from monocular visual odometry is sufficient to solve this problem and investigate how to densify the (possibly semi-dense) depth maps provided by a VO method in a robust fashion.

1.4 Outline

The text is structured as follows. Sections 2 and 3 give brief introductions of visual odometry within visual navigation and surveying and relate our contributions to the state of the art.

Section 4 introduces basic concepts, in particular to fix notations. In section 5, we determine the statistical limits for estimating depth near the singularity at the image

centre with forward-moving monocular cameras and compare them to binocular systems.

In section 6, the main method C-LSD-SLAM is introduced and discussed in detail. Besides the design of C-LSD-SLAM as a filter-based method capable of continuously outputting depth maps, the implementation of C-LSD-SLAM, the mitigation of instability in forward motion and the performance of C-LSD-SLAM are presented.

As C-LSD-SLAM approximates the information (inverse of covariance) matrix of the filter state with a diagonal matrix, section 7 investigates the concept of finding an approximation of the information matrix which better models the joint uncertainty in the scene point estimates caused by camera pose uncertainty.

In section 8, C-LSD-SLAM is evaluated. In particular, the two variants of C-LSD-SLAM dubbed *diagonal C-LSD-SLAM* (as described in section 6) and *extended C-LSD-SLAM* (as described in section 7) are compared and discussed.

Finally, in section 9, we investigate methods for obtaining dense depth maps from the semi-dense depth maps provided by C-LSD-SLAM.

1.5 Abbreviations & definitions

AR	Augmented Reality.
DSP	Digital signal processor.
ECU	Electronic control unit. An embedded system performing a certain function in a vehicle.
FoV	Field of view.
FPGA	Field-programmable gate array. A type of programmable logic device.
GNSS	Global navigation satellite system.
GPS	Global Positioning System. A GNSS system operated by the USA. As GPS is the oldest GNSS system, GPS is often used synonymously with GNSS.
IMU	Inertial measurement unit. A combination of a triaxial gyroscope and a triaxial accelerometer, often in a single sensor package.
LSD-SLAM, C-LSD-SLAM	LSD-SLAM[Engel et al., 2014] is a visual SLAM method. C-LSD-SLAM was developed from LSD-SLAM as part of the work presented here, cf. section 6.
POI	Point of interest.
Pose	Position and orientation of an object (e.g. a camera or a robot). Poses are typically formulated as a Euclidean transformation relative to a reference frame.
RGB-D camera	A combination of an RGB (color) camera and a depth sensor such that the depth sensor provides depth information for the RGB image. Implementations may use e.g. a time-of-flight camera or a structured-light 3D scanner for depth sensing.
SfM	Structure from motion, cf. e.g. section 3.1.
SIMD	Single-instruction, multiple data. This is a class of parallel processing characterized by performing the same operation (instruction) for multiple items (data).
SLAM	Simultaneous localization and mapping, cf. e.g. section 3.1.
VO	Visual odometry, cf. e.g. section 3.1.

2 Introduction to depth estimation and visual navigation

2.1 Animal vision

Many animals have some form of vision. As vision may provide detailed information of the environment over a wide range of distances, it is often an important sense. To that end, visual systems feature a large and complex amount of specialized machinery. For example, it is known that the retinas found in mammals feature both light-sensitive structures and neuronal structures for initial processing. This initial processing already performs extraction of "interesting" components of data such as temporal changes (e.g. due to movement), edge and contrast enhancement.

It is known that mammals and birds both have the capability to perform stabilization of their eyes hardwired into their nervous system via reflexes. In mammals, this is called the vestibulo-ocular reflex. At detection of head movement by the vestibular apparatus in the inner ear, this reflex quickly rotates the eyes in the opposite direction such that overall, the images on the retinas are stabilized.

Birds stabilize their vision by holding their head steady via movement of the neck. Moving the neck is required by the fact that most birds can not move or can only slightly move their eyes. This method gives rise to the head bobbing of e.g. pigeons and chickens when walking on the ground: When walking, the head stays absolutely in the same position while the body moves forward. This means that relative to the body, the head moves backwards. The neck cannot stretch infinitely to the back, so at a certain point the bird will quickly move or "bob" its head forward to a new, stationary position. This method for head stabilization is highly accurate. A well-publicized way of demonstrating this is to pick up a chicken and move its body (in a sufficiently small area) around, whereupon its head will typically stay in exactly the same spot despite the movement of the body.

Animals use vision to both perceive their environment (often in a three-dimensional fashion) and to navigate within their environment. As cameras capture the same information as the vision of animals, it should be possible to also use camera information in a similar fashion. This problem is addressed in the fields of visual odometry, structure from motion and visual SLAM. Note however that while by now, it is possible to compute highly accurate geometrical data from camera images, the algorithms used for this purpose are typically vastly less robust than the perception of animals.

2.2 Brief history of stereo triangulation, visual odometry and structure from motion

2.2.1 Stereo and triangulation

Triangulation has been used by humans as a means for accurately determining distances for a very long time. The necessity and ability of measuring distances precisely can be traced to surveying and construction efforts of ancient cultures. Indeed, the basic tenets of geometry were already established by the time of Thales in early antiquity. Antique geometers strived to invent increasingly more powerful methods, in particular the systematic application of deduction, to be able to solve highly non-trivial questions such as Archimedes' determination of the ratio of the volume ball and the volume of a circumscribed cylinder. Antique mathematics culminated in Euclid's elements, which provided an axiomatic framework for classical, *Euclidean* geometry and codified the concept of a formal proof. In particular, the notion of dividing mathematical texts into *definitions*, *theorems* and *proofs* has remained unchanged since then with the effect that the structure of modern texts bears a striking resemblance to Euclid's elements.

Antique techniques relied heavily on geometrical constructions to solve problems, which can be traced to the fact that the numeric systems (e.g. Roman numerals) prevalent at the time could essentially only perform addition and subtraction in an efficient way. The systematic development of efficient arithmetic methods based on the decimal system during the Middle Ages caused a shift from *construction* to *calculation*, with extensive results on trigonometry being obtained in the medieval Islamic world. The subsequent evolution of mathematical methods necessary for triangulation is mainly centered on the introduction of the usage of overdetermined systems to decrease errors in the 18th century and, since then, an ever-increasing collection of numerical and statistical machinery to describe and solve complex systems.

For most of history, surveying was the main driver for developing triangulation techniques. Large surveying campaigns for mapping whole countries in the late 18th and early 19th century are perhaps the most famous applications of triangulation in surveying. The industrial revolution and its numerous construction efforts increased demand for surveying drastically. Finally, the end of the 19th century saw the introduction of photography sparking the beginnings of photogrammetry.

2.2.2 Photogrammetry, structure from motion and visual odometry

Basically, the usage of cameras for obtaining geometrical information in photogrammetry yields the same kind of information as e.g. a theodolite as both are fundamentally devices

for measuring angles. The qualitative difference (aside from the typically lower angular resolution of cameras) is the fact that cameras are capable of performing many angular observations *at once* and, in the case of e.g. video cameras, are capable of performing many such collective measurements in a short time. In short, cameras are a means to very quickly (and often inexpensively) obtain much information on the captured scene.

However, the usage of cameras as measurement devices raised several issues with some of them unique to photogrammetry and others being vastly amplified versions of problems also present in surveying:

- Automation: The vast data acquisition rate of cameras is a powerful motivator for developing automated solutions.
- Camera calibration: Cameras are not perfect angular measurement devices. The mapping from view ray directions to positions on the image differs from the ideal mapping due to various effects such as distortion by the optics caused by various aberration effects and misalignment and deformation of the components of the optics due to manufacturing tolerances and / or mechanical stress due to a wide selection of sources such as mounting stress, mechanical shock, vibration, thermal deformation, gravity and others. To resolve this discrepancy, this deviation from the ideal mapping is determined in a step called *calibration*. The knowledge of the deviation can then be used to remove the effects of the deviation computationally from the captured images in a step called *rectification* of the images.
- Point association: To perform triangulation, a scene point needs to be found in two or more images. The result of this task is the association of image points in different images as being the image of the same scene point. This association should yield many correct associations while yielding only few misassociations. Geometrically, associations should be as precise as possible. The main approach here is to extract *features* from the image, that is (in a certain sense) easily recognizable points, and then solving the association problem on the extracted feature points. While extraction of features also makes sense from the point of view of data reduction, there is also the approach of *direct matching* in which image patches are compared directly to determine whether they correspond to the same scene part or not.
- Numerical treatment of large nonlinear systems of equations: These arise e.g. in the context of bundle adjustment, where all estimated parameters (scene point positions, camera poses, calibration parameters and others) are fitted to the measurement data. Another aspect of this problem is to find good initial estimates, which leads to the technique of combining exact techniques for determining camera alignment such as the 5-point and 8-point algorithm (cf. e.g. [Nistér, 2004]) with the iterative outlier rejection method RANSAC.

- Tolerance against outliers: There are various components in a photogrammetry pipeline that can be a source of wholly incorrect results, also called outliers. The most important and most well-known source of these are point association errors as they happen frequently and are also unavoidable due to the fact that different real structures can look exactly the same, thus fooling any appearance-based association technique. A common technique to deal with outliers is to use robust objective functions in optimization as these objective functions are less sensitive to outliers than non-robustified objective functions such as the quadratic loss function used in least-squares optimizations.

The advent of digital cameras and a large increase in available computational resources at the end of the 20th and beginning of the 21th century has led to greatly increased adoption of and interest in photogrammetric techniques in particular in areas such as computer vision, robotics and 3D modelling. Here, a particular advance for 3D reconstruction has been the introduction of Semi Global Matching (SGM) [Hirschmüller, 2005] in 2005, which has made dense reconstruction from stereo images practical, sparking a large variety of variants and implementations.

3 Related work

3.1 Overview over Structure from motion, visual odometry and visual SLAM

This section gives a short overview over the problem of determining both the scene geometry and the camera poses from camera images (we call this the SfM problem in the sequel, cf. below for the associated terminology) and over associated terminology.

Note that the individual problems of determining the scene geometry from known camera poses (solved by triangulation) and of determining camera poses from a known scene geometry (solved by resection) are simpler than the problem of solving both subproblems in combination. As the solution of one subproblem affects the solution of the other subproblem and vice-versa, the SfM problem can be considered a chicken-egg problem. In fact, since one cannot distinguish between a "real" scene and an upscaled or downscaled model of the scene from camera images alone, the problem of absolute scale determination is actually unsolvable in SfM without some additional absolute scale reference.

The SfM problem lies at the intersection of the fields photogrammetry, computer vision and robotics. The different points of views have resulted in somewhat different approaches and different terminology, even though the large majority of knowledge and techniques is shared.

The approach of *Structure from Motion* (SfM), which we use here to denote the SfM problem has its roots in photogrammetry. A major aspect of photogrammetry is large-scale surveying (e.g. aerial surveying or surveying of complex objects). Thus typically, methods for processing large collections of unordered images in an offline fashion with a particular focus on accuracy are found under the notion of SfM. The other direction is from the side of robotics and computer vision, giving rise to visual odometry (VO) and visual SLAM. Here, the origins are the desire to obtain the pose of a robot (or other moving object) as well as three-dimensional perception of the environment from the images of a camera mounted on the robot. Hence, VO methods often assume that the images are ordered sequentially by their location in the image stream provided by the camera. Furthermore, the capability of online and real time operation is often found in VO methods caused by the real time nature of robot operation. Visual SLAM is the extension of VO by the SLAM concept, that is by the objective to create a *global* map and to localize the robot within this global map.

There are several other characteristics by which approaches to the SfM problem can be classified:

- Direct methods vs. feature based methods

- Filter based methods vs. keyframe based methods vs. other approaches (e.g. deep learning based methods inspired by biology)
- Monocular vs. stereo methods
- Offline (batch) methods vs. online or real-time methods
- Matching methods, feature descriptors
- Methods for loop closure detection and place recognition in SLAM methods

As our focus is on online direct filter-based monocular methods, the first three characteristics are the most relevant to us. For a more thorough discussion, confer e.g. the surveys [Fuentes-Pacheco et al., 2015, Younes et al., 2016, Yousif et al., 2015].

3.2 Treatment of combined camera pose and scene point uncertainty

If the camera poses are known, the observation and triangulation of scene points can be performed for each scene point individually. If the camera poses are not known, the estimation of the camera poses affects the estimation of scene point locations and vice versa, thus requiring a combined approach.

From the viewpoint of computational complexity, the pose uncertainty affects each scene point estimation, hence this mechanism potentially connects each scene point estimate with each other scene point estimate. As the number of scene point is typically large (or is desired to be large to improve noise rejection in the method), this often contributes essentially to the computational complexity of a VO method.

There are two main branches for the treatment of the combined uncertainty of camera poses and scene points:

The first main branch is *probabilistic filtering* of the scene points and camera poses. In this approach, uncertainty of camera poses and scene point estimates is maintained in a combined fashion in a probabilistic filter such as an extended Kalman filter or a particle filter.

Due to its conceptual simplicity, probabilistic filtering is featured in many older VO methods, cf. [Davison, 2003, Davison et al., 2007, Eade and Drummond, 2006, Holmes et al., 2008, Kwon and Lee, 2010].

The main disadvantage of methods based on probabilistic filtering is their high computational complexity: Methods based on the Kalman filter maintain a covariance matrix for the scene point estimates and the size of this densely populated matrix is quadratic in the number of scene points, yielding at least quadratic complexity in the number of

scene points. Methods based on particle filtering often need many particles to capture the underlying distribution.

A major innovation in this branch is the combination of a Kalman filter with a particle filter in FastSLAM and its successors [Montemerlo et al., 2002, Montemerlo and Thrun, 2007, Lee et al., 2016a, Lee et al., 2016b, Shiguang et al., 2017]. It is based on the observation mentioned above that for fixed camera poses, the scene point observations are independent. I.e. the covariance matrix of the scene point estimates is diagonal, yielding linear complexity in the number of scene points. The uncertainty in the camera pose trajectory is modelled via a particle filter. Each of these particles has its own collection of scene point estimates, yielding a complexity of $O(mn)$, with n the number of scene points and m the number of particles. Hence, the computational efficiency depends mainly on the number of particles required for modelling the camera trajectory.

The observation that under certain conditions, the covariance matrix of an extended Kalman filter can be replaced by a diagonal matrix, is crucial to this text: It facilitates great reductions in computational complexity and thus serves as a main starting point for the research presented in this text.

The second main branch for the treatment of the combined uncertainty of camera poses and scene points is *local bundle adjustment*. Bundle adjustment has its origins in photogrammetry. It formulates the problem of determining the camera poses and scene point positions as a (global) optimization problem. As all relations are between a camera pose and a scene point (and in particular not between different scene points), the problem can be kept sparse at all times [Triggs et al., 1999]. This allows highly efficient implementations. As much of the statistical properties of the problem can be encoded in the objective function and as bundle adjustment aims to find an optimal solution to this optimization problem, highly accurate results can be obtained by bundle adjustment. In visual odometry, bundle adjustment is developed into *local bundle adjustment*. Here, the camera trajectory is divided into short sections (with the start of a new section usually marked by a *keyframe*) and bundle adjustment is performed locally on the sections. This way, the accuracy of bundle adjustment can be leveraged while keeping the individual problem size to the size of the sections and allowing incremental (and online) estimation by simply adding sections successively to the end of the trajectory.

After an influential analysis[Strasdat et al., 2010], which showed that methods based on local bundle adjustment generally outperform methods based on the Extended Kalman filter regarding the ratio of accuracy to computational cost, methods based on local bundle adjustment have become much more popular in visual odometry, with prominent examples being [Klein and Murray, 2007, Forster et al., 2014, Mur-Artal et al., 2015, Engel et al., 2014].

In addition to these two main branches, there are lesser-known branches for the treatment

of the combined uncertainty of camera poses and scene points formed such as biologically inspired methods (e.g. [Milford et al., 2004]) as well as methods employing deep neural networks [Fanani et al., 2017].

3.3 Forward motion

Movement in the view direction is well-known to be problematic in monocular visual odometry. The source of these problems is the fact that in these situations, errors in the estimation of the epipolar direction can cause large errors in the position triangulation of scene points near the epipolar direction. In fact, the objective function for the estimation of the epipolar direction often features multiple singularities near the true epipolar direction[Oliensis, 2005, Vedaldi et al., 2007, Chiuso et al., 2000].

While [Vedaldi et al., 2007] suggests to regularize the problem by constraining the absolute size of the inverse depth of scene point estimates and shows that this makes objective functions become continuous, we follow the suggestion provided e.g. in [Oliensis, 2005] to simply increase the number of tracked scene points. This essentially increases the signal to noise ratio for the camera pose estimation, which increases the reliability of *all* components.

3.4 Occlusion in Augmented Reality

Traditionally, virtual objects are simply drawn over the real scene in Augmented Reality. This is perfectly acceptable as long as all virtual objects are strictly in front of their background. As soon as a virtual object is partially or totally occluded by real objects, immersion and in particular the distance perception for virtual objects is impaired. The latter is especially important for Automotive Augmented Reality applications since correct distance cues are crucial for e.g. interpreting navigation directives presented in the form of navigation arrows correctly and easily.

The resolution of occlusion in Augmented Reality requires to determine for each pixel of the virtual scene whether that pixel is in front or behind the real scene. Since the geometry of the virtual scene is known perfectly as it is generated synthetically, occlusion handling requires knowledge about the geometry of the real scene, in particular its distance relative to the viewer.

There are two approaches for obtaining information on the geometry of the real scene, called the *model-based* and the *depth-based approach*, cf. e.g. [Shah et al., 2012]. In the model-based approach, a 3D model of the real scene is available a-priori, making this ideal for controlled environments without too high complexity. In the depth-based approach,

the Augmented Reality device infers the geometry of the real scene in real time from sensor data. The latter approach combines well with the concept of SLAM: This way, the device maps its surrounding continuously and uses the obtained 3D map for occlusion.

A technique for enhancing low quality or low resolution information on the scene geometry is to use detection of contours in a camera image to increase the edge fidelity of the contours.

Furthermore, the aspect of automatic object placement interacts with occlusion in Augmented Reality. If there is flexibility in the choice of object positions, clever placement strategies may mitigate problems such as occlusion with real objects, self-occlusion of virtual content and bad visibility of virtual objects and cluttering of the scene with virtual content. This problem has been studied extensively in the context of *view management*, with a key objective of finding good placements for annotation to real or virtual scenes.

3.4.1 Model-based

In [Fuhrmann et al., 1999], each moving object of the scene is tracked with a separate tracking module. Previously obtained models of all real objects in the scene are then used for performing occlusion. A similar approach is applied in [Frikha et al., 2016] in the context of AR surgical training for integrating a tracked real instrument into a virtual medical scene.

[Hayashi et al., 2005] proposes a method detecting dynamic objects in the real scene. The model of the real scene includes a set of images of the real scene called keyframes covering the whole scene. In real-time, the camera image of the AR device is compared with a synthetic image for that camera pose generated from a nearby keyframe. Since dynamic objects appear as discrepancies (background subtraction), this is used for extracting the dynamic objects. Their 3D position is then obtained from stereo matching along their contours.

3.4.2 Depth-based

In the system proposed in [Kanbara et al., 2000], a stereo camera is mounted on an HMD to provide occlusion via stereo computation. To decrease computational load, depth estimation is limited to a region surrounding the virtual objects.

An early system for performing (offline) occlusion by foreground objects in a video sequence is described in [Lepetit and Berger, 2000]. The contours of the occluding real objects are outlined manually for a subset of the video frames (i.e. keyframes) and the system uses this information to perform occlusion in the intermediate frames via monocular SfM techniques.

Occlusion handling via the depth-based approach has by now been present in consumer systems for several years. The reason for this is that mapping the surrounding is a component essential to many AR applications. Since targeting the mass market also means providing ease of adoption by developers, such AR systems often already provide a component for mapping, which already provided a certain level of occlusion capability based on the depth-based approach. Well known systems include Microsoft's Kinect/KinectFusion[Izadi et al., 2011] based on a structured-light RGB-D sensor, with a huge amount of applications built on top of this platform, and its later HoloLens which combines a see-through HMD with a time-of-flight RGB-D sensor.

3.4.3 Edge enhancement

The core concept underlying the usage of scene edges is that in certain situations, the occlusion boundaries consist of scene edges. This was first explored in [Berger, 1997] with the observation that if the volumes of real and virtual objects do not intersect, the boundaries regions of occlusion and non-occlusion between real and virtual scene consist entirely of contours of the real objects and of contours of virtual objects. The presented approach then infers the regions of occlusion from the binary occlusion relation on contours obtained from SfM.

In the approach proposed in [Tian et al., 2010], a single real object is allowed to occlude the scene. The occluding object is marked by the user and its silhouette is then tracked and used for occlusion.

In [Du et al., 2016], an RGB-D camera is used for providing an initial estimate of occlusion boundaries. This estimate is then refined by searching for the corresponding contours in the color image via an implicit "edge-snapping" method.

The problem of non-sharp occlusion boundaries is addressed in [Hebborn et al., 2017]. The proposed method uses data from a depth sensor to provide an initial estimate of occlusion. The refinement of this estimate generates translucency values for boundary pixels (alpha matting) to provide translucent, "soft" occlusion for unsharp occlusion boundaries. A refinement of this method is given in [Kremer, 2017], which improves the occlusion estimation by employing optical flow to be able to reuse information across multiple frame and to provide temporal consistency.

3.4.4 View management

A view management system for labels in a mixed reality scene is introduced and analyzed in [Bell et al., 2001]. Label placement near the 3D object centres is compared with placement on or near the visible surfaces of the object. It is shown that combining the latter with

temporally stable label placement and additional techniques such as adaptive switching between internal and external labels yields highly intuitive labels and eases association to the real objects.

[Makita et al., 2009] proposes a multi-user view management system for indoor Augmented reality applications. The labels are placed dynamically but label positions are the same for the different users. The systems minimizes occlusion among labels and collisions with users jointly for the points of view of all users.

In the context of video-based Augmented Reality browsers, an approach for view-management based on image analysis is introduced in [Grasset et al., 2012]. The described method analyses the camera image to be augmented for "interesting" regions (i.e. regions with many features such as edges) and avoids placing labels in interesting regions to avoid covering real objects with labels.

3.4.5 Outdoor methods

As current low cost depth sensors typically do not have suitable range or cannot handle outdoor lighting conditions, most methods for providing occlusion in outdoor AR applications follow the model-based approach for occlusion.

[Kasapakis and Gavalas, 2017] intersects view rays with building footprints provided by OpenStreetMap to obtain a 2D variant of occlusion in urban scenarios. A similar method is used in [Galatis et al., 2016] in conjunction with manually created building footprint data to provide occlusion in a cultural heritage site.

Approximate 3D occlusion is provided in urban scenarios in [Kasperi et al., 2017] from OpenStreetMap data. The proposed method guesses the 3D geometry of buildings from building footprint data and the number of building levels provided by OpenStreetMap. The generated 3D geometry is then used for occlusion.

4 Basic concepts

In this section, we review some basic concepts to provide a clean introduction and to fix notation.

4.1 Geometric concepts

4.1.1 Projective spaces and homogeneous coordinates

We shall use real projective spaces $\mathbb{R}P^n$ for $n \geq 1$. Such a projective space can be modelled as the set of lines in $\mathbb{R}^{n+1}$ passing the origin. More formally, we can model $\mathbb{R}P^n$ as the quotient set $(\mathbb{R}^{n+1} \setminus \{0\})/\sim$, where the equivalence relation $\sim$ is defined for $p, q \in \mathbb{R}^{n+1} \setminus \{0\}$ as follows.

$$p \sim q \qquad :\Leftrightarrow \qquad \exists \lambda \in \mathbb{R} \setminus \{0\} : q = \lambda p$$

Note that for given $p \in \mathbb{R}^{n+1} \setminus \{0\}$, its equivalence class $[p] = \{\lambda p \mid \lambda \in \mathbb{R} \setminus \{0\}\}$ is the line in $\mathbb{R}^{n+1}$ passing through p and the origin point 0, excluding the origin. Hence, we recover our original model.

Given some $p \in \mathbb{R}^{n+1} \setminus \{0\}$, its equivalence class $[p]$ is called a *point in projective space*. The element $p \in \mathbb{R}^{n+1}$ is called a *homogeneous coordinate* for the point $[p]$. Note that $[p] = [\lambda p]$ for all $\lambda \in \mathbb{R} \setminus \{0\}$, i.e. a homogenous coordinate representing a projective point can be rescaled arbitrarily without changing the represented projective point. This fact is often used for rewriting homogeneous coordinates.

Note that we can embed the (affine/Euclidean) n-dimensional space $\mathbb{R}^n$ into $\mathbb{R}P^n$ via the following map.

$$\iota : \mathbb{R}^n \hookrightarrow \mathbb{R}P^n$$
$$x \in \mathbb{R}^n \mapsto \begin{bmatrix} x \\ 1 \end{bmatrix}$$

We will use this embedding canonically. In particular, we will usually assume that the affine space extends continuously to the points at infinity, cf. e.g. [Triggs et al., 1999, section 2.2].

The space $\mathbb{R}P^2$ is also called the *(real) projective plane*. The lines of the projective plane are defined as follows. Suppose given a non-zero linear map $l : \mathbb{R}^{n+1} \to \mathbb{R}$. From l, we obtain a line by

$$g_l := \{[p] \mid p \in \mathbb{R}^{n+1} \setminus \{0\} : lp = 0\}.$$

The collection of all such lines gives the set of lines on the projective plane. Note that if we restrict these lines to the image of the inclusion ι, we obtain exactly the set of lines in $\mathbb{R}^2$.

An important property of projective lines is the fact that any two distinct projective lines intersect at exactly one point. This contrasts the behaviour of distinct lines on the affine plane, which either intersect at one point or are parallel.

4.1.2 Orthogonal projections

Suppose given a real Hilbert space V. For a closed[1] subspace $W \leqslant V$, we denote by π_W the orthogonal projection to W. Likewise, $\pi_{W^\perp}$ is the projection of the orthogonal complement $W^\perp$ of W in V. For a vector $0 \neq v \in V$, we write π_v for $\pi_{\langle v \rangle}$ and $\pi_{v^\perp}$ for $\pi_{\langle v \rangle^\perp}$.

Remark 1. If $\|v\| = 1$ in the above, we have

$$\begin{aligned} \pi_v(w) &= v\langle v, w \rangle \\ \pi_{v^\perp}(w) &= v - v\langle v, w \rangle \end{aligned}$$

for all $w \in V$.

4.2 Camera systems and models

4.2.1 Camera models and rectification

In this text, we assume that cameras observe the scene by central projection to a point, called the camera center. We use the pinhole camera model as well as the spherical camera model to describe the imaging process for such cameras.

Definition 2 (pinhole camera model)**.** Suppose given a point $p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ in the camera coordinate system. Then p is projected by a pinhole camera to the following point on the image plane.

$$f(p) = \begin{pmatrix} u \\ v \end{pmatrix} := \begin{pmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{pmatrix}$$

[1] W is always closed e.g. if W is finite-dimensional

The parameters f_x, f_y are the *focal lengths* of the camera[2]. The point $\begin{pmatrix} c_x \\ c_y \end{pmatrix}$ is the *principal point* of the camera. The z-direction is distinguished being the direction of the *optical axis* of the camera. For real cameras, this is the direction the camera is looking at.

An important property of the pinhole camera model is the fact that the projection can be expressed using projective geometry as follows. We define the camera matrix

$$C := \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}.$$

Then, we have in homogeneous coordinates

$$[f(p)] = \begin{bmatrix} \frac{f_x x + c_x z}{z} \\ \frac{f_y y + c_y z}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x x + c_x z \\ f_y y + c_y z \\ z \end{bmatrix} = [Cp].$$

This rather elegant description often allows calculations to be made explicit when involving cameras modelled as pinhole cameras.

One main use of the pinhole camera model is for theoretical analyses of processes involving image capture. Here, a common variant is to just set $f_x = f_y = 1$ and $\begin{pmatrix} c_x \\ c_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ for simplicity.

The pinhole camera model is also used to unify the behaviour of the wide variety of camera optics by employing *calibration* and *rectification*. In calibration, it is determined how rays of light entering the camera (view rays seen from the camera) are mapped to points on the image plane. This is done in order to determine how to make the camera behave like a pinhole camera by distorting the raw camera images in a suitable way. This distortion process is called rectification. Subsequent processing steps (e.g. photogrammetry) then only need to be capable of processing the rectified images, which means we can hide most of the complexity of the optics of cameras from them. Note that rectification is also done with cameras models other than the pinhole model, e.g. there are specialized models in particular for cameras with very large FoVs.

The pinhole camera model works best for cameras whose FoV does not extend too far from the optical axis of the camera. Indeed, the model becomes singular for view directions perpendicular to the optical axis ($z = 0$). In order to not have constraints on the FoV for

[2]As the focal length is a proportionality factor between angles and image coordinates, it is measured in the units of the image coordinates. E.g. if physical coordinates of the photographic film or image sensors are used as image coordinates, physical sizes (e.g. Millimeters) are used. If pixel coordinates are used as image coordinates, the focal length is given in pixels.

theoretical analysis, we will also use the following omnidirectional model.

Definition 3 (spherical camera model)**.** Suppose given a point $p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ in the camera coordinate system. Then p is projected by a spherical camera to the following point on the image sphere $S^2 \subset \mathbb{R}^3$.

$$f(p) := \frac{p}{\|p\|} \in S^2$$

This model has mainly importance for theoretical analysis. Direct usage in algorithms is prevented by the fact that images are typically stored as rectangular arrays of pixels and the most practical way of covering the sphere with rectangle is cube-mapping. However, cube-mapping means essentially that the above spherical camera model is simply split into six pinhole cameras.

Note that from both the pinhole camera model and the omnidirectional model we obtain the same kind of information, which is the directions of observed scene points relative to the camera. I.e. the information provided differs quantitatively due to different distribution of angular resolution, but not qualitatively. Aside from the singularities of the pinhole model, the difference between both models is essentially a choice of parametrization. Hence, results obtained using one model can be transferred to the other model and vice versa. We will usually choose the model fitting the situation best.

For theoretical analysis, we also introduce the following two-dimensional models.

Definition 4 (2D pinhole camera model)**.** Suppose given a point $p = \begin{pmatrix} x \\ z \end{pmatrix} \in \mathbb{R}^2$ in the camera coordinate system. Then p is projected by the 2D pinhole camera model to the following point on the (one-dimensional) image plane.

$$f(p) = \begin{pmatrix} u \end{pmatrix} = \frac{x}{z}$$

In homogeneous coordinates, we have

$$[f(p)] = \begin{bmatrix} \frac{x}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ z \end{bmatrix} = [p].$$

Note that this model features neither focal length nor a principal point since it is used exclusively for theoretical analysis.

Definition 5 (2D omnidirectional model)**.** Suppose given a point $p = \begin{pmatrix} x \\ z \end{pmatrix} \in \mathbb{R}^2$ in the camera coordinate system. Then p is projected by the circular camera model (2D equivalent

of the spherical camera model) to the following point on the image circle $S^1 \subset \mathbb{R}^2$.

$$f(p) := \frac{p}{\|p\|} \in S^1$$

4.2.2 Inverse depth parametrizations

It is well known in photogrammetry that for the problem of triangulating points from multiple camera observations, the traditional parametrization of points with Cartesian coordinates is often ill-behaved, cf. e.g. [Triggs et al., 1999], [Civera et al., 2008]. The reason for this is that for a far-away point, the view rays to the cameras are always nearly parallel, with variation in the distance of the point causing only little change in how close to parallel these view rays are. Conversely, errors in in the estimates for the directions of the view rays (e.g. due to measurement errors or due to misestimation of the camera poses), cause large effects in the position estimate of the point. This may even cause the position estimate to cross the plane at infinity, thus crossing a singularity when using Cartesian parametrization.

The solution to this issue is the usage of parametrizations which are better suited for triangulation and the kind of information provided by camera observations.

The general idea for such parametrizations is to describe the position of a point by a direction component and an inverse distance component. Using directions for parametrization is completely natural since cameras essentially measure directions. The usage of inverse distances can be motivated by the fact that they smoothly describe distances up to infinity and beyond. Furthermore, the triangulation problem is often roughly linear (cf. e.g. Example 8 for a perfect example) when parametrized by direction and inverse distance. This way, nonlinear effects are reduced. In particular, the uncertainty of coordinate estimation is roughly independent of distance which contrasts the Cartesian case, for which uncertainty in the distance component increases quadratically with distance (cf. Example 8).

The usage of parametrizations featuring inverse distances is a well-known and well established concept in triangulation, cf. e.g. [Triggs et al., 1999], [Civera et al., 2008], [Hoelzer et al., 1978]. In the sequel, we provide formalization to some of these parametrizations, in particular to fix notation.

Definition/Remark 6 (pinhole coordinates[3])**.** Given a point $p \in \mathbb{R}^3$ with Cartesian

[3]There does not seem to be an established name for this parametrization, which is well-known, cf. e.g. [Civera et al., 2008]. In computer graphics, the transform between Cartesian and pinhole coordinates is used in a technique known as reverse z with infinite far plane [Upchurch and Desbrun, 2012, Reed, 2015]

coordinates $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$, we define its *pinhole coordinates* by

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix} := \frac{1}{z} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{1}{z} \end{pmatrix}.$$

Note that u and v give the direction from the coordinate center to p, while d is the inverse distance.

The Cartesian coordinates can be recovered from the pinhole coordinates via

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{\mathbf{d}} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ 1 \end{pmatrix}.$$

Pinhole coordinates can be obtained via a projective transformation from the Cartesian coordinates as follows. In homogeneous coordinates, we have

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{1}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \\ z \end{bmatrix} = \begin{bmatrix} C \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \end{bmatrix}$$

for the matrix

$$C := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Hence, the transformation between Cartesian and pinhole coordinates is a projective transformation and thus can be easily combined with other projective transformations. Furthermore, it preserves planes as projective transformations preserve (hyper)planes.

For a point $q \in \mathbb{R}^2$ with coordinates $\begin{pmatrix} x' \\ y' \end{pmatrix}$, we define analogously its (planar) pinhole coordinates

$$\begin{pmatrix} \mathbf{u}' \\ \mathbf{d}' \end{pmatrix} := \frac{1}{y'} \begin{pmatrix} x' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{x'}{y'} \\ \frac{1}{y'} \end{pmatrix}.$$

As with the 3D case, the transformation between Cartesian coordinates and pinhole coordinates is projective. The hyperplane preservation property of projective transformation

implies that this transformation preserves lines.

Definition 7 (reverse spherical coordinates)**.** Suppose given a point $p \in \mathbb{R}^3 \setminus \{0\}$ with Cartesian coordinates $\begin{pmatrix} x \\ y \\ z \end{pmatrix} =: c_p$. We define its reverse spherical coordinates by

$$(\mathbf{U}, \mathbf{D}) := \left(\frac{c_p}{\|c_p\|}, \frac{1}{\|c_p\|} \right) \in S^2 \times \mathbb{R} \setminus \{0\}.$$

We may recover the Cartesian coordinates via

$$c_p = \mathbf{U} \cdot \frac{1}{\mathbf{D}}.$$

Note that the spherical coordinates $(\mathbf{U}, \mathbf{D})$ and $(-\mathbf{U}, -\mathbf{D})$ denote the same point. This phenomenon appears due to the fact that the region surrounding the plane at infinity is topologically non-orientable, so parametrizations with unique coordinates (which are inherently orientable) are impossible without cutting.

For a point $q \in \mathbb{R}^2 \setminus \{0\}$ with coordinates $\begin{pmatrix} x' \\ y' \end{pmatrix} =: c_q$, we define analogously its reverse polar coordinates by

$$(\mathbf{U}', \mathbf{D}') := \left(\frac{c_q}{\|c_q\|}, \frac{1}{\|c_q\|} \right) \in S^1 \times \mathbb{R} \setminus \{0\}.$$

Example 8 (The triangulation problem for parallel cameras in pinhole coordinates)**.** Suppose given n pinhole cameras at positions $\begin{pmatrix} a_k \\ b_k \\ 0 \end{pmatrix}, k = 1, \ldots, n$ such that all cameras are oriented in the same way, looking in the direction of the z-coordinate. I.e. a point $p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ is observed by the k-th camera at position

$$f_k(p) = \begin{pmatrix} \frac{x - a_k}{z} \\ \frac{y - b_k}{z} \end{pmatrix}$$

of its image plane.

Let us formulate f_k in pinhole coordinates. p has pinhole coordinates $\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\\ \mathbf{d}\end{pmatrix}$. We have

$$\begin{aligned} f_k(p) &= \begin{pmatrix}\frac{x-a_k}{z}\\ \frac{y-b_k}{z}\end{pmatrix} = \begin{pmatrix}\frac{x}{z}-\frac{a_k}{z}\\ \frac{y}{z}-\frac{b_k}{z}\end{pmatrix} \\ &= \begin{pmatrix}\mathbf{u}-\mathbf{d}a_k\\ \mathbf{v}-\mathbf{d}b_k\end{pmatrix}. \end{aligned}$$

Hence, the projection of p on the image plane depends *linearly* on its pinhole coordinates. Thus in this particular variant, the problem of triangulating p from the camera observations becomes a linear problem.

