UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

FABIO IRIGON PEREIRA

High Precision Monocular Visual Odometry

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Microeletronics

Advisor: Prof.Dr. Alatamiro A. Susin

Porto Alegre
September 2018

CIP — CATALOGING-IN-PUBLICATION

Pereira, Fabio Irigon

High Precision Monocular Visual Odometry / Fabio Irigon Pereira. – Porto Alegre: PGMICRO da UFRGS, 2018.

103 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2018. Advisor: Alatamiro A. Susin.

1. Computer Vision. 2. Visual Odometry. 3. Visual SLAM. 4. 3D. I. Susin, Alatamiro A.. II. Título.

AGRADECIMENTOS

Agradeço antes de tudo à minha família. Aos meus pais Paulo e Isa a quem sou eternamente grato. À minha mulher Thelma com quem divido o meu lar, por sua paciência, carinho e apoio e ao meu filho Guilherme alegria dos meus dias. Aos meus irmãos Gabriela, Paulo José e Rafael com quem sempre encontro apoio.

Agradeço ao meu orientador e amigo Altamiro Amadeu Susin pelo direcionamento, disposição e pelos conselhos. Me ensinou muito com suas palavras, mas principalmente com o exemplo.

Aos professores da banca da proposta, Sérgio Bampi, Maurício de Paula, Claudio Jung e Fernando Osório pela revisão e direcionamento de grande valor para o trabalho.

Ao Doutor Marcelo Negreiros, que durante a tese foi meu co-orientador de fato, incentivando ajudando na escrita e mostrando o caminho com a delicadeza que lhe é peculiar. Ao professor Ronaldo Husemann pelo apoio e orientação.

Ao colega Joel Luft com quem dividi a bancada nos últimos anos, pelas histórias e discussões técnico-filosóficas. Aos colegas de pós Thiago Waszak, Igor Hoelscher e Gustavo Ilha pela amizade e pelo clima sadio no laboratório.

Aos bolsistas Arthur Sofiatti, Eduardo da Costa e Mirayr pelas discussões, projetos e companhia no LaPSI. Ao Paulo, Forlin, Ana e Bruna pela zoeira.

Ao programa PGMICRO e á CAPES, por terem me aceitado e pelo financiamento da pesquisa.

Fabio Irigon Pereira

ABSTRACT

Recovering three-dimensional information from bi-dimensional images is an important problem in computer vision that finds several applications in our society. Robotics, entertainment industry, medical diagnose and prosthesis, and even interplanetary exploration benefit from vision based 3D estimation. The problem can be divided in two interdependent operations: estimating the camera position and orientation when each image was produced, and estimating the 3D scene structure. This work focuses on computer vision techniques, used to estimate the trajectory of a vehicle equipped camera, a problem known as visual odometry. In order to provide an objective measure of estimation efficiency and to compare the achieved results to the state-of-the-art works in visual odometry a high precision popular dataset was selected and used. In the course of this work new techniques for image feature tracking, camera pose estimation, point 3D position calculation and scale recovery are proposed. The achieved results outperform the best ranked results in the popular chosen dataset.

Keywords: Computer Vision. Visual Odometry. Visual SLAM. 3D.

Tese de Doutorardo: Odometria Visual Monocular de Alta Precisão

RESUMO

Extrair informação de profundidade a partir de imagens bidimensionais é um importante problema na área de visão computacional. Diversas aplicações se beneficiam desta classe de algoritmos tais como: robótica, a indústria de entretenimento, aplicações médicas para diagnóstico e confecção de próteses e até mesmo exploração interplanetária. Esta aplicação pode ser dividida em duas etapas interdependentes: a estimação da posição e orientação da câmera no momento em que a imagem foi gerada, e a estimativa da estrutura tridimensional da cena. Este trabalho foca em técnicas de visão computacional usadas para estimar a trajetória de um veículo equipado com uma câmera, problema conhecido como odometria visual. Para obter medidas objetivas de eficiência e precisão, e poder comparar os resultados obtidos com o estado da arte, uma base de dados de alta precisão, bastante utilizada pela comunidade científica foi utilizada. No curso deste trabalho novas técnicas para rastreamento de detalhes, estimativa de posição de câmera, cálculo de posição 3D de pontos e recuperação de escala são propostos. Os resultados alcançados superam os mais bem ranqueados trabalhos na base de dados escolhida até o momento da publicação desta tese.

Palavras-chave: Visão Computacional, Odometria Visual, 3D.

LIST OF ABBREVIATIONS AND ACRONYMS

AR	Augmented Reality
BA	Bundle Adjustment
BRIEF	Binary Robust Independent Elementary Features
CV	Computer Vision
DoG	Difference of Gaussian
EL	Epipolar Line
FAST	Feature from Accelerated Segment Test
GN	Gauss-Newton
HoG	Histogram of Gradients
IMU	Inertial Measurement Unit
KL	Kallman Filter
LK	Lucas-Kanade
LM	Levenberg-Marquardt
MVO	Monocular Visual Odometry
OpenCV	Open source Computer Vision library
ORB	Oriented FAST and Rotated BRIEF
PnP	Perspective-n-Point
RANSAC	RANdom SAmple Consensus
RI	Resection-Intersection
SAD	Sum of Absolute Differences
SIFT	Scale Invariant Feature Transform
SfM	Structure from Motion
SLAM	Simultaneous Location And Mapping
SVO	Stero Visual Odometry

UAV	Unmanned Aerial Vehicles
VO	Visual Odometry
VSLAM	Visual Simultaneous Location And Mapping

LIST OF SYMBOLS

$\mathbf{u}_k$	Feature coordinate pair on frame k
w	Arbitrary origin labeled "world frame of reference"
u_k	Feature horizontal coordinate on frame k
v_k	Feature horizontal coordinate on frame k
$\hat{\mathbf{u}}_k$	Projected feature on frame k
$\mathbf{u\prime}$	Feature coordinates after scaling by focal length and shifting by camera center
$\mathbf{U}_k$	matrix of feature coordinates on frame k
$\mathbf{p}$	Point coordinates (x, y, z) on 3D space
$\mathbf{P}$	Matrix of points coordinates on 3D space
$\mathbf{K}$	Matrix of camera intrinsic parameters
$\mathbf{w}$	Vector of camera pose
$\Delta\mathbf{w}$	Increment for camera pose approximation
$\mathbf{T}$	Camera pose as 3×4 rigid-body transformation matrix
θ_x	Camera rotation with respect to axis x
t_x	Camera translation along axis x
$\mathbf{r}$	Vector projection to traced residuals
$\mathbf{J}$	Jacobian matrix of partial derivatives
$\mathbf{H}$	Harris Matrix
$\mathcal{W}$	Image region belonging to a feature
d	Feature depth (distance from camera center)

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

"The only thing worse than being blind is having sight but no vision."

— *Helen Keller*

Vision is one of the base foundations of human civilization. The ability to see, recognize and interpret a scene, perceiving three-dimensional shape is a trivial task even to young children. Providing machines with such abilities, however, although seemed straightforward decades ago, remain as an interesting and unsolved problem.

Computer Vision (CV) is a multidisciplinary field that studies methods to enable automatic understanding of images (RADKE, 2012). The problems that CV scientists try to solve range from detecting people in images, recognizing faces and facial expressions, objects, scene 3D structure, characters and handwriting among many others. To accomplish those objectives, concepts spanning from image processing to algebra, geometry, physics and statistics must be synergistically combined.

During last decades, the exponential expansion in the number of camera equipped mobile phones, decline of consumer camera price, combined to increased processing power of embedded devices, pushed for an increased interest of the scientific community towards the development of vision capable embedded solutions.

This work's main focus is in three-dimensional scene comprehension from two or more bi-dimensional images. Two closely related and interdependent problems must be addressed: the assessment of the scene structure and the definition of camera position and orientation when each image was produced. In the computer vision community the topic is frequently referred to as Visual Simultaneous Location and And Mapping (VSLAM). Monocular systems use a single camera to produce the images while stereo-vision systems use two cameras usually mounted on a rigid rig, where the relative camera positions is usually approximately known. While stereo vision frequently provides better estimation accuracy, as it is a more constrained problem, monocular solutions are appealing due to the simpler hardware and the number of single camera equipped devices already in use in our society.

The camera model describes how a 3D scene is projected on an image plane, an operation where depth information is lost. If more than one image of the same scene are provided, the methods studied in this work, indicate solutions to recover the scene structure up to a well defined ambiguity. If no other information besides the images is provided, the scene scale and absolute position can not be determined, as the image of a planet can be similar to the sketch of a single atom. In those cases, if the projection model is accurate, the scene can be estimated up to what is called a similarity transformation (correct shape up to an unknown absolute scale, position and orientation), otherwise the scene is only estimated up to a projective ambiguity. In some applications, scale can be recovered using information from other sensors (GPS, inertial sensors, ultrasound etc), known image physical characteristics or provided information about camera relative pose.

Despite the fact that the algorithms developed during this work focus on a particular purpose, a broad range of applications, and closely related problems, benefit from advances in scene structure and camera pose estimation. Next section enumerates some of those applications.

1.1 Motivation - From robots to the movies

The problem of three dimensional information extraction from bi-dimensional views of the scene has extensive practical applications in different fields. Although this work targets a singular application (the chosen application is the trajectory estimation of a camera equipped vehicle), most of the proposed contributions are equally qualified for any of the uses listed in this section.

1.1.1 Visual Odometry

Vision Odometry (VO), also called motion tracking or camera solving, tries to estimate the camera position and orientation, i.e. camera pose, when each image produced by the camera was generated. Monocular Visual Odometry (MVO) and Stereo Visual Odometry (SVO) terms are used to differentiate cases where a single camera and a pair of cameras are used respectively.

In order to estimate camera motion, scene structure must be calculated, and errors in the learned scene will translate into errors in the calculated camera pose. Scene structure for VO is an intermediary step not evaluated in the final ego motion - camera relative movement with respect to a static scene (IRANI; ROUSSO; PELEG, 1994).

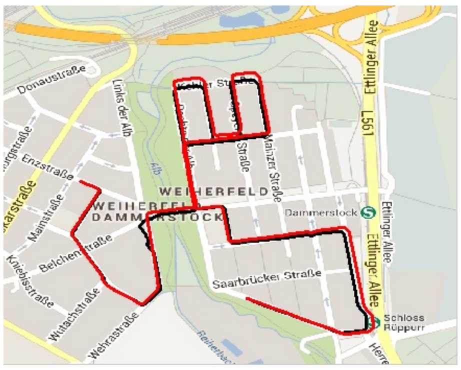
Visual odometry is frequently related to robotics, where a camera equipped robot must find its own location based on the images and possibly other sensors. Autonomous vehicles (FERNANDES et al., 2014), Unmanned Aerial Vehicles (UAVs) (FORSTER; PIZZOLI; SCARAMUZZA, 2014) and even the Mars exploration rover (CHENG; MAIMONE; MATTHIES, 2005) are examples of use cases for VO systems.

In order to gauge the precision of a visual estimation system, a ground truth must be acquired and used to compare to the estimated output. Camera pose can be accurately measured and recorded with relative facility, with high precision sensors (GPS + inertial sensors). Building a high fidelity 3D map of the viewed scene is more complex. Due to the practical efficacy of recording and comparing real camera pose to vision based estimations, VO systems are often used to test vision based systems accuracy.

Figure 1.1 shows a typical trajectory, where the red and black lines show the estimated trajectory and GPS recorded ground-truth and the camera equipped vehicle used in a popular VO dataset (GEIGER; LENZ; URTASUN, 2012b).

Figure 1.1: Visiual Odometry trajectory and vehicle

(a) Visual Odometry Trajectory example



(b) Camera and GPS equipped vehicle

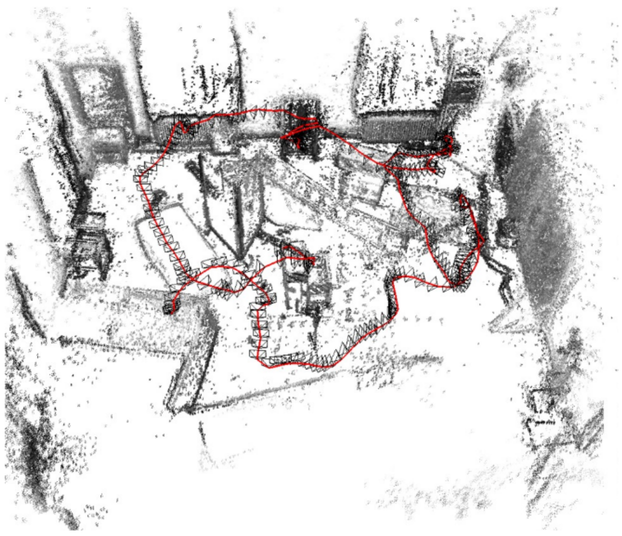
source: (GEIGER; LENZ; URTASUN, 2012b)

1.1.2 3D Reconstruction

Three-dimensional reconstruction refers to the broad problem of discovering the structure of an object, not necessarily using images. The use of images from one or more cameras however, is an important branch in 3D reconstruction work-flow.

Structure from motion (SfM) aims to reconstruct 3D morphology of a scene using the images of a moving camera. Analogous to VO, the camera relative position and orientation must be provided or estimated, as that information is needed for the reconstruction flow, although in this case, camera pose is not evaluated in the final result of the reconstructed object or scene. An example of visual 3D reconstruction is shown in figure 1.2.

Figure 1.2: Visual 3D reconstruction



source: (ENGEL; KOLTUN; CREMERS, 2016)

Discovering an object 3D shape from images, is an important field in geosciences (ALIAKBARPOUR; PALANIAPPAN; SEETHARAMAN, 2017), architecture and civil engineering (MERRELL et al., 2011) and medical applications for both prosthesis and diagnose (BOYKOV; FUNKA-LEA, 2006) among others.

1.1.3 Visual SLAM

Simultaneous Location And Mapping (SLAM) is the classical robotics problem, where a robot must build a map of the environment and simultaneously calculate its own position in the map. Originally several sensors have been used for SLAM, such as radars, ultrasound, laser, Inertial Measurement Units (IMU), GPS among others. When a camera is used, either alone or combined with other sensors, the problem is termed Visual SLAM (VSLAM) and is an important bound between computer vision and robotics.

Visual SLAM could be regarded as a combination of VO and SfM, noting that SfM algorithms usually produce a dense map of the scene or object, while Visual odometry can use a dense map, or a sparse set of a few localized 3D landmarks depending on the application. Autonomous robots are the main example of application, and have been used in autonomous vehicles (FERNANDES et al., 2014), domestic robots, and even inside the human body (PAHLAVAN; BAO; LIANG, 2013).

1.1.4 Augmented Reality

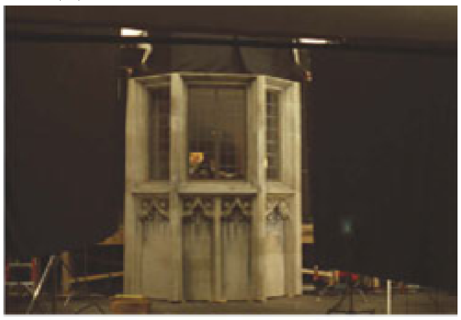
Augmented Reality (AR) is the technology of combining real world images or video with computer-generated information and or imagery. As in other examples, the camera pose and scene structure must be determined. In addition care must be taken to adjust illumination, shadowing and occlusion (when a scene element covers part of the artificially generated image).

Possible uses of AR examples include architecture (to furnish and decorate an empty house), broadcasting (to display statistics information in sports games as if they belonged to the scene for instance) and the game industry (the pokemon mobile phone game is an example of AR in the field).

One of the most important uses of AR is probably in the movies industry, in the AR variant called matchmoving used to provide video sequences with computer generated effects. According to (RADKE, 2012) 43 of the top 50 greatest films of all time are visual effects driven. Figure 1.3 shows an AR example. In the left figure a window on a filming set is shown, and in the right image the final movie scene is pictured, where the window is artificially mounted on the building brick wall.

Figure 1.3: Augmented Reality example (movie "A Beautiful Mind")

(a) Window on a filming set



(b) Final movie scene

source: (RADKE, 2012)

1.2 Objectives of this study

This work focus is on 3D information extraction from 2D images. The monocular problem is especially attractive for several reasons. The relative facility to connect a single camera to embedded hardware when compared to a camera array. The ubiquity of camera equipped processing units such as mobile phones. The challenge of a less constrained problem, more affected by outliers and stability issues (PERSSON et al., 2015) which impose an additional level of difficulty when compared to multiple camera configurations. And finally the fact that the generic methods of monocular vision, can be applied to the particular case of stereo vision with minimal or no modification.

Visual Odometry for camera equipped vehicles was chosen as target application: a camera equipped vehicle generates a sequence of images, and the position and orientation of the camera must be determined for each available image. Good datasets provide a practical way of objective accuracy measurement and comparison to state-of-the-art solutions.

This thesis objective is to elaborate a monocular visual odometry pipeline, able to achieve high accuracy at reduced computational cost, allowing the implementation of a navigation system for real time operation with moderate hardware resources.

1.3 Contributions

During the investigation of visual estimation principles, several algorithms were developed. While solutions to monocular vision odometry can be found in literature, this thesis innovates according to the following contributions:

- An analysis of instability in forward motion estimation able to determine source of local-minima problems is conceived (PEREIRA et al., 2017a).
- A new method of ground feature tracking, using perspective image warping and approximate camera movement is developed (PEREIRA et al., 2017b).
- The use of movement translation for camera nodding approximation and further scale adjustment is designed (PEREIRA et al., 2017b).
- A Multi-attribute ground feature selection procedure is formulated (PEREIRA

et al., 2017b).

- A new two-view bundle adjustment algorithm, based on resection-intersection is created (PEREIRA et al., 2018).

1.4 Notation

As the notation of different books and technical papers in computer vision tend to vary as often happens in multidisciplinary fields, this section briefly introduces the notation used throughout this work.

Scalars are denoted by mixed case italics (s, C), vectors are lower case bold $(\mathbf{v})$ and unless otherwise stated are column wise. Matrices are uppercase bold $(\mathbf{M})$.