Note that in Cartesian coordinates, measurements and stereo disparities are proportional the inverse $\frac{1}{z}$ of the distance z. Hence, the sensitivity of the measurement to variation of z is proportional to $\frac{\mathrm{d}}{\mathrm{d}z}\frac{1}{z} = -\frac{1}{z^2}$. Thus in stereo and multi-view triangulation, the error of the distance estimate increases quadratically with distance.

Example 9 (Reprojection in pinhole coordinates)**.** Suppose given a point p with Cartesian coordinates $p_c = \begin{pmatrix}x\\ y\\ z\end{pmatrix}$ and pinhole coordinates $\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\\ \mathbf{d}\end{pmatrix}$. An Euclidean transformation T given by $(q \in \mathbb{R}^3 \mapsto Rq + t)$ for an orthonormal matrix $R = \begin{pmatrix}R_{11} & R_{12} & R_{13}\\ R_{21} & R_{22} & R_{23}\\ R_{31} & R_{32} & R_{33}\end{pmatrix}$ and a translation vector $t = \begin{pmatrix}t_1\\ t_2\\ t_3\end{pmatrix}$ maps p in Cartesian coordinates to $p'_c = \begin{pmatrix}x'\\ y'\\ z'\end{pmatrix} = Rp_c + t$. In homogeneous coordinates, we have

$$\begin{aligned} \begin{bmatrix}\mathbf{u}'\\ \mathbf{v}'\\ 1\\ \mathbf{d}'\end{bmatrix} &= \begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix} = \begin{bmatrix}Rp_c + t\\ 1\end{bmatrix} \\ &= \left[\begin{pmatrix}R & t\\ 0 & 1\end{pmatrix}\begin{pmatrix}p_c\\ 1\end{pmatrix}\right] = \left[\begin{pmatrix}R & t\\ 0 & 1\end{pmatrix}\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\\ 1\\ \mathbf{d}\end{pmatrix}\right] \end{aligned}$$

By splitting the rotation matrix R into rows $R = \begin{pmatrix} R_{1,-} \\ R_{2,-} \\ R_{3,-} \end{pmatrix}$ and setting $U := \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ 1 \end{pmatrix}$, we obtain

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ 1 \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} R_{1,-}U + t_1\mathbf{d} \\ R_{2,-}U + t_2\mathbf{d} \\ R_{3,-}U + t_3\mathbf{d} \\ \mathbf{d} \end{pmatrix} = (R_{3,-}U + t_3\mathbf{d}) \begin{pmatrix} \frac{R_{1,-}U+t_1\mathbf{d}}{R_{3,-}U+t_3\mathbf{d}} \\ \frac{R_{2,-}U+t_2\mathbf{d}}{R_{3,-}U+t_3\mathbf{d}} \\ 1 \\ \frac{\mathbf{d}}{R_{3,-}U+t_3\mathbf{d}} \end{pmatrix}.$$

Since $\left[\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ \mathbf{d} \end{pmatrix} \right] = \begin{bmatrix} \mathbf{u}' \\ \mathbf{v}' \\ 1 \\ \mathbf{d}' \end{bmatrix}$, we obtain

$$\begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{d}' \end{pmatrix} = \begin{pmatrix} \frac{R_{1,-}U+t_1\mathbf{d}}{R_{3,-}U+t_3\mathbf{d}} \\ \frac{R_{2,-}U+t_2\mathbf{d}}{R_{3,-}U+t_3\mathbf{d}} \\ \frac{\mathbf{d}}{R_{3,-}U+t_3\mathbf{d}} \end{pmatrix} = \frac{1}{R_{3,-}U + t_3\mathbf{d}} \begin{pmatrix} R_{1,-}U + t_1\mathbf{d} \\ R_{2,-}U + t_2\mathbf{d} \\ \mathbf{d} \end{pmatrix}$$

Note that this transformation is perfectly well-behaved near $\mathbf{d} = 0$. Instead, singular behaviour is determined by the denominator $R_{2,-}U + t_2\mathbf{d}$. Note that $R_{2,-}U + t_2\mathbf{d} < 0$ if and only if d' has a different sign than d. I.e. a negative denominator indicates that T moves p from in front of the virtual camera to behind the camera or vice versa. The singular condition $R_{2,-}U + t_2\mathbf{d} = 0$ indicates that p is moved onto the plane perpendicular to the optical axis containing the camera origin.

Note that estimation errors in triangulation may result in negative inverse depths, i.e. it may result in points which lie "beyond infinity". Hence, the inverse depth $\mathbf{d}$ cannot be used to reliably detect whether a point lies in front or behind the virtual camera. Instead, the sign of the denominator above can be used for this purpose: In practice, visible points lie in front of the camera. When reprojecting to new camera poses, a negative sign of the denominator $R_{2,-}U + t_2\mathbf{d}$ then indicates that the point has moved behind the camera.

Remark 10 (The derivative of reprojection with regard to inverse depth and approximations)**.** Note that in the situation of Example 9, we have

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{d}} \begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{d}' \end{pmatrix} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{d}} \frac{1}{R_{3,-}U + t_3\mathbf{d}} \begin{pmatrix} R_{1,-}U + t_1\mathbf{d} \\ R_{2,-}U + t_2\mathbf{d} \\ \mathbf{d} \end{pmatrix}$$

$$
\begin{aligned}
=&\frac{\mathrm{d}}{\mathrm{d}\mathbf{d}}\frac{1}{(R_{3,-}U+t_3\mathbf{d})^2}\begin{pmatrix} t_1(R_{3,-}U+t_3\mathbf{d})-(R_{1,-}U+t_1\mathbf{d})t_3 \\ t_2(R_{3,-}U+t_3\mathbf{d})-(R_{2,-}U+t_2\mathbf{d})t_3 \\ (R_{3,-}U+t_3\mathbf{d})-\mathbf{d}t_3 \end{pmatrix} \\
=&\left(\frac{\mathbf{d}'}{\mathbf{d}}\right)^2\begin{pmatrix} t_1R_{3,-}U-t_3R_{1,-}U \\ t_2R_{3,-}U-t_3R_{2,-}U \\ R_{3,-}U \end{pmatrix} \\
\approx&\left(\frac{\mathbf{d}'}{\mathbf{d}}\right)^2\begin{pmatrix} t_1-t_3\mathbf{u} \\ t_2-t_3\mathbf{v} \\ 1 \end{pmatrix}
\end{aligned}
$$

For the approximation, we use the assumption that the rotation between the current and the new camera pose is small, hence $RU \approx U = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ 1 \end{pmatrix}$.

Example 11 (Direct image alignment in pinhole coordinates)**.** Suppose given a point cloud $p_k = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix}$, $k = 1, \ldots, n$ with pinhole coordinates $\begin{pmatrix} \mathbf{u}_k \\ \mathbf{v}_k \\ \mathbf{d}_k \end{pmatrix}$. Suppose each point has an intensity value j_k. Suppose given an intensity image $J = J(u, v)$ for a pinhole camera with model $f : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} f_x\frac{x}{z}+c_x \\ f_y\frac{x}{z}+c_z \end{pmatrix} = \begin{pmatrix} f_x\mathbf{u} \\ f_y\mathbf{v} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$. The objective is to find an Euclidean transformation $E \in SE(3)$ such that the photometric error

$$\varepsilon(E) := \sum_{k=1}^{n} W\left(J(f(Ep_k), j_k)\right)$$

for some functional W describing photometric similarity is minimized. We assume that a method based on local linearization such as the Gauss-Newton or Levenberg-Marquardt algorithm is used. Hence, the derivative $\frac{\mathrm{d}\varepsilon}{\mathrm{d}E} = \sum_{k=1}^{n} \frac{\mathrm{d}W(J(u,v),j_k)}{\mathrm{d}(u,v)}\,|_{f(Ep_k)}\,\frac{\mathrm{d}f(Ep_k)}{\mathrm{d}E}$ is required. In the following, we will compute the "inner" derivative $\frac{\mathrm{d}f(Ep_k)}{\mathrm{d}E}$, which is the part that does not depend on the similarity functional W or the image J.

Given $E = (p \mapsto Rp + t) = (p \mapsto R(p - R^{-1}t))$, we have the following set of generators of the tangent space of E in $SE(3)$:

$$\frac{\mathrm{d}E(x)}{\mathrm{d}g_1} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \frac{\mathrm{d}E(x)}{\mathrm{d}g_4} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} E(p),$$

$$\frac{\mathrm{d}E(x)}{\mathrm{d}g_2} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \qquad \frac{\mathrm{d}E(x)}{\mathrm{d}g_5} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} E(p),$$

$$\frac{\mathrm{d}E(x)}{\mathrm{d}g_3} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \qquad \frac{\mathrm{d}E(x)}{\mathrm{d}g_6} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} E(p)$$

Here, g_1, g_2, g_3 are the generators of the translation component and g_4, g_5, g_6 are the generators of the rotation component. Note that the transformation function to pinhole coordinates

$$q : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{1}{z} \end{pmatrix} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix}$$

has the following derivatives:

$$\frac{\mathrm{d}q}{\mathrm{d}x} = \begin{pmatrix} \frac{1}{z} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ 0 \\ 0 \end{pmatrix}$$

$$\frac{\mathrm{d}q}{\mathrm{d}y} = \begin{pmatrix} 0 \\ \frac{1}{z} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{d} \\ 0 \end{pmatrix}$$

$$\frac{\mathrm{d}q}{\mathrm{d}z} = -\frac{1}{z^2} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = -\frac{1}{z} \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{1}{z} \end{pmatrix} = -\mathbf{d} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix},$$

I.e. the total derivative is

$$\frac{Dq}{D(x,y,z)} = \mathbf{d} \begin{pmatrix} 1 & 0 & -\mathbf{u} \\ 0 & 1 & -\mathbf{v} \\ 0 & 0 & -\mathbf{d} \end{pmatrix}.$$

Setting $\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix} := q(E(p))$ and $\begin{pmatrix} x \\ y \\ z \end{pmatrix} := E(p)$ and combining the formulas for the derivatives of E and of q, we obtain

$$\frac{\mathrm{d}qE(p)}{\mathrm{d}g_1} = \begin{pmatrix} \mathbf{d} \\ 0 \\ 0 \end{pmatrix} \qquad \frac{\mathrm{d}qE(p)}{\mathrm{d}g_2} = \begin{pmatrix} 0 \\ \mathbf{d} \\ 0 \end{pmatrix} \qquad \frac{\mathrm{d}qE(p)}{\mathrm{d}g_3} = -\mathbf{d} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix}$$

and

$$\frac{\mathrm{d}qE(p)}{\mathrm{d}g_4} = \begin{pmatrix} \mathbf{d} & 0 & -\mathbf{du} \\ 0 & \mathbf{d} & -\mathbf{dv} \\ 0 & 0 & -\mathbf{d}^2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$
$$= \begin{pmatrix} 0 & -\mathbf{du} & 0 \\ 0 & -\mathbf{dv} & -\mathbf{d} \\ 0 & -\mathbf{d}^2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -y\mathbf{du} \\ -y\mathbf{dv} - z\mathbf{d} \\ -y\mathbf{d}^2 \end{pmatrix} = \begin{pmatrix} -\mathbf{uv} \\ -(\mathbf{v}^2+1) \\ -\mathbf{dv} \end{pmatrix}$$
$$\frac{\mathrm{d}qE(p)}{\mathrm{d}g_5} = \begin{pmatrix} \mathbf{d} & 0 & -\mathbf{du} \\ 0 & \mathbf{d} & -\mathbf{dv} \\ 0 & 0 & -\mathbf{d}^2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{du} & 0 & \mathbf{d} \\ \mathbf{dv} & 0 & 0 \\ \mathbf{d}^2 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\mathbf{du} + z\mathbf{d} \\ y\mathbf{du} \\ x\mathbf{d}^2 \end{pmatrix} = \begin{pmatrix} \mathbf{u}^2+1 \\ \mathbf{uv} \\ \mathbf{du} \end{pmatrix}$$
$$\frac{\mathrm{d}qE(p)}{\mathrm{d}g_6} = \begin{pmatrix} \mathbf{d} & 0 & -\mathbf{du} \\ 0 & \mathbf{d} & -\mathbf{dv} \\ 0 & 0 & -\mathbf{d}^2 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$
$$= \begin{pmatrix} 0 & -\mathbf{d} & 0 \\ \mathbf{d} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -y\mathbf{d} \\ x\mathbf{d} \\ 0 \end{pmatrix} = \begin{pmatrix} -\mathbf{v} \\ \mathbf{u} \\ 0 \end{pmatrix}$$

In summary, we have

$$\frac{\mathrm{d}qEp}{\mathrm{d}(g_1, g_2, g_3)} = \mathbf{d} \begin{pmatrix} 1 & 0 & -\mathbf{u} \\ 0 & 1 & -\mathbf{v} \\ 0 & 0 & -\mathbf{d} \end{pmatrix}$$
$$\frac{\mathrm{d}qEp}{\mathrm{d}(g_4, g_5, g_6)} = \begin{pmatrix} -\mathbf{uv} & \mathbf{u}^2+1 & -\mathbf{v} \\ -(\mathbf{v}^2+1) & \mathbf{uv} & \mathbf{u} \\ -\mathbf{dv} & \mathbf{du} & 0 \end{pmatrix}. \tag{1}$$

Note that the derivatives are expressed as low-order polynomials in $\mathbf{u}, \mathbf{v}, \mathbf{d}$, which facilitates analysis and stable and simple implementation.

4.2.3 Monocular and stereo configurations

Camera systems may consist of one or more physical cameras fixed together mechanically by a camera rig. If there is significant stereo overlap between cameras with different camera centers, we call such a system a *stereo system* or *stereo camera* and assume that image capture can be triggered for all cameras simultaneously, yielding a collection of images,

called *stereo image* of the system. We call other systems *monocular*, even if they are e.g. an omnidirectional camera consisting of multiple cameras looking in different directions to cover all directions.

4.3 Estimation

4.3.1 Motivation for alternatives to the Kalman filter

The Kalman filter is often used as some kind of jack of all trades, "tunable" method for estimation, while ignoring that the Kalman filter is designed to estimate a specific class of systems. In this text, we will use more general approaches for estimation. We would also like to use this opportunity to emphasize alternatives to the Kalman filter and in particular the general theory surrounding the Kalman filter which is basically the Kalman filter without the constraints given by its focus on dynamical systems.

Typically, there are two main problems with using the Kalman filter as a generic estimator. Firstly, the Kalman filter assumes that the system to be estimated is a *dynamical system*. Dynamical systems have the property that they are deterministic system subject to some stochastic disturbances. In particular, they are essentially deterministic. Many real-world estimation problems, however, feature some components which are highly deterministic or sometimes even satisfy some constraints exactly and other components which are completely unknown, i.e. there is not even any stochastic property known about them (e.g. deliberate, but unknown external influences such as a human deciding to or deciding to not to influence the system). Using the Kalman filter for estimating such systems typically involves complicated modeling to force the system into the form of a dynamical system, which adds unnecessary complexity and usually makes the estimator suboptimal.

The second problem of the Kalman filter is initialization. The Kalman filter uses a covariance matrix to model uncertainty in the estimation of the state vector. However, situations with zero or low knowledge cannot be modelled well using a covariance matrix since this translates to "infinite covariance" or, in more practical terms, numerical instability due to divisions by zero or almost zero. This situation is always present in initialization, which often causes estimators based on the Kalman filter to have fairly complex initialization routines. This problem can be mitigated e.g. by replacing the covariance matrix in uncertainty tracking by its inverse, which is called the information matrix. Zero knowledge can be modeled using the information matrix by just setting it to zero. Modifying the Kalman filter to use the information matrix yields the *information filter*.

The general estimation theory surrounding the Kalman filter deals with estimation problems with normally distributed uncertainties. For such problems, the log-likelihood (motivated e.g. by maximum-likelihood estimation) is a quadratic functional. Hence, estimation

simply reduces to manipulating quadratic functionals in a suitable way. For example, calculating the maximum-likelihood estimate means determining the minimum of a quadratic functional, which, by Fermat's theorem on stationary points, means simply solving a linear equation. Adding constraints (e.g. observations) means adding quadratic functionals (cf. Remark 15). If components of the estimated vector do not feature in any new constraints any more, they can be eliminated from the estimated vector and from the quadratic functional, which means applying the Schur complement in a certain way to the quadratic term of the quadratic functional.

These three operations (maximum-likelihood estimation, adding constraints and eliminating components of the estimate vector) are everything necessary for dealing with such problems. The Kalman filter is a simple application of this framework.

Remark 12 (Notes on random variables and measurability)**.** Note that a random variable X is by definition a measurable function from a probability space $(\Omega, \mathscr{A}, \mu)$ to a measure space $(E, \mathscr{B})$. Typically, the precise nature of Ω is of no interest. In fact, we may construct the distribution μ_X of X on E by the pushforward construction, which transfers μ to E using X yielding the μ_X as the pushforward measure. This way, all relevant operations can be formulated directly on E thus making the precise nature of Ω irrelevant. Hence, we will omit Ω.

In this section, we will work with (finite-dimensional) vector-valued random variables. Hence, the σ-algebra chosen by us for such a finite-dimensional vector space is the Borel σ-algebra induced by the topology of the vector space. We will construct new random variables only by composing existing random variables with linear transformation. We need to ensure that these new random variables are actually random variables, i.e. we need to ensure that they are measurable (cf. the definition above): Linear transformations on finite-dimensional vector spaces are continuous. Borel-measurability of continuous functions is ensured by the design of the Borel σ-algebra. Hence, linear transformations on finite-dimensional vector spaces are Borel-measurable. As random variables are measurable, the composition of such a linear transformation with a random variable is measurable, so our constructions indeed yield random variables.

4.3.2 Quadratic functionals and maximum likelihood estimation for normal distributions

Remark 13. Suppose given an (unknown) vector of states $x_0 \in \mathbb{R}^n$. Suppose given a linear functional $H : \mathbb{R}^n \to \mathbb{R}^m$ and a (known) vector $y \in \mathbb{R}^m$ such that the *residual* $Hx_0 - y$ constrains x_0, i.e. such that $Hx_0 - y$ is a normally distributed, centered random variable with covariance matrix P for some strictly positive definite symmetric matrix $P \in \mathbb{R}^{m \times m}$.

The functional H can encode various ways of gaining or having knowledge about the vector of states x. Notably, it can encode measurement processes yielding components of y obtained by measurement with normally distributed measurement error. Furthermore, it can encode knowledge about constraints within the system such as relationships between different states subject to random normally distributed influence, e.g. randomly disturbed deterministic behavior.

The probability distribution of the residual has density

$$p(Hx - y) = c \cdot \exp\left(-\frac{1}{2}(Hx - y)^T P^{-1}(Hx - y)\right)$$

for some normalization constant c. Thus, for the quadratic functional

$$q(x) := (Hx - y)^T P^{-1}(Hx - y), \tag{2}$$

the log-likelihood function is

$$\log(p(Hx - y)) = \log(c) - \frac{1}{2}q(x). \tag{3}$$

Maximum-likelihood estimation of x means maximizing the likelihood function (and thus the log-likelihood function due to the monotony of the logarithm), which is equivalent to minimizing q over x. This way, q is a (quadratic) loss function. We are now in the familiar territory of least-squares optimization. The quadratic functional q encodes all our knowledge about x. In particular we can recover the original distribution p via (3) and normalization. Furthermore, a maximum likelihood estimate for x minimizes q by construction. Hence, such a maximum likelihood estimate may be obtained my minimizing the quadratic functional q, that is by performing linear least squares optimization.

Lemma 14 (The inverse of the information matrix is the covariance of the ML-estimator)**.** *Suppose given the situation of Remark 13. Suppose that the information matrix $I := H^T P^{-1} H$ is invertible. Let $\hat{x}$ be the maximum likelihood estimate for x. Let x_0 be the true value for x. Then, the estimation error $e = \hat{x} - x_0$ is a normally distributed, centered random variable with covariance matrix I^{-1}.*

Note that the inverse of the measurement variance, the information P^{-1}, can be thought of as the resolving power of our method in the space of measurements. In this picture, H is the transmission function from state space to measurement space, which means that in order to obtain the resolving power of our method in state space, we simply need to multiply P^{-1} with H (on both sides), giving the information matrix which is the inverse of the variance of the reconstruction error.

However, as probability is much more straightforward concept than the multi-faceted

concept of information, this lemma is stated and proven using concepts from probability theory. This yields a more technical, but conceptually simpler proof.

Proof. As the estimate $\hat{x}$ for x minimizes $q(x)$ as given in Remark 13, Fermat's theorem on stationary points implies that

$$0 = \frac{\mathrm{d}q(x)}{\mathrm{d}x}\Big|_{\hat{x}} = 2H^T P^{-1} H\hat{x} - 2H^T P^{-1} y.$$

As $I = H^T P^{-1} H$ is invertible, we have

$$\hat{x} = (H^T P^{-1} H)^{-1} H^T P^{-1} y.$$

In the following, we reformulate this using the Moore-Penrose inverse, also known as pseudoinverse, cf. [Penrose, 1955]. We will refer to several elementary properties of the pseudoinverse given in Penrose's original text [Penrose, 1955]. Note that these are also found in most treatments of the pseudoinverse and are, in fact, rather unsurprising consequences of the fact that given a singular value decomposition of a matrix $A = USV^T$, its pseudoinverse is simply VS^+U^T, with S^+ obtained from S by transposing S and inverting all non-zero entries.

Since $H^T P^{-1} H = I$ is invertible, the matrix

$$J := P^{-\frac{1}{2}} H$$

has full column rank. Hence, we have

$$(H^T P^{-1} H)^{-1} H^T P^{-\frac{1}{2}} = (J^T J)^{-1} J^T = J^+$$

The last identity can be obtained by combining the fact $J^+ = (J^T J)^+ J^T$ (cf. [Penrose, 1955, Comment after Lemma 1] with the fact that for an invertible matrix such as $(J^T J)$, its pseudoinverse coincides with its inverse, cf. [Penrose, 1955, Lemma 1-3].

I.e. $\hat{x} = J^+ P^{-\frac{1}{2}} y$. As J has full column rank, the matrix J^+ has full row rank. So as Q is invertible, the map $z \mapsto J^+ P^{-\frac{1}{2}} z$ is surjective. As y is a normally distributed random variable with mean Hx_0 and covariance P, $\hat{x}$ is via Lemma 32 a normally distributed random variable with mean $\mu = JP^{-\frac{1}{2}} Hx_0$ and covariance $Q = (J^+ P^{-\frac{1}{2}}) P (J^+ P^{-\frac{1}{2}})^T$. We may reformulate the mean as

$$\mu = J^+ P^{-\frac{1}{2}} Hx_0 = (H^T P^{-1} H)^{-1} H^T P^{-\frac{1}{2}} (P^{-\frac{1}{2}} Hx_0) = (H^T P^{-1} H)^{-1} (H^T P^{-1} H) x_0 = x_0.$$

We may reformulate the covariance as

$$\begin{aligned} Q =& (J^+P^{-\frac{1}{2}})P(J^+P^{-\frac{1}{2}})^T = J^+P^{-\frac{1}{2}}PP^{-\frac{1}{2}}(J^+)^T = J^+(J^+)^T \\ \overset{*}{=}& (J^TJ)^+ = (H^TP^{-\frac{1}{2}}P^{-\frac{1}{2}}H)^+ = I^+ \\ \overset{**}{=}& I^{-1}. \end{aligned}$$

For $(*)$, we have used [Penrose, 1955, Lemma 1-5]. For $(**)$, we have used the fact that the pseudoinverse of an invertible matrix is its inverse. cf. [Penrose, 1955, Lemma 1-3].

Thus, $\hat{x}$ is a normally distributed random variable with mean x_0 and covariance I^{-1}. Hence, applying Lemma 31 to the error $e = \hat{x} - x_0$ completes the proof. □

Remark 15 (Adding constraints)**.** Suppose given a decomposition of the residual space $\mathbb{R}^m = \mathbb{R}^{m_1} \oplus \mathbb{R}^{m_2}$, $m = m_1 + m_2$. Suppose that P has diagonal block structure along that decomposition, that is

$$P = \begin{pmatrix} P_1 & 0 \\ 0 & P_2 \end{pmatrix}$$

with $P_1 \in \mathbb{R}^{n_1 \times n_1}, P_2 \in \mathbb{R}^{n_2 \times n_2}$. This means in particular that the sets of residuals are statistically independent. We denote this situation as the constraints being composed of two independent sets of constraints.

Splitting H and y similarly into blocks $H = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}$, $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, we obtain

$$\begin{aligned} q(x) =& (Hx - y)^TP^{-1}(Hx - y) \\ =& \left(\begin{pmatrix} H_1 \\ H_2 \end{pmatrix} x - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right)^T \begin{pmatrix} P_1 & 0 \\ 0 & P_2 \end{pmatrix}^{-1} \left(\begin{pmatrix} H_1 \\ H_2 \end{pmatrix} x - \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \right) \\ =& \begin{pmatrix} H_1x - y_1 \\ H_2x - y_2 \end{pmatrix}^T \begin{pmatrix} P_1^{-1} & 0 \\ 0 & P_2^{-1} \end{pmatrix} \begin{pmatrix} H_1x - y_1 \\ H_2x - y_2 \end{pmatrix} \\ =& (H_1x - y_1)^TP_1^{-1}(H_1x - y_1) + (H_2x - y_2)^TP_2^{-1}(H_2x - y_2) \\ =& q_1(x) + q_2(x), \end{aligned}$$

with the quadratic functionals of the sets of constraints denoted by

$$\begin{aligned} q_1(x) =& (H_1x - y_1)^TP_1^{-1}(H_1x - y_1) \\ q_2(x) =& (H_2x - y_2)^TP_2^{-1}(H_2x - y_2). \end{aligned}$$

I.e. we may accumulate independent sets of constraints by simply adding their respective quadratic functionals.

Remark 16 (Reducing quadratic loss functionals and recursive estimation)**.** Often, there

are situations where certain parts of a system are of little interest. Consider for example real-time estimation problems (e.g. in the context of control problems): Since measurements and system equations typically relate the system states of different points in time to each other, the (full) state vector is the vector of all system states of all points in time. This seems (and is, as shown by the approach of recursive estimation) excessively complex since one is only interested in the most recent system state. Hence, we need a way to reduce the state vector to "relevant" components:

Suppose that the state vector x consists of two components, that is $x = \begin{pmatrix} u \\ v \end{pmatrix}$, $u \in \mathbb{R}^k, v \in \mathbb{R}^j$, for some k, j. For simplicity, we write $q(x) =: q(u, v)$. We define the reduction of q by v as

$$q_{,v}(u) := \min_v q(u, v).$$

Note that $q_{,v}$ is actually a quadratic functional, cf. e.g. the explicit formulas in Remark 23. Note that

$$\min_u q_{,v}(u) = \min_u \min_v q(u, v) = \min_x q(x) =: q_0,$$

so the minimum values of q and $q_{,v}$ are equal. Hence, for a given u^* we have

$$q_{,v}(u^*) = q \qquad \Leftrightarrow \qquad \min_v q(u^*, v) = q,$$

so u^* is a minimizer of $q_{,v}$ if and only if it is part of a minimizer of q. This allows the usage of $q_{,v}$ for maximum likelihood estimation of the component u. I.e. reducing q to $q_{,v}$ discards only information on v.

Remark 17 (constructing reduced quadratic functionals recursively)**.** So suppose that the (full) vector of states consists of multiple components $x = \begin{pmatrix} r \\ u \\ v \end{pmatrix} = \begin{pmatrix} r \\ \check{x} \end{pmatrix}$. Suppose that the loss function q_1 for x has the form

$$q_1(x) = q_1(r, \check{x}) = q_1(r, u, v) = \hat{q}_1(r, u) + q_0(u, v) \tag{4}$$

for some quadratic functionals $\hat{q}_1, q_0$.

A key example for such a separation (4) is in incremental estimation: First, u is the current system state and v is the previous system state, so there is only the information contained in q_0 available, which relates u to v. At a later point in time, the new system state r is added to the state vector. As information on the system typically either applies to a system state individually (i.e. absolute measurements) or relates consecutive system states

(such as measurements of *change* or differential equations governing the system), the new information gained in the new point in time typically involves r and u, but not v, so it can be described by $\hat{q}_1 = \hat{q}_1(r, u)$.

In the following, we will reduce v from q_0, q_1 and then u.

Let

$$\begin{aligned}\bar{q}_1(r) &:= \min_x q_1(r, x) = \min_{u,v} q_1(r, u, v)\\ \bar{q}_0(u) &:= \min_v q_0(u, v)\end{aligned}$$

Due to (4), we can obtain $\bar{q}_1$ from $\bar{q}_0$ and $\hat{q}$ as follows.

$$\begin{aligned}\bar{q}_1(r) &= \min_{u,v} q_1(r, u, v) = \min_u(\min_v q_1(r, u, v))\\ &= \min_u(\min v(\hat{q}_1(r, u) + q_0(u, v))\\ &= \min_u(\hat{q}_1(r, u) + \bar{q}_0(u))\end{aligned}$$

This means that we do not need to know q_1 explicitly to compute $\bar{q}_1$. Instead, we may compute reduced loss functionals incrementally: First, $\bar{q}_0$ is computed from q_0. Then, $\bar{q}_1$ can be computed from $\hat{q}_1$ and $\bar{q}_0$. In the next time step, we may than compute $\bar{q}_2$ from a $\hat{q}_2$ and $\bar{q}_1$ and so forth.

Note that in our example, the $\bar{q}_i$ only involve the current system state. In particular, they do not involve the system states at previous points in time. I.e. $\bar{q}_i$ is a quadratic functional of the current system state and only of the current system state, which means that the complexity necessary to describe $\bar{q}_i$ (matrix size of the quadratic term, vector size of the linear term) only depends on the dimensionality of the current system state. In particular if the dimensionality of the current system state is fixed (e.g. for dynamical systems), the q_i have fixed complexity, which facilitates implementations requiring only constant computational effort per time step.

So far, we have only described the bare theory. In the sequel, we will explain how to perform the necessary manipulation of quadratic functionals practically.

4.3.3 Quadratic form and square root form

Remark 18 (Quadratic form and square root form)**.** Suppose given a quadratic loss function $q = q(x)$ for $x \in \mathbb{R}^n$, cf. Remark 13. Note that the positive-definiteness of P implies $z^T P^{-1} z \geq 0$ for all $z \in \mathbb{R}^m$, so via equation (2), we obtain

$$q(x) \geq 0 \qquad \text{for all } x \in \mathbb{R}^n. \tag{5}$$

In particular, quadratic loss functions are bounded from below.

Since q is a quadratic functional, it has the form

$$q(x) = x^T I x + lx + d \tag{6}$$

for a quadratic matrix I, a linear functional l and a constant d. Eq. (2) implies $I = H^T P^{-1} H$, so I is symmetric and positive semidefinite since P is symmetric positive definite. The matrix I is called the *information matrix*.

By Lemma 33, there exists a (not necessarily unique) $x^* \in \mathbb{R}^n$ such that $q(x^*) = \inf_{x \in \mathbb{R}^n} q(x)$ and by Lemma 34, we have

$$q(x) = (x - x^*)^T I (x - x^*) + q(x^*). \tag{7}$$

eqs. (6) and (7) are the conventional, "quadratic" form of representing quadratic loss functionals. A main problem of this form is the definiteness of the numerical representation of the information matrix I: By construction, I is positive semidefinite. This is an important property since it ensures the convexity of q, thus making maximum-likelihood estimation via minimization of q possible. However, rounding errors in the numerical representation of I may make the numerical representation of I become indefinite, thus possibly causing failure of algorithms relying on the positive-semidefiniteness of I.

This is solved by the square-root form of q: Since I is positive semidefinite, there exists a matrix S such that $I = S^T S$. One way for obtaining such a matrix S is to use the matrix square root of I, yielding a positive semidefinite square S. Another way is to perform Cholesky factorization on I, yielding a upper or lower (depending on choice of Cholesky flavour) triangular matrix S. Finally, a more constructive way is to simply use the choice

$$S := P^{-\frac{1}{2}} H \tag{8}$$

in (2).

Given such a decomposition $I = SS^T$, equation (7) becomes

$$\begin{aligned} q(x) =& (x - x^*) S^T S (x - x^*) + q(x^*) \\ =& \|S(x - x^*)\|^2 + q(x^*) \\ \overset{Sx^* =: b}{=}& \|Sx - b\|^2 + q(x^*) \end{aligned}$$

The form

$$q(x) = \|Sx - b\|^2 + c \tag{9}$$

for q for a $k \times n$-matrix S, $k \geq 0$, a vector $b \in \mathbb{R}^k$ and a constant c is the *square root form* of q. Note that in this form, the information matrix $I = S^T S$ is always positive semidefinite regardless of any errors made in the computation of S.

Note also that S does not have to be a square matrix, though most methods for constructing S (such as those given above) yield square matrices. A particularly beneficial property of the square root form is that S can be multiplied on the left hand side with an orthogonal matrix $Q \in \mathbb{R}^{k \times k}$ without changing q:

$$\begin{aligned} \|Q(Sx-b)\|^2 =& (Sx-b)^T Q^T Q(Sx-b) = (Sx-b)^T I(Sx-b) \\ =& (Sx-b)^T(Sx-b) = \|Sx-b\|^2 \end{aligned} \tag{10}$$

This is useful for making S into a square matrix if $k > n$, cf. the following remark.

Remark 19 (Simplification of a given square root form)**.** Suppose given a quadratic loss function q in a given square root form (9). Suppose that S is a $k \times n$ matrix with $k > n$. We want to modify the given square root form in such a way that the matrix part S is replaced by a $n \times n$-matrix. We will actually obtain an upper triangular matrix.

We may obtain a QR-decomposition $S = QR$ of S with an orthogonal matrix Q and a upper triangular matrix R. Note that $R = Q^T S$. We have

$$\begin{aligned} q(x) &\overset{(10)}{=} \|Q^T(Sx-b)\|^2 + c = \|Rx - Q^T b\|^2 + c \\ &\overset{b' := Q^T b}{=} \|Rx - b'\|^2 + c. \end{aligned}$$

Now, note that the $k \times n$-matrix R is upper triangular, so $k > n$ implies that R consists of an upper triangular $n \times n$ matrix R_0 and a $(k-n) \times n$ zero matrix Z, i.e $R = \begin{pmatrix} R_0 \\ Z \end{pmatrix}$.

Thus setting $b'' := \begin{pmatrix} b'_1 \\ \vdots \\ b'_n \end{pmatrix}$ and $r := \begin{pmatrix} b'_{n+1} \\ \vdots \\ b'_k \end{pmatrix}$, we have

$$\begin{aligned} q(x) =& \|Rx - b'\|^2 + c = \left\| \begin{pmatrix} R_0 \\ Z \end{pmatrix} x - \begin{pmatrix} b'' \\ r \end{pmatrix} \right\|^2 + c \\ \overset{Z=0}{=}& \|R_0 x - b''\|^2 + \|r\|^2 + c \\ \overset{c'' := c + \|r\|^2}{=}& \|R_0 x - b''\|^2 + c'' \\ \overset{S'' := R_0}{=}& \|S'' x + b''\|^2 + c''. \end{aligned}$$

This way, we have transformed the square root form of q in such a way that the replacement S'' for S is a $n \times n$ upper triangular matrix.

Note that while this is a generic and stable (if suitable QR decomposition methods such as usage of Householder transforms are employed) methods, this is by no means the only method for this task. In particular, knowledge about the structure of S such as sparsity patterns may facilitate the usage of specially tailored simplification methods.

Remark 20 (Adding constraints: implementation)**.** As described in Remark 15, adding constraints in the quadratic form (6) just means adding the quadratic functionals. In the modified quadratic form (7), the minimizer x^* needs to be recomputed.

Concerning adding constraints in square root form, suppose given two quadratic loss functionals q_1, q_2 in square root form:

$$\begin{aligned} q_1(x) =& \|S_1 x - b_1\|^2 + c_1 \\ q_2(x) =& \|S_2 x - b_2\|^2 + c_1 \end{aligned}$$

We have

$$q(x) := q_1(x) + q_2(x) = \left\| \begin{pmatrix} S_1 \\ S_2 \end{pmatrix} x - \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right\| + (c_1 + c_2). \tag{11}$$

I.e. we may add q_1, q_2 by concatenating S_1, S_2 as well as b_1, b_2 and by adding c_1, c_2. Note that the matrix $S := \begin{pmatrix} S_1 \\ S_2 \end{pmatrix}$ is larger than either S_1 or S_2, so it is often beneficial to perform simplification in the style of Remark 19 afterwards.

Lemma 21 (Reducing quadratic loss functions: quadratic form)**.** *Suppose given a quadratic loss function $q = q(x) = q(u, v)$ in the form* (6) *for a decomposition $x = \begin{pmatrix} u \\ v \end{pmatrix}$. We can decompose the description* (6) *of q similarly via splitting I, l*

$$\begin{aligned} I =& \begin{pmatrix} I_{uu} & I_{uv} \\ I_{vu} & I_{vv} \end{pmatrix} \\ l =& \begin{pmatrix} l_u & l_v \end{pmatrix} \end{aligned}$$

along the decomposition of x into u and v. Hence, we have

$$q(x) = q(u, v) = u^T I_{uu} u + 2u^T I_{uv} v + v^T I_{vv} v + l_u u + l_v v + d.$$

If I_{vv} is invertible, the reduced functional $q_{,v}$ has the form

$$q_{,v}(u) = u^T \bar{I} u + \bar{l} u + \bar{c}$$

for

$$\begin{aligned}\bar{I} =& I_{uu} - I_{uv} I_{vv}^{-1} I_{uv}^T \\ \bar{l} =& l_u - l_v I_{vv}^{-1} I_{uv}^T \\ \bar{c} =& d - \frac{1}{4} l_v I_{vv}^{-1} l_v^T.\end{aligned}$$

The matrix $\bar{I}$ is also called the Schur complement *of the block I_{vv} in I.*

Proof. We need to prove the formula for $q_{,v}$. Towards that end, let us examine the minimum of $q(u, v)$ for given u. At the minimal v, we have

$$0 = \frac{\mathrm{d}q(u, v)}{\mathrm{d}v} = 2u^T I_{uv} + 2v^T I_{vv} + l_v.$$

Solving for v yields

$$v = -I_{vv}^{-1} \left(\frac{1}{2} l_v^T + I_{uv}^T u \right).$$

Now, the explicit expression for $q_{,v}$ can be obtained by inserting the above expression for v into the expression for q. □

Remark 22 (Reducing quadratic loss functions: quadratic form in factored form)**.** Suppose that in the situation of Lemma 21, there are matrices E, F such that

$$I = \begin{pmatrix} I_{uu} & I_{uv} \\ I_{vu} & I_{vv} \end{pmatrix} = \begin{pmatrix} E & F \end{pmatrix}^T \begin{pmatrix} E & F \end{pmatrix}$$

and

$$\begin{aligned} I_{uu} &= E^T E \\ I_{uv} &= E^T F \\ I_{vv} &= F^T F. \end{aligned}$$

I.e. the information matrix is given in a factored form, which is already decomposed along the decomposition of x in u and v. Then, we have

$$\begin{aligned}\bar{I} =& I_{uu} - I_{uv} I_{vv}^{+} I_{uv}^T \\ =& E^T E - E^T F (F^T F)^{+} (E^T F)^T \\ =& E^T E - E^T F (F^T F)^{+} F^T E \\ =& E^T (1 - F (F^T F)^{+} F^T) E.\end{aligned}$$

Remark 23 (Reducing quadratic loss functions: square root form)**.** Suppose given a quadratic loss function $q = q(x) = q(u, v)$, $u \in \mathbb{R}^{n_u}$, $v \in \mathbb{R}^{n_v}$ in the square root form (9) for a decomposition $x = \begin{pmatrix} u \\ v \end{pmatrix}$.

We choose a block decomposition

$$S = Q \begin{pmatrix} S_{1u} & S_{1v} \\ 0 & S_{2u} \end{pmatrix}$$

for a orthogonal matrix Q and matrices S_{1u}, S_{1v}, S_{2u} such that the matrix S_{1v} has full column rank, i.e. the map $(v \mapsto S_{1v}v)$ is surjective. Such a decomposition can e.g. be obtained by a rank-revealing QR decomposition (u and v need to be swapped to obtain an upper triangular matrix as the right hand side factor).

Let $b = Q \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ be the decomposition of b along the decomposition of S. We have

$$\begin{aligned} q(u, v) =& \|S \begin{pmatrix} u \\ v \end{pmatrix} + b\|^2 + c = \|Q^{-1}(S \begin{pmatrix} u \\ v \end{pmatrix} + b)\|^2 + c \\ =& \|S_{1u}u + S_{1v}v + b_1\|^2 + \|S_{2u}u + b_2\|^2 + c \end{aligned}$$

Note that since $(v \mapsto S_{1v}v)$ is surjective, we have $\min_v \|S_{1u}u + S_{1v}v + b_1\|^2 = \|0\|^2 = 0$ for all u. Hence, we have

$$q_{,v}(u) = \min_v q(u, v) = \|S_{2u}u + b_2\|^2 + c.$$

This gives the reduction of q in square root form.

4.3.4 Application

Example 24 (Observing a linear dynamical system. Notes on the information filter and on the Kalman filter.)**.** Consider the following time discrete dynamical system.

$$\begin{aligned} x_k =& F_k x_{k-1} + B_k u_k + w_k \\ z_k =& H_k x_k + v_k \end{aligned} \tag{12}$$

Here, x_k is the state vector of the system at time k, z_k is the measurement vector at time k and u_k is the (known) control input at time k. F_k, B_k and H_k are the state transition matrix, the control input matrix and the observation matrix. The vectors w_k, v_k are normally distributed random variables with $w_k \sim N(0, Q_k)$, $v_k \sim N(0, R_k)$ for some positive definite covariance matrices Q_k, R_k. The first equation of (12) describes how the

new system state x_k emerges from the previous system state x_{k-1}, the (known) control input u_k and the (unknown) system disturbance w_k. The second equation in (12) describes the measurement process, with measurement error v_k.

The objective of the information filter and of its dual cousin, the Kalman filter, is to incrementally estimate the system state x_k. We will apply the methods established above to this problem:

Suppose that the current knowledge about the state vector x_{k-1} in the previous time step is in encoded in the quadratic loss functional $q = q(x_{k-1}) = \|Sx - b\|^2 + c$ in square root form.

Let $q_k := Q_k^{-\frac{1}{2}}$, $r_k := R_k^{-\frac{1}{2}}$. Using these, we may add (12) to q: Reformulating (12) as

$$\begin{aligned}(-F_k x_{k-1} + x_k) - (B_k u_k) =& w_k \\ (H_k x_k) - (z_k) =& - v_k,\end{aligned} \tag{13}$$

we obtain via (8) and (11) the extended quadratic loss functional

$$\hat{q}(x_k, x_{k-1}) = \left\| \begin{pmatrix} 0 & S \\ -q_k F_k & q_k \\ r_k H_k & 0 \end{pmatrix} \begin{pmatrix} x_k \\ x_{k-1} \end{pmatrix} - \begin{pmatrix} b \\ q_k B_k u_k \\ r_k z_k \end{pmatrix} \right\|^2 + c.$$

Then, we may perform simplification on $\hat{q}$ using Remark 19 and eliminate x_{k-1} from it using Remark 22 to obtain the quadratic loss function $q' = q'(x_k)$, which encodes our knowledge our x_k. In particular, we may obtain the maximum likelihood estimate for x_k by minimizing q', cf. Remark 13.

Thus we have arrived at a solution to the (recursive) maximum likelihood estimation problem for linear dynamical systems with Gaussian disturbance and measurement error. This solution may be considered as a variant of the square-root form of the *information filter*. Note that as we have basically just plugged the equations of a linear dynamical systems into the general framework, it is a fairly high-level approach. Traditionally, much more explicit forms of the information filter are used and are also commonly found in the literature. The same holds for the Kalman filter, which differs from the information filter mainly in the fact that instead of maintaining an information matrix to quantify the uncertainty of the state estimate, it uses the covariance matrix, which is the inverse of the information matrix.

The key advantage of the approach presented here is that while it can be easily applied to the state estimation problem for linear dynamical systems, it is not restricted to such systems. Indeed, the state transition equation of dynamical systems is quite specific to dynamical systems. For example, the problem of refining estimates of a *static* state

(e.g. when estimating the geometry of a static scene) does not feature a state transition. Another class of problems are problems where parts of the state transition follow complex rules or depend on something unknown to us (e.g. unknown/foreign control input). Here, usage of the Kalman filter requires the problem to be forced into the form of a dynamical system (often at the cost of modelling errors), while in the general framework given above, we can simply restrict the state transition model to the parts which are treatable.

Remark 25 (Error propagation in least-squares estimation and the singular values of the Jacobian)**.** Suppose given a function k which maps a (to be estimated) variable p to the (noise-free) measurements $k(p)$. The examination of the Jacobian $J := \frac{\mathrm{d}k}{\mathrm{d}p}$ of k and in particular of its singular values reveals the sensitivity of the least-squares estimate for p to measurement errors:

Suppose given a 'true' value p_r for p and corresponding noise-free measurements $m_r := k(p_r)$. Given a vector of measurements $m = m_r + \delta$, with measurement noise δ, the least-squares estimate p_{ls} for p is the minimizer for

$$\|k(p) - m\|^2 \to \min .$$

By Fermat's theorem regarding stationary points, p_{ls} satisfies the equation $J^T(k(p_{ls})-m) = 0$. By this equation, $p_{ls} = p_{ls}(m)$ becomes an implicit function of m. The implicit function theorem allows us to compute the derivative of $p_{ls} = p_{ls}(m)$ at m_r, which is

$$\frac{\mathrm{d}p_{ls}}{\mathrm{d}m}\Big|_{m_r} = (J^T J)^{-1} J^T.$$

That is via the linear approximation $p_{ls}(m) = p_{ls}(m_r + \delta) \approx p_{ls}(m_r) + \frac{\mathrm{d}p_{ls}}{\mathrm{d}m}\big|_{m_r}\delta = p_r + (J^T J)^{-1} J^T \delta$, the matrix $(J^T J)^{-1} J^T$ describes the first-order sensitivity of the estimate p_{ls} to measurement noise δ. The matrix $(J^T J)^{-1} J^T$ equals the Moore-Penrose pseudoinverse J^+ of J and can be computed by transposing the singular value decomposition of J and inverting the singular values. In particular, the singular values of $J^+ = (J^T J)^{-1} J^T$ are the inverses of the singular values of J.

Note that here, the matrix $J^T J$ is a variant of the information matrix in the nonlinear context, cf. Remark 13 and Lemma 14.

The linear approximation $p_{ls}(m) = p_{ls}(m_r + \delta) \approx p_r + J^+\delta$ implies that if δ is normally distributed with standard deviation σ (i.e. covariance matrix $\sigma^2 \cdot E$), then the covariance matrix of $p_{ls}(m) - p_r$ can be expected to have the eigenvalues $\sigma^2\mu_1^2, \sigma^2\mu_2^2, \ldots$, where $\mu_1, \mu_2, \ldots$ are the singular values of J^+. As the covariance describes the square of the distance to the mean, the 'linear scale' (i.e. the square root of the largest eigenvalue of the covariance matrix since there is no concept of 'standard deviation' in higher dimensions) of the estimation error is given by $\sigma\mu_{\max} = \sigma\nu_{\min}^{-1}$. Here, $\mu_{\max}$ is the largest singular value

of J^+ and $\nu_{\min}$ is the smallest singular value of J (recall that the singular values of J are the reciprocals of the singular values of J^+).

Variants of the least-squares such as robustified least-squares variants behave slightly differently, but the underlying principle and the importance of J and its pseudoinverse J^+ and their singular values remain the same.

Remark 26 (Application of recursive linear filters to nonlinear problems via linearization)**.** The estimation methods established in the previous sections are designed for linear systems with normally distributed uncertainties.

The presence of nonlinearity in applications motivates extending these methods to *nonlinear estimation problems*.

In this remark, we will describe how to use linearization to apply the previously introduced estimation techniques to nonlinear problems. In the context of dynamical systems, this extends the Kalman filter and information filter to the extended Kalman filter (EKF) and extended information filter (EIF). Note that there exist also alterative techniques for treating nonlinearity such as the *unscented transform* employed by the unscented Kalman filter, which models the nonlinear transformation of distributions by sampling the distributions at discrete points, which can then be transformed directly by such a nonlinear transformation.

Generally, linearization is performed by employing first order Taylor approximation (approximation via the derivative). In the linearized picture, all operations of linear estimation can then be performed as usual. The expansion point for the linearization should be near the real system state in order for the approximation to resemble the real problem closely. Hence, an estimate of the system state (cf. e.g. Remark 13) is typically used as the expansion point.

So suppose given a quadratic loss function in the form

$$q(x) = (x - x_0)^T I (x - x_0) + q(x_0),$$

cf. Lemma 34. The estimate x_0 is used as expansion point for the substitution $x = f(y)$. I.e.

$$\begin{aligned} x \approx & \left.\frac{\mathrm{d}f}{\mathrm{d}y}\right|_{y=y_0} (y - y_0) + x_0 \\ = & \left.\frac{\mathrm{d}x}{\mathrm{d}y}\right|_{y=y_0} (y - y_0) + x_0, \end{aligned}$$

yielding the transformed quadratic loss function $\tilde{q}$ via

$$\begin{aligned} q(x) \approx & \left(\left. \frac{\mathrm{d}x}{\mathrm{d}y} \right|_{y=y_0} (y - y_0) \right)^T I \left(\left. \frac{\mathrm{d}x}{\mathrm{d}y} \right|_{y=y_0} (y - y_0) \right) + q(x_0) \\ = & (y - y_0)^T \left(\left. \frac{\mathrm{d}x}{\mathrm{d}y} \right|_{y=y_0} \right)^T I \left(\left. \frac{\mathrm{d}x}{\mathrm{d}y} \right|_{y=y_0} \right) (y - y_0) + q(x_0) =: \tilde{q}(y). \end{aligned}$$

Note that the derivative $\frac{\mathrm{d}x}{\mathrm{d}y}$ enters quadratically into the transformation of the information matrix I.

5 Long-Range forward triangulation

In this section, we examine the information about the position of stationary objects gained by triangulation under the assumption that the camera movement is perfectly known, which gives an upper bound on the achievable triangulation accuracy for SfM in forward motion. Here, a particular objective is to compare stereo camera systems with monocular systems by distinguishing the influence of the stereo camera baseline and of the baseline caused by vehicle movement.

Forward-facing cameras in (consumer) automobiles are usually mounted in the upper part of the windscreen directly above or behind the rear-view mirror since this is the only position outside the field of view of the driver that is both covered by the windscreen wiper and provides a good position for viewing the outside. Due to the constrained space there, such camera systems are typically either monocular or binocular with short baselines (up to ~ 20cm). See fig. 5 for examples. This is in contrast to research or experimental vehicles, for which a popular method is to stack a sensor platform onto the roof of the vehicle, cf. e.g. [Geiger et al., 2013, Urmson et al., 2004], which allows much larger baselines for stereo camera systems.

Figure 5: Examples of forward facing camera packages in automobiles. Top row: Volvo XC60 (1st generation). Bottom row: Mercedes-Benz S-Class (W222)

The scenes encountered by a forward-facing vehicle-mounted camera are given by the vehicle movement. In particular, the most common situation is driving along a more or less straight road, which means movement along the optical axis. We are mostly interested in

obtaining the geometry of stationary objects, since moving objects cannot be recovered by structure from motion and there exist highly specialized systems for identifying, tracking and locating moving objects in the context of driver assistance systems.

Due to the requirement that a road needs to be (mostly) flat, there are usually no stationary objects on the path of movement. This mitigates the fact that for monocular systems (and to a lesser degree for binocular systems, cf. below), the distance for objects in the direction of movement cannot determined well. If the vehicle moves in a perfectly straight line, distance determination is, in fact, impossible for objects on that line. Furthermore, the obstacle-freeness of the road often causes viewing distances to be quite large. Viewing distances of several hundreds of meters are common in urban situations, with extreme cases of several kilometers (e.g. on highways). This is in contrast to most augmented reality applications where the viewing distance is in the range of meters or a few dozens of meters.

Scene description We make the following assumptions: The vehicle is driving in a straight line with constant speed v and the camera system captures the scene N times consecutively with regularly spaced time-intervals of length Δt in between. The camera system is looking into the direction of travel, i.e. the principal axis is parallel to the direction of movement. It is assumed that the vehicle movement is perfectly known or that at least its error is negligible for the purpose of triangulating distant objects. Furthermore, we assume that all other parameters (camera calibration etc.) are perfectly known, i.e. we only estimate the positions of the triangulated objects. Note that this is actually the worst case for the purpose of triangulating object positions since the monocular case has a singularity along the line of movement and the stereo case features a "smudged" version of that singularity. The baseline of the stereo setup is $B = 2h$. In order to have the same number of images captured by both the stereo and the mono setup (otherwise, the stereo setup has an advantage simply because it captures more images), the monocular setup is assumed to be a stereo setup with zero baseline, i.e. this is the case $h = 0$ (one could similarly assume the monocular setup to have twice the image frequency of the stereo setup).

We use a Cartesian coordinate system which is centered at the middle of the stereo setup at the time capturing the last frame, the z-axis is the principal axis, the x-axis is parallel to the stereo baseline (i.e. horizontal) and the y-axis points upward. In particular, at the i-th scene capture, the vehicle is at position $(0, 0, -w_i)$, with $w_N = 0$ and $w_i \geq 0$ for all i.

For numerical evaluations, the following assumptions are used.

- The focal length is $f \sim 1000$px, which is a typical value for automotive grade cameras.

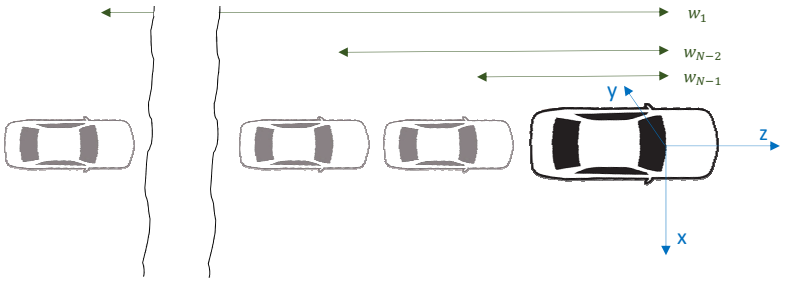


Figure 6: Coordinate system

- The image frequency is 30Hz
- The vehicle travels with $v = 10\text{m/s}$ ($\approx$ speed limit of urban areas)
- Objects are triangulated over a period of 1 second.
- For the stereo setup, the base line is $B = 20\text{cm}$, hence $h = 0.1\text{m}$. For the mono setup, we set $h = 0$.
- We constrain the field of view in such a way that $x^2 + y^2 \leq z^2$ (i.e. the angle of view is constrained to $\leq 90°$). Objects outside this region are at the boundary of or outside the camera field of view. Furthermore, they are at the side of the vehicle rather than in front and thus well outside the singularity along the axis of movement.

The singular values of the Jacobian Using the pinhole camera model, a point p at coordinate $p = (x, y, z)^T$ is projected by the left camera to the image point $k_i^+(p)$ and by the right camera to the image point $k_i^-(p)$, where $k_i^+(p)$ and $k_i^-(p)$ are given by

$$k_i^{\pm}(p) = f \begin{pmatrix} \frac{x \pm h}{z + w_i} \\ \frac{y}{z + w_i} \end{pmatrix},$$

where f is the focal length. Thus, the vector of the i-th measurement is given by

$$k_i(p) = f \begin{pmatrix} \frac{x+h}{z+w_i} \\ \frac{x-h}{z+w_i} \\ \frac{y}{z+w_i} \\ \frac{y}{z+w_i} \end{pmatrix}.$$

The full measurement vector $(k_i(p))_i$ has the Jacobian $(Dk_i(p))_i =: J$. The singular values of J determine the achievable precision when obtaining p by least-squares fitting to the measured data. To obtain these singular values, we compute the normal matrix

$J^TJ = \sum_i (Dk_i(p)^T Dk_i(p))$. We have the following.

$$Dk_i(p) = f \begin{pmatrix} \frac{1}{z+w_i} & 0 & -\frac{x+h}{(z+w_i)^2} \\ \frac{1}{z+w_i} & 0 & -\frac{x-h}{(z+w_i)^2} \\ 0 & \frac{1}{z+w_i} & -\frac{y}{(z+w_i)^2} \\ 0 & \frac{1}{z+w_i} & -\frac{y}{(z+w_i)^2} \end{pmatrix}$$

$$Dk_i(p)^T Dk_i(p) = 2f^2 \begin{pmatrix} \frac{1}{(z+w_i)^2} & 0 & -\frac{x}{(z+w_i)^3} \\ 0 & \frac{1}{(z+w_i)^2} & -\frac{y}{(z+w_i)^3} \\ -\frac{x}{(z+w_i)^3} & -\frac{y}{(z+w_i)^3} & \frac{x^2+y^2+h^2}{(z+w_i)^4} \end{pmatrix}$$

Then, setting

$$S_2 := \sum_i \frac{2}{(z+w_i)^2}, \qquad S_3 := \sum_i \frac{2}{(z+w_i)^3}, \qquad S_4 := \sum_i \frac{2}{(z+w_i)^4}, \tag{14}$$

we have

$$J^TJ = f^2 \begin{pmatrix} S_2 & 0 & -xS_3 \\ 0 & S_2 & -yS_3 \\ -xS_3 & -yS_3 & (x^2+y^2+h^2)S_4 \end{pmatrix}. \tag{15}$$

Note that here, the influence of the direction of the base line has vanished. That is when triangulating point-like features, the direction of the base line is actually irrelevant. Of course, this does not hold for the triangulation of anisotropic features such as edges. In particular, an edge cannot be triangulated at all if the edge is parallel to the camera base line. Note that algebraically, the variable h vanishes from all non-diagonal elements of J^TJ since the sum of the coordinates $(-h, 0)^T$, $(h, 0)^T$ of the two cameras in the plane perpendicular to the direction of movement is zero. In particular, similar results can be obtained for multi-view setups as long as the origin of the coordinate system coincides with the geometric center of the camera positions and as long as the cameras are located in a plane perpendicular to the direction of movement.

Since we eliminated the influence of the direction of the baseline, we may rotate x and y in such a way that the y-component vanishes. I.e. without loss of generality, we may assume that $y = 0$ and that x is the distance of p to the principal axis. This way, J^TJ becomes

$$J^TJ = f^2 \begin{pmatrix} S_2 & 0 & -xS_3 \\ 0 & S_2 & 0 \\ -xS_3 & 0 & (x^2+h^2)S_4 \end{pmatrix}. \tag{16}$$

We immediately obtain the eigenvector $(0, 1, 0)^T$ for the eigenvalue f^2S_2 of J^TJ. This

corresponds to the singular value

$$s_2 = f\sqrt{S_2} = f\sqrt{\sum_i \frac{2}{(z+w_i)^2}}$$

of the Jacobian J. We want to obtain the smaller one of the two remaining eigenvalues. These eigenvalues are solutions of the characteristic equation

$$\chi(\lambda) = \lambda^2 - f^2(S_2 + (x^2+h^2)S_4)\lambda + f^4(S_2S_4(x^2+h^2) - x^2S_3^2).$$

Its discriminant divided by f^4 is

$$\Delta f^{-4} = (S_2 + (x^2+h^2)S_4)^2 - 4(S_2S_4(x^2+h^2) - x^2S_3^2),$$

hence its solutions are given by

$$\begin{aligned}
\frac{\lambda_{1,2}}{f^2} =& \frac{(S_2 + (x^2+h^2)S_4) \pm \sqrt{\Delta f^{-4}}}{2} \\
\frac{\lambda_2}{f^2} =& \frac{(S_2 + (x^2+h^2)S_4) - \sqrt{\Delta f^{-4}}}{2} = \frac{(S_2 + (x^2+h^2)S_4)^2 - \Delta f^{-4}}{2((S_2 + (x^2+h^2)S_4) + \sqrt{\Delta f^{-4}})} \\
=& \frac{4(S_2S_4(x^2+h^2) - x^2S_3^2)}{2((S_2 + (x^2+h^2)S_4) + \sqrt{\Delta f^{-4}})} = \frac{2(S_2S_4(x^2+h^2) - x^2S_3^2)}{(S_2 + (x^2+h^2)S_4)(1+\phi)},
\end{aligned}$$

where $\phi := \frac{\sqrt{\Delta f^{-4}}}{S_2 + (x^2+h^2)S_4}$. Since the matrix J^TJ is the product of a matrix and its conjugate matrix, all eigenvalues are real and nonnegative. Since the eigenvalues are real, the discriminant Δ is nonnegative. Together with $S_2 > 0$, $S_4 > 0$, we obtain $\phi \geq 0$. The term $(S_2S_4(x^2+h^2) - x^2S_3^2)$ is the determinant of a submatrix of J^TJ (cf. eq. 16), so since the eigenvalues of J^TJ are nonnegative, we have $(S_2S_4(x^2+h^2) - x^2S_3^2) \geq 0$, which yields

$$\phi = \frac{\sqrt{(S_2 + (x^2+h^2)S_4)^2 - 4(S_2S_4(x^2+h^2) - x^2S_3^2)}}{S_2 + (x^2+h^2)S_4} \leq \frac{\sqrt{(S_2 + (x^2+h^2)S_4)^2}}{S_2 + (x^2+h^2)S_4} = 1.$$

Overall, we have

$$0 \leq \phi \leq 1. \tag{17}$$

Let us examine the formula for λ_2 further:

$$\lambda_2 = f^2\frac{2(S_2S_4(x^2+h^2) - x^2S_3^2)}{(S_2 + (x^2+h^2)S_4)(1+\phi)} = 2f^2\frac{h^2S_2S_4 + x^2(S_2S_4 - S_3^2)}{(S_2 + (x^2+h^2)S_4)(1+\phi)} \tag{18}$$

Note that

$$S_4 \cdot (x^2 + h^2) = \sum_i \frac{2(x^2 + h^2)}{(z + w_i)^4} \leq \sum_i \frac{2(x^2 + h^2)}{(z + w_i)^2 \cdot z^2} = \frac{x^2 + h^2}{z^2} S_2.$$

We typically have $|h| \ll |z|$. So since we constrain the field of view in such a way that $x^2 + y^2 \leq z^2$ (recall that we have used rotational symmetry to set $y = 0$), we obtain

$$S_4 \cdot (x^2 + h^2) \leq C_h S_2,$$

with $C_h := 1 + \frac{h^2}{z^2} \approx 1$.

Setting

$$\varphi := \frac{S_4(x^2 + h^2)}{S_2},$$

we obtain

$$0 \leq \varphi \leq C_h \tag{19}$$

We can thus refine (18) further to

$$\lambda_2 = 2f^2 \frac{h^2 S_2 S_4 + x^2(S_2 S_4 - S_3^2)}{S_2(1 + \varphi)(1 + \phi)}. \tag{20}$$

So eqs. (17) and (19) imply that the denominator of the right hand side of eq. (20) is asymptotically $S_2 \cdot \theta(1)$. As S_2, S_3, S_4 depend on z, but not on x, y and h, we may use eq. (20) as an asymptotic formula describing the influence of x and h on λ_2, which is the square of the smallest singular value $\nu_{\min}$ of the Jacobian J, that is $\nu_{\min} = \sqrt{\lambda_2}$.

As a motivation for the subsequent examination of formula (20), we will examine some values of $\frac{1}{\sqrt{\lambda_2}} = \frac{1}{\nu_{\min}}$.

In fig. 7, the magnitude of $(\nu_{\min})^{-1} = \lambda_2^{-\frac{1}{2}}$ is plotted. Only values up to 10m/px are plotted since this prevents crowded diagrams and there is a minimum amount of accuracy needed for distance information to be useful for augmented reality applications in urban scenarios.

It can clearly be seen that the monocular setup has a singularity along the axis of movement. No such singularity is present in the stereo setup, but the area near the axis of movement is still a zone where λ_2 has much lower values than in areas more distant from the axis of movement. Remarkably, the values of λ_2 in the mono and stereo setup are almost identical for positions several meters away from the axis of movement.

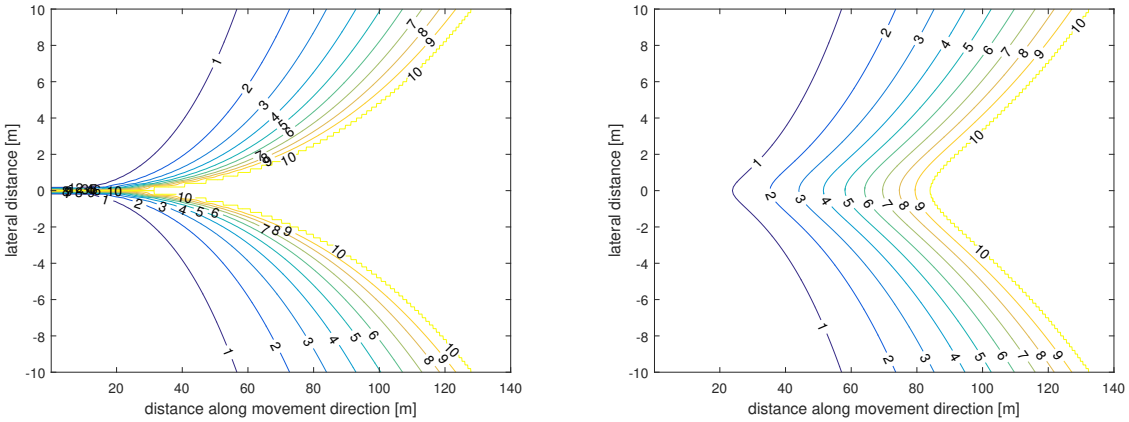


Figure 7: Magnitude of $\frac{1}{\sqrt{\lambda_2}} = \frac{1}{\nu_{\min}}$ [m/px] for monocular (left diagram) and stereo (right diagram) setups

Explicit approximations In the following, the formula (20) will be examined in detail. Here, the main objectives are to find explicit approximation for (20) and to separate the contribution of the stereo baseline from the contribution of the camera movement to the position information, that is to relate the contributions of x^2 and h^2 in (20).

To relate the contributions of the terms with x^2 and h^2 in eq. (20) to each other, we compute their quotient:

$$q := \left(2f^2 \frac{h^2 S_2 S_4}{S_2(1+\varphi)(1+\phi)}\right) \cdot \left(2f^2 \frac{x^2(S_2 S_4 - S_3^2)}{S_2(1+\varphi)(1+\phi)}\right)^{-1} = \frac{h^2 S_2 S_4}{x^2(S_2 S_4 - S_3^2)}$$

In order to evaluate q, we need a more accessible form for S_2, S_3 and S_4. Recall that the vehicle travels with constant speed and captures images at equally spaced time intervals. I.e. we have $w_i = v\Delta t(N-i)$ for $i = 1, \ldots, N$. Using the fact that integrals can be approximated by sums and vice versa (cf. e.g. the Euler-Maclaurin formula), and setting $w_{\min} := -\frac{1}{2}v\Delta t$ and $w_{\max} := (N - \frac{1}{2})v\Delta t$, we obtain approximations for S_2, S_3 and S_4 as follows:

$$\begin{aligned} S_2 &= \sum_{i=1}^{N} \frac{2}{(z+w_i)^2} \approx \int_{-1/2}^{N-1/2} \frac{2}{(z+i\cdot v\Delta t)^2} \mathrm{d}i = \frac{2}{v\Delta t} \int_{w_{\min}}^{w_{\max}} \frac{1}{(z+w)^2} \mathrm{d}w \\ &= \frac{2}{v\Delta t} \left[-\frac{1}{z+w}\right]_{w=w_{\min}}^{w_{\max}} = \frac{2}{v\Delta t}\left(\frac{1}{z+w_{\min}} - \frac{1}{z+w_{\max}}\right) =: I_2 \end{aligned}$$

Analogously, we obtain

$$S_3 = \sum_{i=1}^{N} \frac{2}{(z+w_i)^3} \approx [\ldots] = \frac{2}{v\Delta t} \int_{w_{\min}}^{w_{\max}} \frac{1}{(z+w)^3} \mathrm{d}w = \frac{2}{v\Delta t} \left[-\frac{1}{2(z+w)^2}\right]_{w=w_{\min}}^{w_{\max}}$$

$$
\begin{aligned}
&= \frac{1}{v\Delta t}\left(\frac{1}{(z+w_{\min})^2} - \frac{1}{(z+w_{\max})^2}\right) =: I_3, \\
S_4 &= \sum_{i=1}^{N} \frac{2}{(z+w_i)^4} \approx [\ldots] = \frac{2}{v\Delta t}\int_{w_{\min}}^{w_{\max}} \frac{1}{(z+w)^4}\mathrm{d}w = \frac{2}{v\Delta t}\left[-\frac{1}{3(z+w)^3}\right]_{w=w_{\min}}^{w_{\max}} \\
&= \frac{2}{3v\Delta t}\left(\frac{1}{(z+w_{\min})^3} - \frac{1}{(z+w_{\max})^3}\right) =: I_4.
\end{aligned}
$$

Setting $c := \frac{z+w_{\min}}{z+w_{\max}}$, we obtain

$$
\begin{aligned}
q &= \frac{h^2 S_2 S_4}{x^2(S_2 S_4 - S_3^2)} \\
&\approx \frac{h^2 I_2 I_4}{x^2(I_2 I_4 - I_3^2)} = \frac{h^2 (z+w_{\min}) I_2 (z+w_{\min})^3 I_4}{x^2((z+w_{\min}) I_2 (z+w_{\min})^3 I_4 - ((z+w_{\min})^2 I_3)^2)} \\
&= \frac{h^2 \cdot 2(1-c)\cdot\frac{2}{3}(1-c^3)}{x^2(2(1-c)\cdot\frac{2}{3}(1-c^3) - (1-c^2)^2)} \\
&= \frac{4h^2(1-c)(1-c^3)}{x^2(4(1-c)(1-c^3) - 3(1-c^2)^2)} = \frac{4h^2(1+c+c^2)}{x^2(4(1+c+c^2) - 3(1+c)^2)} \\
&= \frac{4h^2(1+c+c^2)}{x^2(c^2-2c+1)} = \frac{4h^2(1+c+c^2)}{x^2(c-1)^2} \\
&= \frac{4h^2(1+c+c^2)}{x^2(\frac{z+w_{\min}}{z+w_{\max}} - 1)^2} = \frac{4h^2(1+c+c^2)}{x^2(\frac{w_{\min}-w_{\max}}{z+w_{\max}})^2} \\
&= (1+c+c^2)\left(\frac{z+w_{\max}}{x}\frac{2h}{w_{\max}-w_{\min}}\right)^2.
\end{aligned}
$$

Note that for $z > -w_{\min}$, the inequality $w_{\max} > w_{\min}$ implies $0 < c < 1$. For long range situations (i.e. large z), we have $c \approx 1$, thus

$$
q \approx 3\left(\frac{z+w_{\max}}{x}\frac{2h}{w_{\max}-w_{\min}}\right)^2.
$$

Hence, the ratio of the contributions of h and x in eq. (18) depends mainly on the following terms:

- $\left(\frac{z+w_{\max}}{x}\right)^{-1} = \frac{x}{z+w_{\max}}$: This is the tangens of the angle between the direction of movement and the direction which the object is first seen. I.e. this corresponds to distance of the object from the image center.
- $\frac{2h}{w_{\max}-w_{\min}}$: This is the ratio between the stereo baseline and the distance travelled by the vehicle in the capture time.

Clearly, there is a circular cone around the axis of movement (which coincides with the view direction) at whose surface the contributions of h and x in eq. (18) are equal. It has the following properties: For objects inside the cone, the stereo setup gives much better triangulation results than the mono setup; the latter features a singularity there. For

objects outside the cone the mono and stereo setup give similar triangulation performance. The "width" (aperture) of the cone is determined by the ratio $\sqrt{3}\frac{2h}{w_{\max}-w_{\min}}$. Recall that $w_{\max} - w_{\min} = (N+1)v\Delta t \approx Nv\Delta t$ is approximately the distance travelled between the capture of the first camera frame and the capture of the last camera frame. This way, larger velocity or observation time values result in a smaller width of the cone.

Usually, this cone is quite narrow: In the above example with $h = 0.1$m and $w_{\max} - w_{\min} = 10$m ($v = 10$m/s and 1s observation time), we have $\sqrt{3}\frac{2h}{w_{\max}-w_{\min}} \approx 0.035$. That is at a distance of 100m, the cone extends just 3.5m around the line of movement. In particular, the influence of the longitudal movement on triangulation performance dominates the influence of the stereo baseline for most objects encountered in urban scenes for augmented reality applications.