A feature in the k_{th} frame is denoted by $\mathbf{u}_k = (u_k, v_k)^T$, and the set of all features in frame k is grouped in the $n \times 2$ matrix $\mathbf{U}_k$ where each row stores a pair of coordinates. The matrix $\mathbf{P}$ represents a set of points in 3D space, where each row represents a point $\mathbf{p} = (x, y, z)$ that can be mapped to an image coordinate by the camera projection model to position $\hat{\mathbf{u}} = (\hat{u}, \hat{v})^T$. The $\prime$ (prime) symbol is used to denote image coordinate after shifting by the camera center and scaling by the focal length, in homogeneous coordinates: $\mathbf{u}\prime = \mathbf{K}^{-1}\mathbf{u}$ where $\mathbf{K}$ represents the camera intrinsic parameters.

The camera pose is represented by either the vector $\mathbf{w} = [\theta_x, \theta_y, \theta_z, tx, ty, tz]$ containing the three rotations and translations relative to the previous camera pose or by a 3×4 matrix characterizing the rigid-body transformation $\mathbf{T} = \mathbf{R}|\mathbf{t}$ as explained in (RADKE, 2012).

1.5 Outline

This work is structured as follows: chapter 2 introduces the fundamental background and classical algorithms related to 3D estimation from bi-dimensional images. Chapter 3 brings a survey on recent developments in VO. Peculiarities of projection error for translation along optical axis is analyzed in chapter 4. Detection and tracking of scene details on the images are studied in chapter 5. Scale estimation for camera equipped wheeled robots using detected details on the ground is discussed in chapter 6. Calculation of point distance to the camera is covered in chapter 7.

The position of camera and points that minimize reprojection error is computed in chapter 8. Results of a full odometry system using the concepts considered and implemented algorithms are presented in chapter 9. Finally a conclusion is performed in chapter 10.

2 VISUAL NAVIGATION BACKGROUND

"Poetry is as precise as geometry."

Gustave Flaubert

This chapter describes the fundamental concepts and algorithms related to visual navigation. Good references for the reader who wishes a more thorough coverage on the topic are the books "Computer Vision for Visual Effects" (RADKE, 2012) from Richard Radke and "Computer Vision Algorithms and Applications" (SZELISKI, 2010) from Richard Szeliski. For an in depth review of projective geometry the reader may refer to the milestone work "Multiple View Geometry in Computer Vision" (HARTLEY; ZISSERMAN, 2004) from Hartley and Zisserman. Camera pose estimation is extensively covered in the in depth review of bundle adjustment made in (TRIGGS et al., 2000). Practical approaches using OpenCV (BRADSKI, 2000) can be found in (DAWSON-HOWE, 2014) and industry applications on (WOHLER, 2009).

The main classes of algorithms involved in visual navigation can be classified in image processing algorithms and geometric projection methods. In real applications, positions orientations and distances can not be exactly measured, but estimated with a varying level of certainty. Proposed solutions must address both noisy (with small deviations from the real value) and bad (outliers with large errors) measurements.

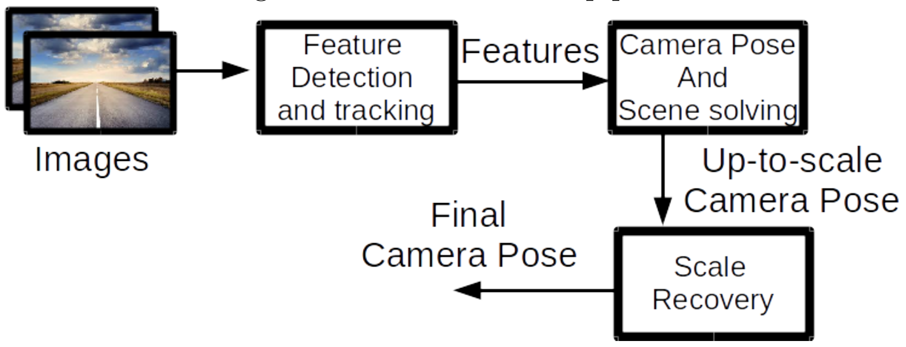
The chapter starts discussing the typical sparse odometry pipeline. Then the most relevant image processing methods used to select distinguishable image regions, and recognize those regions in subsequent images are addressed. The chapter continues with projective geometry, describing the basic model that explains how a 3D scene structure is projected on a 2D image plane. The projection constrains of epipolar geometry and the main ideas on camera pose and feature depth estimation are illustrated. The chapter concludes elucidating ground plane based scale estimation mechanism.

2.1 Sparse Monocular Visual Odometry Overview

Monocular navigation can be split in two interrelated proceedings: the estimation of camera pose at the moment the images were registered and the 3D scene structure inference. Although Monocular Visual Odometry (MVO) results are only related to the position and orientation of the camera, the 3D structure of the viewed scene must be estimated and errors in scene estimation do reflect in localization results.

Dense or semi-dense solutions like (ENGEL; STURM; CREMERS, 2013) try to adjust camera pose and scene structure matching all pixels with significant image gradient. Sparse systems, on the other hand (SONG; CHANDRAKER; GUEST, 2016; Mirabdollah M., Mertshching B. (GET Lab, 2015), select recognizable image regions, called features, and locate those regions in subsequent images. For sparse systems the 3D position is only calculated for those points in the scene (features).

Figure 2.1: Generic MVO pipeline



source: author

The typical sparse MVO pipeline is shown in figure 2.1. The system input is a sequence of images. An image processing unit detects interest points in one image and tracks those points in the remaining images. The point's correspondences are used to estimate scene depth and camera pose, and finally the original scale is approximated. Next sections detail those operations.

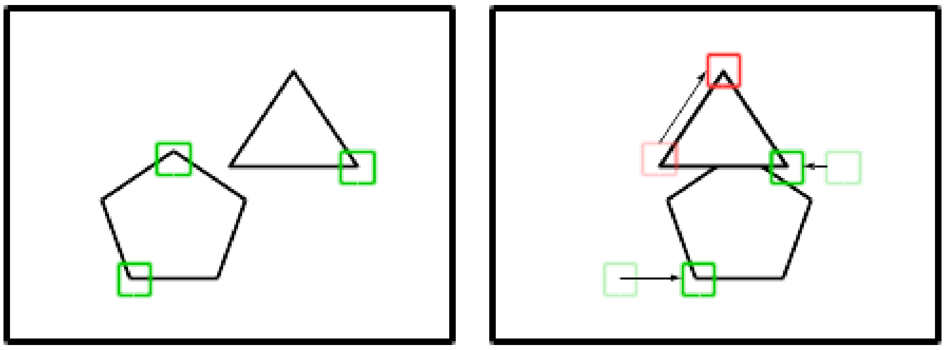
2.2 Feature Detection and Tracking

Feature detection in computer vision refers to methods for computing abstractions of image information, and making decisions whether an image point matches a given interest criterion. Throughout this work, feature detection and tracking is referred as the process of selecting traceable image regions and locating those regions in subsequent frames. The centers of those regions are called features and 3D positions are only estimated for those points. Two classes of image processing algorithms are then relevant and will be presented in this section: the methods to select image regions labeled feature detectors and methods to recognize those regions in other frames identified as trackers and matchers.

A feature correspondence refers to pairs of pixel coordinates belonging to the same scene point. as shown in figure 2.2. The green squares in the left part of figure 2.2 mark three detected features. The squares in the right part of the figure show the matched features. The red square, marks a matching failure due to the occlusion of the top of the pentagon. The arrows represent motion vectors, connecting the feature position in the original image to the tracked position in the second frame. The section starts with detection algorithms followed by tracking and matching techniques.

Figure 2.2: Feature Detection and Tracking

(a) Feature detection example (b) Tracking and motion vectors



source: author

2.2.1 Harris Corner Detector

One of the earliest corner detection algorithms was proposed by Moravec (MORAVEC, 1980) based on the similarity of an image region to overlapping regions in horizontal, vertical and diagonal neighborhood. Harris and Stephens (HARRIS; STEPHENS, 1988) built upon Moravec's ideas using image gradients in stead of shifted image blocks.

The authors defined the positive definite Harris Matrix $\mathbf{H}_a$ as shown in equation 2.1. In the equation $I(u, v)$ is the pixel intensity at coordinates (u, v), $\frac{\partial I}{\partial u}$ is the partial derivative of the intensity (image gradient) along direction u and the sum spam over the currently tested image region $\mathcal{W}$:

$$\mathbf{H}_a = \begin{bmatrix} \sum_{(u,v)\in\mathcal{W}} (\frac{\partial I}{\partial u}(u,v))^2 & \sum_{(u,v)\in\mathcal{W}} \frac{\partial I}{\partial u}(u,v)\frac{\partial I}{\partial v}(u,v) \\ \sum_{(u,v)\in\mathcal{W}} \frac{\partial I}{\partial u}(u,v)\frac{\partial I}{\partial v}(u,v) & \sum_{(u,v)\in\mathcal{W}} (\frac{\partial I}{\partial v}(u,v))^2 \end{bmatrix} \tag{2.1}$$

Harris and Stephens proposed an approximate quality measure was based on the trace of $\mathbf{H}_a$ as shown in equation 2.2 where κ is a tunable sensitivity parameter:

$$Q_l = det(\mathbf{H}_a) - \kappa\, trace\,(\mathbf{H}_a)^2 \tag{2.2}$$

Analyzing the eigenvalues (λ_1, λ_2) of the matrix $\mathbf{H}_a$ a better approximation of corner quality can be estimated. If the pixel intensities inside the box are similar (homogeneous image area), both $\frac{\partial I}{\partial u}$ and $\frac{\partial I}{\partial v}$ will be small and both eigenvalues will present small values. On the occasion that the image region presents a linear edge, one eigenvalue will be large with associated eigenvector perpendicular to the edge, while the other will be near zero. When a corner is present in the image crop, both λ_1 and λ_2 will present non-zero values.

By analyzing the smaller eigenvalue of the Harris Matrix, the quality of the corner can be defined. In 1994, J. Shi and C. Tomasi proposed a small modification in their work (SHI; TOMASI, 1994) according to equation 2.3.

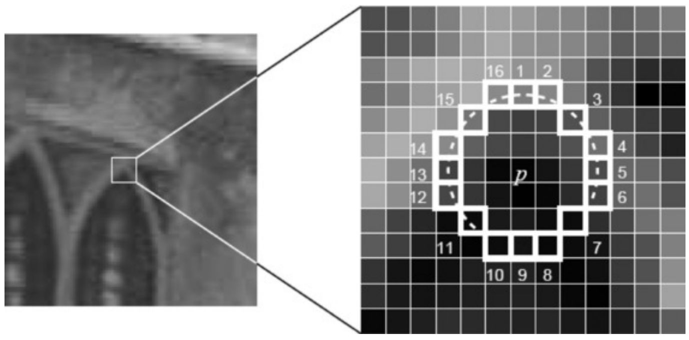
$$Q_l = min((\lambda_1, \lambda_2) \tag{2.3}$$

Corners are tested for all pixel positions in an image and regions with the quality measure above a threshold defined as a fraction of the maximum found quality are returned. To avoid multiple results in a neighborhood, non-maximum suppression is usually applied, where only the highest value in a user-defined radius is retained.

2.2.2 FAST Corner Detector

To avoid calculating Harris matrix and eigenvalues, a low-complexity method was proposed by Rosten and Drummond (ROSTEN; DRUMMOND, 2005) named Features from Accelerated Segment Test (FAST). In the method a pixel is compared to neighbor pixels at a fixed distance r as shown in figure 2.3. If a contiguous arc of n pixels of neighbors are significantly darker or lighter then the central reference, the patch is selected as a corner.

Figure 2.3: FAST corner detector



source: (ROSTEN; DRUMMOND, 2005)

In the original algorithm sixteen pixels around the central pixel where tested and $n = 9$. The algorithm was extended with machine learning approach (ROSTEN; DRUMMOND, 2006) where an extensive set of features were used to build a decision tree based on the intensity difference of the central pixel to the sixteen neighbors. On average fewer than three comparisons are made for each patch candidate to test for corner detection.

2.2.3 Hybrid and Multi-Scale Corner Detector

As FAST verifies a reduced number of pixels around the central feature candidate, regions with high spatial frequency, that should be avoided, may be selected by the feature detector. A common approach in recent solutions is to initially select corners with the FAST methodology, and reject candidates based on the Harris corner algorithm. In this way the slower Harris corner detector is only applied to locations previously selected by the FAST detector, reducing computational complexity and rejecting high spatial frequency candidates in a hybrid solution.

A sensitive point in corner detection is the scale in which corners are searched. If small windows are chosen, fine grained corners are returned, whether larger windows return features in a larger image region. Lindeberg (LINDEBERG, 1998) proposed search for corners in different scales. By searching features in scale-space and scale-normalizing Harris matrix eigenvalues, to prevent larger windows from out weighting smaller ones, it is possible to find multi-scale Harris corners and the natural corner scale.

2.2.4 Lukas-Kanade Tracker

Once distinguishable image regions have been selected, those regions must be located in subsequent images. When images are taken sequentially, features in the new image are expected to be close to the detected features in the original image. A possible method is to use a search procedure to find the image region most similar to the target feature.

Considering a series of images registered by a camera and indexed by time $I(u,v,t)$ it is possible to approximate a block $\mathcal{W}$ of pixels at time $t+1$ by the block at time t shifted by a vector (x,y) as in equation 2.4.

$$I(x+u,y+v,t+1) \approx I(u,v,t) \quad \forall(u,v) \in \mathcal{W} \tag{2.4}$$

Using first order Taylor series approximation it is possible to go from equation 2.4 to equation 2.5 (LUCAS; KANADE, 1981).

$$\mathbf{H}_a \begin{bmatrix} x \\ y \end{bmatrix} = - \begin{bmatrix} \sum_{(u,v)}(\frac{\partial I}{\partial u}(u,v,t)\frac{\partial I}{\partial t}(u,v,t)) \\ \sum_{(u,v)}(\frac{\partial I}{\partial v}(u,v,t)\frac{\partial I}{\partial t}(u,v,t)) \end{bmatrix} \tag{2.5}$$

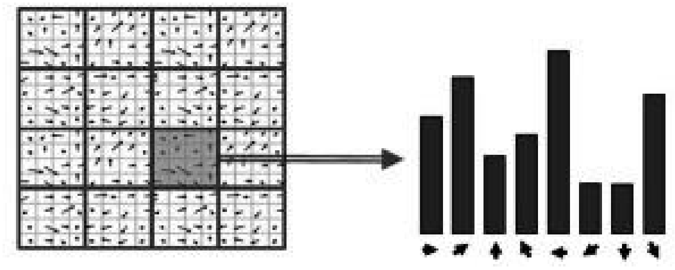
Where $\mathbf{H}_a$ is the Harris matrix, defined in equation 2.1. Multiplying both sides of equation 2.5 by the inverse of the Harris matrix the displacement vector can be iteratively approximated. As the algorithm only works for small displacement where the used Taylor series expansions are meaningful, a pyramidal implementation of the algorithm is usually used as proposed by (BOUGUET, 2000). In that algorithm image pyramids with different resolutions are generated and features are located in the lowest resolution image. Found positions are subsequently refined in higher resolution images until the original full resolution image is processed. An in depth overview of the algorithm can be found in (BAKER; MATTHEWS, 2004).

2.2.5 SIFT Feature Matching

Feature matching is an alternative to tracking. Matching involves computing image descriptors of detected features on both the reference and target images. Then a descriptor comparison strategy indicates if those descriptors refer to the same image feature. The use of descriptors enables algorithms to provide some level of robustness to scale, orientation, illumination and position differences.

The Scale Invariant Feature Transform (SIFT) (LOWE, 1999) is an extremely popular and robust image descriptor (FAN; WANG; WU, 2016) based on histogram of gradients (HoG). The first step for descriptor extraction in SIFT is the feature scale detection. Difference of Gaussian (DoG) for differently scaled versions of the smoothed original image is used and scale with maximum output is chosen. A block of 16x16 pixels around the feature center oriented by the dominating gradient is defined and an 8 bin HoG for each 4x4 sub-blocks is calculated as shown in figure 2.4.

Figure 2.4: SIFT



source: https://gilscvblog.com

The 8 bins of each of the 16 sub-blocks form a vector of 128 elements that are normalized and used to match to candidate descriptors through Euclidean distance comparison.

Scale detection provide robustness to scale differences, the dominant orientation is used to provide invariance to image rotation differences, gradients provide robustness to illumination differences and the feature position is irrelevant for the matching algorithm.

2.2.6 ORB Feature Matching

Although SIFT presents good results, the histogram of gradients and Euclidean distance comparison are computationally demanding for large sets of features. A family of so called binary descriptors have been proposed to provide similar precision of SIFT with a fraction of the computational complexity. Among those methods Binary Robust Independent Elementary Features (BRIEF) (CALONDER et al., 2010) that was later improved in Oriented FAST and Rotated BRIEF (ORB) (RUBLEE et al., 2011) is one of the most relevant.

ORB uses FAST corners, detect and compensate the feature orientation, and use a binary descriptor for feature matching. Each descriptor bit is the result of a binary comparison of pixel intensity of two different positions inside the feature patch as defined by equations 2.6 and 2.7.

$$\tau(I, \mathbf{u}, \mathbf{v}) := \begin{cases} 1 : I(\mathbf{u}) < I(\mathbf{v}) \\ 0 : I(\mathbf{u}) \geq I(\mathbf{v}) \end{cases} \tag{2.6}$$

$$f_n(p) = \sum_{0 \leq i < n} 2^i \tau(p, \mathbf{u}_i, \mathbf{v}_i) \tag{2.7}$$

In the equations $(\mathbf{u}, \mathbf{v})$ are two pixel positions inside the patch, $\mathcal{W}$ is the pixel intensity inside the patch at position $\mathbf{u}$ and $f_n(\mathcal{P})$ is the code-word for patch. The positions of pixel comparison inside the patch $(\mathbf{u}_i, \mathbf{v}_i)$ have been defined using a training set of 300.000 features. The original work used $n = 256$ bit codes for descriptor size.