Let us return to (20). Inserting the approximations I_2, I_3, I_4 for S_2, S_3, S_4 yields

$$\begin{aligned}
\lambda_2 \approx & 2f^2 \frac{h^2 I_2 I_4 + x^2(I_2 I_4 - I_3^2)}{I_2(1+\varphi)(1+\phi)} \\
= & 2f^2 \frac{h^2(z+w_{\min})I_2(z+w_{\min})^3 I_4 + x^2\left((z+w_{\min})I_2(z+w_{\min})^3 I_4 - ((z+w_{\min})^2 I_3)^2\right)}{(z+w_{\min})^4 I_2(1+\varphi)(1+\phi)} \\
= & 2f^2 \frac{h^2 \frac{2}{v\Delta t}(1-c)\frac{2}{3v\Delta t}(1-c^3) + x^2\left(\frac{2}{v\Delta t}(1-c)\frac{2}{3v\Delta t}(1-c^3) - \left(\frac{1}{v\Delta t}(1-c^2)\right)^2\right)}{(z+w_{\min})^3 \frac{2}{v\Delta t}(1-c)(1+\varphi)(1+\phi)} \\
= & \frac{2f^2}{v\Delta t} \frac{h^2 \frac{4}{3}(1-c)^2(1+c+c^2) + x^2\left(\frac{4}{3}(1-c)^2(1+c+c^2) - (1-c)^2(1+c)^2\right)}{2(z+w_{\min})^3(1-c)(1+\varphi)(1+\phi)} \\
= & \frac{2f^2(1-c)}{v\Delta t} \frac{h^2 \frac{4}{3}(1+c+c^2) + x^2\left(\frac{4}{3}(1+c+c^2) - (1+c)^2\right)}{2(z+w_{\min})^3(1+\varphi)(1+\phi)} \\
= & \frac{2f^2(1-c)}{v\Delta t} \frac{h^2 \frac{4}{3}(1+c+c^2) + \frac{x^2}{3}(1-2c+c^2)}{2(z+w_{\min})^3(1+\varphi)(1+\phi)} \\
= & \frac{f^2(1-c)}{3v\Delta t} \frac{B^2(1+c+c^2) + x^2(1-c)^2}{(z+w_{\min})^3(1+\varphi)(1+\phi)}.
\end{aligned}$$

We want to perform long-range asymptotics for this formula. Note that

$$\begin{aligned}
1 - c = & 1 - \frac{z+w_{\min}}{z+w_{\max}} = \frac{(z+w_{\max}) - (z+w_{\min})}{z+w_{\max}} \\
= & \frac{w_{\max} - w_{\min}}{z+w_{\max}}.
\end{aligned}$$

Hence setting $u := \frac{x}{z}$ and $d := \frac{1}{z}$, we have

$$\begin{aligned}
\lambda_2 \approx & \frac{f^2(1-c)}{3v\Delta t} \frac{B^2(1+c+c^2) + x^2(1-c)^2}{(z+w_{\min})^3(1+\varphi)(1+\phi)} \\
= & \frac{f^2(w_{\max} - w_{\min})}{3v\Delta t(z+w_{\max})} \frac{B^2(1+c+c^2) + x^2\left(\frac{w_{\max}-w_{\min}}{z+w_{\max}}\right)^2}{(z+w_{\min})^3(1+\varphi)(1+\phi)}
\end{aligned}$$

$$
\begin{aligned}
\approx & \frac{f^2(w_{\max}-w_{\min})}{3v\Delta t(z+w_{\max})}\frac{B^2(1+c+c^2)+u^2\left(w_{\max}-w_{\min}\right)^2}{(z+w_{\min})^3(1+\varphi)(1+\phi)} \\
\approx & \frac{f^2(w_{\max}-w_{\min})d^4}{3v\Delta t}\frac{B^2(1+c+c^2)+u^2\left(w_{\max}-w_{\min}\right)^2}{(1+\varphi)(1+\phi)} \\
= & \frac{(N+1)f^2d^4}{3}\frac{B^2(1+c+c^2)+\left(u(w_{\max}-w_{\min})\right)^2}{(1+\varphi)(1+\phi)} \\
\approx & Nf^2d^4\left(B^2+\frac{1}{3}\left(u(w_{\max}-w_{\min})\right)^2\right).
\end{aligned}
$$

Hence, we obtain

$$
\frac{1}{\sigma_{\min}} = \frac{1}{\sqrt{\lambda_2}} \approx \frac{1}{\sqrt{N}fd^2}\frac{1}{\sqrt{B^2+\frac{1}{3}\left(u(w_{\max}-w_{\min})\right)^2}}.
$$

This means that the asymptotic sensitivity of the estimation error to the measurement error depends on the following variables:

- number of captured frames (information increases linearly with N)
- focal length (linear contribution)
- the inverse of the distance (quadratic contribution)
- stereo baseline B (linear contribution when dominant in sum)
- product of distance from image center u and traveled distance $(w_{\max}-w_{\min})$ (linear contribution when dominant in sum)

This concludes the theoretical analysis of the geometry of the triangulation problem.

Numerical simulation In the following, we perform some basic numerical simulations to evaluate the accuracy of the monocular triangulation of positions in front of the vehicle from noisy observations. We use the same assumptions on focal length, image frequency, vehicle speed etc. as before. Furthermore, it is assumed that the measurement noise is normally distributed (in practice, this assumption must be weakened e.g. due to mismatching of features, cf. e.g. [Triggs et al., 1999, section 3], but it is a suitable simplification for the present geometrical analysis) and we use least squares estimation for position triangulation. We let the standard deviation of the noise be $\sigma = 1$px. For least-squares estimation, a slightly damped variant of the Gauss-Newton algorithm is used. We evaluate the performance at object positions arranged in a regularly spaced two-dimensional grid (recall that we may eliminate the third dimension by suitably rotating the scene around the line of movement). For each position, we perform a Monte-Carlo simulation with 2000 samples to infer suitably precise estimates of bias and covariance of the triangulation. We calculate the experimental bias by subtracting the true position of the triangulated point from the

sample mean of the position estimates. We use the (unbiased) sample variance as estimate of the covariance.

In Remark 25, we established that the sensitivity of the triangulation error to measurement noise can be approximated by $\nu_{\min}^{-1}$, where $\nu_{\min}$ is the smallest singular value of J (recall that in our case, we have $\nu_{\min} = \sqrt{\lambda_2}$). Hence, we expect the triangulation error covariance to be about $\sigma\nu_{\min}^{-1}$. Fig. 8 in combination with $\sigma = 1$px shows that this expectation holds,

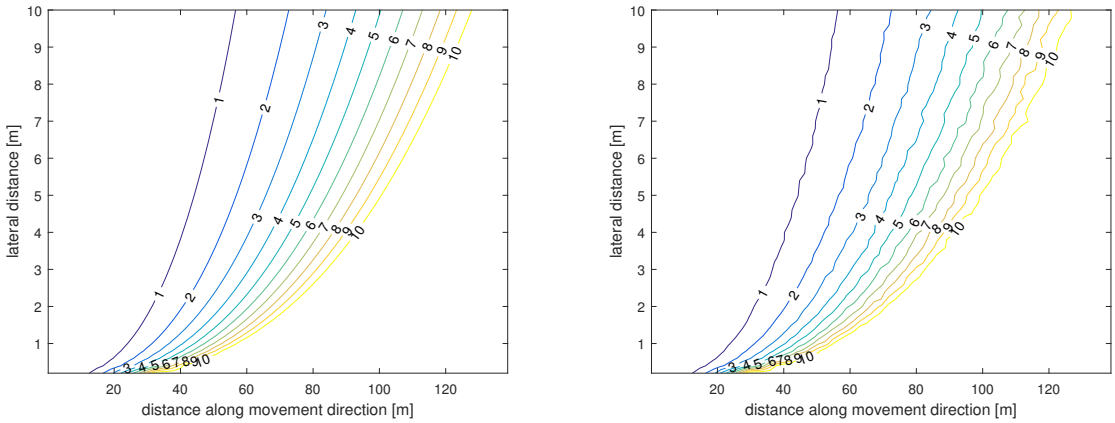


Figure 8: Left: Magnitude of $\frac{1}{\sqrt{\lambda_2}}$ [m/px] for monocular setup. Right: Monte Carlo estimation of the squareroot of the largest eigenvalue of the covariance matrix in meter.

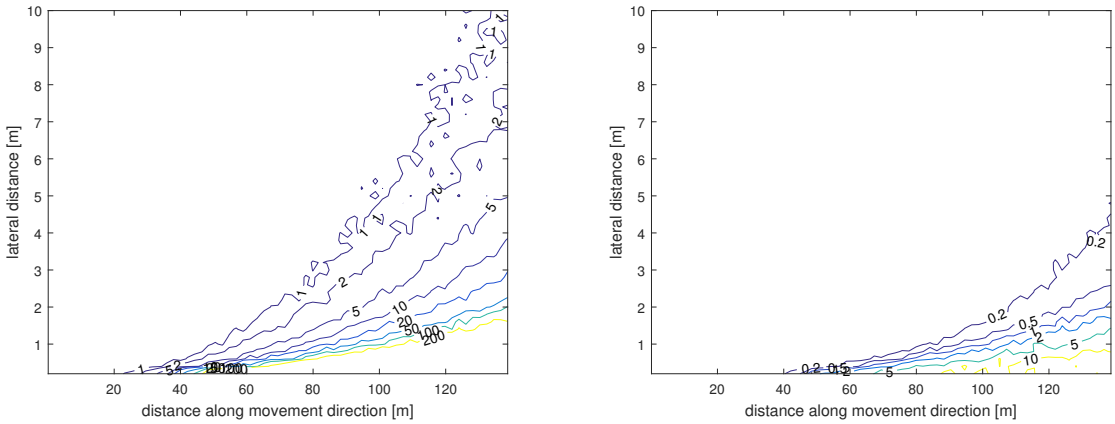


Figure 9: Bias of the monocular setup obtained from Monte Carlo simulation [m]. Left: bias component along movement direction. Right: lateral bias component

which supports the approximations made above.

The Monte Carlo simulation was also used in fig. 9 to experimentally determine the bias of the triangulation. Note that in case the triangulated object is several meters apart from the line of movement, the bias is moderate. However, near the line of movement the bias

quickly grows to severe overestimation of the true distance. This can be attributed to the fact that the singularity along the line is the center of a zone of strong nonlinearity surrounding it, which skews the centered measurement error distribution via nonlinear effects into the non-centered estimation error distribution. Note that the triangulation problem is generally much more linear in pinhole coordinates (cf. Example 8), so in situations where triangulated positions are averaged (e.g. for smoothing a depth map), pinhole coordinates should be used for averaging to reduce the influence of nonlinearity.

Summary In the above, the triangulation of distant objects for forward-facing vehicle-mounted monocular and binocular cameras has been analyzed. In particular, the (worst) case when the vehicle is driving with constant speed in a straight line has been scrutinized in detail. Despite the nonlinear nature of the measurement process, several quite explicit formulas for the Jacobian and values derived from it such as its smallest singular values have been obtained. A key technique for facilitating the analysis of the Jacobian is the approximation of sums via integrals, which allowed us to simplify the sums appearing in the Jacobian to a great degree and to interpret the results in simple, geometrical terms. In particular, we have shown that the geometrical advantage of stereo setups over monocular setups is mostly restricted to objects in a narrow cone around the principal axis. Outside this cone, the advantage of a stereo setup is mostly caused by the fact that a stereo camera captures twice as many images as a monocular camera. We argued that for objects of interest to triangulation in vehicular Augmented Reality applications, there is typically several meters distance to the singularity at the principal axis. We have shown that at this distance to the principal axis, we can recover distance information with sufficient precision to e.g. distinguish buildings from each other up to a viewing distance of about 100m, even with a monocular setup.

6 Structure from motion in forward motion

6.1 Introduction

Monocular structure from motion has long been known to be problematic in the case of forward motion. This becomes already apparent when the subproblem of triangulation is considered, since this problem features a singularity along the movement direction, which is right in the middle of the camera image. This issue is analyzed in detail in section 5. When moving to the full SfM problem in forward motion, there arise poles in the objective function that are caused by the great sensitivity of point estimates near the epipolar direction to errors in the estimation of the epipolar direction. This phenomenon is an intrinsic part of the SfM problem in forward motion and lies arguably at the root of why so many visual odometry methods fail when faced with monocular forward motion. A basic idea behind our approaches to the SfM problem is the concept that this problem can be made smaller if the error in the estimation of the epipolar direction can be reduced relative to the image resolution. We take the somewhat simplistic approach of involving as many features in the camera image as possible into the estimation of the camera movement. This means involving (almost) every pixel in the image in visual odometry, hence the algorithms used need to scale up easily to large amounts of features.

6.2 Rationale for approximating the information matrix in filter-based visual odometry

Recall that filter-based visual odometry methods maintain a covariance estimate for the errors of the distance estimates. This covariance estimate typically is in the form of a covariance matrix or information matrix (for simplicity, we will assume to use the information variant in the sequel), whose number of rows and columns depend linearly on the number of tracked features. Thus, the size of this information matrix is quadratic in the number of features.

An approach for circumventing this is to simply replace the information matrix by a matrix which is less complex. From estimation theory, we know that we may discard information by subtracting a positive semidefinite matrix from the information matrix. Hence, the replacement described above is allowed as long as the difference matrix between the covariance matrix and the replacement is positive semidefinite. In the following, we will provide some rough calculations which show that the information matrix becomes nearly diagonal for large numbers of features, which motivates replacing the information matrix by a diagonal matrix or a similar structure that can be maintained with low complexity.

The residual of the VO problem for feature position estimates p_i and camera pose estimates

k_j is

$$r := \sum_{i,j} \|m_{ij} - f \cdot K_0(p_i, k_j)\|^2. \tag{21}$$

Here, f is the camera focal length, m_{ij} is the observation of point p_i at the j-th camera position and the function K_0 models the observation process. We separate the focal length from K since we want to examine the influence of the image resolution. Defining the vectors $m := (m_{ij})_{i,j}$, $k = (k_j)_j$, $p = (p_i)_i$ and the vector valued function $K = K(p, k) = (K_0(p_i, k_j))_{i,j}$, we obtain

$$r = \|m - fK(p, k)\|^2. \tag{22}$$

Let $fE := \frac{\mathrm{d}r}{\mathrm{d}p} = f\frac{\mathrm{d}K}{\mathrm{d}p}$ be the block derivative of all feature positions and $fF := \frac{\mathrm{d}r}{\mathrm{d}k} = f\frac{\mathrm{d}K}{\mathrm{d}k}$ be the block derivative of all camera poses. The extended information matrix is $f^2 \begin{pmatrix} E^T \\ F^T \end{pmatrix} \begin{pmatrix} E & F \end{pmatrix}$, cf. Remark 25. By Remark 22, the information matrix of the pose estimates is

$$f^2 E^T (1 - F(F^T F)^+ F^T) E.$$

I.e. this is the matrix $E^T E$, which has diagonal structure (different point estimates are independent) disturbed by a term describing the pose estimation uncertainty. Let us consider varying the image size.

Firstly, note that increasing the focal length f (which is a major effect of increasing the image resolution) increases the overall information. However, changing f does *not* change the ratio between the per-point information and the uncertainty due to pose uncertainty.

The second major effect of increasing the image resolution is that the number of scene points usable for VO is increased, both for feature-based methods and for direct methods. Let us extend E and F accordingly: E is replaced by a matrix $\bar{E} = \begin{pmatrix} E & 0 \\ 0 & \hat{E} \end{pmatrix}$, F is replaced by $\bar{F} = \begin{pmatrix} F \\ \hat{F} \end{pmatrix}$. This way, we have $\bar{F}^T \bar{F} = F^T F + \hat{F}^T \hat{F} \geq F^T F$ as quadratic forms. If F and $\bar{F}$ have the same column rank (which is satisfied if the increasing the number of scene points does not change the unobservable degrees of freedom of the system), this implies $(\bar{F}^T \bar{F})^+ \leq (F^T F)^+$, cf. e.g. [Birman and Solomjak, 1980, 10.2, Theorem 6]. This way, the influence of the camera pose uncertainty relative to the per-point information is smaller in $\bar{f}^2 \bar{E}^T (1 - \bar{F}(\bar{F}^T \bar{F})^+ \bar{F}^T) \bar{E}$ than in $f^2 E^T (1 - F(F^T F)^+ F^T) E$. Hence, increasing the number of scene points makes the information matrix more closely similar to a diagonal matrix.

This analysis shows that approximating the information matrix by a matrix with diagonal

structure is a valid approach for sufficiently large number of scene points. This result is encouraging since using high resolution images and as many scene points as possible is already a prerequisite for achieving high accuracy. Furthermore, matrices with diagonal structure can be maintained much more efficiently computationally than the full information matrix due to the former having linear complexity in the number of scene points and the latter having quadratic complexity.

Note however, that in addition to the number of scene points, the efficiency of the above approximation of the information depends also on the number of observations per scene point. Hence, there is motivation to develop more exact approximations of the information matrix, in particular for high speed situations which are common for vehicle-mounted cameras.

Furthermore, while second-order effects such as bias may be reduced by increasing the number of scene points, there is probably no simple way of eliminating them. This contrasts VO methods based on bundle adjustment since for these, (semi-)global optimization implicitly removes second and higher order effects.

6.3 Introduction to C-LSD-SLAM

The visual odometry method we developed is called C-LSD-SLAM. As its name implies, it is based on LSD-SLAM, which is a method developed by researchers at TU Munich, cf. [Engel et al., 2014]. Basing C-LSD-SLAM on LSD-SLAM allowed us to start from an existing framework, which avoids bootstrapping issues. Furthermore, LSD-SLAM produces semi-dense depth maps, which is already close to the dense depth map required by the intended application. Finally as LSD-SLAM is a direct method, the computation steps typically involve iteration over all pixels, which facilitates parallelization. This contrasts feature based methods since feature extraction and matching are difficult to parallelize.

LSD-SLAM is a keyframe based method. As such, semi-dense depth maps are only available for keyframes, which are a strict subset of all frames. As most operations (tracking, mapping, keyframe reprojection) are performed relative to the most recent keyframe, this approach is problematic for the fast motion encountered by a vehicle-mounted camera since there, the distance between a keyframe and its succeeding keyframe may be too large to give a sufficient overlap. Furthermore, our intended application needs a depth map for each frame. Hence, the most obvious change from LSD-SLAM to C-LSD-SLAM is the fact that in C-LSD-SLAM, every frame is a keyframe. Instead of the asynchronous tracking and mapping used in LSD-SLAM styled after PTAM, C-LSD-SLAM performs the following steps for each frame in a synchronous fashion. Given the last, "old" frame, which is a keyframe since it has a depth map, a "new" frame is processed as follows

(1) Tracking of the new frame relative to the old frame

(2) Updating of the depth map of the old frame via triangulation with the new frame.

(3) Reprojection of the depth map of the old frame to the new frame. This way, the new frame obtains a depth map and becomes a keyframe.

In between, there are several smoothing and regularization steps for the depth map, which we have omitted for simplicity. To simplify the method, the SLAM functionality present in LSD-SLAM has been removed in C-LSD-SLAM. So despite its name, C-LSD-SLAM is not a SLAM method but a visual odometry method though arguably the SLAM functionality could easily be re-integrated.

6.4 C-LSD-SLAM: Implementation

C-LSD-SLAM is a filter-based visual odometry method, so its core state is the filter state. The filter state is encoded in a central structure ("DepthMap"), which features some global filter data and, much more importantly, certain per-pixel data. There are the following core per-pixel data for maintaining the estimated geometry of the scene in front of the camera:

- inverse depth estimate (float value)
- information value for inverse depth estimate (float value)
- validity flag

The validity flag indicates whether the pixel is valid, that is whether there is actually an estimate for the pixel present since there are various conditions that can prevent the method from assigning an estimate to the pixel. Note that it might make sense from the perspective of data size to encode the validity into the information value (e.g. positive values indicate a valid pixel). However, as validity is required for several routines without needing to know the information value and since testing a Boolean value is typically a well-optimized operation on modern processors (in particular compared to examining a float value), it does not make much sense in practice.

In addition to the above data, there are the following auxiliary per-pixel data:

- smoothed inverse depth estimate (float value)
- information value of smoothed inverse depth estimate (float value)
- validity counter (integer)

The map of smoothed depth estimates are obtained from the map of "raw" depth estimates via smoothing. While smoothing is not absolutely necessary, several routines (image alignment, depth estimation) benefit from the outlier suppression of regularizing the depth

map. The validity counter is an auxiliary value that is used for filling gaps in the depth map.

6.4.1 Some notes on parallelization techniques

Many visual odometry methods follow the parallel tracking and mapping paradigm for parallelization (e.g. [Forster et al., 2014, Engel et al., 2014, Mur-Artal et al., 2015]), which was popularized by the method aptly called parallel tracking and mapping (PTAM) [Klein and Murray, 2007] in 2007. A major reason given for this is the ability to decouple tracking and mapping in the context of real-time processing. In particular, this allows different timing constraints for tracking and mapping. E.g. while tracking usually runs in a real-time fashion, the mapping component can perform more complex processing on select keyframes or can process batches of consecutive frames in the manner of local bundle adjustment. Running tracking and mapping in separate threads which can run concurrently firstly allows the tracking thread to preempt other computations, which may be used to meet real-time constraints, and secondly improves usage of available resources in multi-core systems. We argue that the efficient usage of multi-core systems alone does not justify the high amount of complexity required by this style of concurrent processing any more. Indeed, since typically tracking uses a map as reference and mapping uses tracking results for generating and updating maps, tracking and mapping operate on and modify the very same data, which requires fairly complex synchronization methods to prevent race conditions without overly harming efficiency. On the other hand, processing units with multiple cores, each featuring simultaneous multithreading and/or SIMD vectorization, have as of 2017 become ubiquitous. E.g. in the desktop segment, current high-end systems have dozens of logical cores and the newest SIMD variant of the x86 architecture (AVX-512) features 512-bit vector registers. Multi-core systems are standard in the mobile sector, with SIMD provided e.g. by ARM's NEON instruction set. GPUs have been using massively parallel processing for decades, with GPUs featuring high numbers of wide and highly multithreaded SIMD units.

Thus, in order to effectively make use of these kind of parallelized computing units, each significant processing step of a visual odometry method should be fully parallelized and vectorized, otherwise that processing step will become a bottleneck. This way, each processing step can make usage of all of the available processing power. The choice whether different processing steps run concurrently or sequentially is then mainly a choice of latency and of thread synchronization complexity, not of total processing time.

6.4.2 Frame tracking

The frame tracking component determines the pose of a new frame relative to a given keyframe, that is relative to a frame for which a depth map is available.

As in the original LSD-SLAM, frame tracking proceeds via direct image alignment. That is, the valid pixels of the keyframe are reprojected to the pose candidate and the pixel intensity of the old and new frame are compared. The intensity difference is robustified via the Huber loss function and summed over all pixels to obtain the reprojection residual. Minimization of the nonlinear residual proceeds via Gauss-Newton iteration. To further facilitate the nonlinear minimization, this is performed in a coarse to fine scheme. That is, the image resolution (and depth resolution) is halved multiple times and the first instance of Gauss-Newton optimization is performed on the most coarse resolution. The pose estimate resulting from this optimization is then used as the initial guess for the optimization on the next-large resolution. This is repeated, yielding finer and finer estimates until the native image resolution is reached.

The result of this estimation are the following.

- A pose estimate of the new frame relative to the keyframe
- The reprojection residual
- An estimate of the difference of lighting conditions of the new frame relative to the keyframe. This estimation is performed simultaneously with the pose estimation and used in both frame tracking and depth estimation to compensate for lighting changes.

Compared with LSD-SLAM, the tracking module of C-LSD-SLAM features the following innovations. Firstly, the reprojection is streamlined via projective formulation. LSD-SLAM uses Cartesian coordinates in an intermediate step for reprojection, which requires several excess operations and more importantly is vulnerable to divisions by zero for far away points. Instead, C-LSD-SLAM performs the transformation fully in pinhole coordinates. For these, the transformation from old to new frame is a projective transformation, hence the reprojection is essentially performed by multiplying the pinhole coordinates of a pixel with a 4×4 homogeneous transformation matrix and performing a normalization step for the resulting homogeneous coordinates. Key advantages of this variant is that it is simple, it is fast and works with *any* inverse depth such as points which lie on the plane at infinity (zero inverse depth) and works even with points which lie beyond the plane at infinity. This way, the algorithm allows freedom to choose what kind of pixel estimates are used for image tracking.

Concerning a fast implementation, note that the steps required for each pixel are the following:

- Reproject the pixel (cf. Example 9)
- Check whether it is still in the camera FoV and still in front of the camera (cf. Example 9 for the latter check), discard the pixel otherwise.
- Calculate residuals and derivatives of residuals

These steps are independent for each pixel, so they lend themselves easily to parallelization. The resulting residuals and derivatives then have to be summed over all pixels. This feat is known as a parallel reduction step and is a standard problem in High Performance Computing. There exist efficient solutions for parallel reduction for almost any system featuring parallel processing. In the present implementation, both multithreading and SSE2 SIMD vectorization are used for parallelization. Accumulation in thread-local accumulation variables is used for parallel reduction. Due to the small number of CPU threads available, the computational cost of summing over the thread-local variables in the end is negligible.

To further reduce the computation time for frame tracking, a subsampling scheme is used to involve less pixels of the keyframe in the image alignment. This is motivated by the fact that due to various error sources, image alignment is an inexact process. Indeed, the information provided by the depth map of the keyframe is much more information than can be used for image alignment. Hence, image alignment is virtually unaffected if the collection of input pixels is thinned. Decimation is performed for the various resolution levels in such a way that for each non-decimated pixel there is a non-decimated pixel in the next-coarser level covering it. This way, the objective function remains mostly consistent between resolution levels. Among resolution levels, decimation starts with a decimation factor of 1 (no decimation) and, beginning with a certain level, the decimation factor is doubled for each succeeding level. This way, the levels with highest resolution have also the highest decimation factor. In practice, we decimate by eliminating complete image rows. That is a decimation factor of 1 means every image row is used. A decimation factor of 2 means that every other image row is skipped. A decimation factor of 4 means that only every fourth row is used and so on. Skipping whole rows has the advantage that the above rule that every non-decimated pixel should have a "parent" in the previous resolution level can be implemented easily. Furthermore, it results in good data access patterns, thus improving caching and performance.

In practice, we e.g. obtain a speedup of a factor of about 4 over the variant without decimation when using a decimation factor of 8 at the level with highest resolution.

Recall that the tracking of a new frame relative to the last frame proceeds by direct image alignment. In particular for scenes with repetitive structures, the success of this optimization scheme depends on availability of a reasonable initial guess. To improve the performance in situations with high turn rates, C-LSD-SLAM uses the pose change between the last frame and its predecessor as the initial guess for the pose change between

the last frame and the new frame.

6.4.3 Reprojection to a new keyframe

Reprojection is performed when the estimated geometry of a keyframe (i.e. its depth map) is to be transferred to a new keyframe. For this step, the pose of the new keyframe relative to the old keyframe needs to be known, e.g. from frame tracking. The reprojection step reprojects the pixels from the depth map of the old keyframe to the new keyframe, performs information reweighting according to the extended information filter or a variant thereof and resolves collisions of reprojected pixels.

Reprojection itself is straightforward (cf. Example 9). We discuss information reweighting in detail in section 6.6. Resolving pixel collisions is the main issue affecting implementation. By pixel collision we denote the phenomenon that in reprojection, multiple source pixels may be projected to the same target pixel slot. Without collision handling, the last of these source pixels would simply fill the target pixel slot, thus overwriting the other source pixels and destroying their information content unconditionally. LSD-SLAM detects collisions by checking at reprojection of a source pixel whether there is already a pixel present in the target pixel slot. Upon detection of a collision, it performs one of the following operations. If the two pixels are far enough apart that the information values of their depth estimates (e.g. the inverses of the estimation covariances) indicate that the points are distinct, the situation is regarded as occlusion, so the closer one of the two pixels is retained while the more distant one is discarded. Otherwise, the two estimates are merged as two independent estimates of the same point.

Note that this variant for resolving pixel collisions requires that the source pixels are processed sequentially since otherwise, several source pixels might arrive at a target pixel slot concurrently thus breaking the sequential nature of the above collision handling. In fact, the minimum requirement for this is that for each target pixel, the source pixels arriving at this target pixel need to be serialized.

There is no efficient way to perform this serialization in parallel: One problem preventing this is the fact that such a serialization would require parallel computation units to detect collisions among themselves. Since parallel units generally do not have suitable means for communicating directly with each other, such a detection would require the computation units to detect a collision from their interaction with the target pixel slot. I.e. it would require some sort of locking mechanism for each target pixel. While this might be feasible with CPU technology with some efficiency decrease, it would likely severely affect efficiency for other processor types such as GPUs.

The approach used in C-LSD-SLAM uses an approximate variant of collision resolution which allows it to write a reprojected source pixel to a target slot *without checking for*

collision. We denote this approach *stochastic collision avoidance*. For this, each target pixel does not feature one target slot, but multiple target slots called *bins*. The number of bins per pixel is constant (e.g. 2 bins per pixel or 4 bins per pixel). When a reprojected source pixel is to be written to a target pixel, one of the bins of the pixel is selected in a pseudorandom fashion and the pixel is then written unconditionally to that bin. The pseudorandom bin selection is designed in such a fashion that only few source pixels are written to the same bin (in practice this mostly boils down to preventing collisions between adjacent pixels in the source depth map). After reprojecting all source pixels in this fashion, a second step is then performed for actually resolving collisions. For each target pixel, this step regards which bins of the target pixel have been written with transformed source pixels and then resolves these collisions sequentially. Note that there are no dependencies between the target pixels, so this step can be performed in parallel for all target pixels.

A technical detail of writing to bins is that if there is concurrent writing to a bin from multiple source pixels, then the end result should correspond to either of the source pixels and should not be a mixture of the source pixels. I.e. writing to a bin needs to be an atomic operation. Hence, the size of a bin should be at most the maximum atomic write size of the processor (otherwise locking is required, which incurs a performance penalty). Nowadays, many processors are capable of writing at least 64 bits atomically (in particular, this is a standard feature of 64bit processors). Larger atomic write sizes are much rarer, so the data of a bin needs to be fit into 64 bits, which requires a fair amount of data squeezing.

6.4.4 The frame processing loop

We have already established that in order to process a new frame, several steps including tracking, mapping and keyframe reprojection are performed. In the following, the frame processing loop is explained in detail.

At the beginning of a frame cycle, the most recent frame is a keyframe, i.e. it has a depth map. Let us call this the "old" keyframe. At the arrival of a new frame, the following steps are performed:

(1) Preprocessing of the new frame (format conversion, rectification etc.)

(2) Tracking of the new frame relative to the old keyframe.

(3) Updating the depth map of the old keyframe by triangulation between the old keyframe and the new frame

(4) Regularization of the depth map of the old keyframe

(5) Reprojection of the depth map of the old keyframe to the new frame. This way, the

new frame becomes the next keyframe.

(6) Regularization of the depth map of the new frame and hole filling

The first regularization (i.e. smoothing) step is necessary since the regularized inverse depth values of the pixels are used for reprojection. However, note that in order to conserve the amount of information, the raw (non-regularized) information values of the inverse depth estimates have to be used for reprojection.

After reprojection, the depth map typically has various types of irregularities (gaps, effects of spurious occlusions etc.), so the second regularization step mitigates this and provides a smoothed depth map for both frame tracking and for triangulation.

6.4.5 Initialization

For visual odometry systems, initialization is a delicate step as misestimation of the initial state may cause the system to diverge due to the nonlinearity of the problem. For initialization, LSD-SLAM initializes the depth map of the first frame randomly. C-LSD-SLAM basically follows the same approach, but chooses a different distribution for the initial candidates. In particular, LSD-SLAM chooses the candidates for the inverse depths uniformly between a positive minimum value (maximum distance) and a maximum value (minimum distance). C-LSD-SLAM sets the minimum inverse depth to zero (i.e. infinite distance). This way, candidates for all distances are provided by the initialization (the minimum distance has no practical meaning since the scale factor is chosen arbitrarily by the system) and thus the initial state fits a much wider range of scenes. Notably for vehicle mounted cameras facing forwards, the region the camera is moving towards is the road in front of the vehicle, so it typically features large distances. The variant used in LSD-SLAM does not have viable candidate depths for this region, so the system needs to first reject all of these candidates before it can create valid depth estimates in this region, which is a process that often takes many frames. In the C-LSD-SLAM variant, convergence is much quicker, which also reduces the likelihood of the system diverging at initialization.

C-LSD-SLAM additionally features an initialization technique specifically targeted at car-mounted cameras: A car cannot turn without driving a curve, so a car-mounted camera (approximatively) does not rotate while moving in a straight line. Moving in straight line is the most difficult situation for initialization. C-LSD-SLAM assumes that the camera orientation does not change in the first few frames. If the camera orientation indeed stays approximatively the same (i.e. the vehicle is driving in a straight line), this greatly helps the initialization process. If the camera turns, the vehicle does not drive in a straight line, which is a much easier initial situation where the system can usually initialize successfully despite having made incorrect assumptions.

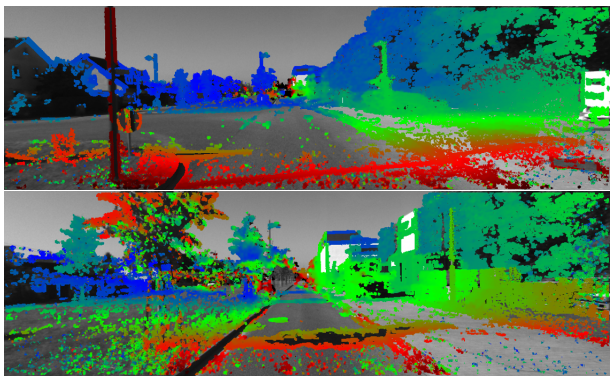


Figure 10: Instability of non-robustified C-LSD-SLAM on sequence 06 of the KITTI odometry challenge. Distance estimates of the depth maps are colored from blue (far) over green to red (near).

6.5 Instability in forward motion

In its bare implementation, C-LSD-SLAM suffers from unstable behaviour in forward motion. Since forward motion is the default situation of our targeted application, this is important.

In the following we will first describe and analyse this behaviour. Then, our mitigation method is introduced and analyzed, which is based on a reweighting strategy when reprojecting pixels from a keyframe to a new keyframe.

6.5.1 Description

For sufficiently fast forward motion, C-LSD-SLAM suffers from a type of instability where erroneous depth estimates propagate over the image, eventually causing tracking divergence or failure. For example in the sequence given in fig. 10, the upper image shows invalid distance estimates appearing near the epipolar direction, which then propagate mainly to the left. The lower image shows the situation several frames later. Here, invalid distance estimates have overtaken a large part of the image, causing tracking divergence several frames later.

The basic mechanism underlying this behaviour is as follows. Near the epipolar direction, the relative object motions are very small and their interpretation in terms of object distance is extremely sensitive to errors in the estimate for the epipolar direction, cf. Remark 27. Hence, distance estimates near the epipolar direction are often highly erroneous. Note that we assume that we are driving forward, i.e. right in the epipolar direction. I.e. we drive towards this zone of erroneous estimates near the epipolar direction. Among these erroneous estimates, those which are much closer than the correct distance are particularly problematic: Due to their large inverse depth, their predicted apparent motion is large, so

they move rather quickly away from the epipolar direction. If such an estimate happens to be confirmed by a mismatch (which is not uncommon, in particular if the images patches used for matching are small such as in LSD-SLAM and C-LSD-SLAM), the large apparent motion causes the incorrect estimate to have a high information value. Such incorrect estimates with high confidence often accumulate on one side of the epipolar direction (often covering a large area of the image), which influences the tracking. Tracking errors in turn cause more errors near the epipolar direction, which tends to cause runaway error amplification, with the ultimate result being the divergence of the method.

6.5.2 Analysis

In the following, we describe two effects which, in combination, may cause runaway error amplification as explained below.

Remark 27 (Disparity poles caused by an incorrect epipolar direction, cf. also [Vedaldi et al., 2007])**.** In this part, we describe an effect which causes poles of the disparity estimate near the epipolar direction. This is particularly important in forward motion since in forward motion, the epipolar direction and thus the poles are in the center of the camera image. In sideway motion, these poles are usually outside the image and thus less problematic.

For simplicity, we describe this in the 2D case. Suppose that the camera moves in a straight line forward. Suppose that the camera moves along two lines which are parallel to the movement direction, one on each side of the camera. In the 3D case, this is roughly equivalent to moving along structures parallel to the movement direction such as opposing building facades and the street surface. We will model the geometry in front of the camera via pinhole coordinates. As we parametrize the visible scene parts, we can use the image coordinate $\mathbf{u}$ as independent coordinate and model the inverse distance of the visible scene parts as $\mathbf{d} = \mathbf{d}(u)$. The transform from Cartesian coordinates to (2D) pinhole coordinates preserves lines cf. Definition/Remark 6, so the two lines parallel to the movement direction are also lines in pinhole coordinates. Hence, we may parametrize these lines via the equations

$$\begin{aligned} d_1(\mathbf{u}) =& a_1\mathbf{u} + b_1 \\ d_2(\mathbf{u}) =& a_2\mathbf{u} + b_2. \end{aligned}$$

We assume that the first line is to the left of the camera ($\mathbf{u} \leq 0$) and the second is to the right ($\mathbf{u} \geq 0$). Since the two lines intersect at infinity straight in front of the camera, we have $d_1(0) = d_2(0) = 0$, hence $b_1 = b_2 = 0$. For simplicity, we assume that the lines have both a "unit" distance to the camera, that is $a_1 = -1$ and $a_2 = 1$. Hence, we may

combine d_1 and d_2 to the following scene geometry description:

$$\mathbf{d}(\mathbf{u}) = |\mathbf{u}|$$

This concludes the set up of the scene in front of the camera.

Now, suppose that the camera moves infinitesimally with translation $\begin{pmatrix}\Delta x\\ \Delta z\end{pmatrix}$ and rotation angle $\Delta\alpha$. By the infinitesimal transformations given in Example 11, the points of the scene are reprojected to

$$\begin{pmatrix}\mathbf{u}\\ \mathbf{d}\end{pmatrix} + \Delta x \begin{pmatrix}\mathbf{d}\\ 0\end{pmatrix} + \Delta z \begin{pmatrix}-\mathbf{du}\\ -\mathbf{d}^2\end{pmatrix} + \Delta\alpha \begin{pmatrix}\mathbf{u}^2+1\\ \mathbf{du}\end{pmatrix}.$$

Since the first component determines the projection to the camera, the transformed points are visible on the image line at position

$$\mathbf{u} + \Delta x\mathbf{d} - \Delta z\mathbf{du} + \Delta\alpha(\mathbf{u}^2+1)$$

and the observed displacement from the original position $\mathbf{u}$ is

$$D(u) = \Delta x\mathbf{d} - \Delta z\mathbf{du} + \Delta\alpha(\mathbf{u}^2+1).$$

Now, suppose that the real motion is $\Delta x = 0$, $\Delta\alpha = 0$ and $\Delta z = 1$, i.e.

$$D_r(u) = -\mathbf{d}_r\mathbf{u}.$$

Performing triangulation from this displacement yields the following equation for the estimated inverse depth $\mathbf{d}$.

$$-\mathbf{d}_r\mathbf{u} = \Delta x\mathbf{d} - \Delta z\mathbf{du} + \Delta\alpha(\mathbf{u}^2+1)$$

Solving for $\mathbf{d}$ yields

$$\mathbf{d} = \frac{\mathbf{d}_r\mathbf{u} + \Delta\alpha(\mathbf{u}^2+1)}{\Delta z\mathbf{u} - \Delta x}.$$

I.e. the estimate $\mathbf{d}$ has a pole at $\mathbf{u} = -\frac{\Delta x}{\Delta z}$.

In fig. 11, we have simulated the resulting inverse depth estimate. It clearly shows that the continuous ground truth inverse disparity function is distorted into a discontinuous estimate. In particular, there is a section where the estimate diverges to negative infinity, meaning that the scene is estimated to be far behind the plane at infinity. For comparison, fig. 12 shows development of such a singularity in C-LSD-SLAM.

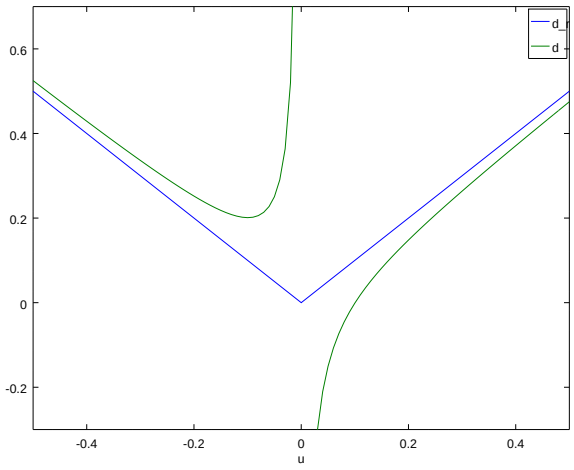


Figure 11: Plot of pole development for estimated camera movement $\alpha = -0.01$, $\Delta z = 1$, $\Delta x = 0$

Remark 28 (Mechanism for error retention)**.** Note that if the camera pose is misestimated, its effect on triangulation will tend to make future camera pose estimates consistent with the original error. Consider the case of forward motion, that is the epipolar direction is at the center of the image. Consider the case that between two frames, there is a misestimation of the camera yaw motion with the effect that the apparent object motion relative to a rotation-compensated pose is overestimated to the left of the epipolar direction and underestimated to the right of the epipolar direction. This will cause the inverse depths on the left image half to be overestimated and the inverse depth on the right image half to be underestimated. Note that the uncertainty of the pose estimate causing these errors of the inverse depth estimates is not regarded in future steps of the algorithm. In particular, the frame tracking for the succeeding frame will try to find a pose estimate which fits the incorrect inverse depth estimates as close as possible. Generally, this will replicate the error of the pose estimate between the original frames to some degree since this choice of pose estimate generally replicates the errors in the inverse depth estimates.

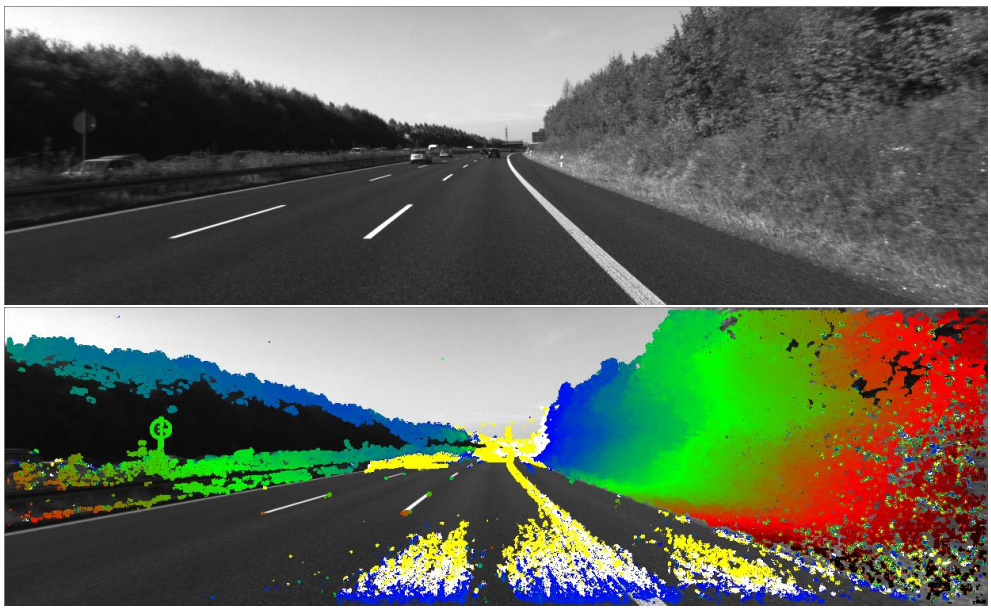


Figure 12: Pole development in the KITTI odometry benchmark challenge 21. color changes from near to far points from red over green to blue. Points at or behind infinity are white. Points far beyond infinity are yellow

The combination of the effects described in Remark 27 and Remark 28 yield divergent behaviour. While the effect described in Remark 27 means that even small errors of the pose estimate may cause singularities in the inverse depth estimates, that is potentially *unbounded* errors, the effect described in Remark 28 means that these effects can only slowly be eliminated. So a small error in the estimation of the epipolar direction may cause large errors near the epipolar direction and also (as described in Remark 28) a systematic error in the inverse depth estimates of the whole depth map. In combination, the result of these effects is often that the error of the estimation of the epipolar direction is even worse in the next frame, ultimately leading to the divergence of the method.

6.5.3 Mitigation strategies against divergent behaviour

The analysis above suggests two possible approaches for preventing divergent behaviour. The first is to mitigate the emergence of singularities near the epipolar direction. The second is to better remove the effects of incorrect pose estimates from the inverse distance estimates. In the sequel, we will pursue the former approach. The latter approach will be discussed in detail in section 7.

6.6 Reweighting scheme against singular behaviour near the epipolar line

The concept underlying our method for mitigating the behaviour described in Remark 27 is to prevent the noisy distance estimates near the epipolar direction from gaining influence in larger image parts by suitable weighting and depth estimation strategies.

Erroneous distance estimates near the epipolar direction can be separated into two classes, distinguished by whether they are further away or closer than the real distance. The estimates which are too far away tend to stay close to the epipolar direction, i.e. they stay within the region of high uncertainty surrounding the epipolar direction. Our only measure targeting these estimates is disabling the usage of negative inverse depth results in the depth estimation component, since negative inverse depth estimates can be caused by incorrect estimation of the epipolar direction.

On the other hand, distance estimates which are too close tend to move rapidly away from the epipolar direction, i.e. to a region where depth estimation uncertainties are much lower than near the epipolar direction. If it happens to be that the bogus estimate is confirmed in the depth estimation step (by a mismatch), we have an incorrect depth estimate on which the system places high confidence. Due to their closeness and high confidences, incorrect depth estimates of this type have the capability of strongly affecting the tracking component, in particular the estimation of the epipolar direction. Misestimation of the epipolar direction in turn causes systematic underestimation of pixel distances on one side of the epipolar direction (on the other side, distances are overestimated), which can cause error runaway.
To prevent this, we penalize all approaching pixels when reprojecting them from a keyframe to a new keyframe. In particular, we multiply the information estimate for their inverse depth by the factor

$$F(dist_{old}, dist_{new}) := \left(\frac{dist_{new}}{dist_{old}}\right)^c = \left(\frac{d}{d'}\right)^c, \tag{23}$$

where the exponent c is a positive number and $d = \frac{1}{dist_{old}}$, $d' = \frac{1}{dist_{new}}$ are the inverse depths of the old (original) and new (reprojected) depth values. For distance estimates which are too small, the fraction $\left(\frac{dist_{new}}{dist_{old}}\right)$ is smaller than the "true" value, so these estimates are penalized more heavily than correct estimates or depth estimates which are too large.

Experimentally, we have found values from $c = 8$ up to $c = 13$ to give good results. In this section, we use $c = 13$ for evaluation. Higher values for c tend to make the whole system more stable when faced with high movement speeds or scenes with moving objects near the epipolar direction. However, it should also be noted that decreasing depth estimate confidences means increasing the confidence region of the depth estimate. This causes the

depth estimation module to perform stereo matching for longer intervals of the epipolar line, thereby increasing the required computational effort.

Remark 29 (Effect of reweighting strategy on information efficiency)**.** Note that the reweighting strategy described above actively discards information. Here, we will give an approximation on how much information is actually discarded.

Note that if a inverse depth d is reprojected with longitudal pose difference Δz, the new inverse depth is

$$\begin{aligned} d' =& \frac{1}{z'} = \frac{1}{z - \Delta z} = \frac{1}{\frac{1}{d} - \Delta z} \\ =& \frac{d}{1 - d\Delta z}, \end{aligned}$$

so the reverse is

$$d = \frac{d'}{1 + d'\Delta z}$$

with the derivative

$$\begin{aligned} \frac{\mathrm{d}d}{\mathrm{d}d'} =& \frac{(1 + d'\Delta z) - d'\Delta z}{(1 + d'\Delta z)^2} = \frac{1}{(1 + d'\Delta z)^2} \\ =& \left(\frac{1}{1 + d'\Delta z}\right)^2 = \left(\frac{d}{d'}\right)^2. \end{aligned}$$

The reweighting factor in an orthodox extended information filter is $\left(\frac{\mathrm{d}d}{\mathrm{d}d'}\right)^2 = \left(\frac{d}{d'}\right)^4$, cf. Remark 26, which corresponds to a exponent of $c = 4$ in eq. (23). Reweighting approaching pixels with a larger exponent means placing lower weight on more distant estimates than the extended information filter suggests.

So let us approximate the total information. Suppose that $d = \frac{1}{z}$ is the inverse depth at the last observation. Suppose that the distances of all observations are $z_k = z + k\Delta z$, i.e. the vehicle is moving with constant speed toward the object. This yields the inverse depths

$$d_k := \frac{1}{z_k} = \frac{1}{z + k\Delta z} = \frac{1}{\frac{1}{d} + k\Delta z} = \frac{d}{1 + kd\Delta z}. \tag{24}$$

Suppose that the estimation of the object position starts with the estimation of d_n and an information of i. Suppose that in each succeeding estimation step, the information $i_n := i$ is gained from observation and fused with (i.e. added to) the reweighted information from the previous step. I.e.

$$i_k = i + \left(\frac{d_{k+1}}{d_k}\right)^c i_{k+1}$$

for $k = n-1, \dots, 0$. Unrolling this recursion, we obtain

$$i_k = \sum_{j=k}^{n} \left(\frac{d_j}{d_k}\right)^c i$$

for $k = 0, \dots, n$. Using approximation of sums by integrals, this yields the following:

$$\begin{aligned}
i_0 =& \sum_{j=0}^{n} \left(\frac{d_j}{d_0}\right)^c i \\
\overset{(24)}{=}& i \sum_{j=0}^{n} \left(\frac{1}{d}\frac{d}{1+jd\Delta z}\right)^c = i \sum_{j=0}^{n} \left(\frac{1}{1+jd\Delta z}\right)^c \\
\approx& i \int_{j=0}^{n} \left(\frac{1}{1+jd\Delta z}\right)^c \mathrm{d}j = \frac{i}{1-c}\left[\frac{1}{(1+jd\Delta z)^{c-1}}\right]_{j=0}^{n} \\
=& \frac{i}{1-c}\left[1 - \frac{1}{(1+nd\Delta z)^{c-1}}\right]
\end{aligned}$$

Note that the term $\frac{1}{1-c}$ is monotonously decreasing in c and that for positive Δz (which is always the case for approaching pixels), the term $\left[1 - \frac{1}{(1+nd\Delta z)^{c-1}}\right]$ is monotonously increasing in c. I.e. the decrease of $\frac{1}{1-c}$ is greater than the total information decrease. Hence, we can obtain a rough upper bound on the information loss caused by increasing c by examining the behaviour of the term $\frac{1}{1-c}$. Recall that the extended information filter suggests $c = 4$. Hence, the factor of information decrease caused by setting $c > 4$ can be roughly bounded by

$$q \approx \frac{\frac{1}{1-c}}{\frac{1}{1-4}} = \frac{3}{c-1}$$

Hence, the (rather large) choice $c := 10$ roughly decreases the information by a factor of $q \approx \frac{1}{3}$, which is quite significant, but much less dramatic as the appearance of c in the exponent of (23) might suggest at first glance.

6.7 Evaluation

We use the sequences of the KITTI odometry benchmark[Geiger et al., 2012] for evaluation. The main reason for this choice is that these datasets feature very long sequences, hence the stability of C-LSD-SLAM can be investigated which was a key objective of the development of C-LSD-SLAM.

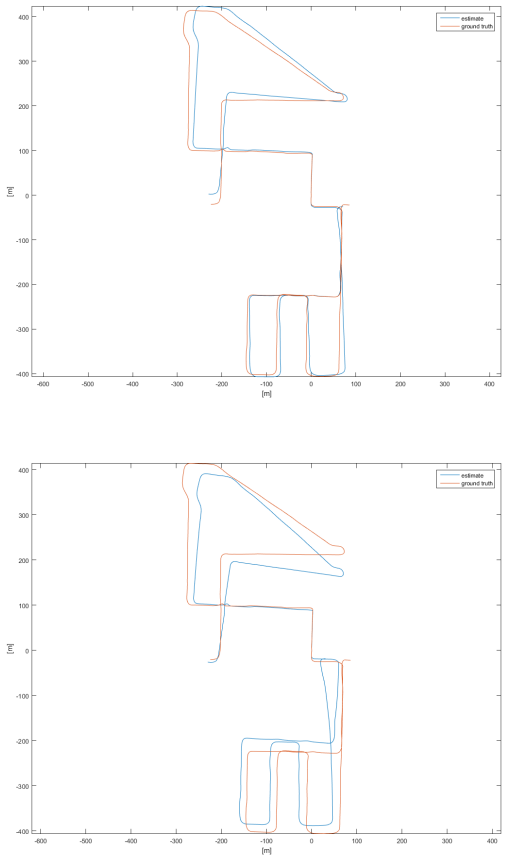


Figure 13: Tracking result for sequence 08 of the KITTI odometry challenge [Geiger et al., 2012]. Upper plot: full resolution input (1226×370). Lower plot: half resolution input (584×184). The trajectories are aligned at the middle of the sequence. We use a simple variant of ground plane estimation (cf. e.g. [Song and Chandraker, 2014]) to determine the absolute scale.

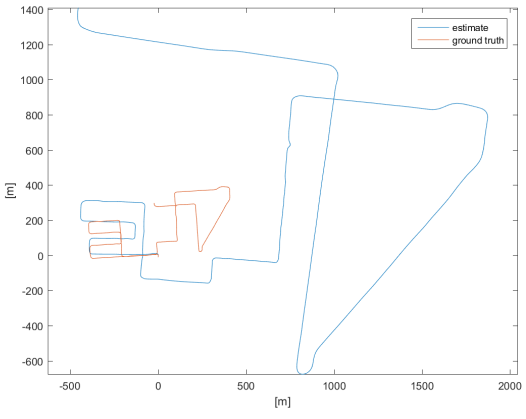
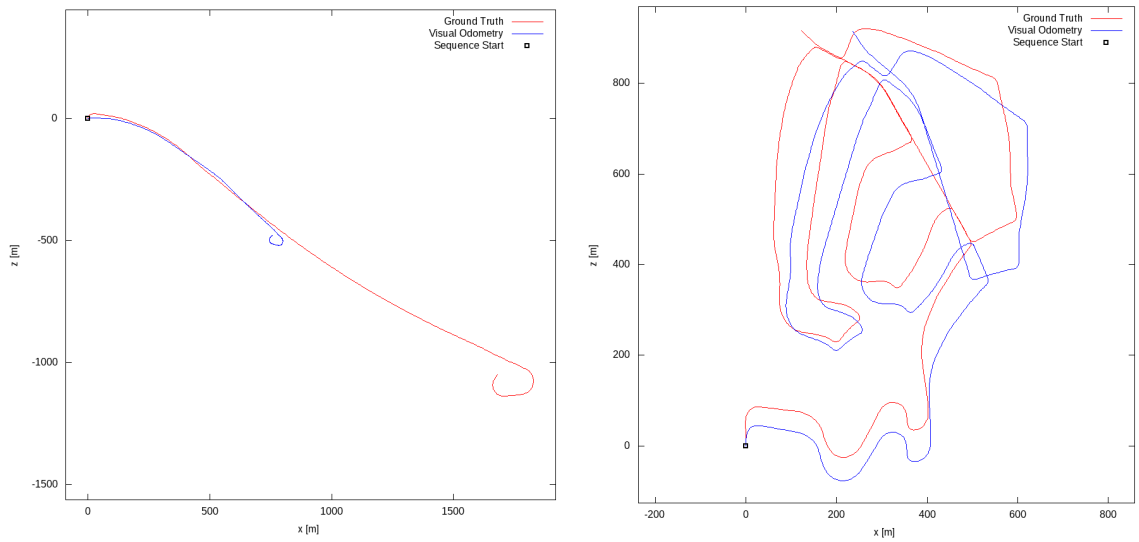
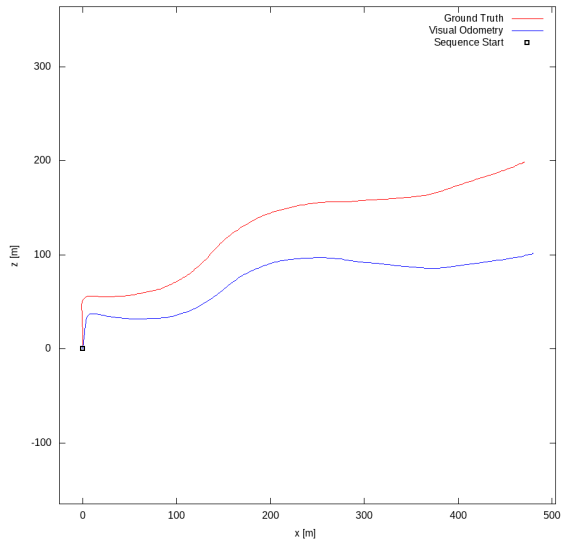


Figure 14: Tracking result for the sequence 08 of the KITTI odometry challenge, with absolute scale estimation enabled only at the beginning of the sequence. The scale drifts towards larger scales.

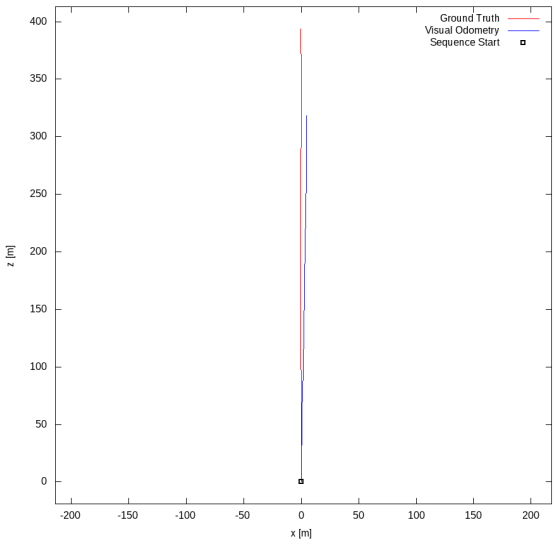


(a) dataset 01

(b) dataset 02

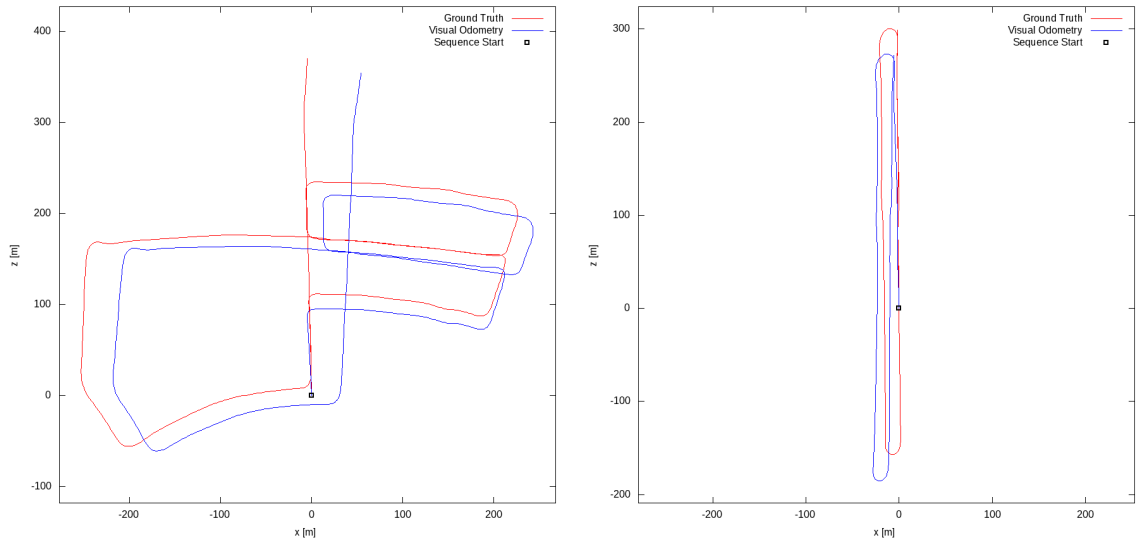


(c) dataset 03



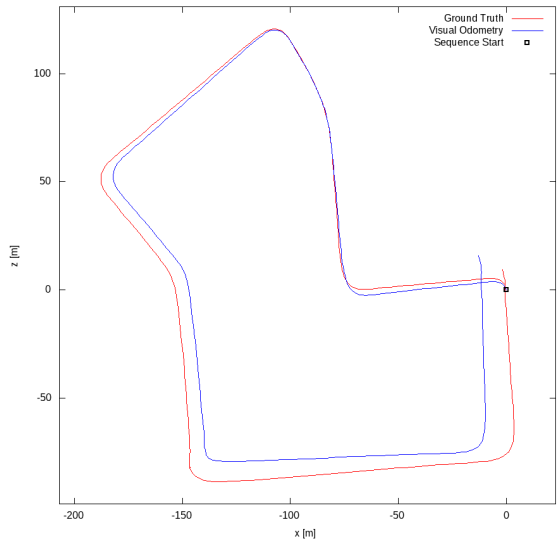
(d) dataset 04

Figure 15: Trajectories generated by C-LSD-SLAM at full resolution on the training datasets 01 to 04 of the KITTI odometry challenge

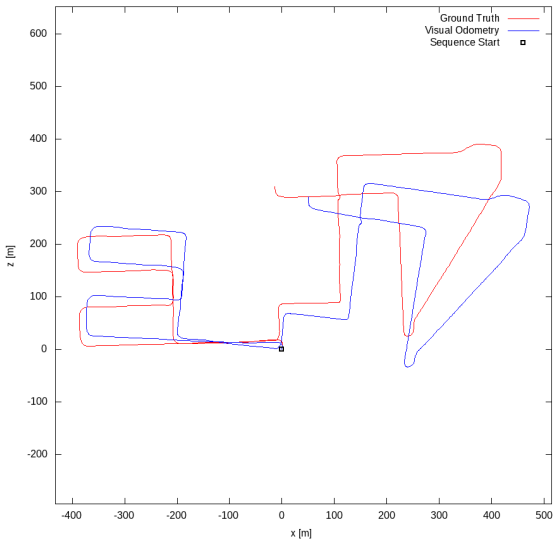


(a) dataset 05

(b) dataset 06

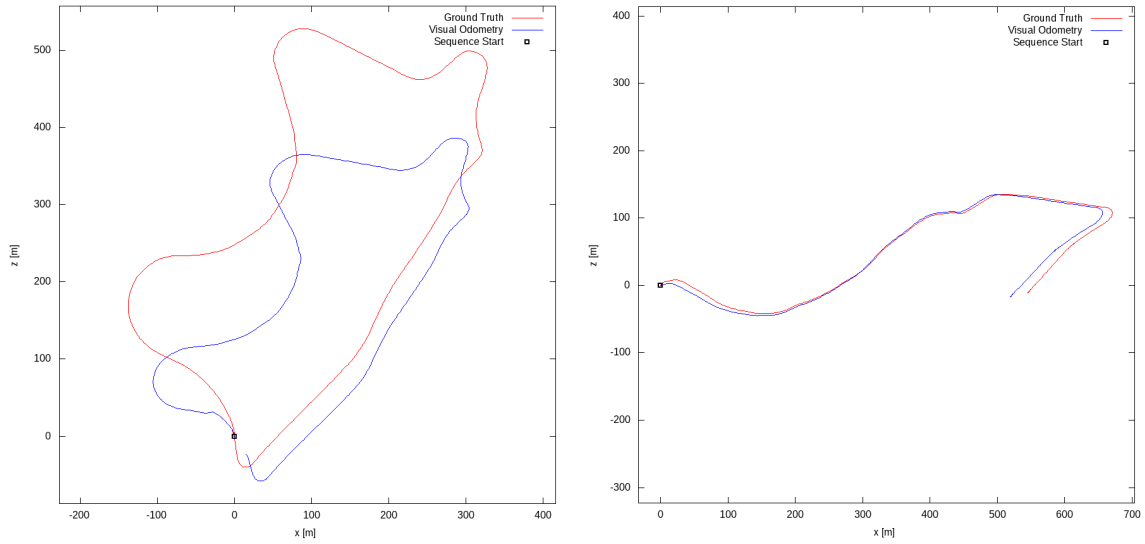


(c) dataset 07



(d) dataset 08

Figure 16: Trajectories generated by C-LSD-SLAM at full resolution on the training datasets 05 to 08 of the KITTI odometry challenge



(a) dataset 09 (b) dataset 10

Figure 17: Trajectories generated by C-LSD-SLAM at full resolution on the training datasets 09 and 10 of the KITTI odometry challenge

6.7.1 Tracking stability

Generally, C-LSD-SLAM is highly stable. We tested C-LSD-SLAM on all 21 sequences of the KITTI odometry challenge. There are three sequences (01, 12 and 21) on which C-LSD-SLAM shows signs of instability, that is significant areas near the epipolar direction with incorrect depth estimates. All of these are on highway scenes, with regions of instability typically initiating near moving vehicles near the epipolar direction. Only in sequence 21, the system becomes actually unstable. In sequence 21, the system switches several times between periods of correct operation and divergent behaviour. This behaviour may be attributed to the fact that highway scenes have very low scene structure, hence it is difficult for the system to eliminate the influence of moving objects. C-LSD-SLAM performs very well in urban scenes. In particular situations where the vehicle turns around corners, which are difficult for the original LSD-SLAM due to the fast change of perspective, are handled easily by C-LSD-SLAM. In fact, during turns C-LSD-SLAM reliably recovers even from a completely diverged tracking state (e.g. due to initialization failure).

6.7.2 Scale drift and absolute scale determination

Without a method to determine the absolute scale, C-LSD-SLAM shows a consistent scale drift towards increasing scales (cf. fig. 14). In order to make reasonable comparisons with ground truth trajectories, we use a simple variant of ground plane estimation (cf. e.g. [Song and Chandraker, 2014]) to determine the absolute scale. It is performed by fitting a plane to the depth estimates in a small image area corresponding to the road patch in

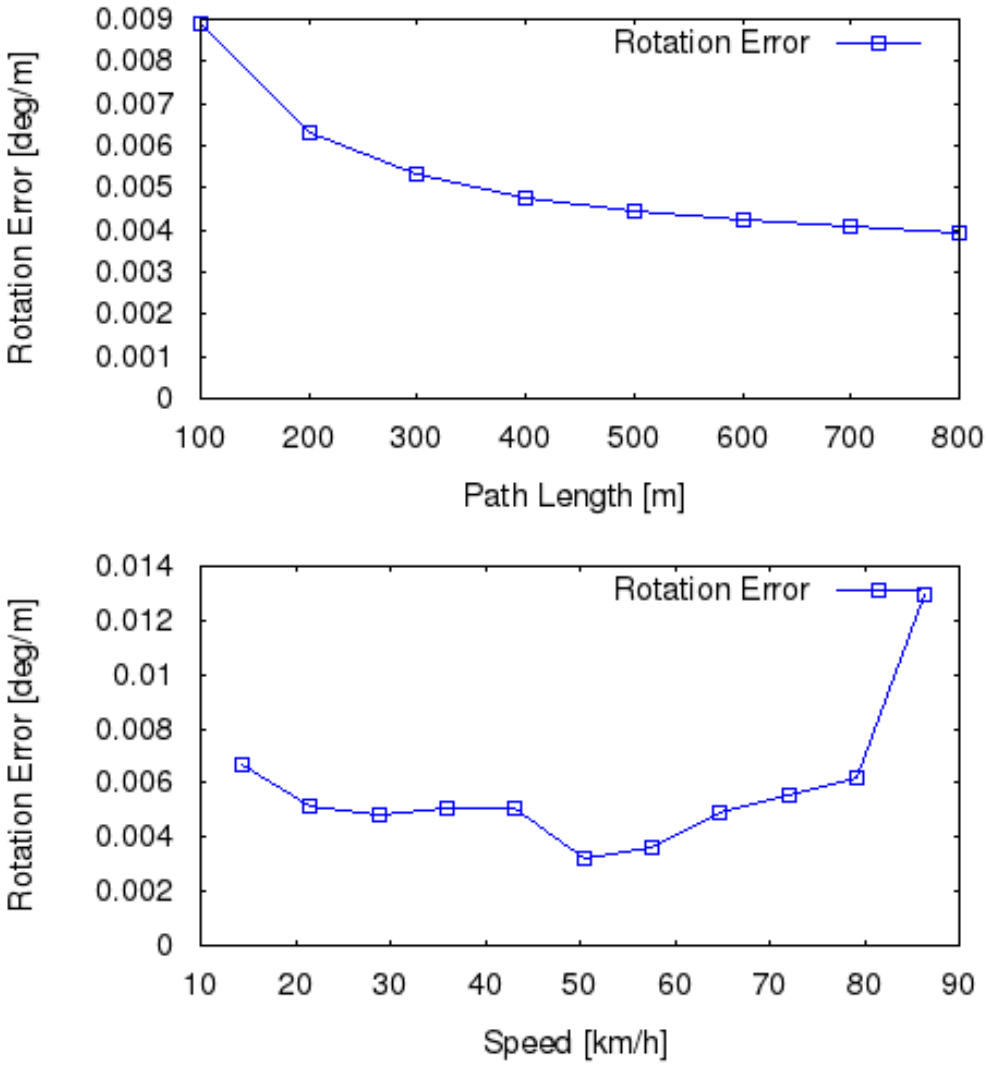


Figure 18: Rotational accuracy of C-LSD-SLAM on the datasets 01 to 10 of the KITTI odometry challenge as computed by the KITTI development kit. Top: rotational accuracy relative to travelled distance. Bottom: rotational accuracy relative to speed

front of the vehicle and then using this plane to estimate the height above the ground. This method works only if the road in front of the vehicle has sufficient texture to allow depth estimates, which fails e.g. for very smooth road surfaces or high speeds (motion blur).

6.7.3 Accuracy

In figs. 15 to 17, we have plotted the trajectories estimated by C-LSD-SLAM for the training datasets of the KITTI odometry challenge over the ground truth trajectories. Note that for dataset 01, the absolute scale estimation fails, which is an effect of the road surface being too smooth for depth estimation by C-LSD-SLAM. The absolute scale

estimation also gives incorrect results in dataset 09.

As plotted in fig. 18, the rotational error for longer sub segments of the training datasets of the KITTI datasets at full resolution is around 0.005deg/m. This is neither particularly good nor particularly bad for a monocular slam method. Translational errors can be quite high (e.g. ca. 50% in dataset 01), which is caused by failures of the absolute scale estimation as described above.

6.7.4 Performance

At full resolution (ca. 1200×370px) of the KITTI datasets, C-LSD-SLAM runs at about 7fps on a (rather old) Intel Core i5 M540 (dual core, 2.53GHz) and at about 30 fps at half resolution (ca. 600×180px). On a more recent Intel Core i7-6820HQ (quad core, 2.7GHz), we obtain about 20fps at full resolution.

6.8 Conclusion

C-LSD-SLAM is a variant of LSD-SLAM following the filter-based visual odometry paradigm. By employing a reweighting scheme for approaching pixels, C-LSD-SLAM has been made robust for situations with fast forward motion. In particular, C-LSD-SLAM features stable tracking on the sequences of the KITTI odometry challenge, which is in the original variants of LSD-SLAM a feature exclusive to the stereo variant of LSD-SLAM[Engel et al., 2015].

C-LSD-SLAM performs very well in situation with fast changes of perspective as often encountered in urban scenes. However, it struggles with scenes with little structure, in particular if there are moving objects present.

7 Efficient approximation of the information matrix

In section 6, we have obtained an computationally efficient filter-based direct visual odometry method by approximating the information matrix of the scene point estimates with a diagonal matrix. As this is a somewhat crude way to handle the information matrix, we are interested in more exact techniques for approximating the information matrix, in particular with regard to the correlation of the scene point estimates introduced by uncertainty of the camera pose estimates. In this section, we introduce a solution to this which retains the linear complexity featured by the approximation with diagonal matrices.

Given the vector space H of inverse depth estimates of the scene points, the core idea underlying this approach is to "compress" the correlations caused by the pose uncertainty into a subspace L of H of fixed dimension (in our case $\dim L = 15$). A technical challenge with this approach is the fact that as L needs to be large enough to accommodate all kinds of pose estimation uncertainties in a unified fashion, it is usually "too large" for the pose estimation uncertainty encountered in a concrete situation. More precisely, this means that the important operations typically happen in a strict subspace of L. Preventing degenerate behaviour in the remaining, unattended components of L is solved in our approach by some careful manipulation of explicitly rank-deficient matrices.

7.1 Introduction

In section 6.2, we have motivated approximating the information matrix. More specifically, note that mitigation of the effect described in Remark 28 means being able to efficiently correct errors of the pose estimation a posteriori. A crucial part for this is being able to model the effects of these errors of the pose estimates in the approximation of the information matrix used. In particular this means we need to obtain a more sophisticated approximation of the information than the approximation by a diagonal matrix used in section 6:

In eq. (22), the function K models the observation process in the VO problem. Its total derivative DK has the structure

$$\frac{\mathrm{d}K}{\mathrm{d}(p,k)} = \begin{pmatrix} \frac{\mathrm{d}K}{\mathrm{d}p} & \frac{\mathrm{d}K}{\mathrm{d}k} \end{pmatrix}$$

obtained by separating the derivatives with respect to the scene point estimates from the derivatives with respect to the camera pose estimates.