2.3 Camera Pose and Scene Structure Estimation

Once feature correspondences have been established, point 3D positions and the camera poses can be approximated in order to match calculated projections to found features. This section starts with the camera projection model, followed by epipolar constraints, and finally introduces camera pose estimation and scene reconstruction algorithms.

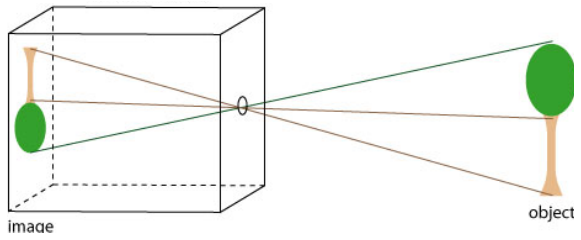
2.3.1 Pinhole Projection Model

This section introduces the model that maps a three-dimensional scene into a bi-dimensional image. The simple pinhole camera model is explained and relationship to real cameras is briefly highlighted.

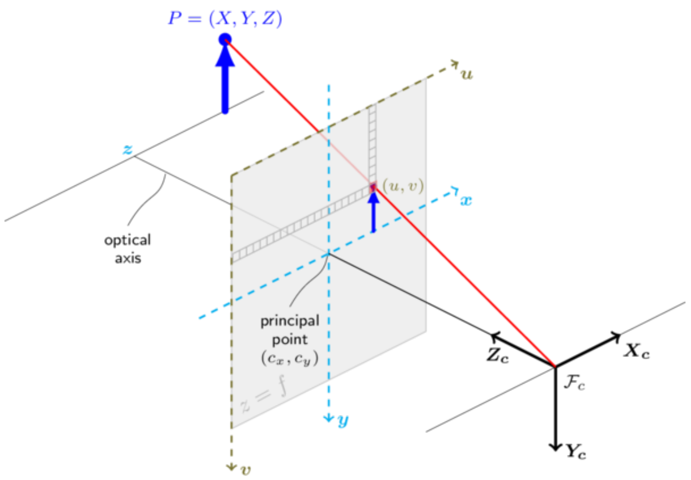
In the pinhole camera model, light from the scene passes through a single point (the pinhole or principal point) and project an inverted image on a plane as shown in figure 2.5 (a). For convenience, a virtual image plane, in front of the principal point is used and a non inverted projection is formed as shown in figure 2.5 (b).

Figure 2.5: Pinhole Camera Model

(a) Pinhole camera



(b) Pinhole Camera Model



source: (BRADSKI, 2000)

Figure 2.5 (b) shows the used orientation convention, where the optical axis is aligned with the z axis, and axis x and y are aligned to the image plane. A point with coordinates $(x, y, z)^T$ is projected to image coordinates $(u, v)^T$ by the projection equation 2.8.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.8}$$

On equation 2.8 variables $(u, v)^T$ are image coordinates in pixels, (f_x, f_y) is the focal length in pixels, (c_x, c_y) are pixel coordinates of the camera center, and the vector $(x, y, z)^T$ is the point coordinate in units of distance (meters for example). Equation 2.8 can be written in the simple form of equations 2.9 and 2.10.

$$u = \frac{x}{z} f_x + c_x \tag{2.9}$$

$$v = \frac{y}{z} f_y + c_y \tag{2.10}$$

The set of parameters (f_x, f_y, c_x, c_y) define the main characteristics of the pinhole camera model and are referred to as intrinsic camera parameters. When points position use a different coordinate reference than the camera, the point's coordinates must be rotated and translated to the camera frame of reference, before being projected onto the image plane as shown in equation 2.11.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.11}$$

The coefficients r_{ij} are coefficients of a $3x3$ rotation matrix and t_i are the x, y and z translation parameters. The matrix with the intrinsic camera parameters is denoted by $\mathbf{K}$, replacing the rotation matrix by $\mathbf{R}$ and translation vector by $\mathbf{t}$, and using homogeneous coordinates the proportionality of equation 2.11 becomes the identity on equation 2.12. The rotation and translation are called camera extrinsic parameters or camera pose.

$$\tilde{\mathbf{u}} = \mathbf{K} \begin{bmatrix} \mathbf{R} & | & \mathbf{t} \end{bmatrix} \tilde{\mathbf{x}} \tag{2.12}$$

Homogeneous coordinates are useful to represent points at infinity and non-linear relations in a simpler form. The operations on point $\tilde{\mathbf{x}}$ on equation 2.12 can be combined in the matrix $\mathbf{T}_{(k,w)}$ as shown in equation 2.13. In equation 2.13 subscripts (k, w) indicate that a point in the world frame of reference w (an arbitrary fixed reference) will be projected by $\mathbf{T}_{(k,w)}$ in the camera image plane at instant k.

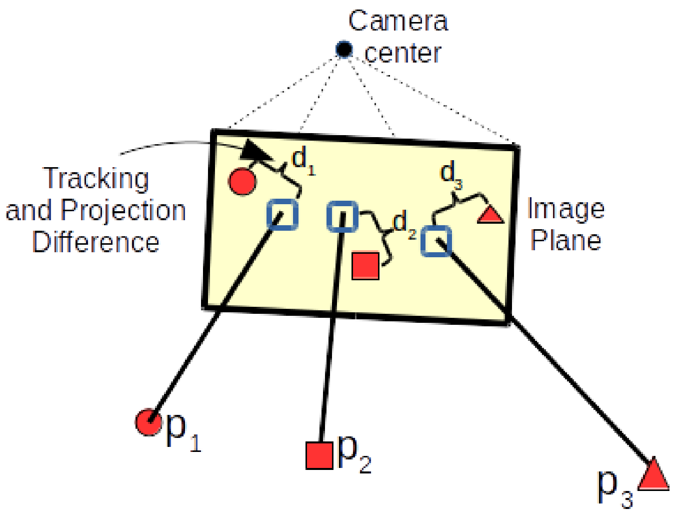
$${}_k\tilde{\mathbf{u}} = \mathbf{T}_{k,w} \cdot {}_w\tilde{\mathbf{x}} \tag{2.13}$$

Projection error minimization for pose estimation aims to find a transformation $\mathbf{T}_{k,w}$ that minimizes the difference between viewed features, and the calculated projection on the image plane as shown in equation 2.14.

$$\mathbf{T}_{k,w} = \arg\min_{\mathbf{T}_{(k,w)}} \sum_{i \in n} ||\mathbf{u}_i - \mathbf{T}_{(k,w)} \cdot {}_w\tilde{\mathbf{x}}_i||^2 \tag{2.14}$$

Figure 2.6 exemplifies the projection errors. In the figure three 3D points p1, p2 and p3 are projected in the blue squares in the image plane. The tracked feature positions are marked in the plane as the red circle, square and triangle. The distances d_1, d_2 and d_3 are the difference from projected position to tracked feature position called projection error. Projection minimization algorithms try to adjust the points 3D positions and camera pose in order to minimize the projection errors.

Figure 2.6: Projection Error - difference from calculated projections to tracked coordinates

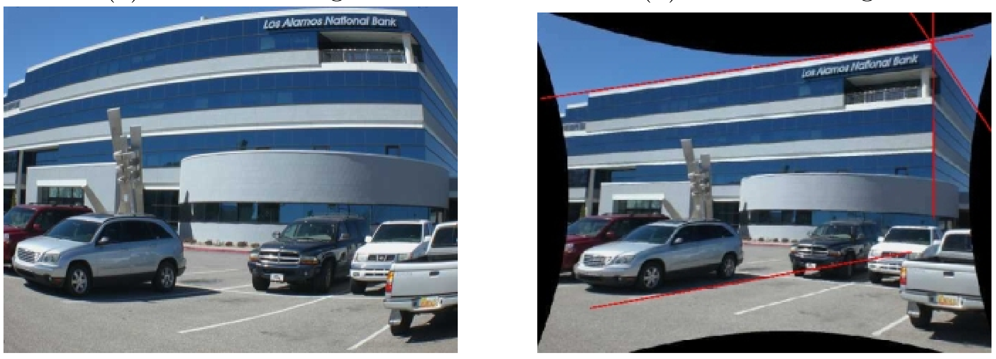


source: author

Real cameras employ lenses to allow more light through the principal point, which commonly cause distortions, mainly in radial direction. Both the intrinsic parameters and distortion coefficients can be discovered by the process called camera calibration. Distortion arising from camera lenses can be corrected by image rectification as shown in figure 2.7 and only the main intrinsic parameters of matrix **K** used from there on.

Figure 2.7: Image distortion correction

(a) Distorted Image (b) Rectified Image



source: (www.edwardrosten.com/work/)

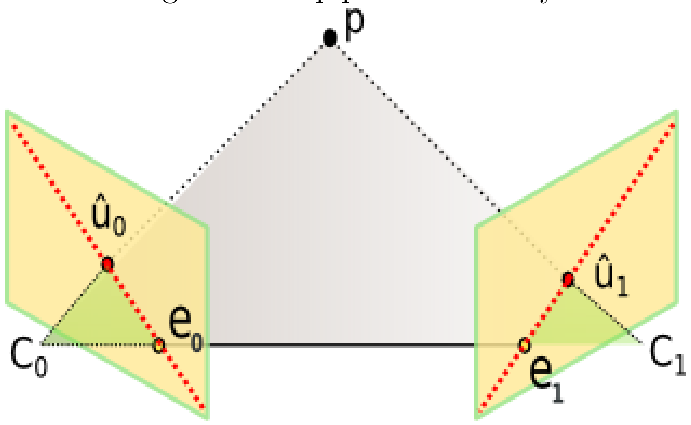
Once a camera is calibrated, and distortion removed, the pinhole camera model can be used as an accurate projection model for real cameras.

2.3.2 Epipolar Geometry

This section presents the projection constraints for a scene viewed from two different perspectives. In stereo vision two cameras are used, usually side by side, to simultaneously acquire what is called a stereo image pair. The geometric difference from stereo to monocular configurations is that in stereo case the camera intrinsic parameters (focal distance, camera center and distortion coefficients) may be different but the relative camera position is usually known (at least approximately). While in the monocular case, intrinsic camera parameters are the same, but camera relative pose must be estimated for each frame. Extrinsic parameters however (camera position and orientation) have equivalent properties, and the topics covered in this section are equally valid for both situations.

Consider a 3D point $\mathbf{p}$ viewed by two cameras with known intrinsic parameters. The point and both camera centers form a plane in space, and the intersection of that plane with the image planes forms a line in each image, the epipolar lines (EL) as shown in figure 2.8. In the figure $\mathbf{p}$ is a point in 3D space, C_0 and C_1 are the camera centers, $\hat{\mathbf{u}}_0$ and $\hat{\mathbf{u}}_1$ are projections of the point on the image planes, $\mathbf{e}_0$ and $\mathbf{e}_1$ are the epipoles and the dashed red lines are the epipolar lines.

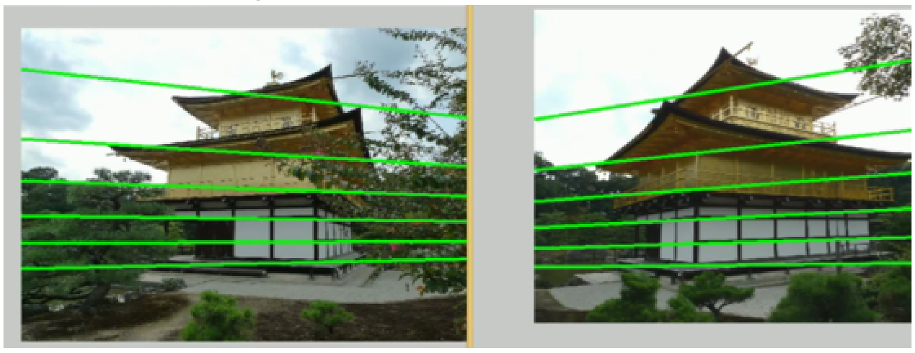
Figure 2.8: Epipolar Geometry



source: author

Once the relative camera pose is known, the point projection in one image ($\hat{\mathbf{u}}_0$ for example), constrains the image region where the point projection in the other image ($\hat{\mathbf{u}}_1$) can be located to the epipolar line. The position along the EL where the second projection is located, defines the point's depth (distance from point to the camera center). Figure 2.9 shows a real example, where a temple is photographed from different perspectives. The green lines are epipolar lines. It is possible to note that the top line of the left figure crosses the top roof and intercepts a tip of the second roof. The same points are also located in the top line of the image on the right. The same happens to all the lines in both images, any image point in a particular line in one image, will be found along the corresponding line in the other image (provided it is visible in both images).

Figure 2.9: Epipolar lines example



source: (RADKE, 2012)

The epipole is the point of intersection of the line joining the camera center and the image plane, it is the position in one image of the camera generating the other image. All epipolar lines intersect at the epipole.

2.3.3 Triangulation

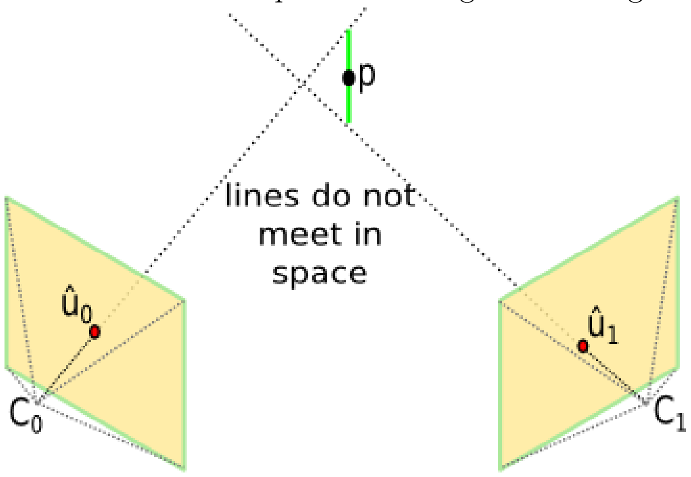
Triangulation is the algorithm used to find the 3D position of a point from its projection in images. Triangulation assumes known relative camera position and orientation and known intrinsic camera parameters. In this section triangulation from two images, i.e. two-view triangulation is briefly revised.

In the absence of errors exact triangulation can be calculated by finding the 3D intersection of the lines that connect the camera centers to the image feature position as shown in figure 2.8.

Several sources of errors such as occlusions, false-positives on tracking, moving objects in the scene, motion-blur and other artifacts, cause errors in feature image position and camera pose estimation. In the presence of errors the lines no longer intersect in space and inexact triangulation algorithms must be used.

The naive approach, used when the lines from the camera centers trough the tracked features do not intercept in space, is to find the mid of the line segment that is perpendicular to both lines. Figure 2.10 exemplifies the situation.

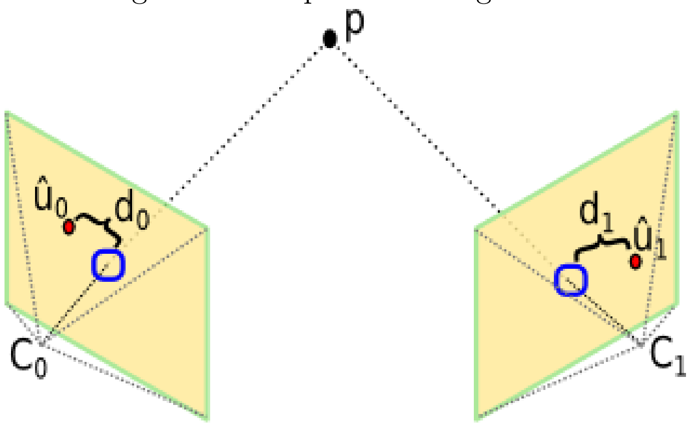
Figure 2.10: Mid of Perpendicular Segment Triangulation



source: author

Hartley and Sturm in the work "Triangulation" (HARTLEY; STURM, 1997) proposed the theoretical optimal method which finds the 3D point that minimizes the distance of projected and viewed feature in both images as shown in figure 2.11. In the figure the distances d_0 and d_1 measured in pixels are jointly minimized. As the method involved solving a sixth degree polynomial, Kanatani et al. (KANATANI; SUGAYA; NIITSUMA, 2008) later proposed an iterative non-linear optimization solution. In both works the method that finds the midpoint of the perpendicular line segment is proven not to be a good solution.

Figure 2.11: Optimal Triangulation



source: author

Lindstrom with "Triangulation Made Easy" (LINDSTROM, 2010), provided a non-iterative improvement for Kanatani's method finding the optimal step size for non-linear optimization. Both (KANATANI; SUGAYA; NIITSUMA, 2008) and (LINDSTROM, 2010) lack an analysis of camera relative pose estimation error in the estimated depth.

Chapter 7 presents a fast triangulation method based on a simplification of optimal triangulation by minimizing the reprojection error in a single image.

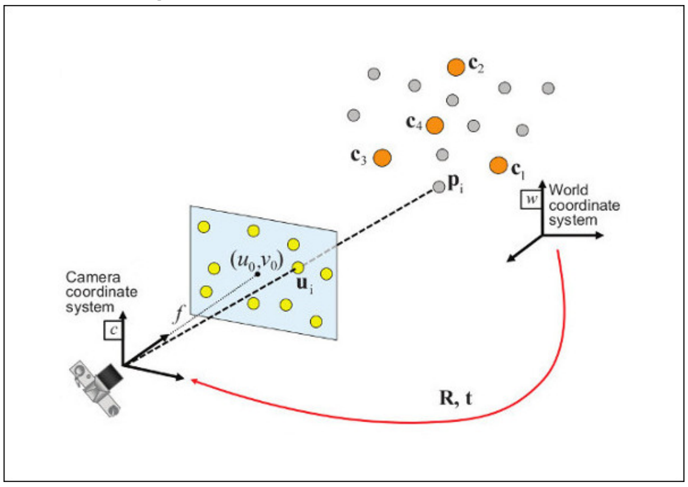
2.3.4 Camera Pose Estimation

The problem of non-linear optimization for jointly estimating scene structure and camera position is termed Bundle Adjustment (BA). The name is inspired in the bundle of light rays entering the camera lenses. When the camera pose is the only parameter to be estimated, a term often used is motion-only BA.

Structure-only BA uses non-linear minimization to estimate the scene structure when camera poses are immutable. It is similar to the triangulation viewed in last section, with the difference that multiple views are usually used.