As the extended information matrix is $f^2 \left(\frac{\mathrm{d}K}{\mathrm{d}(p,k)}\right)^T \left(\frac{\mathrm{d}K}{\mathrm{d}(p,k)}\right)$, cf. Remark 25, the matrix $f(\frac{\mathrm{d}K}{\mathrm{d}(p,k)})$ is a square root of the information matrix, cf. Remark 18, thus we can use the simplification and reduction techniques of the square root form, cf. Remarks 19 and 23.

Note that for fixed camera pose estimates, the observations of the scene points are independent. In practice, this means that we have $\left\langle \frac{\partial K}{\partial p_i}, \frac{\partial K}{\partial p_{i'}} \right\rangle = 0$ for $i \neq i'$, i.e. the columns of $\frac{\mathrm{d}K}{\mathrm{d}p}$ are orthogonal, which can be verified by eq. (21).

Thus we may find an orthogonal transformation Q of the columns of $f\frac{\mathrm{d}K}{\mathrm{d}p}$ such that $Qf\frac{\mathrm{d}K}{\mathrm{d}p}$ has the form

$$Qf\frac{\mathrm{d}K}{\mathrm{d}p} = \begin{pmatrix} E \\ 0 \end{pmatrix}$$

for a diagonal matrix E (note that notion of E here differs from the one in section 6.2). Applying this to the total derivative of K yields the block structure

$$Qf\frac{\mathrm{d}K}{\mathrm{d}(p,k)} = \begin{pmatrix} E & F_0 \\ 0 & A_0 \end{pmatrix}$$

for some matrices F_0, A_0.

Note that F_0 has as many columns as there are degrees of freedom in all pose estimates combined. Now, if we can find a subspace L such that all columns of F_0 (approximatively) lie in L, we can choose a basis B_L of L and represent the columns of F_0 approximately as linear combinations of the elements of B_L. Interpreting B_L as a matrix and collecting the coefficients of the linear combinations in a matrix C_0, we obtain

$$F_0 \approx B_L C_0.$$

Hence, we have

$$Qf\frac{\mathrm{d}K}{\mathrm{d}(p,k)} \approx \begin{pmatrix} E & B_L C_0 \\ 0 & A_0 \end{pmatrix} =: S_0$$

By Remark 19, we may replace the square root of the information matix $f\frac{\mathrm{d}K}{\mathrm{d}(p,k)}$ with $Qf\frac{\mathrm{d}K}{\mathrm{d}(p,k)}$, so we may replace it approximately with S_0. Consider the case that there are many camera poses, i.e. the number of degrees of freedom in the camera poses (i.e. the number of columns in C_0) exceeds the fixed dimension $\dim L$ (which is the number of rows of C_0). Hence C_0 has more columns than rows. By using the LQ decomposition (dual of QR decomposition), we may find an orthogonal transformation $\tilde{Q}$ such that

$$C_0 = \begin{pmatrix} C & 0 \end{pmatrix} \tilde{Q}$$

for an $(\dim L) \times (\dim L)$-matrix C. For the square root S_0, we obtain

$$S_0 = \begin{pmatrix} E & B_L \begin{pmatrix} C & 0 \end{pmatrix} \tilde{Q} \\ 0 & A_0 \end{pmatrix} = \begin{pmatrix} E & B_L \begin{pmatrix} C & 0 \end{pmatrix} \\ 0 & A_0 \tilde{Q}^T \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q} \end{pmatrix}.$$

We are now ready to replace the uncertainty of the camera pose estimates in S_0 with an abstract structure in L. Transforming S_0 on the right hand side with the matrix $\begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q}^T \end{pmatrix}$ (thus reparametrizing the camera poses) and splitting this parametrization of the camera poses along the decomposition of the matrix $\begin{pmatrix} C & 0 \end{pmatrix}$, we obtain

$$\begin{aligned} \tilde{S}_0 :=& S_0 \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q}^T \end{pmatrix} = \begin{pmatrix} E & B_L C & B_L 0 \\ 0 & A'_0 & A''_0 \end{pmatrix} \\ =& \begin{pmatrix} E & B_L C & 0 \\ 0 & A'_0 & A''_0 \end{pmatrix} \end{aligned} \tag{25}$$

for some matrices A'_0, A''_0. The columns of $\tilde{S}_0$ are now decomposed into three blocks. The first block consists of the parameters of the scene point estimates. The middle block has $\dim L$ components, and represents the "relevant" components of the pose parameters, whereas the last block represents "excess" components of the pose parameters.

The last form given in (25) shows that reducing the "excess" components (cf. Remark 22) from the state vector only affects A''_0 (which is eliminated by the reduction) and A'_0 (giving a reduced matrix A'), yielding the reduced form

$$\tilde{S} = \begin{pmatrix} E & B_L C \\ 0 & A' \end{pmatrix}$$

Recall that C is a $(\dim L) \times (\dim L)$-matrix, hence A'_0 has $\dim L$ columns. By simplifying A' via Remark 19 until it has only $\dim L$ rows or by adding zero-rows to A' until it has $\dim L$-rows, we may modify $\tilde{S}$ into the form

$$S = \begin{pmatrix} E & B_L C \\ 0 & A \end{pmatrix} \tag{26}$$

for a $(\dim L) \times (\dim L)$-matrix A.

Let n be the number of scene point estimates. Note that for the square root of the (reduced) information matrix given in eq. (26), the submatrix E is an $n \times n$ diagonal matrix, the matrices A and C are $(\dim L) \times (\dim L)$-matrices and the matrix B_L describing a basis of L is an $n \times (\dim L)$-matrix, giving overall linear complexity in n. As we will see below, our variant actually does need not store B_L explicitly.

Note that from a high-level perspective, the essential steps in obtaining this form were:

(1) Approximating the columns of the matrix F_0 (which describes how variations of the pose estimates affect the scene point estimates) with elements of the linear space L, with L being a subspace of all scene point estimates of fixed dimension.

(2) Reducing the space of camera pose estimates in such a way that the remaining degrees of freedom have the same dimension as L and can be described using L. These remaining degrees of freedom describe the uncertainty of the scene pose estimates incurred by uncertainty of the camera poses in a unified, abstract fashion.

7.2 Implementation in pinhole coordinates for direct methods

In the following we describe how to apply the concept introduced above to the VO parametrization in pinhole coordinates. In particular:

- The subspace L is defined.
- A method for updating a square root form eq. (26) with the observations of a new image is given.
- A method for reprojecting a square root form eq. (26) to a new keyframe is given (square root forms are given relative to a camera pose, cf. below).

7.2.1 Perspective parametrization

In the following, we parametrize all scene point estimates via their 2D-coordinate on the current camera image $u := \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$, with $\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix}$ being the pinhole coordinates of a scene point in pinhole coordinates. While the method does not prevent the usage of more generally suited parametrizations of the scene points, we use this variant since it simplifies notation greatly.

For a scene point p with pinhole coordinates $\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix}$ relative to a certain camera pose, let

$$P_u := \begin{pmatrix} -\mathbf{uv} & 1+\mathbf{u}^2 & -\mathbf{v} \\ -(1+\mathbf{v}^2) & \mathbf{uv} & \mathbf{u} \end{pmatrix}$$
$$S_u := \begin{pmatrix} 1 & 0 & -\mathbf{u} \\ 0 & 1 & -\mathbf{v} \end{pmatrix}$$

$$Q_u := \begin{pmatrix} \mathbf{d} & 0 & -\mathbf{ud} \\ 0 & \mathbf{d} & -\mathbf{vd} \end{pmatrix}$$
$$= \mathbf{d} S_u.$$

P_u is used for describing uncertainty in the rotation part of camera poses, Q_u is used for the translation part. Note that these formulas can be motivated from the derivatives given in eq. (1) and we will use eq. (1) to that end later.

Let $I(u)$ be the image intensity at the image coordinate u. Let $g(u)$ be the image gradient at the image coordinate u.

For $\delta C_T, \delta C_R \in \mathbb{R}^3 \otimes \mathbb{R}^3$ and $d = d(u)$ being the inverse depth estimate of the point denoted by u, let

$$F(u,d) \begin{pmatrix} \delta C_T \\ \delta C_R \end{pmatrix} := d((g(u) f S_u) \otimes (g(u) f S_u))(\delta C_T) + ((g(u) f S_u) \otimes (g(u) f P_u))(\delta C_R). \tag{27}$$

Here, the row vectors $(g(u) f S_u)$ and $(g(u) f P_u))$ are regarded as linear maps $\mathbb{R}^3 \to \mathbb{R}$, with their tensor products being linear maps $\mathbb{R}^3 \otimes \mathbb{R}^3 \to \mathbb{R}$. The usage of the tensor product here is a crucial technical step which allows us to reformulate a bilinear map as a linear map. For example, the bilinear map $(c, c') \mapsto (g(u) f S_u) c \cdot (g(u) f S_u) c'$ can be reformulated as the linear map $(c \otimes c') \mapsto ((g(u) f S_u) c \otimes (g(u) f S_u) c')(c \otimes c') = (g(u) f S_u) c \cdot (g(u) f S_u) c'$.

Recall that the diagonal matrix E in eq. (26) has one entry diagonal per estimated scene point. We denote these entries by $i(u)$ and assume by convention that $i(u) > 0$ for all u (otherwise, we may change the signs of a row in eq. (26)).

We define the vectors

$$\left(\frac{1}{i(u)} (F(u,d) \begin{pmatrix} \delta C_T \\ \delta C_R \end{pmatrix} \right)_u \qquad \text{for} \quad \delta C_T, \delta C_R \in \mathbb{R}^3 \otimes \mathbb{R}^3$$

to be the elements of the subspace L. As the tensor product $\mathbb{R}^3 \otimes \mathbb{R}^3$ has dimension 9, L has dimension at most 9+9 = 18. Note however that the tensor product $((g(u) f S_u) \otimes (g(u) f S_u))$ in eq. (27) is symmetric, so we may restrict δC_T to symmetric tensors of $\mathbb{R}^3 \otimes \mathbb{R}^3$ without changing L. As the space of symmetric tensors in $\mathbb{R}^3 \otimes \mathbb{R}^3$ has dimension 6, the linear space L has dimension $6 + 9 = 15$.

In this venue, we suppose the columns of V in eq. (26) to be vectors $\begin{pmatrix} \delta C_T \\ \delta C_R \end{pmatrix}$, with δC_T being a symmetric tensor in $\mathbb{R}^3 \otimes \mathbb{R}^3$ (and using a parametrization with 6 parameters) and with δC_R being an element of $\mathbb{R}^3 \otimes \mathbb{R}^3$.

7.2.2 Update in perspective parametrization

Suppose we have a square root form (26). By the above, its precise form is

$$S = \begin{pmatrix} \operatorname{diag}(i(u))_u & \left(\frac{1}{i(u)} F(u,d)V\right)_u \\ & A \end{pmatrix}$$

We call such a square root form a *standard form*. Note that we will sometimes regularize the denominator of $\frac{1}{i(u)}$ with a small regularization term $\varepsilon > 0$, yielding the fraction $\frac{1}{i(u)+\varepsilon}$.

For convenience, we introduce the shorthand

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)} F(u,d)Vc \\ & Ac \end{pmatrix} \tag{28}$$

for S.

This shorthand uses the convention that if u appears in a block in the first row, the block is a diagonal matrix, with the diagonal entries given by the formula in the block and letting u run over all estimated scene points. If u appears in a block in any other row, the rows of the block are indexed by letting u run over all scene points; a row is then given by evaluating the formula of the block at its index u. Blocks without u are written literally. Zero blocks are left empty. Furthermore, instead of writing the degrees of freedom as a vector multiplied at the right of S, the degrees of freedom $\delta d(u)$ and c (and later c' and d') are written directly into the blocks for better legibility.

We want to update the standard form S with the observations obtained from a new frame.

To add the new information, the basic concept is to compute the total derivative (Jacobian) of the function modelling the noise-free observation process and to concatenate the Jacobian at the end of S, cf. Remark 20. To obtain the update of S in standard form, we cannot use the Jacobian directly but need to use an approximation.

Suppose the transformation of Cartesian coordinates relative to the current camera pose to Cartesian coordinates relative to the new camera pose is $(x \mapsto Rx + t)$.

Let $\pi = \pi(u) := fu + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$ be the projection of a scene point to its image coordinate in the current frame. Similarly, $\pi(u') = fu' + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$ is the projection to its image coordinates in the new frame. Let I, I' be the intensity functions of the current and new frame and let g, g' be their gradient.

Suppose given a point p with image coordinate $u = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$ and inverse depth $\mathbf{d}$ relative

to the current camera pose. Setting $U := \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ 1 \end{pmatrix}$, Example 9 shows that the pinhole coordinates reprojected to the new camera pose are

$$\begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{d}' \end{pmatrix} = \frac{1}{R_{3,-}U + t_3\mathbf{d}} \begin{pmatrix} R_{1,-}U + t_1\mathbf{d} \\ R_{2,-}U + t_2\mathbf{d} \\ \mathbf{d} \end{pmatrix}.$$

The derivative of the image coordinates with respect to $\mathbf{d}$ is

$$\begin{aligned} \frac{\mathrm{d}\pi'(u')}{\mathrm{d}\mathbf{d}} =& \frac{\mathrm{d}}{\mathrm{d}\mathbf{d}} f \begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \end{pmatrix} \\ \overset{\text{R.10}}{\approx}& f \left(\frac{\mathbf{d}'}{\mathbf{d}}\right)^2 \begin{pmatrix} t_1 - t_3\mathbf{u} \\ t_2 - t_3\mathbf{v} \end{pmatrix} \\ \approx& f \left(\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} - \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} t_3 \right) \\ =& f S_u t. \end{aligned}$$

For the approximations, we use the assumption of Remark 10 that the rotation between the current and the new camera pose is small and also that the translation between the poses is small, yielding $\mathbf{d}' \approx \mathbf{d}$.

For the derivatives of $\pi'(u')$ with respect to the pose difference between new and current camera pose, we use similarly the assumption of small camera movement to obtain $\begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{d}' \end{pmatrix} \approx \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix}$. Inserting this into (1) yields the approximate derivatives

$$\begin{aligned} \frac{\mathrm{d}\pi'(u')}{\mathrm{d}t} \approx& f\mathbf{d} S_u \\ \frac{\mathrm{d}\pi'(u')}{\mathrm{d}R} \approx& f P_u. \end{aligned}$$

Now, noting that in direct VO methods, we measure the image intensity I', which has the gradient $g'(\pi'(u')) \approx g(\pi(u))$, we obtain the approximation of the Jacobian

$$\begin{pmatrix} (gfS_u t)\delta d(u) & \mathbf{d}(u)(gfS_u)(\delta t) + gfP_u(\delta R) \end{pmatrix}$$

for variation in d, R and t.

Adding this to S yields

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ & Ac \\ (gfS_ut)\delta d(u) & \mathbf{d}(u)(gfS_u)(\delta t)+gfP_u(\delta R) \end{pmatrix}$$

Setting $j(u) := |gfS_ut|$ yields

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ & Ac \\ j(u)\delta d(u) & \frac{1}{j(u)}\left(\mathbf{d}(u)(gfS_ut)(gfS_u(\delta t))+(gfS_ut)(gfP_u(\delta R))\right) \end{pmatrix}$$

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ & Ac \\ j(u)\delta d(u) & \frac{1}{j(u)}(F(u,d)(t\otimes(\delta t),t\otimes(\delta R)) \end{pmatrix}$$

Choose $W := ((\delta t,\delta R)\mapsto(t\otimes\delta t,t\otimes\delta R))$, which is a rank deficient operator, substitute $(\delta t,\delta R)$ with $d\in\mathbb{R}^6$:

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ & Ac \\ j(u)\delta d(u) & \frac{1}{j(u)}F(u,d)Wd \end{pmatrix}$$

Apply rotation matrices $\frac{1}{\sqrt{i(u)^2+j(u)^2}}\begin{pmatrix} i(u) & j(u) \\ j(u) & -i(u) \end{pmatrix}$ on the left hand side to rotate the blocks of the left column onto a single block:

$$\begin{pmatrix} \sqrt{i(u)^2+j(u)^2}(\delta d(u)) & \frac{1}{\sqrt{i(u)^2+j(u)^2}}F(u,d)Vc & \frac{1}{\sqrt{i(u)^2+j(u)^2}}F(u,d)Wd \\ & Ac & \\ & \frac{j(u)}{i(u)\sqrt{i(u)^2+j(u)^2}}F(u,d)Vc & -\frac{i(u)}{j(u)\sqrt{i(u)^2+j(u)^2}}F(u,d)Wd \end{pmatrix}$$

Regularize denominators:

$$\begin{pmatrix} \sqrt{i(u)^2+j(u)^2}(\delta d(u)) & \frac{1}{\sqrt{i(u)^2+j(u)^2}}F(u,d)Vc & \frac{1}{\sqrt{i(u)^2+j(u)^2}}F(u,d)Wd \\ & Ac & \\ & \frac{j(u)}{(i(u)+\varepsilon)\sqrt{i(u)^2+j(u)^2}}F(u,d)Vc & -\frac{i(u)}{(j(u)+\varepsilon)\sqrt{i(u)^2+j(u)^2}}F(u,d)Wd \end{pmatrix}$$

Simplify lower row (cf. Remark 19), with the matrices $\hat{G}$, $\hat{G}'$ each having $15+6=21$ rows:

$$\begin{pmatrix} \sqrt{i(u)^2+j(u)^2}(\delta d(u)) & \frac{1}{\sqrt{i(u)^2+j(u)^2}}F(u,d)Vc & \frac{1}{\sqrt{i(u)^2+j(u)^2}}F(u,d)Wd \\ & Ac & \\ & Gc & G'd \end{pmatrix}$$

Find orthogonal transformation $\begin{pmatrix} c \\ d \end{pmatrix} = Q\begin{pmatrix} c' \\ d' \end{pmatrix}$, $c'\in\mathbb{R}^{15}, d'\in\mathbb{R}^6$

such that $\begin{pmatrix} V & W \end{pmatrix} Q = \begin{pmatrix} V' & 0 \end{pmatrix}$ via an LQ decomposition of $\begin{pmatrix} V & W \end{pmatrix}$:

$$\begin{pmatrix} \sqrt{i(u)^2 + j(u)^2}(\delta d(u)) & \frac{1}{\sqrt{i(u)^2+j(u)^2}} F(u,d)V'c' \\ & \hat{G}c' \qquad\qquad \hat{G}'d' \end{pmatrix}$$

Eliminate d' by reduction (cf. Remark 23). As d' is already isolated, only $\hat{G}$ and $\hat{G}'$ are affected:

$$\begin{pmatrix} \sqrt{i(u)^2 + j(u)^2}(\delta d(u)) & \frac{1}{\sqrt{i(u)^2+j(u)^2}} F(u,d)V'c' \\ & A'c' \end{pmatrix}$$

Note that this is in standard form, hence it gives the update of S.

7.2.3 Rescaling in perspective parametrization

Note that C-LSD-SLAM (as does LSD-SLAM) tracks the scene scale explicitly by periodically rescaling the scene point estimates in such a way that their average inverse depth is 1 and by tracking these scale modifications in the pose graph. In the following, we show how to perform such rescaling for a standard form S.

Rescaling proceeds by rescaling the inverse distance estimates via a substitution

$$d'(u) := r \cdot d(u)$$

for some scale factor $r \in \mathbb{R}^*$. We have $\delta d' = r\delta d$, hence

$$d = \frac{1}{r} d, \qquad\qquad \delta d = \frac{1}{r} \delta d'.$$

For convenience, we split the matrix V of the standard form (cf. (28)) along the decomposition $\begin{pmatrix} \delta C_T \\ \delta C_R \end{pmatrix}$ of $F(u, \mathbf{d})$ in (27) into $V = \begin{pmatrix} V_t \\ V_R \end{pmatrix}$.

This yields

$$\begin{aligned} & \begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)} F(u,d)Vc \\ & Ac \end{pmatrix} \\ = & \begin{pmatrix} \frac{i(u)}{r}(\delta d'(u)) & \frac{1}{i(u)} F(u,d)Vc \\ & Ac \end{pmatrix} \\ \overset{i'(u):=\frac{i(u)}{r}}{=} & \begin{pmatrix} i'(u)(\delta d'(u)) & \frac{1}{ri'(u)} F(u,d)Vc \\ & Ac \end{pmatrix} \end{aligned}$$

$$= \begin{pmatrix} i'(u)(\delta d'(u)) & \frac{1}{ri'(u)} d(u)((gfS_u) \otimes (gfS_u))(V_t c) + ((gfS_u) \otimes (gfP_u))(V_R c) \\ & Ac \end{pmatrix}$$
$$= \begin{pmatrix} i'(u)(\delta d'(u)) & \frac{1}{i'(u)} d'(u)((gfS_u) \otimes (gfS_u))(\frac{1}{r^2} V_t c) + ((gfS_u) \otimes (gfP_u))(\frac{1}{r} V_R c) \\ & Ac \end{pmatrix}$$
$$= \begin{pmatrix} i'(u)(\delta d'(u)) & \frac{1}{i'(u)} F(u, d') V' c \\ & Ac \end{pmatrix}$$

for $V' := \begin{pmatrix} \frac{1}{r^2} V_t \\ \frac{1}{r} V_R \end{pmatrix}$.

7.2.4 Reprojection

Suppose that we want to transform points as well as a standard form from an (old) camera pose to a new camera pose. Suppose that the transformation from coordinates relative to the old camera pose to coordinates relative to the new camera pose is given by the Euclidean transformation $E = (x \mapsto Rx + t)$ for a rotation matrix R and translation vector t.

Suppose given $p \in \mathbb{R}^3$ to be reprojected, with pinhole coordinates $\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{d} \end{pmatrix}$. Let $p' := E(p)$ be the transformed point, with pinhole coordinates $\begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{d}' \end{pmatrix}$. See also Example 9 for a description of Euclidian transformations in pinhole coordinates. Let $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ be the projection of a point to its image coordinates, i.e. we have $\pi(p) = f \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$.

Let $I, I' : \mathbb{R}^2 \to \mathbb{R}$ be the (intensity) images at the old (I) and new (I') camera positions. Let g, g' be the derivatives in image coordinates.

For a "perfect" image alignment and if p is correctly estimated, we have

$$I(\pi(p)) \approx I'(\pi(p')) \tag{29}$$

due to the assumption that correctly associated points look similar.

I.e. for on correctly estimated pixels, we have

$$I \circ \pi|_{\{\text{correct estimates}\}} \approx I' \circ \pi \circ E|_{\{\text{correct estimates}\}}$$

We denote by $\exp_{\mathfrak{se}(3)}$ the exponential map from the Lie algebra $\mathfrak{se}(3)$ of the Euclidean

group $SE(3)$ to $SE(3)$. I.e. for $\mathfrak{g} \in \mathfrak{se}(3)$, $\exp_{\mathfrak{se}(3)}(\mathfrak{g})$ is an element of $SE(3)$. We obtain

$$
\begin{aligned}
& g(\pi(p)) f(d(u) S_u \delta t + P_u \delta R) \\
\overset{\text{E.11}}{=} & g(\pi(p)) \frac{\mathrm{d}\pi(\exp_{\mathfrak{se}(3)}(\mathfrak{g})(p))}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)] \\
= & \frac{\mathrm{d} I(\pi(\exp_{\mathfrak{se}(3)}(\mathfrak{g})(p)))}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)] \\
\approx & \frac{\mathrm{d} I'(\pi(E(\exp_{\mathfrak{se}(3)}(\mathfrak{g})(p))))}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)] \\
= & g'(\pi(p')) \frac{\mathrm{d}\pi(E(\exp_{\mathfrak{se}(3)}(\mathfrak{g})(p)))}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)] \\
= & g'(\pi(p')) \frac{\mathrm{d}\pi((E \circ \exp_{\mathfrak{se}(3)}(\mathfrak{g}) \circ E^{-1})(p'))}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)] \\
= & g'(\pi(p')) f(d'(u') S_{u'}(-) + P_u(-)) \frac{\mathrm{d}(E \circ \exp_{\mathfrak{se}(3)}(\mathfrak{g}) \circ E^{-1})}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)].
\end{aligned}
$$

Regarding the term $\frac{\mathrm{d}(E \circ \exp_{\mathfrak{se}(3)}(\mathfrak{g}) \circ E^{-1})}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)]$: An element of $\mathfrak{h} \in \mathfrak{se}(3)$ acts on $\mathbb{R}^3$ via $(x \mapsto r \times x + w)$ for some rotation speed $r \in \mathbb{R}^3$ and linear velocity $w \in \mathbb{R}^3$. To compute the term $\frac{\mathrm{d}(E \circ \exp_{\mathfrak{se}(3)}(\mathfrak{g}) \circ E^{-1})}{\mathrm{d}\mathfrak{g}}[(\delta t, \delta R)]$:, we need to represent $\mathfrak{h}$ as the derivative of an one-parameter family of operators tangentially to $0 \in SE(3)$. One way of doing this is the family of operators $(x \mapsto (\lambda r) \times x + \lambda w)$, $\lambda \in \mathbb{R}$ near $\lambda = 0$.

We obtain (note that $E^{-1} = (x \mapsto R^{-1}(x - t))$)

$$
\begin{aligned}
& \frac{\mathrm{d}(E \circ \exp_{\mathfrak{se}(3)}(\mathfrak{g}) \circ E^{-1})}{\mathrm{d}\mathfrak{g}}[\mathfrak{h}] \\
= & \frac{\mathrm{d}}{\mathrm{d}\lambda} \left[E \circ (x \mapsto (\lambda r) \times x + \lambda w) \circ E^{-1} \right] \\
= & \frac{\mathrm{d}}{\mathrm{d}\lambda} \left[(x \mapsto Rx + t) \circ (x \mapsto (\lambda r) \times x + \lambda w) \circ \left(x \mapsto R^{-1}(x - t) \right) \right] \\
= & \frac{\mathrm{d}}{\mathrm{d}\lambda} \left[(x \mapsto Rx + t) \circ \left(x \mapsto (\lambda r) \times (R^{-1}(x - t)) + \lambda w \right) \right] \\
= & \frac{\mathrm{d}}{\mathrm{d}\lambda} \left(x \mapsto R[(\lambda r) \times (R^{-1}(x - t)) + \lambda w] + t \right) \\
= & \left(x \mapsto R[(r) \times (R^{-1}(x - t)) + w] + 0 \right) \\
= & \left(x \mapsto (Rr) \times (x - t) + Rw \right) \\
= & \left(x \mapsto (Rr) \times x + Rw - (Rr) \times t \right).
\end{aligned}
$$

Hence, the matrix transforming the coefficients $\begin{pmatrix} w \\ r \end{pmatrix}$ of $\mathfrak{h}$ to the coefficients $\begin{pmatrix} w' \\ r' \end{pmatrix}$ of

$\frac{\mathrm{d}(E\circ\exp_{\mathfrak{se}(3)}(\mathfrak{g})\circ E^{-1})}{\mathrm{d}\mathfrak{g}}[\mathfrak{h}]$ is

$$\begin{pmatrix} R & t\times(R(-)) \\ 0 & R \end{pmatrix}.$$

Note that from Remark 10, we obtain

$$(\delta\mathbf{d}') \approx \left(\frac{\mathbf{d}'}{\mathbf{d}}\right)^2 (\delta\mathbf{d})$$

for small camera rotations R.

Now we are able to proceed to the reprojection: We set $i'(u') := \left(\frac{\mathbf{d}(u)}{\mathbf{d}'(u')}\right)^2 i(u)$. Starting from a standard form, we obtain

$$\begin{aligned}
&\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ & Ac \end{pmatrix} \\
=&\begin{pmatrix} i(u)\left(\frac{\mathbf{d}(u)}{\mathbf{d}'(u')}\right)^2(\delta d'(u')) & \frac{1}{i(u)}F(u,d)Vc \\ & Ac \end{pmatrix} \\
=&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{\mathbf{d}(u)^2}{\mathbf{d}'(u')^2}\frac{1}{i'(u')}F(u,d)Vc \\ & Ac \end{pmatrix} \\
=&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{\mathbf{d}(u)}{\mathbf{d}'(u')^2}\frac{1}{i'(u')}[g(\pi(p))f(d(u)S_u(-)] \\ & \otimes[g(\pi(p))f(d(u)S_u(\delta t)+P_u(\delta R))]Vc \\ & Ac \end{pmatrix} \\
\approx&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{\mathbf{d}(u)}{\mathbf{d}'(u')^2}\frac{1}{i'(u')}[g'(\pi(p'))f(d'(u')S_{u'}(-))\frac{\mathrm{d}(E\circ\exp_{\mathfrak{se}(3)}(\mathfrak{g})\circ E^{-1})}{\mathrm{d}\mathfrak{g}}\circ(\delta t\mapsto(\delta t,0))] \\ & \otimes[g'(\pi(p'))f(d'(u')S_{u'}(-)+P_u(-))\frac{\mathrm{d}(E\circ\exp_{\mathfrak{se}(3)}(\mathfrak{g})\circ E^{-1})}{\mathrm{d}\mathfrak{g}}]Vc \\ & Ac \end{pmatrix} \\
=&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{\mathbf{d}(u)}{\mathbf{d}'(u')}\frac{1}{i'(u')}F(u',d')\left(\frac{\mathrm{d}(E\circ\exp_{\mathfrak{se}(3)}(\mathfrak{g})\circ E^{-1})}{\mathrm{d}\mathfrak{g}}\circ(\delta t\mapsto(\delta t,0))\right. \\ & \left.\otimes\frac{\mathrm{d}(E\circ\exp_{\mathfrak{se}(3)}(\mathfrak{g})\circ E^{-1})}{\mathrm{d}\mathfrak{g}}\right)Vc \\ & Ac \end{pmatrix} \\
=&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{\mathbf{d}(u)}{\mathbf{d}'(u')}\frac{1}{i'(u')}F(u',d')\left((l\mapsto Rl)\otimes\right. \\ & \left.((\delta t,\delta r)\mapsto(-(R(\delta r))\times t+R(\delta t),R(\delta r)))\right)Vc \\ & Ac \end{pmatrix} \\
\overset{V':=(\ldots)}{=}&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{\mathbf{d}(u)}{\mathbf{d}'(u')}\frac{1}{i'(u')}F(u',d')V'c \\ & Ac \end{pmatrix} \\
\approx&\begin{pmatrix} i'(u')(\delta d'(u')) & \frac{1}{i'(u')}F(u',d')V'c \\ & Ac \end{pmatrix}.
\end{aligned}$$

For the last approximation, we assume small camera movement, yielding $\mathbf{d}' \approx \mathbf{d}$.

As the last form is in standard form and uses terms relative to the new camera pose, the procedure given above is a method for reprojecting a standard form to a different camera pose.

Concerning the regularization by reweighting as described in section 6.6, we simply apply to the square root information values $i(u)$ the regularization analogously to section 6.6, which means using the square root of the regularization factor described in section 6.6 since we have a square root form of the information matrix here.

7.2.5 Splitting diagonal terms off

The aim of this section is to modify standard forms in such a way that we have information terms for the pixels which are completely independent of the pose uncertainty. I.e. we want to obtain a form

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ 0 & Ac \\ i^*(u)(\delta d(u)) & 0 \end{pmatrix}$$

Since the diagonal terms $i^*(u)$ are independent of the (complicated) pose uncertainty, they can be manipulated in a more simple fashion than the coefficients $i(u)$ since for the latter, there is an intricate connection to the terms of the block $\frac{1}{i(u)}F(u,d)Vc$.

So how do we obtain such a form: Consider a standard form

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ 0 & Ac \end{pmatrix}$$

If nothing is known about c (i.e. $A = 0$) then the "dangling" degrees of freedom modelled by c (these are then effectively zero eigenspaces of the information matrix) prevent us from splitting of positive definite diagonal terms since this would imply the information matrix to be positive definite. Hence, the information contained in A must restrict c sufficiently to allow diagonal terms:

Suppose that $A^TA = (A')^TA' + \sum_u (\frac{\lambda}{i(u)}F(u,d)V)^T(\frac{\lambda}{i(u)}F(u,d)V)'$ for some $\lambda > 0$ and a matrix A'. I.e. (as motivated above) we assume that we have some "excess" information in the pose uncertainty. Then, we may modify the standard form as follows:

$$\begin{pmatrix} i(u)(\delta d(u)) & \frac{1}{i(u)}F(u,d)Vc \\ 0 & A'c \\ 0 & \frac{\lambda}{i(u)}F(u,d)Vc \end{pmatrix}$$

Application of the "block row rotation" $\frac{1}{\sqrt{1+\lambda^2}}\begin{pmatrix}1 & & \lambda\\ & \sqrt{1+\lambda^2} & \\ \lambda & & -1\end{pmatrix}$ gives

$$\begin{pmatrix} \frac{i(u)}{\sqrt{1+\lambda^2}}(\delta d(u)) & \frac{\sqrt{1+\lambda^2}}{i(u)}F(u,d)Vc \\ 0 & A'c \\ \frac{\lambda}{\sqrt{1+\lambda^2}}i(u)(\delta d(u)) & 0 \end{pmatrix}$$

I.e. we have obtained a diagonal block $i^*(u)(\delta d(u)) := \frac{\lambda}{\sqrt{1+\lambda^2}}i(u)(\delta d(u))$ by siphoning off information from the i and from A.

7.3 Implementation

As of the time of writing, there is a partially functioning implementation of the methods introduced in this chapter available. In the sequel, we denote the variant described in section 6 as *diagonal C-LSD-SLAM* since it uses a diagonal matrix to approximate the information matrix. The more intricate variant described here in section 7 will be denoted as *extended C-LSD-SLAM*.

As manipulations of the information matrix should be performed near the maximum likelihood estimate, cf. e.g. Remark 26, the frame tracking module does not just optimize the pose of the new frame as in diagonal C-LSD-SLAM, but optimizes both the pose of the new frame and the degrees of freedom described by c in the standard form (cf. eq. (28)) when performing direct image alignment. This is implemented by first performing the (normal) hierarchical pose optimization and then performing a final optimization step involving the pose and c on the full resolution, unsmoothed depth map of the keyframe.

The other steps are performed where appropriate: The update step described in section 7.2.2 is performed when updating a depth map with the triangulations from a new frame. Reprojection and rescaling as described in sections 7.2.3 and 7.2.4 are implemented as part of the normal reprojection step. The separation of information described in section 7.2.5 is performed once per frame.

A remaining issue of the implementation is the fact that currently, there is a numerical stability issue which affects situations where the camera nearly does not move, leading to system crashes. The problem manifests itself as overflow of the double precision floating point numbers used, which hints at some issue of the numerical implementation.

8 Evaluation of C-LSD-SLAM

8.1 Datasets

We use the KITTI datasets for testing and evaluation[Geiger et al., 2013]. The KITTI datasets were captured with a car-mounted sensor platform on roads in and surrounding Karlsruhe, Germany. To us, the most relevant features of the sensor platform are four forward facing cameras (two monochrome cameras and two color cameras) whose global shutters are triggered synchronously with a frequency of 10 Hz. Furthermore, the sensor platform features a precision IMU/GPS module for capturing a ground truth vehicle trajectory. The KITTI datasets provide pre-rectified camera sequences, with each rectified camera stream having a resolution of about 1200×370 pixels and a horizontal FoV of about 90° (about 700px focal length).

The set of four cameras is particularly interesting for the purpose of evaluation since it allows us to effectively test the same situation with different data.

For evaluation, we will use the training datasets of the KITTI odometry challenge [Geiger et al., 2012]. We select these datasets since they feature pre-synchronized ground truth trajectory data and, more importantly, they are among the longest of the KITTI datasets (most sequences consist of several thousand frames), which is particularly suited for testing and evaluating the stability of the VO method.

8.2 Terminology and default parameters

Recall that we denote the variant of C-LSD-SLAM described in section 6 as *diagonal C-LSD-SLAM* whereas the variant described in section 7 is denoted as *extended C-LSD-SLAM*.

In developing C-LSD-SLAM, most of the method parameters have been left unchanged from the default parameters of LSD-SLAM. These include the (assumed) pixel noise, the minimum intensity gradient for including a pixel in the semi-dense depth map and the threshold of the huber loss function used in the tracking component for robustifying pixel matching. In the tracking component, the set of resolution level used for hierarchical direct image alignment have been extended to include the highest resolution level (native image resolution) whereas LSD-SLAM stops alignment at half of the native resolution. Furthermore, C-LSD-SLAM also uses more resolution levels on the coarse end of the resolution pyramid since the default configuration of LSD-SLAM bases its resolution level selection on the assumption that the full image size is about 600×400 pixels, which is too small for image sizes of the KITTI datasets.

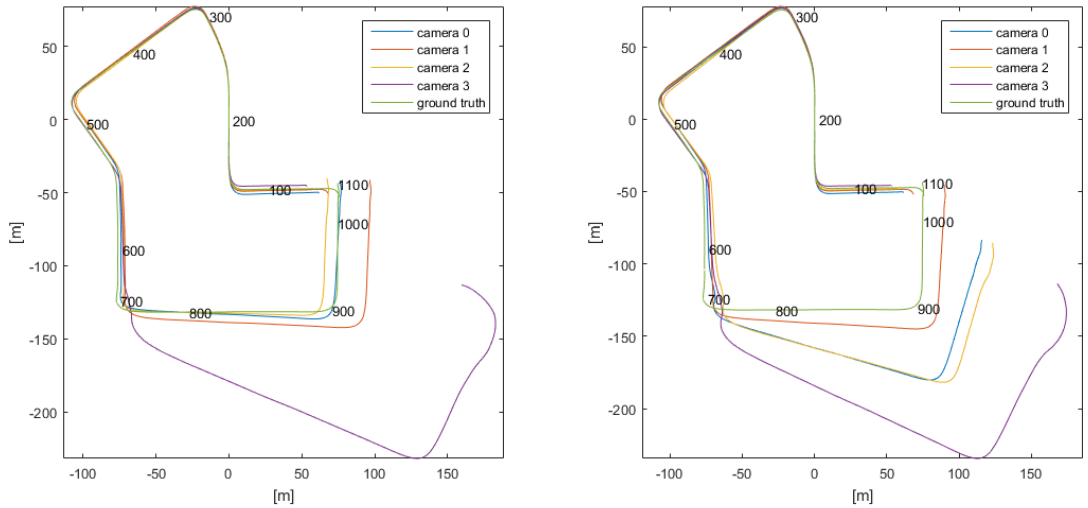


Figure 19: Trajectories generated by C-LSD-SLAM at full resolution on the training dataset 07 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM. The frame numbers are plotted along the ground truth trajectory. The trajectories are aligned at the 200th frame.

C-LSD-SLAM introduces several new parameters which are described below.

parameter	default value	notes
row decimation in the tracking module	8/4/2/1/1/1/...	On the full resolution level, every eighth image row is used for tracking, at half resolution level every fourth image row is used and so forth.
number of bins used for stochastic collision avoidance in the reprojection step	4 per pixel	two bins per pixel is the minimum usable number of bins
regularization reweighting exponent	11	cf. eq. (23)
regularization constant	$\varepsilon = 0.1$	this curtails huge entries of the standard form, cf. section 7.2.2. For comparison, the information value of a pixel is typically much larger than 1.

8.3 Robustness and repeatability

Note that the current implementation of the extended C-LSD-SLAM suffers from a numerical corruption which causes it to sometimes crash, cf. section 7.3. As this behaviour is specifically restricted to situations where the vehicle nearly does not move and thus

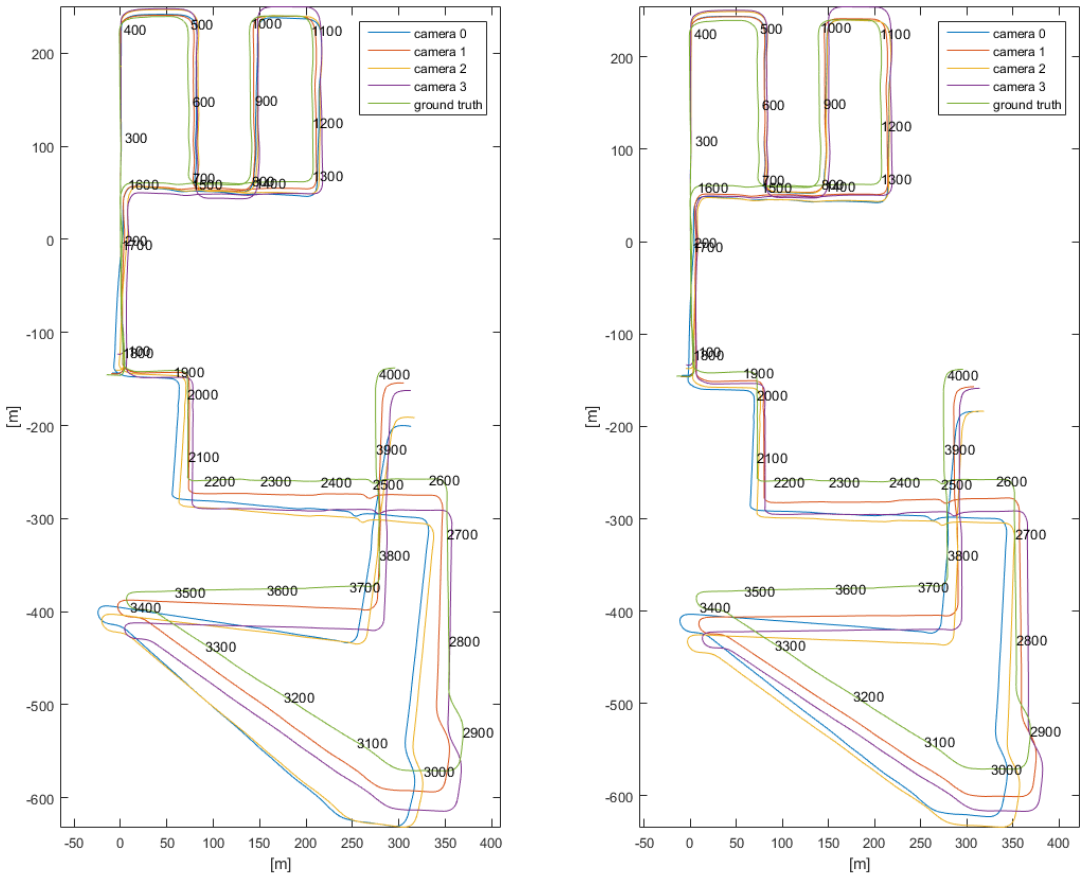


Figure 20: Trajectories generated by C-LSD-SLAM on the training dataset 08 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

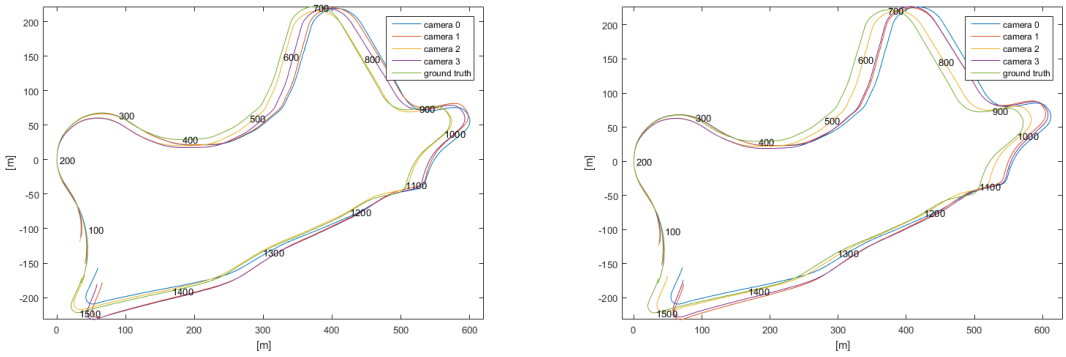


Figure 21: Trajectories generated by C-LSD-SLAM on the training dataset 09 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

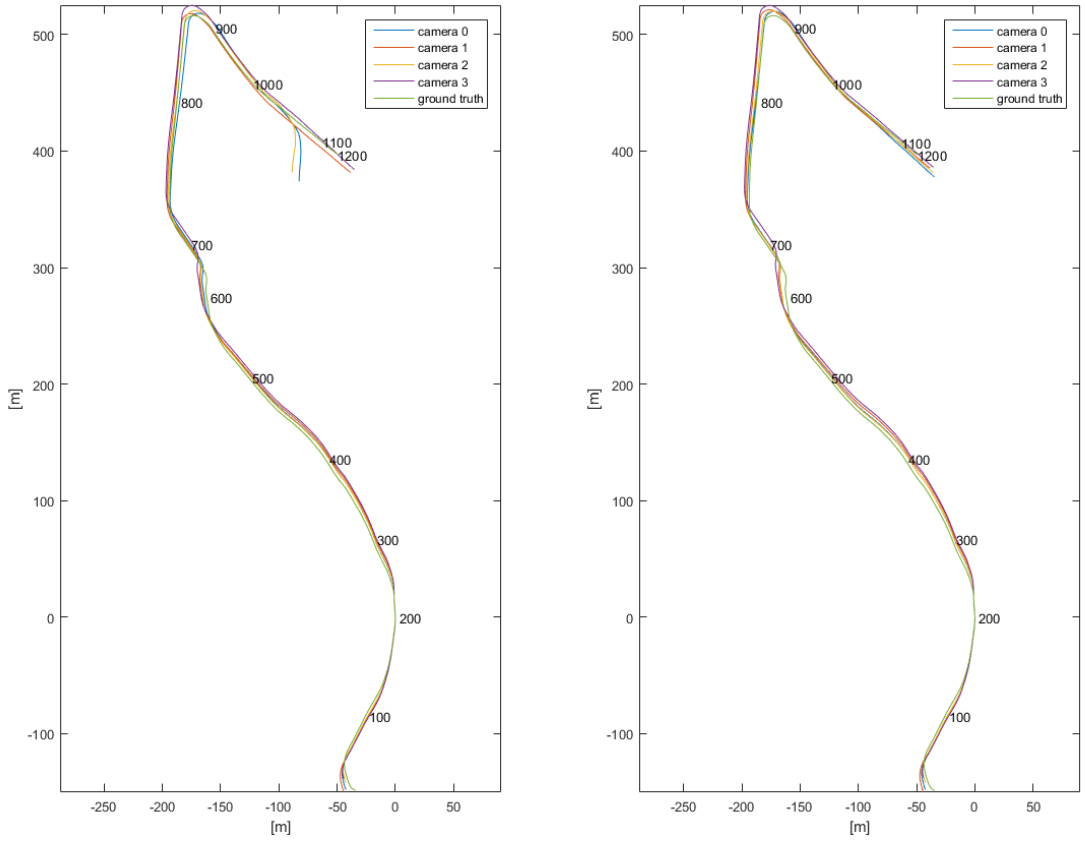


Figure 22: Trajectories generated by C-LSD-SLAM on the training dataset 10 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

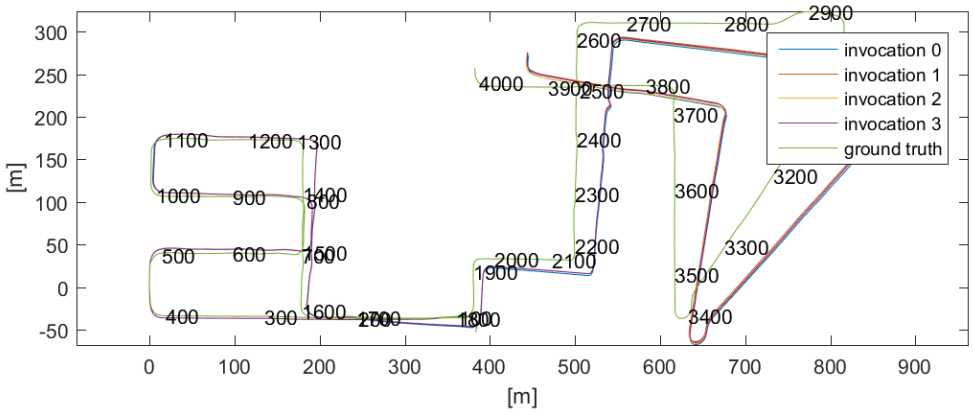


Figure 23: Result of multiple invocation of diagonal C-LSD-SLAM on the same dataset (camera 0 of dataset 08)

seems to be a bug of the current implementation instead of a property of the method itself, we have discarded results from tests in which the extended C-LSD-SLAM crashed and simply repeated the particular test to obtain clean results.

Note that there are only two effects that introduces randomness into the behaviour of the method. The first one is the initialization of the method, which is performed by random initialization of the depth map, cf. section 6.4.5. The second one is the non-deterministic behaviour introduced by the usage of multiple processing threads in several tasks: Although this effect has been minimized greatly in comparison to the original LSD-SLAM, it manifests itself to a small degree in rounding behaviour as well as the technique of statistical collision avoidance. In statistical collision avoidance used for keyframe reprojection (cf. section 6.4.3), this effect is present by design. The differences in rounding behaviour stem from summing certain floating point values over all pixels of a frame (e.g. in frame tracking, the coefficients of the normal equation linearized problem are calculated this way). Here, each thread processes a certain subset of all pixels and computes a thread-local sum for the pixels it has visited. Afterwards, the sums of the threads are added to obtain the total sum. Since the pixels a certain thread visits depends on the run-time behaviour and since the order of floating point additions affects the result due to rounding, the total sum varies slightly due to run-time thread behaviour.

In fig. 23, the diagonal C-LSD-SLAM was invoked several times on the same data and the resulting trajectory estimates plotted. Note that in this dataset, even the convergence phase at the beginning, where the system refines the initial random depth map to a valid depth map, is almost invisible. Afterwards, the behaviour is nearly deterministic. In particular, the variance between different invocations is much smaller than the variance resulting from different camera positions as seen in fig. 20.

In figs. 19 to 22, diagonal C-LSD-SLAM and extended C-LSD-SLAM have been invoked separately on all four cameras of several datasets of the KITTI odometry challenge. Note that the cameras are arranged in two pairs of forward facing stereo cameras with each having a baseline of about 50cm, cf. [Geiger et al., 2013]. Cameras 0 and 1 are monochrome cameras (camera 0 is the left camera). Cameras 2 and 3 are color cameras (camera 2 is the left camera). The two stereo pairs are stacked directly on top of each other, hence cameras 0 and 2 as well as 1 and 3 are close to each other. The variants of C-LSD-SLAM were invoked separately on the four cameras (i.e. each invocation processes the data of one of the four cameras) in order to compare the results which allows us to investigate the effect of different input data in the same situation. As a first result, note that in figs. 19 to 22, the behaviour of camera 0 is usually similar to the behaviour of camera 2 (blue and yellow graphs). Also, the behaviour of camera 1 is usually similar to the behaviour of camera 3 (red and purple graphs). This implies that the property whether the method uses monochrome or color camera data is less important than the mounting

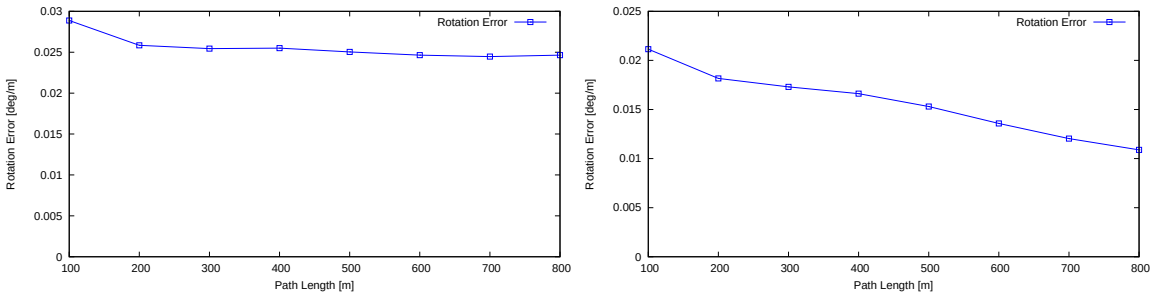


Figure 24: Rotation error versus track length on dataset 01 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

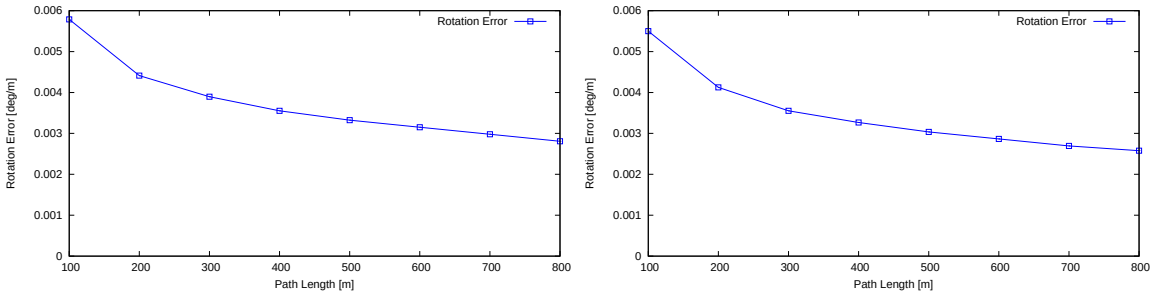


Figure 25: Rotation error versus track length on dataset 02 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

position of the cameras for the behaviour of the methods since, as described above, the cameras with similar behaviour are mounted closely to each other. The second result is that there are two distinct modes of behaviour: In the first mode, the system state is converged. Here, the result of the method accumulates errors only gradually. The second mode is system divergence. Here, the depth map estimate of the current frame is inconsistent to the geometry in the current frame, thus causing large estimation errors of the camera trajectory in a very short time. Examples are given by the corners in the trajectory estimates in fig. 19 as well as the turns at the end of fig. 22. Typically, the method recovers quickly from divergent behaviour and resumes convergent operation.

8.4 Accuracy

For evaluating the accuracy numerically, we apply the evaluation tools of the KITTI odometry challenge to the training datasets of the odometry challenge. To obtain a notion on how the error depends on path length, the evaluation tool does not just compute the end-to-end error of the whole trajectory, but also of subsections of the trajectory: For a given path length, the subsections of the trajectory of that length are created. The error for this path length is obtained by averaging the errors of the individual subsections.

Note that we focus our evaluation on the rotational error since we use a very simple technique for absolute scale estimation with limited accuracy, cf. section 6.7.2.

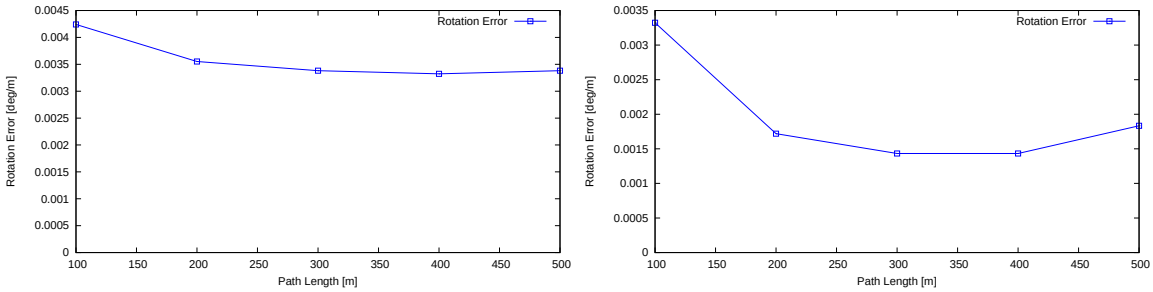


Figure 26: Rotation error versus track length on dataset 03 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

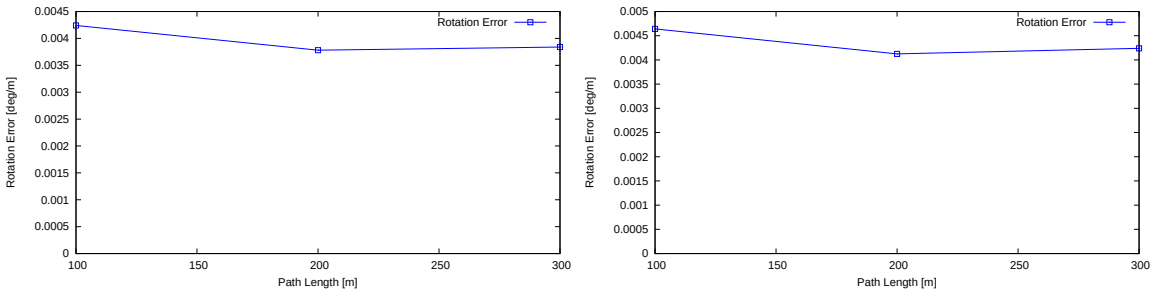


Figure 27: Rotation error versus track length on dataset 04 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

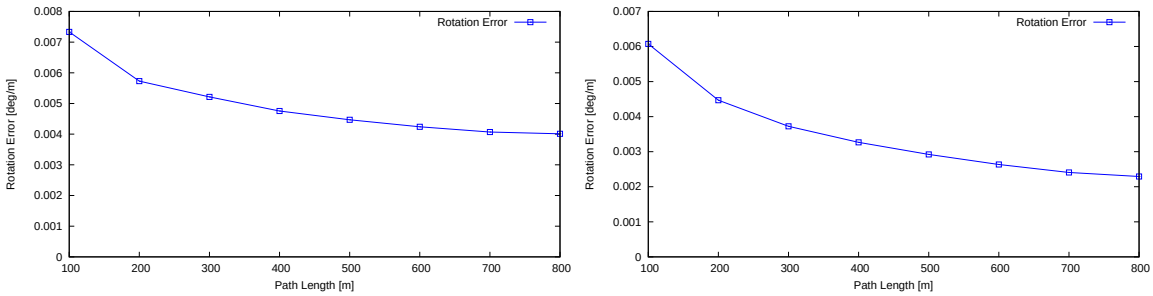


Figure 28: Rotation error versus track length on dataset 05 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

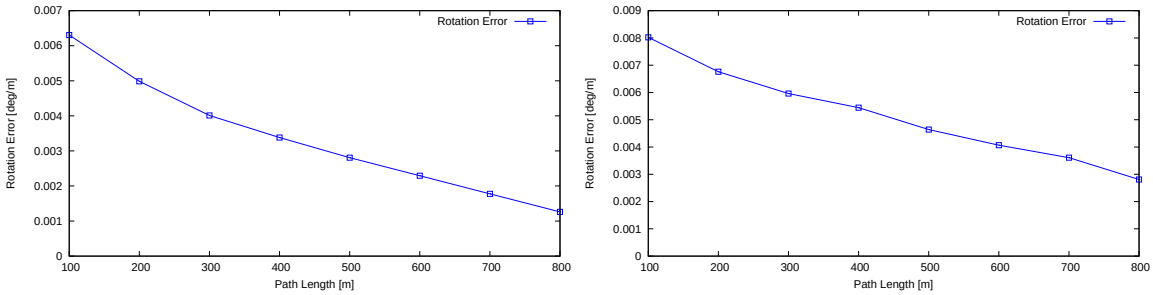


Figure 29: Rotation error versus track length on dataset 06 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

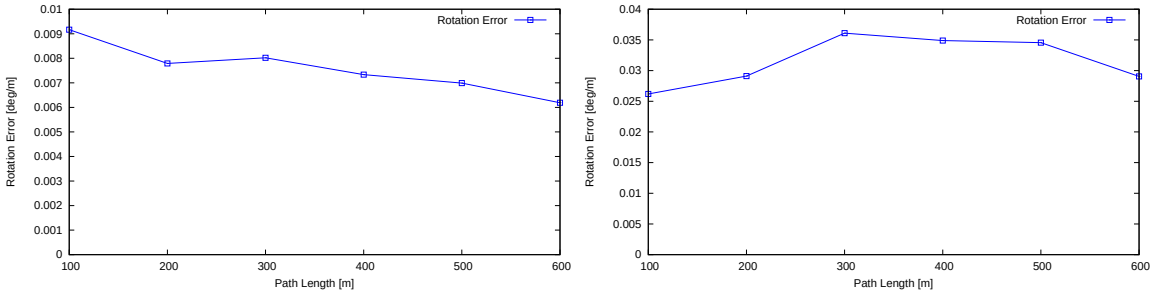


Figure 30: Rotation error versus track length on dataset 07 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

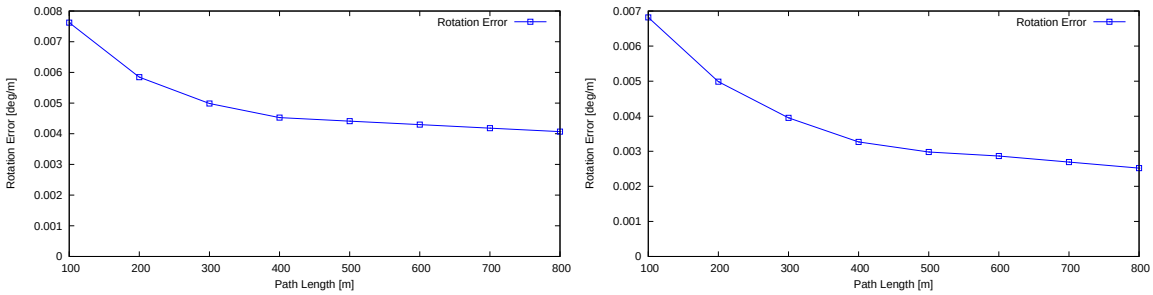


Figure 31: Rotation error versus track length on dataset 08 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

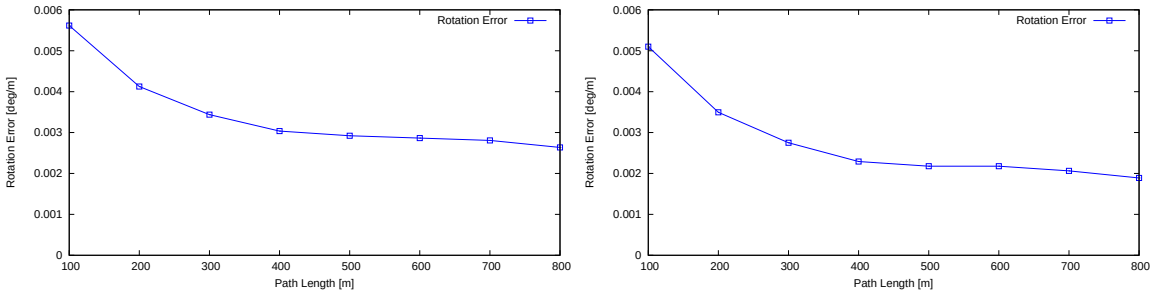


Figure 32: Rotation error versus track length on dataset 09 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

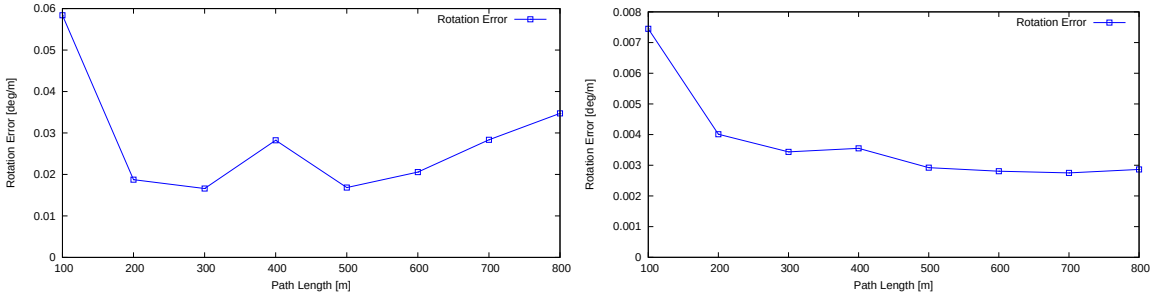


Figure 33: Rotation error versus track length on dataset 10 of the KITTI odometry challenge. Left: diagonal C-LSD-SLAM. Right: extended C-LSD-SLAM

Let us compare the results of the two variants of C-LSD-SLAM on the datasets:

Dataset	Comment
01	Both variants of C-LSD-SLAM take long for initial convergence, resulting in large total errors.
02	Extended C-LSD-SLAM is slightly better.
03	Extended C-LSD-SLAM is better.
04	Diagonal C-LSD-SLAM is slightly better.
05	Extended C-LSD-SLAM is slightly better.
06	Diagonal C-LSD-SLAM is better.
07	Diagonal C-LSD-SLAM is better due to a divergent section in the operation of extended C-LSD-SLAM.
08	Extended C-LSD-SLAM is better.
09	Extended C-LSD-SLAM is better.
10	Extended C-LSD-SLAM is better due to a divergent section in the operation of Diagonal C-LSD-SLAM.

As both variants of C-LSD-SLAM diverge once in the ten datasets, there is no clear winner regarding stability. In terms of accuracy, there is a clear tendency that the extended C-LSD-SLAM gives more accurate results. This provides justification for the increased complexity of the extended C-LSD-SLAM over the diagonal C-LSD-SLAM.

9 Densification

9.1 Introduction

Stereo methods estimate distances by performing stereo matching between different images. While there are many ways of performing stereo matching, they all rely on some form of image (dis)similarity to accept or reject candidate matches. The existence of dissimilarity depends on image texture, so stereo matching is fundamentally impossible in untextured image regions and difficult in regions with low texture. In particular, two surfaces with different geometrical structure may look exactly the same if they are untextured. In street scenes, there are many untextured regions such as building facades or the sky.

Thus, there are image regions for which stereo methods can not provide depth estimates. The image area for which a stereo method actually does not provide depth estimates is typically even larger due to technical limitations. For example, LSD-SLAM and its derivates restrict depth estimation to pixels with sufficient intensity gradient, which roughly means the pixels of edges in the image. This is the semi-dense depth map giving LSD-SLAM its name. Feature-based methods typically provide even more sparse depth maps.

Since we want to obtain a distance value for each pixel of an image, we need to densify the semi-dense depth map provided by LSD-SLAM. Thus we need some way to assign distance values to untextured regions. The core motivation for this section is to show that we can successfully densify the depth maps provided by LSD-SLAM with reasonable effort.

Note that the above examination of untextured regions means that we cannot obtain the structure of an untextured surface by observing it with a camera, so assigning depth values to an untextured region is ultimately some form of guesswork. So while some guessing errors will be unavoidable, reasonably accurate guesses can be made by usage of prior knowledge about the scene such as scene structure.

Thus, a natural approach to develop guessing strategies is to make the guesses consistent with the geometry of typical scenes. This way, the guessing strategies should produce correct results when applied to such a typical scene (i.e. prior knowledge is used). Here, traffic scenes have the property that image regions are usually either planar (road surface, building faces etc.) or they are well textured (e.g. trees). Note that the sky is an exception to this since it is both often completely untextured and it has infinite distance.

9.2 Raycasting method

For each pixel without depth estimate (we call this a target pixel), this method searches in several directions on the image for the nearest pixel with a depth estimate. These reference pixels are then used to find out whether the surrounding of the target pixel seems to be a plane segment by examining how well these reference pixels can be fitted to a plane. If this succeeds (i.e. the method decides that the target pixel is part of a plane segment), a depth estimate is created from the reference pixels via interpolation (cf. below). Otherwise, the method refrains from creating a depth estimate for the target pixel. This conservative approach to creating estimates is intended to minimize the number of grossly incorrect estimates created by the raycasting method. This helps limiting the influence of depth outliers unavoidably present in the original depth map.

9.2.1 Basic method

For each pixel without a depth estimate (we call such a pixel a target pixel), the raycasting method checks whether the surrounding of the target pixel seems to be a planar section. This proceeds in two steps.

Step one: In the first step, several rays are cast from the target pixel into several directions. The methods searches along each ray for the first pixel with a valid depth estimate, which is used as a depth reference for the target pixel.

Step two: In the second step, the algorithm decides whether the surrounding of the target pixel seems to be a planar section by analyzing the depth references collected for the target pixel in the previous step. Toward that end, it fits a plane to the depth references, with the confidences of the depth references used as weights. Then, the plane is re-evaluated at depth references and compared to the original depth values, with the confidences of the depth references used to generate confidence intervals. If the re-evaluated depth values are within such a confidence interval, we call the depth reference consistent with the plane, otherwise inconsistent. If the number of inconsistent depth references is too large, no depth estimate is generated for the target pixel. Otherwise, a (new) plane is fitted to the consistent depth references and the evaluation of this new plane at the target pixel is used as depth estimate for the target pixel. Furthermore, the new plane is evaluated at the consistent depth references and compared to the reference value; the largest difference is used as covariance estimate of the newly created depth estimate of the target pixel.

Multiple iterations The above method will obviously leave pixels without a depth estimate. To obtain better coverage, the basic method described above is repeated a fixed

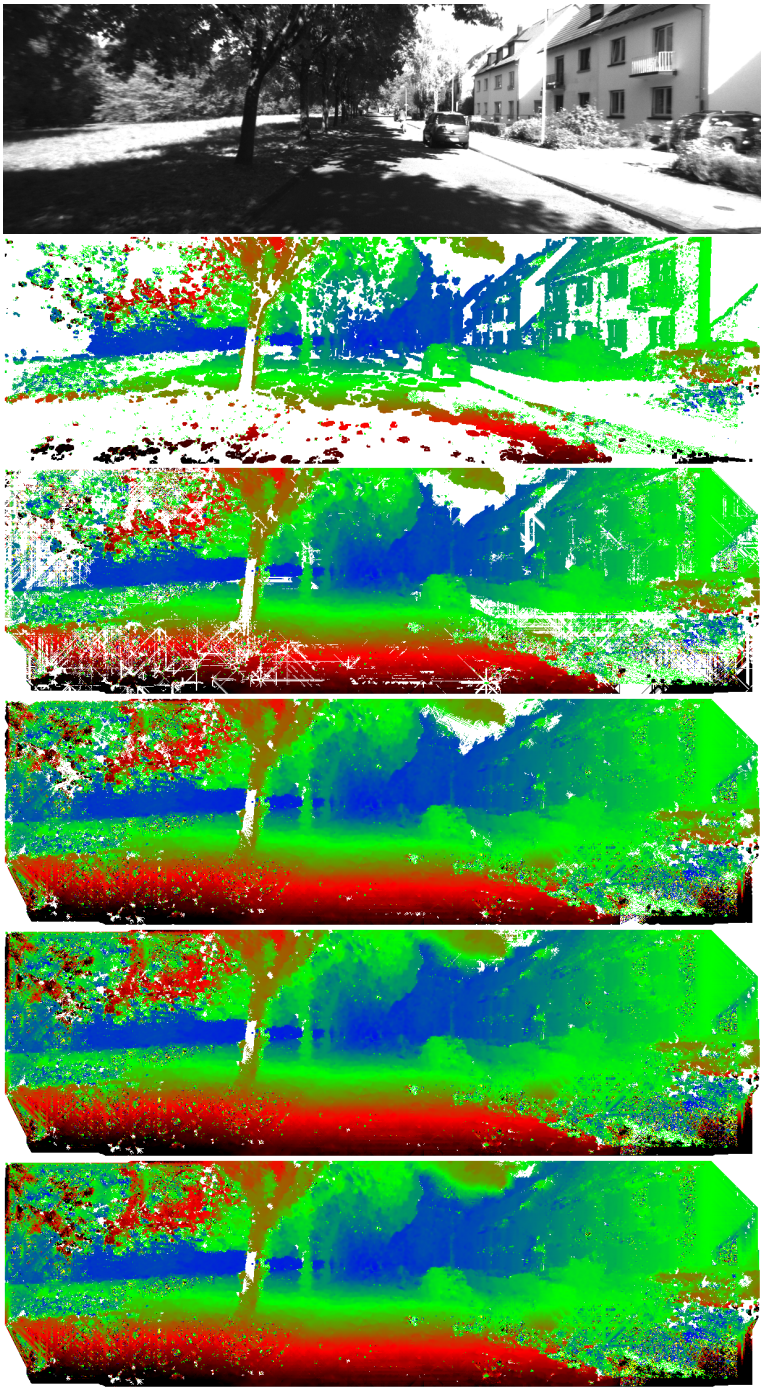


Figure 34: Densification via raycasting method on a depth map of sequence 08 of the KITTI odometry benchmark. We use 8 search directions, cf.fig. 37. Top image: camera image. 2nd image: original depth map provided by C-LSD-SLAM (color goes from near to far from dark red (black) over green to blue. The remaining images show densified depth map after each densification iteration.