Closely related to Motion-only BA is Perspective-n-Point (PnP), which can be defined as the process of, given a set of 3D point coordinates and their projection on the image plane, find the camera position and orientation that minimizes the projection error (FISCHLER; BOLLES, 1981). Figure 2.12 demonstrates the context. On the simplest form three points are used in the P3P algorithm (GAO et al., 2003). Other solutions have been proposed in literature such as EPnP (LEPETIT; MORENO-NOGUER; FUA, 2008) and P5P (NISTER, 2004), as well as non-linear iterative methods for arbitrary number of points such as Gauss-Newton (GN) and Levenberg-Marquardt (LM) (BJORCK, 1996).

Figure 2.12: Perspective-n-Point



source: (BRADSKI, 2000)

When camera intrinsic parameters are not known, reconstruction can be performed up to a projective ambiguity, where reconstructed lines are not guarantee to remain parallel or perpendicular. When cameras are calibrated, or parameters estimated accurately, Euclidean or metric reconstruction is possible, where the estimations only differ from the ground truth by and unknown similarity transformation (i.e. scale and absolute position) (HARTLEY; ZISSERMAN, 2004).

Section 8.2 proposes a new method for two view camera pose estimation based on alternation of triangulation and non-linear minimization of projection residuals.

2.3.5 Numerical Optimization

Optimization is an important field in decision making research and physical systems analysis. In scientific context optimization means the selection of the best alternative with respect to a specified criterion.

This criterion must be objectively defined as a measure of system performance (such as time, profit, cost etc) and is dependent on system variables. System modeling is the process of identification of the relationship between objective function and system variables. Once the model is defined the optimization algorithm aims to adjust those variables in order to optimize the objective.

In the optic-geometric problems studied in this work, the data available is the tracked feature position and the calculated projection of points, from the camera model. The obvious objective is to minimize the distance from tracked to projected coordinates as was shown in figure 2.6. The 3×4 matrix $\mathbf{T} = \mathbf{R}|\mathbf{t}$ is replaced by the vector containing the three camera rotations and three translations ($\mathbf{w} = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z]$). The Rodrigues formula (BRADSKI, 2000) can be used to translate between the rotation matrix and the three rotation angles.

The objective function s that will be minimized for a set of m features is the sum of squared residuals $\mathbf{r} = (r_0, \dots r_{2m-1})$ as shown:

$$s(\mathbf{w}) = \sum_{i=0}^{2m-1} r_i^2(\mathbf{w}) \tag{2.15}$$

Where the residuals are defined as the horizontal and vertical difference from projection to tracked feature coordinate:

$$r_{2n} = \hat{u}_n - u\prime_n; \;\; r_{2n+1} = \hat{v}_n - v\prime_n \mid \forall n \in [0, m-1] \tag{2.16}$$

The Gauss-Newton (GN) algorithm is an approximation of the Newton's method to minimize a sum of squared function values with the advantage that second order derivatives calculations are not needed.

Starting from an initial estimation $\mathbf{w}^{(0)}$ the algorithm iteratively approximate the system variables according to:

$$\mathbf{w}^{(s+1)} = \mathbf{w}^{(s)} - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}(\mathbf{w}) \tag{2.17}$$

Where $\mathbf{J}$ is the Jacobian matrix defined as:

$$\mathbf{J}_{(i,j)} = \frac{\partial r_i(\mathbf{w})}{\partial \mathbf{w}_j} \tag{2.18}$$

Gauss-Newton algorithm presents a rapid conversion rate, but is prone to stability issues. Levenberg-Marquardt (LM) algorithm is a damped variation of GN as shown in equation 2.19. In LM robustness in the situations where the inverse of the approximate Hessian $(\mathbf{J}^T\mathbf{J})^{-1}$ is ill conditioned is improved by the use of the damping term $\lambda_d\mathbf{I}$.

$$\mathbf{b}^{(n+1)} = \mathbf{b}^{(n)} - (\mathbf{J}^T\mathbf{J} + \lambda_d\mathbf{I})^{-1}\mathbf{J}^T\mathbf{r} \tag{2.19}$$

Where λ_d is an adjustable damping parameter. Values closer to zero approximate LM to GN, and increasing the parameter value shifts the solution closer to gradient-descent. A common variation is to replace the identity matrix $\mathbf{I}$ by the diagonal of the approximate Hessian $(\mathbf{J}^T\mathbf{J})$.

Thrust region methods like Dog-Leg (CONN; GOULD; TOINT, 2000) try to robustly balance gradient-descent and Gauss-Newton. Although such methods have been proposed for multi-view triangulation (HEDBORG; ROBINSON; FELSBERG, 2014), they are less used in practice for BA.

2.4 Scale Recovery

As mentioned in section 2.3 the feature position and camera relative pose can be estimated up to a similarity ambiguity. That is the translation and depth scale, and absolute camera orientation can not be recovered directly. Figure 2.13 exemplifies the situation showing a model of a road. By looking at the picture it is not possible to know the absolute size and orientation of the road.

Figure 2.13: Scale Ambiguity

source: https://br.all.biz/maquete-de-estradas-de-cidade-g90239

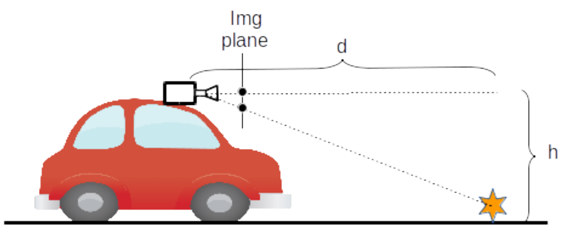
Since scale information is absent from monocular image series, a strategy to infer the correct scale must be adopted. Monocular systems must rely on either other sensors or a previously known image physical characteristic to recover the movement scale. The common approach used in camera equipped vehicles applications is to use the camera height and relative ground position to estimate the scene depth and movement scale as explained in (SCARAMUZZA et al., 2009) and overview in the next subsection.

2.4.1 Scale Extraction from Camera Height

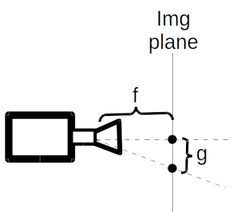
Figure 2.14 (a) show the basic setup of the application in study. The camera matches a feature on the ground where d is the feature depth and h the camera height with respect to the ground. The image plane in detail is shown in image 2.14 (b).

Figure 2.14: Scale from Camera Height

(a) Setup



(b) Ground Feat. on the Img Plane



source: Author

From figures 2.14 (a) and (b) a triangle similarity can easily be devised as shown in equation 2.20. In the equation f is the focal length (in pixels), g is the difference from the image center to the found feature position (in pixels), d is the feature depth, and h is the camera height.

$$\frac{d}{h} = \frac{f}{g} \tag{2.20}$$

Isolating d:

$$d = \frac{fh}{g} \tag{2.21}$$

The absolute scene size can be adjusted approximating the depth d calculated with triangulation to the depth calculated with equation 2.21. The equation assumes negligible rotation along z axis (roll). Rotation along y axis (yaw - camera side turning) have no impact on scale estimation. This simple model oversights rotations on x axis (pitch or camera nodding) even though this parameter present a heavy impact on scale estimation.

A capital step in ground based scale estimation is robust and accurate ground feature tracking. Another issue that must be carefully addressed is the detection and compensation for angular oscillations around x axis (camera nodding).

Chapter 6 introduces developed algorithms for ground feature tracking, ground feature selection and camera nodding compensation.

2.5 Chapter Conclusion

This chapter presented an overview of the main operations related to visual navigation. The three main operations related to monocular visual odometry are feature detection and tracking, camera pose and scene structure approximation and scale recovery.

Feature detection and tracking uses image processing algorithms. Feature detection selects distinguishable image regions and tracker locates those regions in following frames. An alternative to tracking is to use feature descriptors where image patches are transformed in a code-word and those codes are compared to decide if they belong to the same scene point. Once feature correspondences coordinates are available, the considered geometric techniques use re-projection error minimization to find camera pose and feature 3D position that harmonize the tracked position and calculated feature projections.

Finally the camera height and tracked ground features are used to recover the true scale of both scene depth and camera translation.

Table 2.1 relates the most relevant methods over-viewed in this section to their corresponding category.

Table 2.1: Algorithms summary

Method	Category
Harris/Shi-Tomasi	Detection
FAST	Detection
Lucas-Kanade	Tracking
SIFT	Matching
BRIEF/ORB	Matching
Triangulation	Scene Structure
PnP	Camera Pose
BA	Structure and Pose
GN/LK	Optimization

3 VISUAL ODOMETRY IN THE LITERATURE

"Those who do not remember the past are condemned to repeat it."

George Santayana

This chapter reviews some of the most relevant works in monocular visual odometry and summarizes the advances achieved in last decades.

In 1985, Chatila and Laumond (CHATILA; LAUMOND, 1985) proposed performing localization and mapping in a concurrent manner, what was later called SLAM. Early visual navigation systems used a binocular stereo system (SE; LOWE; LITTLE, 2002). A successful monocular localization and mapping technique that influenced many authors is MonoSLAM (DAVISON et al., 2007), where an extended Kalman filter (EKF) was used to update a probabilistic 3D scene map. Filtering approach became a popular technique at the time, shared by implementations like (CHEKHLOV et al., 2006) and (MENG; HONG; CHEN, 2009).

The milestone work PTAM from Klein and Murray (KLEIN; MURRAY, 2007) was the first work to split image processing (tracking) and geometrical processing (camera pose and scene structure) in separate threads. Using a keyframe-based framework that influenced many solutions (FORSTER; PIZZOLI; SCARAMUZZA, 2014), (SONG; CHANDRAKER; GUEST, 2013) and is still a reference for present implementations (SONG; CHANDRAKER; GUEST, 2016), (PERSSON et al., 2015).

Dense or semi-dense propositions (ENGEL; STURM; CREMERS, 2013) use the whole image, acting directly on pixel intensities, warp the image to produce a dense 3D structure of the scene. Those methods valuable when the objective is to build a detailed scene map, as depth is estimated for every pixel presenting relevant graphical gradient. When used for camera tracking however, they tend to be more susceptible to outliers in non-static scenes.

Sparse systems on the other hand, must select and recognize traceable image regions, and commonly use a key-frame based approach (KLEIN; MURRAY, 2007; FORSTER et al., 2015; SONG; CHANDRAKER; GUEST, 2016). Keyframes are a subset of the generated images, and new feature are only detected in those frames.

As full Bundle-Adjustment (BA) for a large set of images is computationally demanding, most solutions tend to use a subset of the images in what is called local BA. ORB-SLAM (MUR-ARTAL; MONTIEL; TARDóS, 2015) use a co-visibility graph to select frames for local BA, while the most common approach in VO is to use the last frames as used by (ENGEL; KOLTUN; CREMERS, 2016), (SONG; CHANDRAKER; GUEST, 2013) and (Mirabdollah M., Mertshching B. (GET Lab, 2015).

The robustness and generality of monocular techniques are assessed in the work from Persson et al. (PERSSON et al., 2015). In that work, the authors adapted monocular algorithms like tracking by matching and long optimization windows to a binocular stereo vision system. The work achieved remarkable results outperforming other solutions proposed at the time.

In order to compare the developed solutions to other implementation, this work uses a public dataset. The KITTI dataset for odometry is a collection of 22 image sequences recorded with a camera equipped vehicle. For 11 of the sequences the GPS recorded path is supplied, and may be used for system calibration. Results of the remaining 11 sequences can be uploaded to the site and used for comparison. The following sections detail the monocular works that achieved most notorious results when tested with the Dataset. More details about the sequences will be dicusssed in chapter 9.

3.1 VISIO-2-M

Originally focused on stereo scene reconstruction, Visio 2-M was the first work that tried monocular estimation on the KITTI dataset (GEIGER; ZIEGLER; STILLER, 2011). Image descriptors were not used and sum of absolute difference (SAD) over an 11x11 block of pixels with SSE instructions were employed for feature tracking and matching. A circular matching algorithm searched for detected features in the next frame, and found features were traced back in the original frame. Features that presented a mismatch from the original position to the back-traced position are discarded as possible outliers.

Non-maximum and non-minimum suppression have been used to reduce the total number of features to values between 200 and 500. Camera pose is calculated through non-linear reprojection error minimization using Gauss-Newton adjusting the six pose parameters (three translation distances and three rotation angles) and depth is calculated through triangulation.

The original stereo system was adapted to monocular case, and produced interesting results at the time, with some of its ideas inspiring more recent solutions with better estimation results.

3.2 MLM-SFM

Song et al (SONG; CHANDRAKER; GUEST, 2013) presented a multi-threaded system largely enhancing the monocular odometry precision in 2013. The work was later improved in 2016 (SONG; CHANDRAKER; GUEST, 2016). FAST corners and ORB descriptors were used for detection and matching and 4 point PnP (LEPETIT; MORENO-NOGUER; FUA, 2008) used for camera pose calculation. The pose with largest consensus in a RANSAC framework was further refined with Levenberg-Marquard non-linear optimization.

Camera pose and 3D point positions were additionally improved with a sliding window resection-intersection (RI) bundle adjustment with the last 10 frames. The SBA package (LOURAKIS; ARGYROS, 2009) was used, in a sequence of operations that alternate triangulation and pose estimation. Scale was estimated by triangulating points on the ground and using a fixed ground projection.

In the more recent version (SONG; CHANDRAKER; GUEST, 2016) scale estimation was enhanced by the combination of multiple cues. A dense homography was used for ground plane projection. Objects detected in the scene were used to find vertical lines and approximate camera pitch angle. And triangulation of ground points improved the ground plane position estimation. The error distribution of each cue was approximated to a Gaussian distribution and a Kalman Filter (KF) was adopted to combine the different cues in a statistical framework.

When proposed for the first time, this solution presented precision comparable to stereo systems of the time, with results that outperformed all other published MVO system so far.

3.3 FTMVO

MLM-SFM remained as the most accurate VO system tested on the KITTI dataset until 2015 when Mirabdollah and Mertsching published FTMVO (Mirabdollah M., Mertshching B. (GET Lab, 2015).

That solution used the five point algorithm (NISTER, 2004) for initial camera pose estimation. Point positions were calculated with a probabilistic multiple view triangulation. Errors in camera pose estimation and feature tracking were modeled as Gaussian random variables, and the maximum of their approximated probability distribution used to calculate feature depth.

To improve camera pose estimation and reduce scale uncertainty, a window with the last 10 observations of features was used in a local bundle adjustment step. Features were divided in two groups, with low and high parallax (far and close features). Epipolar based error is used for far points and a projective error for close points.

The system runs in a single thread, using Harris corners (HARRIS; STEPHENS, 1988) and Lucas-Kanade (BOUGUET, 2000) for feature detection and tracking.

With the innovative approach of treating differently far and close features, and the probabilistic triangulation framework, the achieved precision estimation on camera pose improved over previously published solutions.

3.4 PMO / PbT-M2

The interesting work of Fanani Et al. (FANANI et al., 2016), innovated using only two frames for camera pose estimation, renouncing RANSAC for outlier removal and not using re-projection error minimization.

Most of the works based on sparse visual navigation isolate image processing and geometric re-projection error. That is after the use of images for feature tracking pixel intensities are no longer used.

In (FANANI et al., 2016) a regular feature detection/tracking is used to find an initial camera pose. And the found pose is refined using non-linear optimization. The big difference is that the residual function used in minimization is not re-projection error, but pixel intensity difference of pixel positions around the feature centers in both the original and projected pixel coordinates.

This idea was previously explored in the work DSO from Engel Et al. (ENGEL; KOLTUN; CREMERS, 2016). The use of pixel intensity minimization provides robustness at the cost of a higher computational complexity due to two reasons. A single feature generates several pixel residuals. And several pixel positions must be interpolated at every optimization cycle.

The robustness provided by the proposed optimization enabled (FANANI et al., 2016) to replace RANSAC by sequential outlier removal.

3.5 Chapter conclusion

This chapter presented a brief review of most relevant works in monocular visual navigation that use the KITTI dataset (GEIGER; LENZ; URTASUN, 2012a) as evaluation platform. The seminal works MonoSLAM (DAVISON et al., 2007) and PTAM (KLEIN; MURRAY, 2007) inspired solutions that achieved remarkable results. Table 3.1 lists significant methods used in those solutions.

Table 3.1: MVO solutions summary

SOLUTION	detection	tracking	pose est.	struc. est.	Transl Err[%]
VISIO-2 M	SAD	SAD+circ	GN	triang.	11.9
MLM-SFM	FAST	ORB	wBA[EPnP]	wndTriang	2.54
FTM-VO	KLT	LK	P5P	mvTriang	2.24
PMO	KLT	LK	LM (pixel int)	triang	2.05

source: author

In the table the detection and tracking methods are named as SAD for sum of absolute difference, FAST and HARRIS are the corner detector from sections 2.2.2 and 2.2.1 and KLT is the Kanade-Lucas-Tomasi expression that combines Shi-Tomasi (SHI; TOMASI, 1994) detector and Lucas-Kanade (LUCAS; KANADE, 1981) tracker.

The pose estimation and structure estimation algorithms GN and LM are the classical non-linear optimization Gauss-Newton and Levenberg-Marquardt. Triang refers to the triangulation method from section 2.3.3, windowed methods wndTriang use a set of frames. Perspective-n-point (EPnP, P5P) and windowed Bundle-Adjustment (wBA) are used for camera pose estimation as overviewed in section 2.3.4.

The last column brings the average translation error as measured by the

Dataset developers, for an idea of the relative accuracy of each solution.

4 ANALYSIS OF PROJECTION ON MOVEMENT ALONG THE OPTICAL AXIS

"Life can only be understood backwards; but it must be lived forwards."

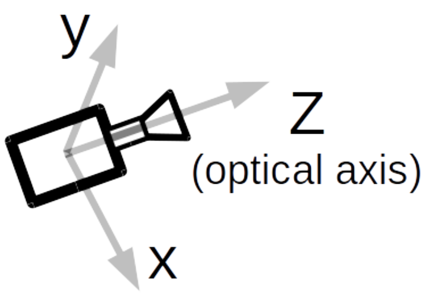
Søren Kierkegaard

It is well understood that different camera translations have different impacts on camera pose estimation. Movement along the optical axis (forward or backward) produces lower parallax than movement perpendicular to the optical axis (sideways translations). This chapter presents an analysis of projection characteristics of a camera moving along the optical axis and suggests a backward looking strategy for camera pose estimation. This investigation was presented in (PEREIRA et al., 2017a) and was selected as a best paper on the conference.

4.1 Projection Error in Backward Movement

The camera frame of reference has the camera center as origin and the z axis perpendicular to the image plane as shown in figure 4.1.

Figure 4.1: Optical Axis



source: Author

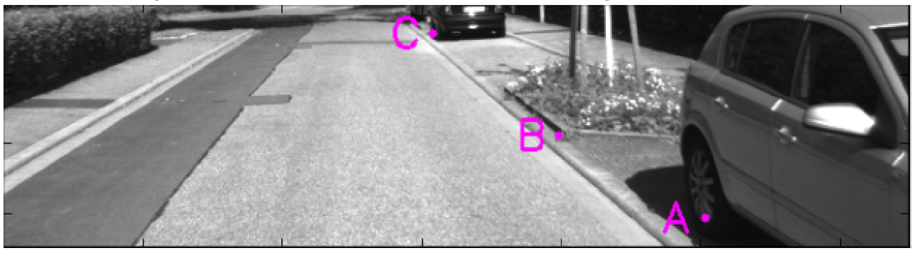
When a camera moves sideways, objects closer to the camera present a larger image displacement when compared to a further away background. In motions along the z axis (forward or backward), closer features near the image borders present a higher parallax when compared to points distant to the camera and closer to the image center. As a consequence, the same error in estimated depth produces a larger image projection difference for closer points. Accordingly, the 3D position of points closer to the camera can be estimated with superior accuracy for the same image tracking precision.

Figures 4.2 and 4.3 taken from the KITTI database (GEIGER; LENZ; URTASUN, 2012a) exemplify this situation: In the upper image area three points 'A', 'B' and 'C' are marked. Since the camera relative position and orientation, as well as camera calibration, are given by the database developers, the points can be tracked in the next image and the depth can be estimated by triangulation (HARTLEY; STURM, 1997). In the present case they are 7.5m, 12.8m and 72.2m.

The blue line segments in figure 4.3 are the projected position interval, when the estimated depth is within a ±25% interval of the estimated depth, whose length in pixels are respectively 16.2, 6.5, 0.54. The difference in the interval in pixels illustrate that the same error in depth generates a larger projection error for closer points.

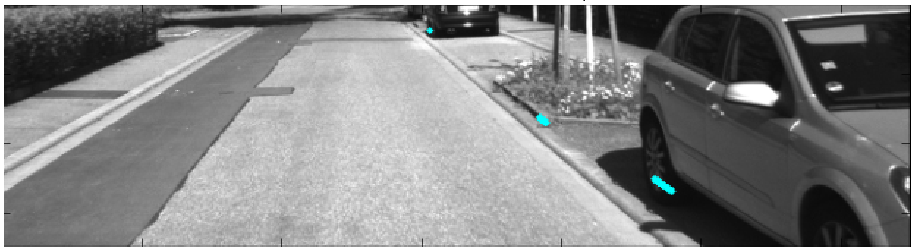
It is also possible to calculate the depth accuracy for a given image tracking precision. If for instance it is supposed that the image tracker finds features within a circle of radius of 0.2 pixel, the estimated depth in meters of feature 'A', 'B' and 'C' would be inside the intervals $7.5 \pm 0.3\%$, $12.8 \pm 1.2\%$ and $72.2 \pm 11\%$.

Figure 4.2: Initial points in tracking environment



source: (GEIGER; LENZ; URTASUN, 2012a) modified by author

Figure 4.3: Predicted position with +/- 25% depth range



source: (GEIGER; LENZ; URTASUN, 2012a) modified by author

On a forward moving camera, new features are usually detected for the first time in their most distant position from the camera. Depth is estimated, and those features are used for further camera pose estimation. When backward movement is used, the camera moves away from the scene, and new features are located for the first time in their closest position to the camera. In the latter case, the feature three dimensional position is estimated with superior accuracy, improving future camera pose calculations. In the example of figure 4.2 point 'A' would shortly move out of sight in forward movement, while remaining visible in following images if backward movement is used.

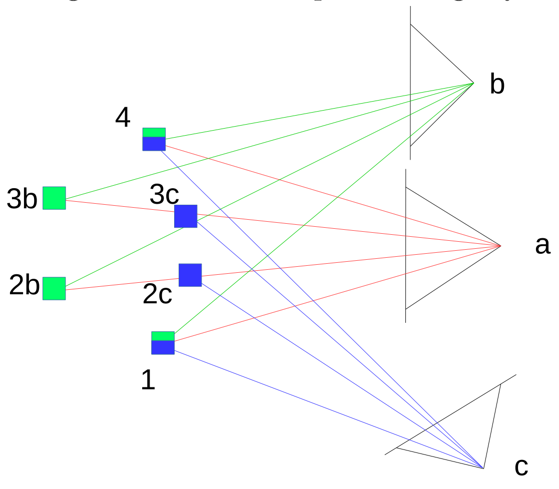
The second and complementary effect of backward movement is the projection error amplification on approaching features. Because of the same discussed characteristics of forward movement, the projection error for small depth inaccuracies is amplified when a feature approaches the camera. This effect is intensified in a RANSAC framework (FISCHLER; BOLLES, 1981), where good close features may be ignored due to projection errors as a result of small estimated depth inaccuracies. Points with low parallax present projections sensitive to camera rotation but their image position is slightly affected by camera translation, especially on the z axis. To recover z translation, close points are of paramount importance, and should not be ignored.

When good points are ignored and points with larger position estimation errors are used to calculate camera pose, a third effect may occur: camera pose estimation and scene structure can converge to local minima not corresponding to the correct pose as shown in figure 4.4.

In the figure a camera at position a 'sees' four features. In a second moment the same features move to the left on the image plane where the center features move slightly less then the outer features. Two possible camera movements and feature positions are then shown. Either the camera moved to the right to position 'b' and the center features are further away from the camera than the other features (in position $2b$ and $3b$), or the camera translated to the left an rotated to position 'c' and the center features ($2c$ and $3c$) are closer to the camera than the outer features (1 and 4).

Backward movement, for up-to-scale camera pose estimation, improves precision due to superior accuracy for depth calculation of new points because they are usually closer to the camera. Robustness is improved as points close to the camera are hardly skipped in a RANSAC scheme reducing the chances of Local-minima

Figure 4.4: Camera pose ambiguity



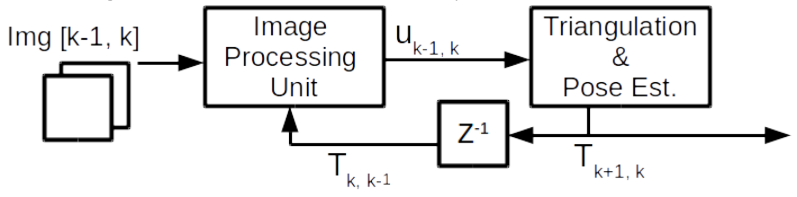
source: author

problems on camera pose and scene structure estimation. Finally well estimated points are used in following estimations as they remain in sight until they are farther from the camera. Those characteristics allow a system that rely on only the last two views for camera pose estimation and feature depth calculation. Next section presents a simple visual navigation system that proves these conclusions.

4.2 Backward Based Visual Odometry

This section presents a backward moving visual odometry system that takes advantage of the principles discussed in last section. The system is based on simple camera pose estimation using reprojection error minimization followed by two-view triangulation. The backward movement is simulated by processing images in reversed order. Figure 4.5 shows the system architecture.

Figure 4.5: Backward VO System Architecture



source: author

In the proposed application, features are detected in the current camera frame and tracked in the new frame. A feature is described by its coordinates in both images and estimated depth with respect to the current frame. Features not located are removed from the feature set, and new features are detected in every new frame.

4.2.1 Implementation

This section provides an overview of the system used to validate the conclusions of the last section. The reader interested in implementation details may refer to the conference paper (PEREIRA et al., 2017a). The Image Processing Unit is responsible for feature detection and tracking, and the Triangulation & Pose Estimation module finds the camera pose by re-projection error minimization. The last estimated camera rotation and translation $T_{k,k-1}$ is used as initial estimate of current camera relative pose.

The Image Processing Unit used Shi-Tomasi corners (SHI; TOMASI, 1994) and Lucas-Kanade tracker (LUCAS; KANADE, 1981) to find feature correspondences, as this combination presented lower re-projection errors than the descriptor based methods. The unit output is a matrix of feature coordinates $U_{k-1,k}$.

The system used a list of outliers (non-static features) that should be removed from camera estimation. When the tracked image position does not remain along the epipolar line the feature is marked as a bad feature. It continues to be tracked, but is not considered for pose estimation. If in future localization, the feature is found to be in the EL, it is removed from the bad features list, and can be used in the pose estimation algorithm.

The pose estimation used the EPnP algorithm (LEPETIT; MORENO-NOGUER; FUA, 2008) followed by non-linear Levenberg-Marquardt optimization. The optimal triangulation proposed by (HARTLEY; STURM, 1997) is used. Both algorithms have implementations available in OpenCV (BRADSKI, 2000).

4.2.2 Backward Visual Odometry Results

The proposed system was implemented in Python, using SciPy (JONES et al., 2001–) and OpenCV (BRADSKI, 2000). The achieved results on the KITTI dataset (GEIGER; LENZ; URTASUN, 2012a) are presented in this section.

Minimum feature distance used was 15 pixels; LK used 3 pyramid levels for normal features and 2 levels for ground points. Newer features that move inside an occupied 15x15 pixels grid position are removed to prevent excessive clustering. Found features are traced back in the original picture to avoid bad matches and occlusions.

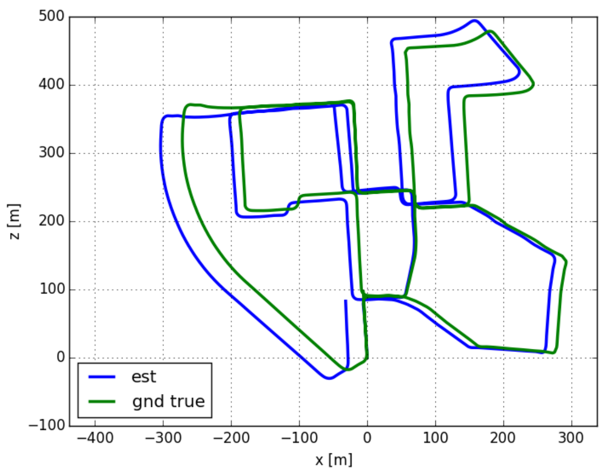
The localization step minimized the reprojection error in sets of 10 features, and $nR = 45$ repetitions. Far features are added to the blacklist if projection error is above 0.5 pixels, and removed from the list if the projection error falls below 0.1 pixels. The median re-projection error after scene-pose stabilization was in the order of 0.2 pixels.

Table 4.1: Per trajectory results

sequence	00	01	02	03	04	05	06	07	08	09	10	Avg
rot [10^{-3}deg/m]	4.8	2.6	3.9	2.3	3.2	3.0	2.8	2.8	3.7	3.0	3.6	3.2

source: author

Figure 4.6: Estimated trajectory and ground truth for the first sequence



source: author

Table 4.1 shows the results rotation results for the first 11 routes. Translation errors are heavily dependent on scale estimation and are outside the scope of this chapter. Figure 4.6 shows the car path in meters for the first trajectory (estimated in blue and ground truth in green) for a qualitative idea of estimation precision.

The errors in table 4.1 are measured in 10^{-3} degrees per meter. The values in the table present low orientation drift, that can be visually verified in figure 4.6. The figure shows the car trajectory over several hundred meters and very accurate orientation estimation, for the simple backward proposed system.

The high level Python implementation running in a single thread, in an Intel core i7-3770K CPU @ 3.5GHz achieved real time performance (0.09s per frame on average). Since there is no key-frame selection, complex feature descriptors or windowed bundle adjustment, memory requirements should be convenient to lower capacity hardware.

4.3 Chapter Conclusion

In this chapter the advantages of pose estimation for a camera looking at features that move away from the camera were discussed. This effect can be achieved with a backward looking camera or processing images in reverse order.

A Monocular backward VO system was implemented to validate the drawn conclusions. The system used a simple two-view feature tracking and pose estimation approach. A list of not trusted features was used to improve robustness to outliers.

The python and OpenCV implementation, achieved real time performance of 10 frames per second on an Intel core i7. A low level C implementation targeted to an embedded platform, with parallel processing of feature correspondence and geometrical estimation, with a thorough analysis of worst case execution time remains as a future work.

The concepts presented in this chapter provide the conceptual base over which the ideas and algorithms presented in the following chapters were founded.

5 DETERMINING IMAGE CORRESPONDENCES

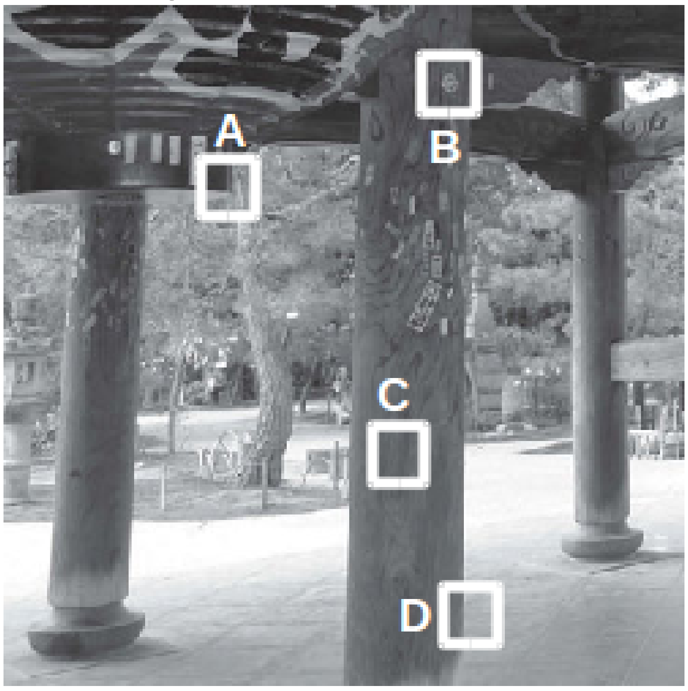
“Do not go where the path may lead, go instead where there is no path and leave a trail.

— *Ralph Waldo Emerson*

The first step in classical sparse navigation pipeline is to find image correspondences. That is, to find the position in more than one image where the same scene point is projected. A process that is divided in two steps, detection of interest points and tracking or matching of those points in another image.

Interesting regions must present an image intensity gradient different than zero, as uniform areas are not distinct form neighbors. Pixel gradients must occur in more than one direction, otherwise there would be a single border in the image area, and displacements along the border are not noticeable. Those interesting regions are called image features and are classified in two main categories: blobs and corners.

Figure 5.1: Corner Candidates



source: (RADKE, 2012)

Figure 5.1 shows four feature candidates. Region *A* is a clear corner, region *B* is a blob, region *C* is a homogeneous region that is similar to neighboring regions and region D is a border in which vertical displacement can hardly be noticed.

It is clear that regions A and B are preferable over candidates C and D. Once interesting regions have been recognized, those regions must be identified in following images. Two classes of algorithms are used to accomplish this task, matching and tracking as explained in section 2.2.

Matching refers to the process of detecting interest regions in both the original and the target frames, and using a strategy to decide if they belong to the same scene point. The naive approach would be Sum of Absolute Difference (SAD). The transformation of the image area in a feature descriptor is able to provide some level of robustness to orientation, scale and illumination variation.

In the target application, where a camera mounted on a vehicle produces several images per second, the images present a high degree of similarity and feature displacement between consecutive frames are of the order of a few pixels. In that context tracking algorithms that iteratively search the neighboring area, tend to present higher localization performance and fewer outliers when compared to matching solutions. Tracking methods are better suited for low-frequency corners. For those reasons, in table 3.1 most of the analyzed solutions combine corner detection with the pyramidal implementation of Lucas-Kanade tracker (BOUGUET, 2000). The tests performed during the course of this work have shown lower computational accuracy when FAST corner detector (ROSTEN; DRUMMOND, 2005) and ORB feature tracker (RUBLEE et al., 2011) where used, when compared to Shi-Tomasi (SHI; TOMASI, 1994) corner detector and pyramidal LK tracker (BOUGUET, 2000). The second configuration have shown lower number of outliers and higher detection accuracy, and for those reasons were chosen for feature correspondence generation. Next sections explain the strategies used to further improve detection accuracy, remove outliers and prevent feature clustering.

5.1 Grid Based Clustering Prevention

Corner detection algorithms calculate a score for every pixel position in the image. It is common that pixel positions close to a corner present high scores in image areas overlapping the detected feature. To prevent overlapping feature detection, or even feature clustering (many features detected close to each other) non-maximum suppression is usually used.

Classical non-maximum suppression save the highest score position and set the area where the scores was located (up to a predefined distance) to a minimum value guaranteeing neighbors won't be selected. The process repeats finding the next highest score and performing the same actions. The algorithm guarantees that best corners will be selected and a minimum distance between features will be respected.

Since the camera is moving and some scene objects may not be static, the feature position change from one frame to the next, either approaching other features or moving away from them. Camera pose estimation benefits from spread features over the image area and excessive clustering should be avoided. A faster alternative for the iterative non-maximum suppression algorithm mentioned above is explained in this section.