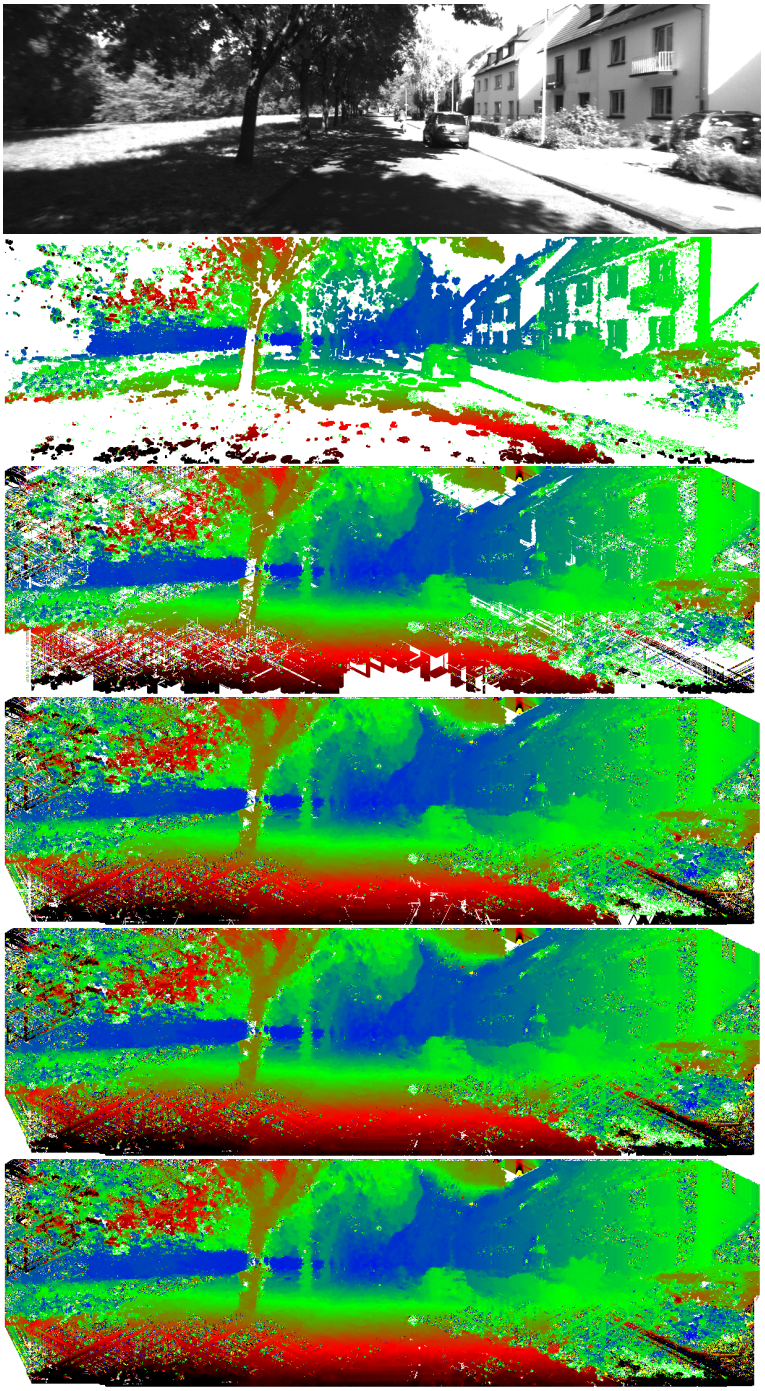


Figure 35: Densification via raycasting method on a depth map of sequence 08 of the KITTI odometry benchmark. We use 6 search directions, cf.fig. 38. Top image: camera image. 2nd image: original depth map provided by C-LSD-SLAM (color goes from near to far from dark red (black) over green to blue. The remaining images show densified depth map after each densification iteration.

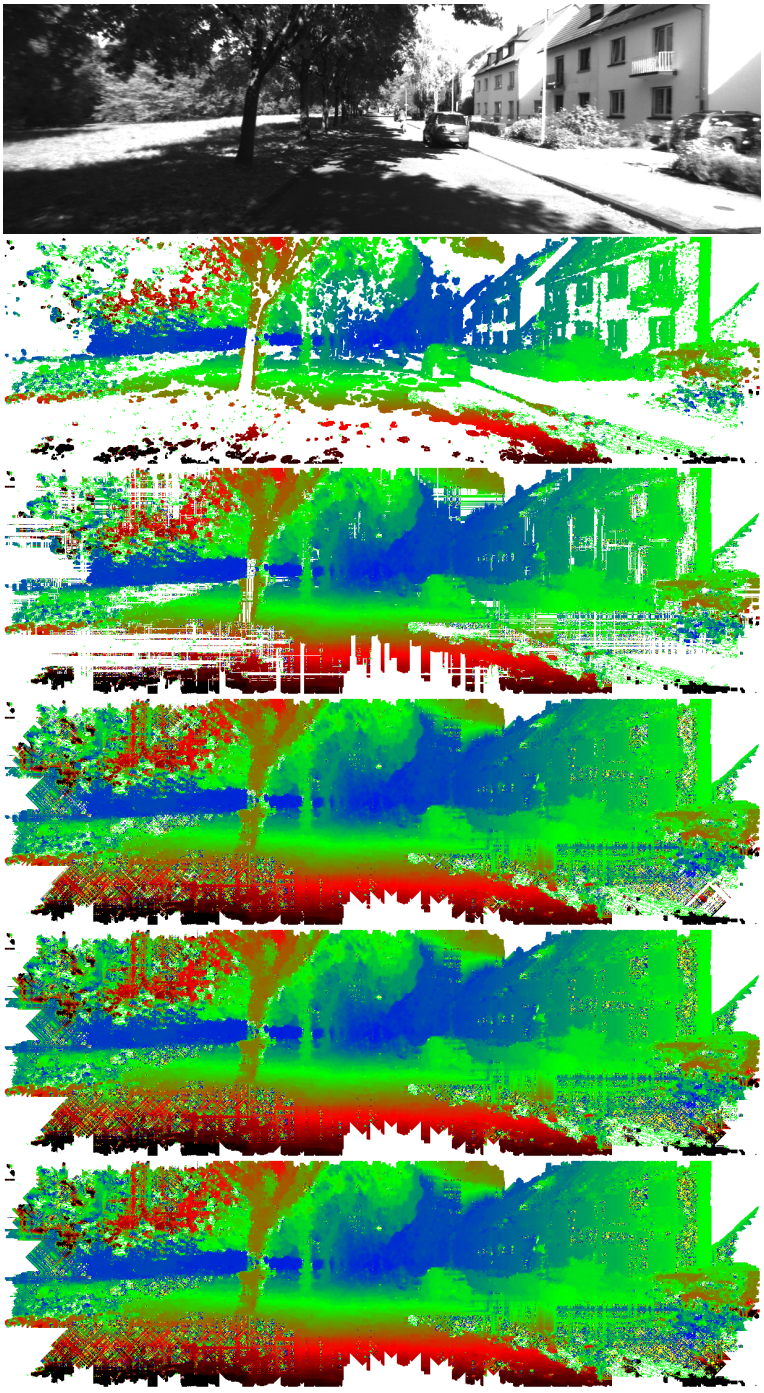


Figure 36: Densification via raycasting method on a depth map of sequence 08 of the KITTI odometry benchmark. We use 4 search directions, cf.fig. 39. Top image: camera image. 2nd image: original depth map provided by C-LSD-SLAM (color goes from near to far from dark red (black) over green to blue. The remaining images show densified depth map after each densification iteration.

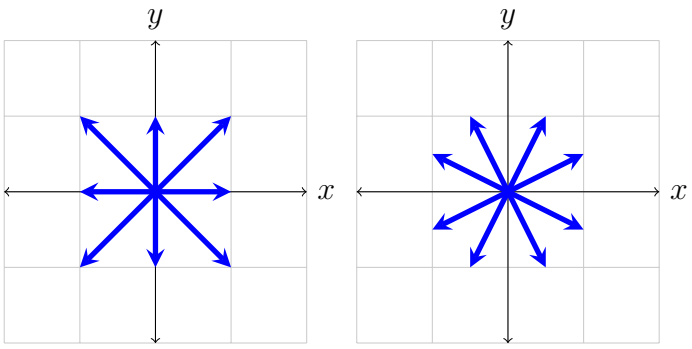


Figure 37: Two sets of search directions, each consisting of 8 directions.

low number of times to increase the coverage successively. For the different iterations, different sets of ray directions are used to vary the depth references.

9.2.2 Implementation

For implementing step one, we reverse the raycasting: Instead of starting from a target pixel, the algorithm starts from the boundaries of the image in reverse ray direction. When going along a ray, the algorithm keeps the location of the last pixel encountered that has a valid depth map in a variable and disperses this reference to the next valid pixel in ray direction along the ray to each target pixel encountered as a depth reference (this approach has some resemblance to the dynamic programming approach used in Semi-Global Matching, albeit without dynamic programming). This way, the algorithm needs to visit each pixel only once per ray direction. Furthermore, it can be well parallelized and vectorized.

For implementing step two, we process each target pixel individually, with a standard Gauss solver used for plane fitting.

For the view ray directions, we use patterns of 4, 6 or 8 ray directions. The ray vectors are chosen in such a way that one component is ± 1 and the other is a rational number with small denominator, which simplifies finding consistent ways of rounding to integral pixel coordinates when moving along a ray, without the formation of gaps between adjacent rays. For example, we have the sets of vectors consisting of $(\pm 1, 0)$, $(0, \pm 1)$, $(\pm 1, \pm 1)$ and consisting of $(\pm 1, \pm 0.5)$, $(\pm 0.5, \pm 1)$ as 8-ray direction patterns, cf. fig. 37. We alternate between these two patters when doing multiple iterations.

9.2.3 Discussion

Consider fig. 34 as it shows the basic properties of densification via the raycasting method: Note that planar patches can be successfully filled with new depth estimated provided the

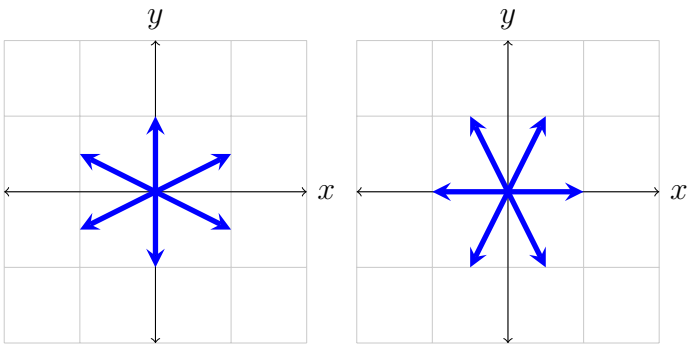


Figure 38: Two sets of search directions, each consisting of 6 directions.

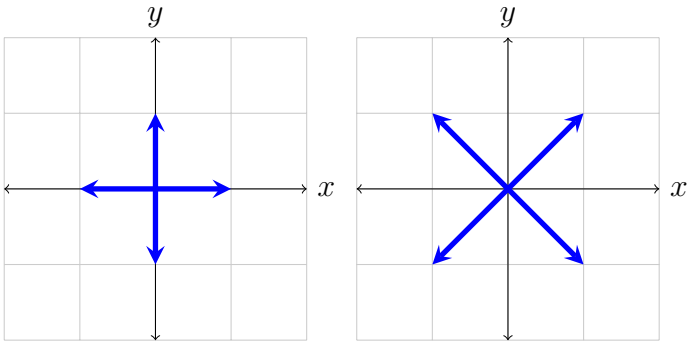


Figure 39: Two sets of search directions, each consisting of 4 directions.

boundary and inside of the patch have sufficient numbers of correct depth estimates. This works even for large patches. Isolated outliers prevent the creation of new estimates in the ray directions starting from their location. This way, there is typically a star pattern without new estimates surrounding such outliers after the first iteration of densification. Since the search direction patterns are alterated between densification iterations and since newly created depth estimates also serve as reference pixels in succeeding iterations, such outliers will be successively surrounded with correct depth estimates.

Densification via the raycasting method does not work well in regions with high outlier density (e.g. lower right corner in fig. 34). Furthermore, as the sky has distance infinity, it does not fit the assumption of planarity of patches with some correct distance estimates at the boundary of the patches. Hence, the estimates created for sky pixels are generally incorrect.

Note that eight search directions is about the minimum number of search directions: Reduction to six or four search directions reduces the level of redundancy for detecting whether a target pixel is within a planar patch or not, thus increasing the sensitivity to outliers significantly, cf. figs. 35 and 36.

The method can be easily parallelized and is computationally reasonably inexpensive. In our implementation, the configuration of eight search directions and two iterations yield

on the KITTI datasets (ca. 500kPx) a runtime of about 40ms per frame on a Intel Core i5 M540 (dual core, 2.53GHz).

10 Summary

We have investigated the adaption of monocular visual odometry for the usage in in-car Augmented Reality infotainment systems. To that end, the geometry of the triangulation problem was analyzed in detail. Using explicit approximations, it has been shown that the stereo baseline caused by vehicle movement contributes essentially to the capability of resolving distant features appearing frequently in traffic scenes.

Concerning visual odometry itself, it has been shown that (semi-)dense direct techniques can be combined in a scalable fashion with the filter-based approach to visual odometry. Such VO methods produce semi-dense depth maps with low-latency, which makes these kind of methods ideally suited for high-speed Augmented Reality applications.

Here, the key obstacle was that traditional filter-based approaches scale poorly to large number of scene point estimates: Filter-based visual odometry methods traditionally maintain a (dense) covariance matrix for the uncertainty of the scene point estimates. The complexity of this covariance matrix is quadratic in the number of scene point estimates, which prevents scaling filter-based VO methods to large numbers of pixels. Our solution to this problem is to replace the (dense) inverse of the covariance matrix, which is called the information matrix, with a structure that has linear complexity in the number of scene point estimates, thus enabling dense or semi-dense depth estimation. We have proposed two variants for this: In the first variant, the information matrix is replaced with a diagonal matrix, thus discarding the correlation between scene point estimates introduced by the uncertainty of the estimation of the camera poses. In the second variant, the correlation between scene point estimates caused by camera pose uncertainty is described in a unified, abstract fashion. This way, linear complexity in the number of scene point estimates is maintained despite maintaining a notion of the correlation of the scene point estimates caused by the uncertainty of all pose estimates.

An implementation of these techniques, called C-LSD-SLAM, has been developed based on the existing visual SLAM method LSD-SLAM. Care has been taken that all relevant routines of the method are fully parallelized, thus allowing easy scale up to the massive level of parallelization found in current processing units.

As fast forward motion causes instability in C-LSD-SLAM, a simple reweighting scheme has been introduced which is capable of stabilizing the system against erroneous scene point estimates near the epipolar direction.

Finally, C-LSD-SLAM has applied to and evaluated on the odometry benchmark datasets of the KITTI suite, verifying the suitability of the method on real-world data.

11 Appendix

11.0.1 Linear transformation of normal distributions

Lemma 30 (Linear transformation of normal distributions 1)**.** *Suppose given a normally distributed random variable X on $\mathbb{R}^n$, $n \geq 1$ with mean $\mu \in \mathbb{R}^n$ and covariance matrix $P \in \mathbb{R}^{n \times n}$.*

Suppose given an invertible $n \times n$-matrix L.

Then $Y = LX$ is a normally distributed random variable with mean $L\mu$ and covariance matrix LPL^T.

Proof. Note that by Remark 12, $Y = LX$ is a random variable.

For convenience, let

$$\begin{aligned} \nu :=& L\mu \\ Q :=& LPL^T. \end{aligned}$$

The probability density function of the normally distributed random variable X on $\mathbb{R}^n$ is

$$\mathrm{d}X(x) = \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left(-\frac{1}{2}(x-\mu)^T P^{-1}(x-\mu)\right),$$

with $|P|$ being the absolute value of the determinant of P.

Substituting $x = L^{-1}y$, we obtain via the substitution rule of integration the probability density function of Y:

$$\begin{aligned} \mathrm{d}Y(y) =& \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left(-\frac{1}{2}(L^{-1}y-\mu)^T P^{-1}(L^{-1}y-\mu)\right) \cdot |L^{-1}| \\ =& \frac{1}{\sqrt{(2\pi)^n |L| \cdot |P| \cdot |L^T|}} \exp\left(-\frac{1}{2}(L^{-1}(y-\nu))^T P^{-1}(L^{-1}(y-\nu))\right) \\ =& \frac{1}{\sqrt{(2\pi)^n |LPL^T|}} \exp\left(-\frac{1}{2}(y-\nu)^T (L^{-1})^T P^{-1}(L^{-1})(y-\nu)\right) \\ =& \frac{1}{\sqrt{(2\pi)^n |Q|}} \exp\left(-\frac{1}{2}(y-\nu)^T (LPL^T)^{-1}(y-\nu)\right) \\ =& \frac{1}{\sqrt{(2\pi)^n |Q|}} \exp\left(-\frac{1}{2}(y-\nu)^T Q^{-1}(y-\nu)\right) \end{aligned}$$

This is the probability density of a normally distributed random value with mean ν and covariance matrix Q. □

Lemma 31 (Adding a constant to a normal distribution)**.** *Suppose given a normally*

distributed random variable X on $\mathbb{R}^n$, $n \geq 1$ with mean $\mu \in \mathbb{R}^n$ and covariance matrix $P \in \mathbb{R}^{n \times n}$. Suppose given $a \in \mathbb{R}^n$.

Then $Y = X + a$ is a normally distributed random variable on $\mathbb{R}^n$ with mean $\mu + a$ and covariance matrix P.

Proof. The probability density function of the normally distributed random variable X on $\mathbb{R}^n$ is

$$\mathrm{d}X(x) = \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left(-\frac{1}{2}(x-\mu)^T P^{-1}(x-\mu)\right),$$

with $|P|$ being the absolute value of the determinant of P.

Substituting $x = y - a$ yields the probability density function

$$\begin{aligned}\mathrm{d}Y(y) =& \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left(-\frac{1}{2}(y-a-\mu)^T P^{-1}(y-a-\mu)\right) \\ =& \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left(-\frac{1}{2}(y-(\mu+a))^T P^{-1}(y-(\mu+a))\right),\end{aligned}$$

of Y, which is the probability density function of a normally distributed random variable with mean $\mu + a$ and covariance matrix P. □

Lemma 32 (Linear transformation of normal distributions 2)**.** *Suppose given a normally distributed random variable X on $\mathbb{R}^n$, $n \geq 1$ with mean $\mu \in \mathbb{R}^n$ and covariance matrix $P \in \mathbb{R}^{n \times n}$.*

Suppose given $m \geq 1$. Suppose given an $m \times n$-matrix L with full row rank (i.e. the linear transformation $L : \mathbb{R}^n \to \mathbb{R}^m, x \mapsto Lx$ is surjective).

Then $Y = LX$ is a normally distributed random variable with mean $L\mu$ and covariance matrix LPL^T.

Proof. For $i \geq 0$, let E_i be the $i \times i$-identity matrix. For $i, j \geq 0$, let $0_{i \times j}$ be the $i \times j$-zero matrix.

Let A be the kernel of $LP^{\frac{1}{2}}$. Let $k := \dim A$. Choose a rotation matrix R on $\mathbb{R}^n$ such that $R\langle e_1, \ldots, e_k\rangle = A$, i.e. R maps the span of the first k coordinate vectors to A. In particular, this implies $\ker(LP^{\frac{1}{2}}R) = \langle e_1, \ldots, e_k\rangle =: B$. Note that for the orthogonal complement $B^\perp$ of B in $\mathbb{R}^n$, this implies $B^\perp = \langle e_{k+1}, \ldots, e_n\rangle$.

Let $Z := R^{-1}P^{-\frac{1}{2}}(X - \mu)$, which is by Lemmas 30 and 31 a normally distributed variable on $\mathbb{R}^n$ with mean 0 and covariance matrix $R^{-1}P^{-\frac{1}{2}}P(R^{-1}P^{-\frac{1}{2}})^T = R^{-1}(R^{-1})^T = (R^T R)^{-1} = E_n^{-1} = E_n$. Note that we have $Y = LX = (LP^{\frac{1}{2}}R)Z + L\mu$.

Let V be the projection of Z to the first k components in $\mathbb{R}^n$. Let W be the projection of Z to the remaining components $k+1$ to n in $\mathbb{R}^n$. I.e. we have the orthogonal decomposition $Z = V + W$ along the orthogonal decomposition of $\mathbb{R}^n$ in B and $B^\perp$.

Note that as the covariance matrix of Z is E_n (note in particular that its components in $\mathbb{R}^n$ are mutually independent random variables), V is on $\mathbb{R}^k$ a normally distributed random variable with covariance matrix E_k and mean 0 and W is on $\mathbb{R}^{n-k}$ a normally distributed random variable with covariance matrix E_{n-k} and mean 0.

Let ι_B be the submatrix of E_n consisting of the first k columns of E_n, that is

$$\iota_B := \begin{pmatrix} E_k \\ 0_{n-k\times k} \end{pmatrix}.$$

Note that ι_B is the matrix of the canonical inclusion map $B \hookrightarrow \mathbb{R}^n$. Let $\iota_{B^\perp}$ be the submatrix of E_n consisting of the first $n-k$ columns of E_n, that is

$$\iota_{B^\perp} := \begin{pmatrix} 0_{k\times n-k} \\ E_{n-k} \end{pmatrix}.$$

Note that $\iota_{B^\perp}$ is the matrix of the canonical inclusion map $B^\perp \hookrightarrow \mathbb{R}^n$.

We obtain

$$\begin{aligned} \iota_B \iota_B^T + \iota_{B^\perp} \iota_{B^\perp}{}^T &= \begin{pmatrix} E_k \\ 0_{n-k\times k} \end{pmatrix} \begin{pmatrix} E_k & 0_{k\times n-k} \end{pmatrix} + \begin{pmatrix} 0_{k\times n-k} \\ E_{n-k} \end{pmatrix} \begin{pmatrix} 0_{n-k\times k} & E_{n-k} \end{pmatrix} \\ &= \begin{pmatrix} E_k & 0_{k\times n-k} \\ 0_{n-k\times k} & 0_{n-k\times n-k} \end{pmatrix} + \begin{pmatrix} 0_{k\times k} & 0_{k\times n-k} \\ 0_{n-k\times k} & E_{n-k} \end{pmatrix} \\ &= E_n. \end{aligned} \tag{30}$$

As $\operatorname{im} V = \ker(LP^{\frac{1}{2}}R)$, we have

$$LP^{\frac{1}{2}}R\iota_B = 0. \tag{31}$$

Also, the decomposition $Z = V + W$ implies

$$Y = (LP^{\frac{1}{2}}R)Z + L\mu = (LP^{\frac{1}{2}}R\iota_{B^\perp})W + L\mu. \tag{32}$$

The matrix L has full row rank, so the map $x \mapsto Lx$ is surjective (i.e. an epimorphism). Furthermore, P is invertible and B is the kernel of $LP^{\frac{1}{2}}R$, so the matrix $LP^{\frac{1}{2}}R\iota_{B^\perp}$ is invertible. In particular, this implies $n - k = m \geq 1$.

Hence, Lemmas 30 and 31 together with eq. (32) imply that Y is a normally distributed

variable with mean $L\mu$ and covariance

$$\begin{aligned}
Q =& (LP^{\frac{1}{2}}R\iota_{B^\perp})E_{n-k}^{-1}(LP^{\frac{1}{2}}R\iota_{B^\perp})^T \\
\overset{(31)}{=}& (LP^{\frac{1}{2}}R\iota_{B^\perp})(LP^{\frac{1}{2}}R\iota_{B^\perp})^T + (LP^{\frac{1}{2}}R\iota_B)(LP^{\frac{1}{2}}R\iota_B)^T \\
=& (LP^{\frac{1}{2}}R)(\iota_B\iota_B^T + \iota_{B^\perp}{\iota_{B^\perp}}^T)(LP^{\frac{1}{2}}R)^T \\
\overset{(30)}{=}& (LP^{\frac{1}{2}}R)E_n(LP^{\frac{1}{2}}R)^T = LP^{\frac{1}{2}}RR^TP^{\frac{1}{2}}L^T = LP^{\frac{1}{2}}P^{\frac{1}{2}}L^T \\
=& LPL^T.
\end{aligned}$$

□

11.0.2 Quadratic functionals

Lemma 33 (semidefinite bounded quadratic functionals attain their minimum)**.** *Suppose given a quadratic functional $q(x) = x^TAx + lx + c$, $x \in \mathbb{R}^n$. Suppose that A is symmetric and positive semidefinite. Suppose that q is bounded from below, i.e. there exists $d \in \mathbb{R}$ such that $q(x) \geq d$ for all $x \in \mathbb{R}^n$.*

Then there exists $x_0 \in \mathbb{R}^n$ such that $q(x_0) = \min_{x\in\mathbb{R}^n} q(x)$, i.e. q attains its minimum.

Proof. Since A is symmetric, it has orthogonal eigenspaces by the principal axis theorem. Thus via choice of a suitable basis of $\mathbb{R}^n$, we may assume without of generality that A is a diagonal matrix. I.e. $A = \operatorname{diag}(a_1, \ldots, a_n)$, with all $a_i \geq 0$ since A is positive semidefinite. Hence, introducing the conventions $l = (l_1, \ldots, l_n)$ and $x = (x_1, \ldots, x_n)^T$, we obtain

$$q(x) = c + \sum_{i=1}^{n} \underbrace{(a_ix_i^2 + l_ix_i)}_{=:q_i(x_i)}$$

I.e. q is the sum of the scalar functionals q_i (plus the constant c), so we may minimize q by minimize each q_i separately. Since q is bounded from below, each q_i is bounded from below. For each $i \in \{1, \ldots, n\}$, there are two cases:

$a_i = 0$: Assume $l_i \neq 0$. This would imply q_i to be a non-zero linear function. In particular, it would imply q_i to be unbounded, which is impossible. Hence, we have $l_i = 0$. Thus, $q_i(x_i) = \text{const} = 0$ for all $x_i \in \mathbb{R}$. Thus, q_i attains its minimum e.g. at $x_i^* := 0$.

$a_i \neq 0$: Since A is positive semidefinite, we actually have $a_i > 0$. By completing the square, we have $q_i(x_i) = a_i\left(x_i + \frac{l_i}{2a_i}\right)^2 - \frac{l_i^2}{4a_i}$, hence q_i attains its unique global minimum at $x_i^* := -\frac{l_i}{2a_i}$.

Setting $x_0 := (x_1^*, \ldots, x_n^*)^T$, we have $q(x_0) = \min_{x\in\mathbb{R}^n} q(x)$ by construction of x_0. □

Lemma 34. *Suppose given a quadratic functional $q(x) = x^T Ax + lx + c$, $x \in \mathbb{R}^n$. Suppose given a global minimum $x_0 \in \mathbb{R}^n$ of q, that is $q(x_0) = \min_{x \in \mathbb{R}^n} q(x)$. Then*

$$q(x) = (x - x_0)^T A(x - x_0) + q(x_0) \text{ for all } x \in \mathbb{R}^n. \tag{33}$$

Proof. Since x_0 is a minimum of q, Fermat's theorem on stationary points implies

$$0 = \frac{\mathrm{d}q}{\mathrm{d}x}\Big|_{x=x_0} = 2x_0^T A + l,$$

hence

$$x_0^T A = -\frac{1}{2}l. \tag{34}$$

We have

$$\begin{aligned}
(x - x_0)^T A(x - x_0) + q(x_0) =& x^T Ax - x_0^T Ax - x^T Ax_0 + x_0^T Ax_0 + q(x_0) \\
\overset{(34)}{=}& x^T Ax + \frac{1}{2}lx + \frac{1}{2}x^T l^T + x_0^T Ax_0 + q(x_0) \\
=& x^T Ax + lx + x_0^T Ax_0 + x_0^T Ax_0 + lx_0 + c \\
=& q(x) + 2x_0^T Ax_0 + lx_0 \\
\overset{(34)}{=}& q(x) - lx_0 + lx_0 \\
=& q(x).
\end{aligned}$$

□

12 Bibliography

[Bell et al., 2001] Bell, B., Feiner, S., and Höllerer, T. (2001). View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 101–110. ACM.

[Berger, 1997] Berger, M. O. (1997). Resolving occlusion in augmented reality: a contour based approach without 3D reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR) 1997*, pages 91–96. IEEE.

[Birman and Solomjak, 1980] Birman, M. S. and Solomjak, M. Z. (1980). *Spectral Theory of Self-Adjoint Operators in Hilbert Space*. D. Reidel Publishing Company, Dordrecht, Holland.

[Chiuso et al., 2000] Chiuso, A., Brockett, R., and Soatto, S. (2000). Optimal structure from motion: Local ambiguities and global estimates. *International journal of computer vision*, 39(3):195–228.

[Civera et al., 2008] Civera, J., Davison, A. J., and Montiel, J. M. (2008). Inverse depth parametrization for monocular SLAM. *IEEE transactions on robotics*, 24(5):932–945.

[Davison, 2003] Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision (ICCV) 2003*, pages 1403–1210 vol.2. IEEE.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067.

[Du et al., 2016] Du, C., Chen, Y.-L., Ye, M., and Ren, L. (2016). Edge snapping-based depth enhancement for dynamic occlusion handling in augmented reality. In *International Symposium on Mixed and Augmented Reality (ISMAR) 2016*, pages 54–62. IEEE.

[Eade and Drummond, 2006] Eade, E. and Drummond, T. (2006). Scalable monocular SLAM. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2006*, volume 1, pages 469–476. IEEE Computer Society.

[Engel et al., 2014] Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV) 2014*, pages 834–849. Springer.

[Engel et al., 2015] Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct SLAM with stereo cameras. In *International Conference on Intelligent Robots and Systems (IROS) 2015*, pages 1935–1942. IEEE.

[Fanani et al., 2017] Fanani, N., Stürck, A., Ochs, M., Bradler, H., and Mester, R. (2017).

Predictive monocular odometry (PMO): What is possible without RANSAC and multi-frame bundle adjustment? *Image and Vision Computing*, 68:3–13.

[Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *International Conference on Robotics and Automation (ICRA) 2014*, pages 15–22. IEEE.

[Frikha et al., 2016] Frikha, R., Ejbali, R., and Zaied, M. (2016). Handling occlusion in augmented reality surgical training based instrument tracking. In *International Conference of Computer Systems and Applications (AICCSA) 2016*, pages 1–5. IEEE.

[Fuentes-Pacheco et al., 2015] Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81.

[Fuhrmann et al., 1999] Fuhrmann, A., Hesina, G., Faure, F., and Gervautz, M. (1999). Occlusion in collaborative augmented environments. *Computers & Graphics*, 23(6):809–819.

[Galatis et al., 2016] Galatis, P., Gavalas, D., Kasapakis, V., Pantziou, G., and Zaroliagis, C. (2016). Mobile augmented reality guides in cultural heritage. In *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services*, pages 11–19. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) Brussels, Belgium.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237.

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2012*, pages 3354–3361]. IEEE.

[Grasset et al., 2012] Grasset, R., Langlotz, T., Kalkofen, D., Tatzgern, M., and Schmalstieg, D. (2012). Image-driven view management for augmented reality browsers. In *International Symposium on Mixed and Augmented Reality (ISMAR) 2012*, pages 177–186. IEEE.

[Hayashi et al., 2005] Hayashi, K., Kato, H., and Nishida, S. (2005). Occlusion detection of real objects using contour based stereo matching. In *Proceedings of the 2005 international conference on Augmented tele-existence*, pages 180–186. ACM.

[Hebborn et al., 2017] Hebborn, A. K., Höhner, N., and Müller, S. (2017). Occlusion matting: Realistic occlusion handling for augmented reality applications. In *International Symposium on Mixed and Augmented Reality (ISMAR) 2017*, pages 62–71. IEEE.

[Hirschmüller, 2005] Hirschmüller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2005*, volume 2, pages 807–814. IEEE.

[Hoelzer et al., 1978] Hoelzer, H., Johnson, G., and Cohen, A. (1978). Modified polar coordinates-the key to well behaved bearings only ranging. *IBM R&D Report 78-M19-0001A*.

[Holmes et al., 2008] Holmes, S., Klein, G., and Murray, D. W. (2008). A square root unscented kalman filter for visual monoSLAM. In *International Conference on Robotics and Automation (ICRA) 2008*, pages 3710–3716. IEEE.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011). Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM.

[Kanbara et al., 2000] Kanbara, M., Okuma, T., Takemura, H., and Yokoya, N. (2000). A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In *Proceedings IEEE Virtual Reality 2000*, pages 255–262. IEEE.

[Kasapakis and Gavalas, 2017] Kasapakis, V. and Gavalas, D. (2017). Occlusion handling in outdoors augmented reality games. *Multimedia Tools and Applications*, 76(7):9829–9854.

[Kasperi et al., 2017] Kasperi, J., Edwardsson, M. P., and Romero, M. (2017). Occlusion in outdoor augmented reality using geospatial building data. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, page 30. ACM.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR) 2007*, pages 225–234. IEEE.

[Kremer, 2017] Kremer, K. (2017). Tiefen-basierte Verdeckung in Augmented Reality mittels natural image matting. Master's thesis, Universität Koblenz-Landau.

[Kwon and Lee, 2010] Kwon, J. and Lee, K. M. (2010). Monocular slam with locally planar landmarks via geometric rao-blackwellized particle filtering on lie groups. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2010*, pages 1522–1529. IEEE.

[Lee et al., 2016a] Lee, S.-H., Eoh, G., and Lee, B. H. (2016a). Relational FastSLAM: an improved rao-blackwellized particle filtering framework using particle swarm characteristics. *Robotica*, 34(6):1282–1296.

[Lee et al., 2016b] Lee, S.-H., Oh, J. H., and Lee, B. H. (2016b). Improved unscented

FastSLAM using geometric information of particles. *International Journal of Mechanical Engineering and Robotics Research*, 5(1):43.

[Lepetit and Berger, 2000] Lepetit, V. and Berger, M.-O. (2000). A semi-automatic method for resolving occlusion in augmented reality. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2000*, volume 2, pages 225–230. IEEE.

[Makita et al., 2009] Makita, K., Kanbara, M., and Yokoya, N. (2009). View management of annotations for wearable augmented reality. In *International Conference on Multimedia and Expo (ICME) 2009*, pages 982–985. IEEE.

[Milford et al., 2004] Milford, M. J., Wyeth, G. F., and Prasser, D. (2004). RatSLAM: a hippocampal model for simultaneous localization and mapping. In *International Conference on Robotics and Automation (ICRA) 2004*, volume 1, pages 403–408. IEEE.

[Montemerlo and Thrun, 2007] Montemerlo, M. and Thrun, S. (2007). Fastslam 2.0. In *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pages 63–90. Springer.

[Montemerlo et al., 2002] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI Conference on Artificial Intelligence 2010*, pages 593–598.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Nistér, 2004] Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *Transactions on pattern analysis and machine intelligence*, 26(6):756–770.

[Oliensis, 2005] Oliensis, J. (2005). The least-squares error for structure from infinitesimal motion. *International Journal of Computer Vision*, 61(3):259–299.

[Penrose, 1955] Penrose, R. (1955). A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press.

[Reed, 2015] Reed, N. (2015). Depth precision visualized. `http://web.archive.org/web/20170921100911/https://developer.nvidia.com/content/depth-precision-visualized`. Accessed: 2017-09-21.

[Shah et al., 2012] Shah, M. M., Arshad, H., and Sulaiman, R. (2012). Occlusion in augmented reality. In *International Conference on Information Science and Digital Content Technology (ICIDT) 2012*, volume 2, pages 372–378. IEEE.

[Shiguang et al., 2017] Shiguang, W., Mingde, Y., Chengdong, W., and Jun, L. (2017). An improved FastSLAM2.0 algorithm based on ant colony optimization. In *29th Chinese*

Control And Decision Conference (CCDC) 2017, pages 7134–7137. IEEE.

[Song and Chandraker, 2014] Song, S. and Chandraker, M. (2014). Robust scale estimation in real-time monocular SFM for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2014*, pages 1566–1573. IEEE.

[Strasdat et al., 2010] Strasdat, H., Montiel, J., and Davison, A. J. (2010). Real-time monocular SLAM: Why filter? In *International Conference on Robotics and Automation (ICRA) 2010*, pages 2657–2664. IEEE.

[Tian et al., 2010] Tian, Y., Guan, T., and Wang, C. (2010). Real-time occlusion handling in augmented reality based on an object tracking approach. *Sensors*, 10(4):2885–2900.

[Triggs et al., 1999] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer.

[Upchurch and Desbrun, 2012] Upchurch, P. and Desbrun, M. (2012). Tightening the precision of perspective rendering. *Journal of Graphics Tools*, 16(1):40–56.

[Urmson et al., 2004] Urmson, C., Anhalt, J., Clark, M., Galatali, T., Gonzalez, J. P., Gowdy, J., Gutierrez, A., Harbaugh, S., Johnson-Roberson, M., Kato, H., et al. (2004). High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-04-37*.

[Vedaldi et al., 2007] Vedaldi, A., Guidi, G., and Soatto, S. (2007). Moving forward in structure from motion. In *Conference on Computer Vision and Pattern Recognition (CVPR) 2007*, pages 1–7. IEEE.

[Younes et al., 2016] Younes, G., Asmar, D., and Shammas, E. (2016). A survey on non-filter-based monocular visual slam systems. *arXiv preprint arXiv:1607.00470*.

[Yousif et al., 2015] Yousif, K., Bab-Hadiashar, A., and Hoseinnezhad, R. (2015). An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, 1(4):289–311.

Acknowledgements

I would like to thank my colleagues at Daimler Research & Development, in particular Team Augmented Reality. The time during my tenure as PhD student at Daimler has often been quite busy, but also fun, insightful and very rewarding. I am thankful of being given the opportunity to contribute pushing the boundaries in such a productive environment. I would like to thank my advisor Prof. Dieter Fritsch of the Institute for Photogrammetry, University of Stuttgart for his feedback and support, encouraging me to apply my background from pure mathematics to and and integrate it into engineering methods for Augmented Reality. I would like Prof. Dr. Luc Van Gool of ETH Zürich for agreeing to serve as a referee for this thesis. Finally, I would like to thank my family for their constant support.