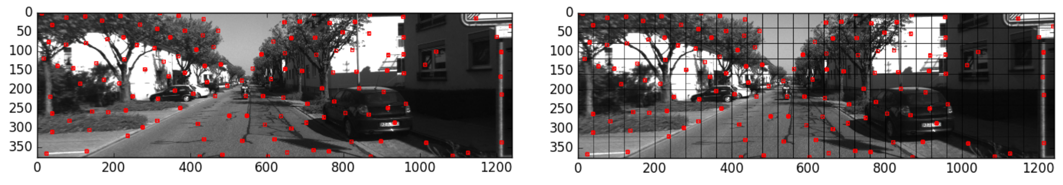
The grid based clustering prevention is a fast approximation of non-maximum suppression. In the algorithm features are sorted by their score value, and the image is divided in a square grid (blocks of $n \times n$ pixels). Then, the block address of every feature is calculated according to equation 5.1. In the equation B_a is the linear block address, u and v are feature coordinates, n is the block size and nB_x is the number of block columns (horizontal picture resolution divided by the block size).

$$B_a = \frac{v * nB_x}{n} + \frac{u}{n} \tag{5.1}$$

The final step is to go through the sorted feature list, and mark only the first feature of each block as a valid feature, removing the remaining occurrences. Figure 5.2 shows an example of the classical non-maximum suppression (a) and the fast grid based clustering prevention (b).

Figure 5.2: Clustering Prevention

(a) Non-maximum suppression (b) Grid Based



source: KITTI Dataset (GEIGER; LENZ; URTASUN, 2012a) modified by author

In the figure features are marked with red dots, and the grid lines forming image blocks are shown in the left image (b). The algorithm prevents more than one feature per image block.

The draw back is that features detected close to a grid border can present a short distance o another feature in a neighboring block. This, however, is not a major problem. The used algorithm ensures that the best features will be selected and prevents excessive clustering at a fraction of the computational cost.

5.2 Perspective Warping for Tracking Enhancement

Features on the ground are especially important for scale estimation, and especially difficult to track due to low gradients on asphalt and high graphical distance for higher vehicle speeds. The ground plane however, remains in a virtually constant position. Figure 5.3 (*a*) shows an image example from the KITTI dataset (GEIGER; LENZ; URTASUN, 2012a) where the ground zone is delimited. Figure 5.3 (*b*) is the subsequent image in the sequence, where two features are marked.

Figures 5.3 (*c*) and (*d*) show warped versions of the first image. In figure 5.3 (*c*), the original image 0 is warped to match the ground zone of image 1 (the reader can notice the green squares in similar positions in both images). In figure 5.3 (*d*) the original image 0 is warped to match mid depth details of image 1 (as highlighted by the similar yellow circle position in both images).

In order to calculate the perspective warping, four virtual control points are used. In ground plane matching, the virtual points are placed in the approximated constant ground position, and for a constant depth approximation, the virtual points are placed at a constant distance from the camera.

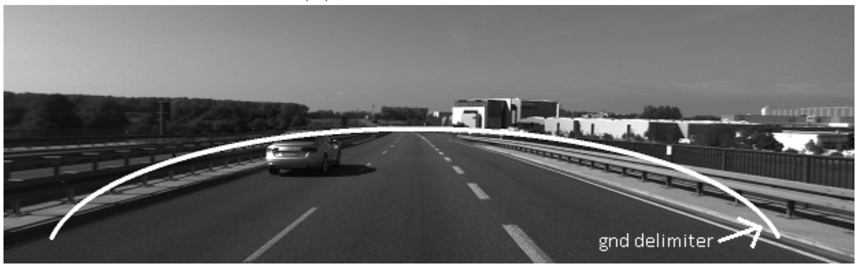
The last relative camera pose $T_{k,k-1}$ is used to project the control points into the next frame. Then using four projections in both frames, a perspective warping transformation can be calculated. No pixel intensities are used in this operation, only geometric point projections.

Feature coordinates tracked in the warped image are translated to their true positions by the inverse of the perspective warping used in the image.

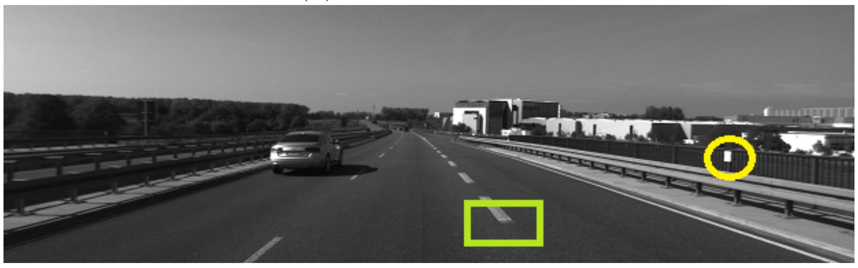
A problem that occurs especially when the car drives at higher speeds, are similar patterns along epipolar lines. On figure 5.3 (a) the guard rails on both sides of the road are very similar and thus difficult to differentiate for correct matching. In stereo cases, the stereo calculated depth can be used to help choosing the correct match, but on monocular situations, the correct correspondence is very difficult to find, as different estimated depths provide different matches with equivalent re-projection errors.

Figure 5.3: Perspective Warping for Tracking Enhancement

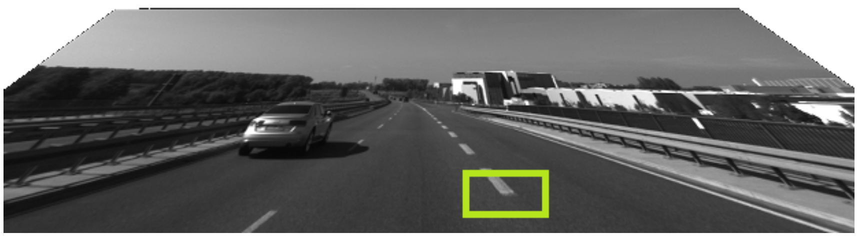
(a) Original Image 0



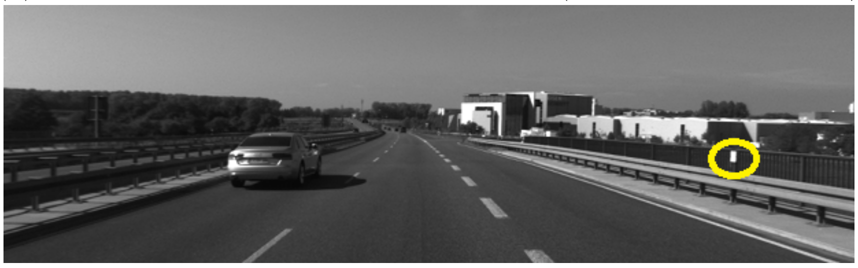
(b) Original Image 1



(c) Img 0 warped for ground matching (as shown by the box)



(d) Img 0 warped for close object matching (as shown by the circle)



source: (GEIGER; LENZ; URTASUN, 2012a) modified by the author

The warping used in image 5.3 (d) can help to mitigate this problem. By warping the image for two different depths and checking if features are located to corresponding locations outliers due to repetitive patterns can be identified and eliminated. In the figure, a small difference in the warping of figure 5.3 (d) is not likely to disturb the tracking of the feature marked with the yellow circle, but would probably change the tracking position of features along the guard-rails.

5.3 Chapter Conclusion

This chapter focused in the used feature generation mechanism, illustrating why the Shi-Tomasi (SHI; TOMASI, 1994) and pyramidal LK (BOUGUET, 2000) were chosen as detector and tracker.

An algorithm to avoid feature clustering that approximate non-maximum suppression at lower computational cost was described.

Finally the use of image warping for tracking enhancement was unfolded. In the algorithm four virtual points are projected in both images, and the projections are used to calculate a perspective image transformation, used to improve feature tracking.

6 GROUND BASED ABSOLUTE SCALE ESTIMATION

"Lice don't seem so small when they get in your hair."

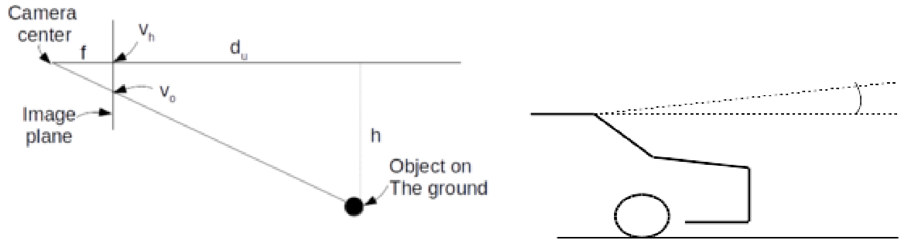
Marty Rubin

In monocular vision systems, absolute scale is an inherently absent information, and must be recovered either by other sensors (odometers, GPS, laser scanners, ultrasound, inertial sensors, etc) or inferred from known image characteristics. In the case of autonomous vehicles, the camera distance to the ground can be measured, as it remains nearly constant, and used to estimate the scene scale as explained in (SCARAMUZZA et al., 2009). It is important to note that scale estimation is not dependent on up to scale camera pose estimation as those steps are loosely coupled. This chapter describes the algorithms developed to ameliorate scale estimation.

The absolute distance of a point on the ground can be calculated by triangle similarity as explained in 2.13. Ignoring usually negligible rotations around optical axis (roll) and rotations around the vertical axis (yaw), the depth of a ground point can be calculated from equation 6.1 and figure 6.1 (a) (repeated here for convenience). By comparing the depth from triangulation and the calculated ground depth, the whole scene scale can be approximated. In the equation f is the focal length, v_o and v_h are the vertical position of the object and the horizon in the image in pixels. The objects depth is d_u and h is the camera height. The position of the horizon is initially set to the center of the image (considering the camera optical axis is perfectly parallel to the ground plane, or pitch is zero). Figure 6.2 shows an example of a point marked in the ground and the horizon line and the vertical position of the object.

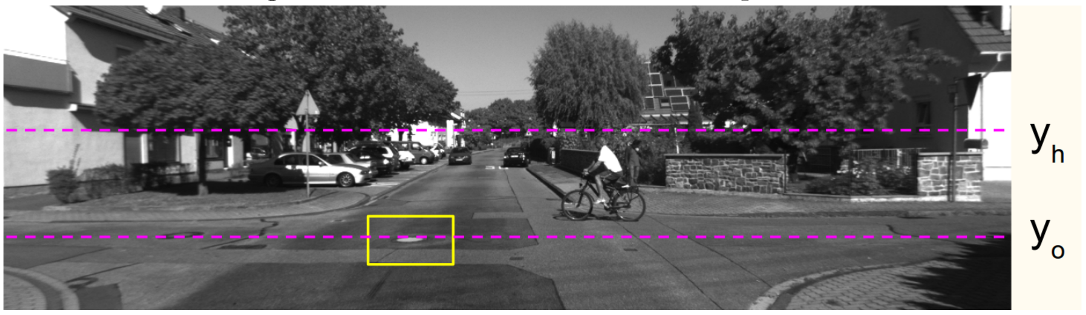
Figure 6.1: Scale estimation

(a) Triangle similarity (b) Pitch angle



source: author

Figure 6.2: Detail of feature vertical position



source: (GEIGER; LENZ; URTASUN, 2012a) modified by author

$$d_u = \frac{fh}{v_h - v_o} \tag{6.1}$$

The developed scale enhancement mechanism is based on pitch angle recovery and multi-attribute ground feature selection. Following sections cover these topics.

6.1 Pitch Recovery

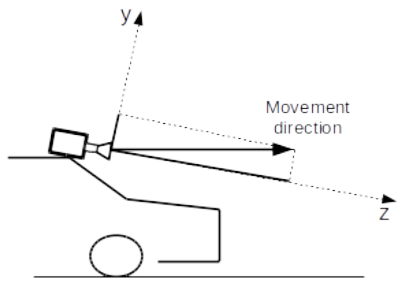
For a camera equipped wheeled robot, scale estimation is less affected by small oscillations in camera height (distance from the ground) or rotation around z axis (roll), than it is by rotations on x axis (pitch or camera nodding), as shown in figure 6.1(b), relative to the ground plane. Pitch angle variations affect the position of the horizon v_h in equation 6.1.

Average pitch angle is not constant from trajectory to trajectory as small difference in the car weight (another passenger, or whether the gas tank is filled or not), may pose small differences to v_h that cause scale offsets. The proposed algorithm estimates camera average orientation with respect to x axis, based on translation movement.

When z camera axis is parallel to ground, average vertical translation (t_y) should be zero and v_h of figure 6.1 (a) is on the calibrated camera center c_y. By the ratio of forward and vertical movement $\Delta t_y/\Delta t_z$ the angle between z axis and the ground plane can be approximated and v_h adjusted. Figure 6.3 shows the translation decomposition when the optical axis is not parallel to the ground.

Since y translation Δt_y is also affected by short time oscillations in camera height, a low-pass filtering is applied and in every new frame a smoothed horizon line image position v_h is updated through equations 6.2 and 6.3, where λ_v is a smoothing

Figure 6.3: Camera Movement Decomposition



source: Author

factor between 0 and 1:

$$nv_h = \frac{ft_y}{t_z} + c_y \tag{6.2}$$

$$v_h = \lambda_v.nv_h + (1 - \lambda_v)v_h \tag{6.3}$$

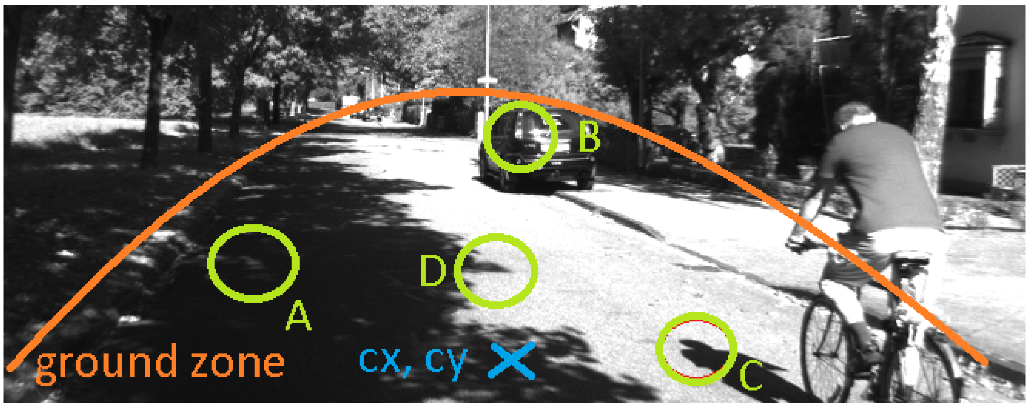
Although algorithms that take into account ground pitch angle have been proposed in (KITT et al., 2011), (ZHOU; DAI; LI, 2016) and (SONG; CHANDRAKER; GUEST, 2016) those works try to estimate the angle from features on the ground, generally using an homography. This work applies a different approach that relies in the ratio of forward and vertical translation movement for ground plane orientation estimation.

6.2 Ground Point Selection

The success in scale estimation is based on absolute distance estimation of points on the ground. It is therefore evident that the selection of a static point that is actually on the ground and correctly tracked is mandatory. Although this might seem a minor task at a first glance, distinguishing correct static matches on the ground from non ground objects (like parked cars or trees), incorrect matches (in low textured asphalt) and non-static objects (pedestrians, bicycles etc) can prove to be non-trivial. A good ground point should be in the bottom center area of the image, should present a good image gradient along the epipolar line to enhance matching precision and is expected to be close to the estimated ground.

On figure 6.4 a typical example is shown, where points A, B, C and D show ground candidates. The line is the ground area, and the cross is the center ground point at (cx, cy). Point A is a ground point with low image gradient and therefore prone to tracking errors. Feature B lies in a parked car and is not on the ground. Candidate C has good gradient and is on the ground but is not static since the feature will follow a moving shadow. Finally candidate D is a good ground point.

Figure 6.4: Ground Point Selection



source: (GEIGER; LENZ; URTASUN, 2012a) modified by the author

A multi-objective cost function is used to elect the best ground point candidate. The function, shown in equation 6.4, calculates a cost based on three characteristics, and the point with lowest cost score is selected. In the equation, c_i is the combined cost, parameter l_i, is the feature cost related to image position, g_i is the score related to graphical tracking quality and s_i is a height related cost of the i^{th} candidate.

$$c_i = l_i + g_i + s_i \tag{6.4}$$

The cost calculation follow a two step procedure, first a raw cost rl_i, rg_i and rs_i are calculated, and then a sigmoid function is used for scaling and smoothing of cost scores.

6.2.1 Positional Cost

The positional raw cost rl_i penalizes points far from the bottom center of the image with the equation 6.5.

$$rl_i = s(\sqrt{(v - c_y)^2 + (u - c_x)^2 / q_u}) \tag{6.5}$$

In the equation c_x and c_y are coordinates of a constant point in the center-bottom of the image. The feature coordinates are u and v and q_u is a parameter used to widen the ground area.

6.2.2 Tracking Quality Cost

Tracking quality is measured by calculating SAD of pixel intensity difference over two image areas displaced one pixel along the epipolar line. Different depths result in different point projection along the epipolar line.

The first image area is a box centered in the detected feature location. The estimated relative camera pose $\mathbf{T}_{k+1,k}$ is used to find the epipolar line, and calculate a 1 pixel displacement along the line (the second box center). The position of the second box can be calculated applying a variation to the feature depth, projecting the feature and normalizing the vector from the first box position to the new projection to length 1 pixel.

Both positions present sub-pixel accuracy and interpolation is used to find pixel intensities inside the box.

6.2.3 Feature Height Cost

The last quality measure is the feature height with respect to the estimated constant ground.

Both static features not in the ground (trees, parked cars or guard-rails), will present a position far from the estimated ground plane.

Non-static features present a high error in depth triangulation. Those features are also penalized by the feature height estimation.

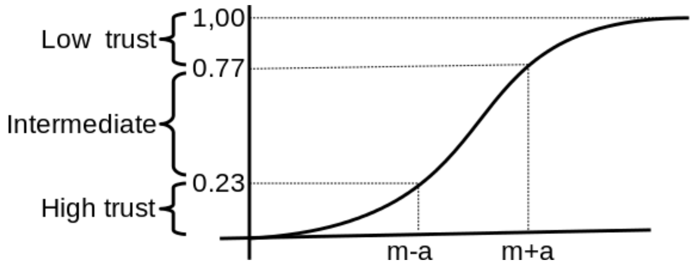
6.2.4 Sigmoid Scaling

A smooth transition function of the form of equation 6.6 is used to scale and smooth the calculated costs. Costs are bounded in the $[0, 1]$ interval and a smooth threshold is used to distinguish good ground point from low confidence ones.

$$l = \frac{1}{1 + e^{\frac{rl-m}{a}}} \tag{6.6}$$

Parameters m the mean and a the aperture, are used to calibrate the sigmoid scaling as shown in figure 6.5.

Figure 6.5: Sigmoid function



source: from author

The sigmoid function parameters depend on application specific conditions, such as image illumination, camera intrinsic parameters, camera position with respect to the ground among others. In the test database, for half of the sequences the GPS recorded camera position is given, to be used for system calibration. Those sequences were used to find the sigmoid parameters. In this work the values for m and a for each of the costs were: $m_l = 80$, $a_l = 40$, $m_g = 8$, $a_g = 4$, $m_s = 0.5$, $a_s = 0.25$. The used weights were $w_l = 2$, $w_s = 0.5$ and $w_g = 0.5$.

6.3 Chapter Conclusion

This chapter presented a scale recovery strategy for camera equipped wheeled robots. The algorithm uses points in the ground and measured camera height to estimate absolute scene scale.

The estimated scale is corrected by a translation based pitch angle recovery algorithm. The method uses the forward and vertical translation ratio to continuously track the estimated horizon position in the image.

The ground point selection algorithm combines three different criteria to calculate a selection cost. The scores are based on point position in the image, graphical tracking quality and estimated point height. Calculated scores are smoothed and scaled with a sigmoid function and added. The candidate with lowest cost score is selected.

Implementation and results of the developed algorithms are presented in chapter 9.

7 FAST TRIANGULATION

"A man only becomes wise when he begins to calculate the approximate depth of his ignorance."

Gian Carlo Menot

As explained in section 2.3.3, triangulation is the algorithm used to find the 3D position of a point from its projection in two images, for which the camera relative position is given. It is the method used to recover the sparse scene depth. In this chapter the low complexity method proposed in (PEREIRA et al., 2018) is explained.

In the absence of errors exact triangulation can be calculated by finding the 3D intersection point of the lines that connect the camera centers to the image feature position. Due to errors in feature tracking and camera pose estimation, the lines from the camera center through feature positions usually do not meet in space. In those cases inexact triangulation must be used.

The optimal method from section 2.3.3, proposed by Hartley and Sturm in "Triangulation" (HARTLEY; STURM, 1997), finds the 3D point that minimizes the distance of projected and viewed feature in both images.

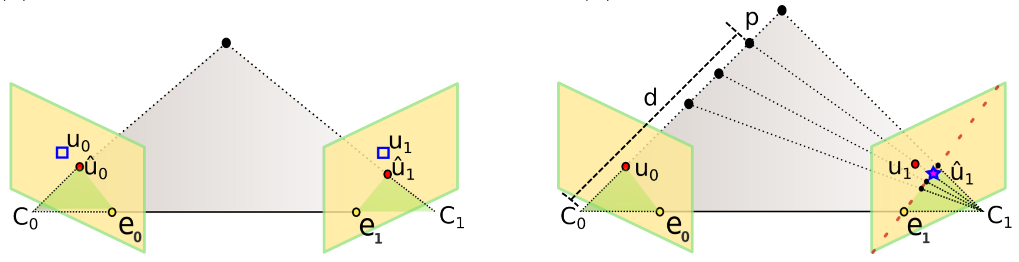
7.1 The Proposed Fast Triangulation

The proposed fast triangulation method is based on Euclidean distance minimization of projection error in a single image. The method is shown in figure 7.1. In the figure two cameras at positions $\mathbf{C}_0$ and $\mathbf{C}_1$ locate a 3D point $\mathbf{p}$ in positions $\mathbf{u}_0$ and $\mathbf{u}_1$. The point depth d is the distance from the point to the center of camera 0. The epipolar line is marked as the dashed line on image plane 1.

Inexact triangulation methods minimize the combined distance that both viewed points must be moved along the image planes to ensure that the rays of light meet on space. The method described in this chapter, restricts the position of the point to be along the line that passes through the feature position viewed by camera 0 $\mathbf{u}_0$, and finds the depth d that minimizes the distance from the point viewed by camera 1 $\mathbf{u}_1$ to the epipolar line as shown in equation 7.1. The ' $\prime$ ' (prime) symbol

Figure 7.1: Triangulation Methods

(a) Standard Two-View Triangulation (b) Proposed Fast Triangulation



source: Author

indicates origin in the camera center and unit focal distance as mentioned in section 1.4.

$$\arg\min_{d} g(d) = [(\hat{u\prime}_1(d) - u\prime_1)^2 + (\hat{v\prime}_1(d) - v\prime_1)^2] \tag{7.1}$$

The known rigid body transformation matrix $\mathbf{T} = \mathbf{R}|\mathbf{t}$, changes the reference frame of a 3D point from camera 0 (C_0) to camera 1 (C_1). The point viewed in camera 0 will be transported by T to $(\hat{x}_1, \hat{y}_1, \hat{z}_1)^T$ as:

$$\begin{bmatrix} \hat{x}_1 \\ \hat{y}_1 \\ \hat{z}_1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} \hat{x}_0 \\ \hat{y}_0 \\ \hat{z}_0 \\ 1 \end{bmatrix} \tag{7.2}$$

where:

$$\mathbf{T} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \tag{7.3}$$

And from the pinhole camera model (HARTLEY; ZISSERMAN, 2004):

$$\hat{x}_0 = u\prime_0 d; \ \hat{y}_0 = v\prime_0 d, \ \hat{z}_0 = d \tag{7.4}$$

Isolating $\hat{u\prime}_1$ and $\hat{v\prime}_1$:

$$\hat{u\prime}_1 = \frac{\hat{x}_1}{\hat{z}_1} \tag{7.5}$$

$$\hat{v\prime}_1 = \frac{\hat{y}_1}{\hat{z}_1} \tag{7.6}$$

Equations 7.7 and 7.8 can be rewritten using equations [7.2-7.4]:

$$\hat{u\prime}_1 = \frac{r_{00}u\prime_0 d + r_{01}v\prime_0 d + r_{02}d + t_x}{r_{20}u\prime_0 d + r_{21}v\prime_0 d + r_{22}d + t_z} \tag{7.7}$$

$$\hat{v\prime}_1 = \frac{r_{10}u\prime_0 d + r_{11}v\prime_0 d + r_{12}d + t_y}{r_{20}u\prime_0 d + r_{21}v\prime_0 d + r_{22}d + t_z} \tag{7.8}$$

For simplicity of notation the rotation part of matrix **T** multiplied by the viewed point in homogeneous coordinates, is introduced as the known constant vector **k**:

$$\begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} = \begin{bmatrix} r_{00}\,{}_0u + r_{01}\,{}_0v + r_{02} \\ r_{10}\,{}_0u + r_{11}\,{}_0v + r_{12} \\ r_{20}\,{}_0u + r_{21}\,{}_0v + r_{22} \end{bmatrix} \tag{7.9}$$

And Equations 7.7 and (7.8) may be rewritten as:

$$\hat{u\prime}_1 = \frac{k_x d + t_x}{k_z d + t_z} \tag{7.10}$$

$$\hat{v\prime}_1 = \frac{k_y d + t_y}{k_z d + t_z} \tag{7.11}$$

There is a restriction that $k_z\,{}_0z + t_{03}$ can not be zero, and actually it has to be positive, otherwise the point would not be visible on camera 1 (would be behind it).

Equation (7.1) can be solved by replacing the results of (7.10) and (7.11) on (7.1) and setting the derivative with respect to ${}_0z$ to zero. Isolating ${}_0z$ results in:

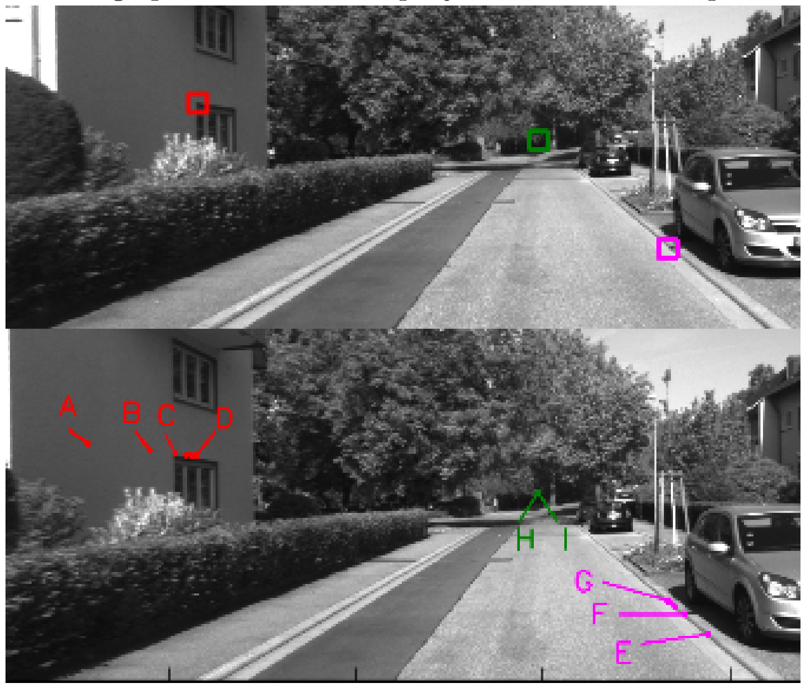
$$d = \frac{(t_x - u\prime_1 t_z)(k_z t_x - k_x t_z) + (t_y - v\prime_1 t_z)(k_z t_y - k_y t_z)}{(k_x - k_z u\prime_1)(k_x t_z - k_z t_x) + (k_y - K_z v\prime_1)(k_y t_z - k_z t_y)} \tag{7.12}$$

7.2 Discussion

While in theory, optimal methods provide a better approximation of the real distance from point to camera, in practice both methods yield very similar results as will be shown in the next section. Equation 7.12 requires only 39 floating point operations to estimate depth (22 additions, 16 multiplications and one division), a negligible computational cost when compared to the standard optimal method and less then half the complexity of other published solutions such as (KANATANI; SUGAYA; NIITSUMA, 2008; LINDSTROM, 2010).

For pure camera rotations (translation vector is zero), there is no parallax, and no depth information can be extracted. Similarly, when the epipole lies inside the image region (when the camera moves along the optical axis for instance), points whose image coordinates are close to the epipoles present low parallax. On those cases the numerator and denominator of equation (7.12) will become very small and depth estimation accuracy decreases. Another critical situation happens when the point is close to the infinity, when only the denominator of equation (7.12) tends to zero, or is found to be beyond the point at infinity (due to measurement errors), and triangulation will find negative depths. Infinity problems can be avoided with the use of inverse depths (flipping numerator and denominator of equation 7.12).

Figure 7.2: Triangulation example (Top image shows three manually selected points, and the bottom image present difference projection based on depth variation)



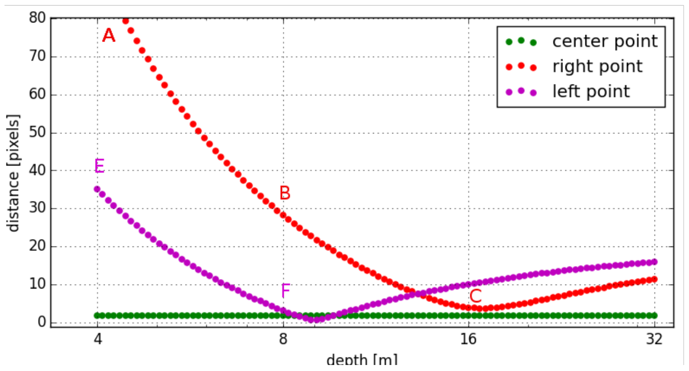
source: (GEIGER; LENZ; URTASUN, 2012a) modified by the author

To illustrate those points, Fig. 7.2 shows a real example of triangulation. The two first images from sequence 3 of the KITTI dataset (GEIGER; LENZ; URTASUN, 2012a) for which the absolute camera displacement is given by the database developers, are displayed. In this case the camera moved forward approximately 96 centimeters. Three traceable points have been marked on the first image and were projected on the second (using equations [7.2 - 7.8] with arbitrary depths $d = (4, 8, 16...512)$ meters.

Points (A, B, C, D) on the lower image, are the projected position of the upper left window corner (marked with a red box in the top image) if the distance from the feature to the camera were respectively (4, 8, 16, 512) meters. If the distance from the window marked with the left most square in the top of figure 7.2 from the camera were 4 meters, the window's top left corner in the bottom image would be on the position marked by point 'A'. Noticing that the window's in the bottom image corner is actually near point 'C', the true distance from that point to the camera should be around 16 meters. Point 'D' is close to the projection at infinity as large increases in depth will have little impact on projected position.

The same is true for points (E, F, G) for depths (4, 8, 512) meters and points (I, H) for d (4, 512) meters. For the feature in the asphalt, point F is closer to the searched point, while the center's point depth can not be calculated by triangulation as different distances from the camera generate the same projection.

Figure 7.3: Distance from the tracked points on fig 7.2 to their reprojections for various depths



source: author

Figure 7.3 shows the distance in pixels from the same projected points to the tracked feature positions, for different depths. The minimum of those curves is determined by Equation 7.12. If looked closely we can see that the red curve does not reach zero, as a small re-projection error is always present, as the tracked point does not rest exactly on the EL.

Table 7.1: Simulation test configuration

Parameter	value
Resolution	1024^2
Focal length	512
Number of feats	1000
Angular error σ_a [deg]	$[0.005, 0.01, 0.02, 0.04, 0.08]$
Translation error σ_t [dist u]	$[0.0025, 0.005, 0.01, 0.02, 0.04]$
graphical error σ_p [pixel]	$[0.1, 0.2, 0.4, 0.8, 1.6]$

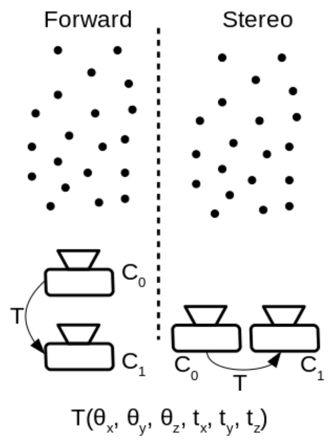
7.3 Results on Synthetic Data

This section presents synthetic data results for depth calculation of the proposed algorithm, when compared to the theoretical optimal method proposed by Hartley (HARTLEY; STURM, 1997) in two different camera pose configurations.

For this test two virtual cameras are created, a set of 3D coordinates are generated and projections on both image planes are calculated. No shapes or colors are generated, only 3D coordinates and 2D image positions. Then noise is added to camera poses or 2D feature positions and the original depth is compared to the triangulation output.

In the first case, cameras are placed side by side as in a stereo rig, while the in the second, both cameras are aligned along the optical axis (second camera moved forward with respect to the first) as shown in fig 7.4.

Figure 7.4: Test case scenarios: stereo and forward camera configuration

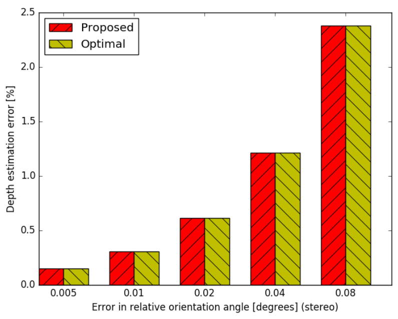


source: author

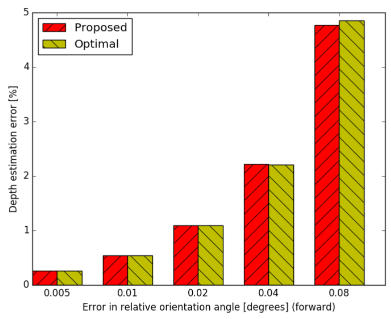
In order to test the estimation robustness to errors in camera relative orientation, camera relative translation and graphical tracking, Gaussian error was added to those parameters, and the error in depth estimation is compared.

Figure 7.5: Triangulation depth error comparison

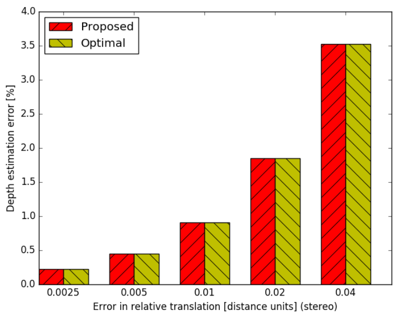
(a) Stereo case: depth error for angular noise on cam relative pose



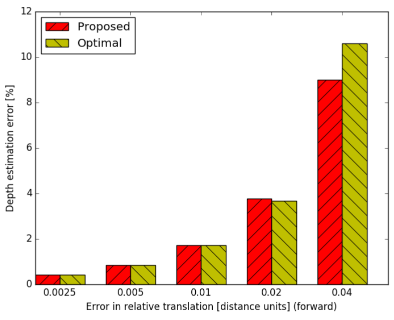
(b) Forward case: depth error for angular noise on cam relative pose



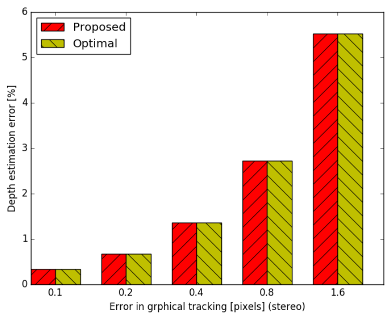
(c) Stereo case: depth error for transl. noise on cam relative pose



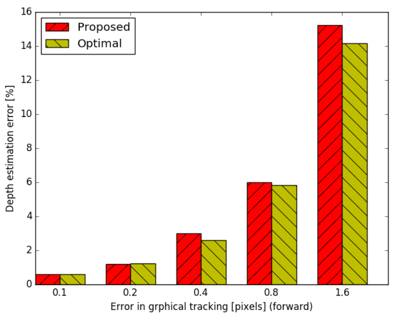
(d) Forward case: depth error for transl. noise on cam relative pose



(e) Stereo case: depth error for tracking noise in pixel location



(f) Forward case: depth error for tracking noise in pixel location



source: author

Camera relative pose is composed of six parameters, three orientation angles σ_a and three translation displacements σ_t as shown in equations 7.13 and 7.14, one for each axis. Gaussian noise was added to each orientation angle with zero mean and standard deviation degrees. Error in camera relative translation used standard deviation of units of distance.

$$\sigma_a = [0.005, 0.01, 0.02, 0.04, 0.08] \tag{7.13}$$

$$\sigma_t = [0.0025, 0.005, 0.01, 0.02, 0.04] \tag{7.14}$$

The third main source of error in triangulation is inaccuracy in feature tracking. To test robustness for image processing errors Gaussian noise was added to the located pixel coordinates $\sigma_p = [0.1, 0.2, 0.4, 0.8, 1.6]$ pixels in both images.

The test used a virtual camera with resolution of 1024^2 pixels and focal length of 512 pixels. In each iteration, a random point set of 1000 points were generated in front of the cameras, with distances ranging uniformly from 5 to 25 units of distance. Distance between cameras is one unit of distance.

Table 7.1 resumes the test set parameters. The test was repeated 1000 times. Average depth error is plotted in the charts of figure 7.5. On the charts the yellow bars show the depth error of the optimal method and the red bars the errors of the proposed method for the different noise levels in different parameters. In the absence of errors both methods yield results that are equal up to the sixth decimal digit.

It is possible to note that for the stereo case (figs 7.5 a, c and e), the proposed and optimal methods have equivalent depth estimations, and in fact the average error in all stereo cases are in accordance in at least up to the fifth decimal digit. Those results show that for stereo applications that aim at finding feature distance from the camera, the proposed method can replace the optimal method with virtually no loss of precision.

The forward case is much less stable as can be seen by the error scale of figures 7.5 b, d and f. The proposed method shows larger discrepancies from the optimal method for forward motion in most cases, the error increase however is not significantly worse. In some cases, probably due to stability issues when finding polynomial roots, the proposed method slightly outperforms the optimal triangulation method.

7.4 Chapter Conclusion

This chapter presented a simple light-weight triangulation method proposed in (PEREIRA et al., 2018) that is suitable for practical applications in which the system was tested.

The chapters starts with the development of the fast triangulation methods, followed by a comparison of the estimated depth to the optimal method developed in (HARTLEY; STURM, 1997).

For the error levels of camera pose estimation seen in the performed tests (10^{-3} degrees per meter and translation errors in the order of 1%) both the proposed and the optimal methods present coincident depth estimations. The proposed triangulation method's utility is addressed in section 8.2. Results of the fast triangulation with real data are analyzed in chapter 9.

8 CAMERA POSE ESTIMATION

"There is nothing insignificant in the world. It all depends on the point of view."

Johann Wolfgang von Goethe

This chapter considers the estimation of camera relative translation and rotation. Section 2.3.4 mentions the Perspective-n-Point (PnP) problem, which is the estimation of camera pose based on the 3D positions of points and their projections on the image plane.

In this application the scene structure, and therefore the 3D position of points, is not known and must be estimated along with the camera pose. The method that alternates triangulation and PnP is called Resection-Intersection (RI).

In RI an initial camera pose is used for triangulation and initial point positions are estimated. The calculated point positions are then used to update the camera pose. And the algorithm repeats the process until a stop criterion is met.

As mentioned in section 1.4, camera pose is represented by either a vector $\mathbf{w}$ containing three rotations and three translations relative to the last estimated camera pose, or by the 3×4 matrix $\mathbf{T}$. The Euler-Rodrigues formula (BANCHOFF, 1990) can be used to convert between the vector $\mathbf{w}$ and the matrix $\mathbf{T}$. In the next sections two methods based on triangulation followed by PnP are addressed.

8.1 Cyclic Three-View Pose Estimation: CYC_MVO

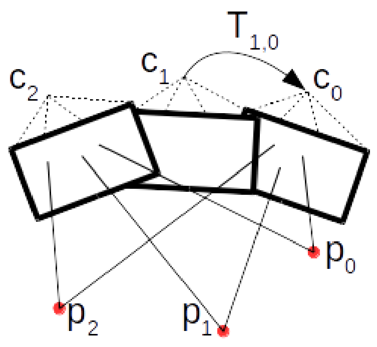
Regular RI suffers from slow convergence and local minima problems as analyzed in (TRIGGS et al., 2000). An alternative to two-view RI is to use a sequence of frames and perform triangulation and camera pose estimation in pairs of frames inside the window. The work (SONG; CHANDRAKER; GUEST, 2013; SONG; CHANDRAKER; GUEST, 2016) uses groups of 10 frames for pose estimation.

The method presented in this section CYC_MVO was proposed in (PEREIRA et al., 2017b). It combines the use of feature correspondences in a small set of three frames and the ideas from chapter 4 to estimate camera pose.

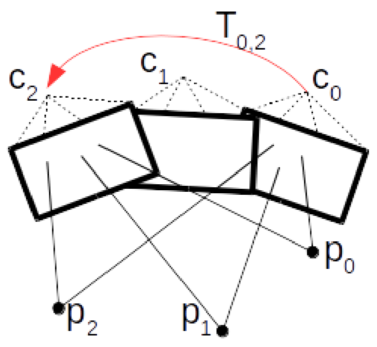
The module receives as inputs the (u, v) image coordinates of corresponding matched image regions, and the last camera displacement to calculate the new camera displacement as shown in figure 8.1.

Figure 8.1: Cyclic Estimation

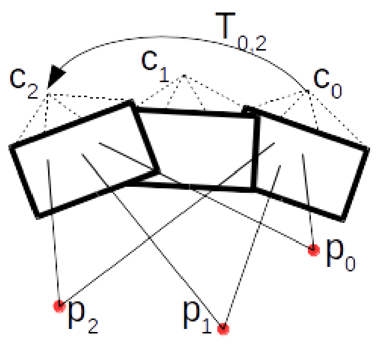
(a) Estimate points using $\mathbf{T}_{1,0}$



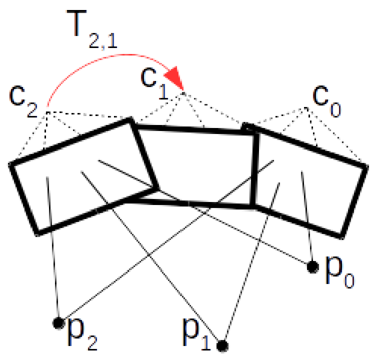
(b) Estimate $\mathbf{T}_{0,2}$ using points



(c) Estimate points using $\mathbf{T}_{0,2}$



(b) Estimate $\mathbf{T}_{2,1}$ using points



source: author

In figures 8.1 three cameras with centers in (c_0, c_1, c_2) view three points (p_0, p_1, p_2). The setup is equivalent to a single camera in three different moments. The parameter being adjusted is marked in red in each image.

In the first moment (a) the last relative camera pose $T_{1,0}$ is used to triangulate the points 3D positions. In the second moment (b) the calculated points are used to estimate the relative pose from camera 0 to camera 2. Another triangulation step is performed in image (c) to update the point's positions. And the relative pose from camera 2 to camera 1 is finally calculated in (d).

The algorithm then can continue to alternate triangulation and PnP in a cyclic approach. If it is assume that (c_0, c_1, c_2) represent the center of the same forward-moving camera at moments $(0, 1, 2)$, where c_2 is closer to the features, then the algorithm behaves as a backward moving estimator inside a three images batch.

In the algorithm 1 features in pairs of images are used in a cyclic way, that is, first correspondences in images $k, k-1$ are used, then $k-1, k+1$ and finally $k+1, k$.

Algorithm 1 Cyclic BA algorithm

```
function CYC_BA($T_{k,k-1}, U_{k-1,k,k+1}$)
    num_cyc ← 0,
    rep ← max_rep
    threshold ← init_threshold
    $T_{k-1,k} \leftarrow Invert(T_{k,k-1})$
    while rep>threshold & num_cyc<max_cyc do
        $d_{k-1} \leftarrow triangulate(T_{k-1,k}, U_k, U_{k-1})$
        $T_{k+1,k-1} \leftarrow SolvePnP(U_{k-1}, d_{k-1}, U_{k+1})$
        $d_{k+1} \leftarrow triangulate(T_{k+1,k-1}, U_{k-1}, U_{k+1})$
        $T_{k,k+1} \leftarrow SolvePnP(U_{k+1}, d_{k+1}, U_k)$
        $d_k \leftarrow triangulate(T_{k,k+1}, U_{k+1}, U_k)$
        $T_{k-1,k} \leftarrow SolvePnP(U_k, d_k, U_{k-1})$
        $rep \leftarrow GetReproj(U_{k-1}, U_k, d_k, T_{k-1,k})$
        num_cyc, threshold ← num_cyc+1, threshold + $\delta$
    end while
    return $Invert(T_{k,k+1})$
end function
```

source: author

The triangulation method of section 2.3.3 is used to calculate depth. SolvePnP uses Levenberg-Marquardt non-linear optimization in a RANSAC approach to estimate camera displacement. At each iteration a subset of features is randomly selected and camera pose is estimated based on the subset. The median error is calculated for all features and the solution with the lowest error is elected.

Small errors in camera pose estimation, cause larger re-projection errors in close features due to large displacement along an incorrect epipolar line, and RANSAC can select only far points which are less sensitive to camera translation. To avoid this situation, the system divides the points in two sets with the same number of points, based on their distance to the camera. The median is calculated for close points and far points and the average of medians is used as the fit criterion for RANSAC.

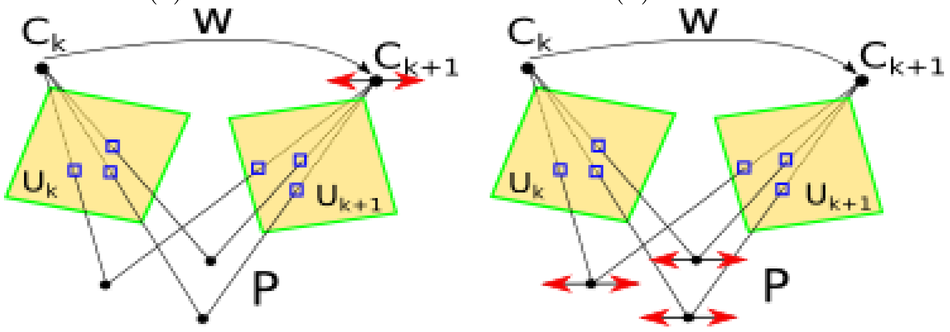
When system is initialized the first relative camera pose estimation must be performed without previous knowledge of motion. Three initial guesses of forward motion are made, with three different speeds. The solution with lowest reprojection error is selected and scale of camera translation is set to the first recovered depth to ground estimation, with the method outlined in chapter 6. Results of camera pose estimation will be performed in the chapter 9.

8.2 Two-View Resection-Intersection: RI_MVO

Classical Bundle-Adjustment (TRIGGS et al., 2000) attempts to simultaneously optimize the 3D feature position and camera poses as a single sparse optimization problem. The Bundle-Adjustment variation that alternates optimization of camera pose and point position is called Resection-Intersection (RI) (TRIGGS et al., 2000) as illustrated in figure 8.2. In the figure, C_k represents the camera position and orientation at time k, $\mathbf{w}$ is the vector containing the camera relative translation and rotation from time k to time $k+1$, $\mathbf{P}$ is a set of 3D points and $\mathbf{U}$ is the set of features on the image planes k and $k+1$.

Figure 8.2: Resection Intersection Cycle

(a) Resection (b) -Intersection



source: author

This section addresses resection-intersection technique for two-view camera pose estimation introduced in (PEREIRA et al., 2018) here referred as RI_MVO. The key algorithmic change of the proposal is to account for the variation of the 3D position of points when calculating the incremental adjustments in camera pose.

The technique makes an intense use of triangulation (6 operations for each camera pose step update), making triangulation a critical element of camera pose estimation. In order to reduce the cost of feature 3D position estimation, the fast projective invariant, inexact triangulation method from chapter 7 is used.

8.2.1 Gauss-Newton Applied to Projection Minimization

This subsection describes how nonlinear optimization, in particular the Gauss-Newton (GN) algorithm (BJORCK, 1996), is used for projection error minimization. GN is an iterative optimization algorithm that adjusts the parameters $\mathbf{w}$ in order to minimize the residual function f as shown in equation 8.1. In each step, the algorithm calculates an update increment according to equation 8.2. In the equation, $\mathbf{r}$ is the residual to be minimized, $\mathbf{J}$ is the Jacobian matrix of partial derivatives of $\mathbf{r}$ with respect to each parameter of $\mathbf{w}$ and s is the iteration step.

$$\arg\min_{\mathbf{w}} f(\mathbf{w}) = \sum_{j=0}^{n-1} r_j^2(\mathbf{w}) \tag{8.1}$$

$$\mathbf{w}^{(s+1)} = \mathbf{w}^s + \Delta\mathbf{w} = \mathbf{w}^s - (\mathbf{J}^T\mathbf{J})^{-1}\,\mathbf{J}^T\mathbf{r}(\mathbf{w}) \tag{8.2}$$

Although robust residual functions have been proposed (TRIGGS et al., 2000), in this work the standard projection difference is used as error. That is $\mathbf{r} = (\Delta_0 u, \Delta_0 v, \Delta_1 u, ...\Delta_{(n-1)} v)^T$ where $\Delta_i u$ and $\Delta_i v$ are the horizontal and vertical differences from projection to viewed position for the i^{th} feature: $\Delta_i u =_i u -_i \hat{u}$. For a set of n features $\mathbf{r}$ is a column vector with $2n$ elements, and $\mathbf{J}$ is a $2n \times 6$ matrix where each column $i \in [0, 5]$ is the partial derivative of $\mathbf{r}$ with respect to the i^{th} element of the parameter vector $\mathbf{w}$.

8.2.2 Residual and Partial Derivatives Calculation

Algorithm 2 presents the calculation of the residuals and Jacobian matrix in each iteration. In the algorithm the residuals are calculated in three steps: (1) triangulation to update the 3D position of points, (2) projection of triangulated positions onto the image plane, and (3) difference of projection from the image localized feature. No prior information about the point 3D position is used. For simplicity of notation the constant camera intrinsic parameters matrix $\mathbf{K}$ was suppressed. The vectorization function *vec* transforms an $n \times 2$ matrix in a vector of size $2n$.

Matrix $\mathbf{J}$ is calculated numerically, a small variation is applied to each component of vector $\mathbf{w}_k$ to approximate the partial derivative $\partial\hat{\mathbf{U}}/\partial\mathbf{w}_k[i]$. In the algorithm $\mathbf{w}_{\Delta,i}$ is a vector whose i^{th} element is a small value in order of 10^{-6} and every other element is zero. Functions *triangulate* and *project* are the fast triangulation and regular projection of a point into an image.

To calculate one update step seven triangulation and projection operations are carried out, one for the residual and one for each of the six parameters of the camera pose. A modification that reduces one triangulation per update cycle and

Algorithm 2 Residuals and Jacobian Calculation

```
1: function GET_J_R($\mathbf{w}_k, \mathbf{U}_k, \mathbf{U}_{k+1}$)
2:     /*residuals: triangulation, projection and difference*/
3:     $\mathbf{P} = triangulate(\mathbf{w}_k, \mathbf{U}_k, \mathbf{U}_{k+1})$
4:     $\hat{\mathbf{U}}_{k+1} = project(\mathbf{w}_k, \mathbf{P})$
5:     $\mathbf{r} = vec(\hat{\mathbf{U}}_{k+1} - \mathbf{U}_{k+1})$
6:
7:     /*for each Jac col: calc ∂U/∂w[i]*/
8:     while $i \in [0, 5]$ do
9:         $\mathbf{w}_{k,i} = \mathbf{w}_k + \mathbf{w}_{\Delta,i}$
10:        $\mathbf{P}_{\Delta,i} = triangulate(\mathbf{w}_{k,i}, \mathbf{U}_k, \mathbf{U}_{k+1})$
11:        $\hat{\mathbf{U}}_{k+1,i} = project(\mathbf{w}_{k,i}, \mathbf{P}\Delta, i)$
12:        $\mathbf{J}_i = vec(\hat{\mathbf{U}}_{k+1,i} - \hat{\mathbf{U}}_{k+1})/(\mathbf{w}[i]_{\Delta,i})$
13:    end while
14:    return $(\mathbf{r}, \mathbf{J})$
15: end function
```

the handling of outliers in the feature tracking are further described in section 8.2.3.

8.2.3 Implementation Issues

The proposed resection-intersection algorithm uses feature correspondence to estimate each new relative camera pose. Non-static scene objects and tracking failure generate bad feature correspondences that are traditionally dealt with a RANSAC scheme. In this implementation a sequential backward selection approach was used. At each iteration the projection error for all features is calculated and the features with largest errors are removed from the set of features used for camera pose estimation. The update loop stops when the median projection error of the features used for estimation falls below a threshold.

Estimating both camera pose and points positions represents an undetermined problem since different scales of translation movement and distances from points to camera have exactly the same projection characteristics. There is no guarantee that the scale of the movement will be maintained when a new camera pose is estimated. In the present application, the dominant translation direction is along the z axis (forward movement), and abrupt changes in car speed during the 0.1 seconds inter-frame interval are not expected. To remove the extra degree of freedom and maintain a scale close to the previously estimated scale, the last calculated relative camera pose $\mathbf{w}_{k-1}$ is used as initial pose estimation, and the translation on

z axis t_z is kept constant. The update step is calculated for the first five elements of the relative pose, the last column of the Jacobian is not calculated, and six triangulations and projections are performed for each update cycle. Naturally the car speed is not constant, but varies smoothly. The true translation is adjusted based on the absolute scale extraction from chapter 6.

8.3 Chapter Conclusion

This chapter discussed the camera pose estimation from feature correspondences. Two algorithms that minimize the difference from 3D projection to tracked features are considered.

The Cyclic Three-View Pose Estimation methods use a set of three frames and emulate backward movement within the frame set. RANSAC is used to detect and remove outliers.

The Two-View Resection-Intersection uses only two frames and a modified Jacobian calculation to improve RI convergence rate. The method uses several triangulations in each camera pose update step, in a Gauss-Newton non-linear optimization loop.

The convergence rate allowed sequential backward selection to be used in stead of RANSAC for outlier removal. Implementation and analysis of the methods are carried out in chapter 9.

9 MVO SYSTEM IMPLEMENTATION AND RESULTS

"I love fools' experiments. I am always making them."

Charles Darwin

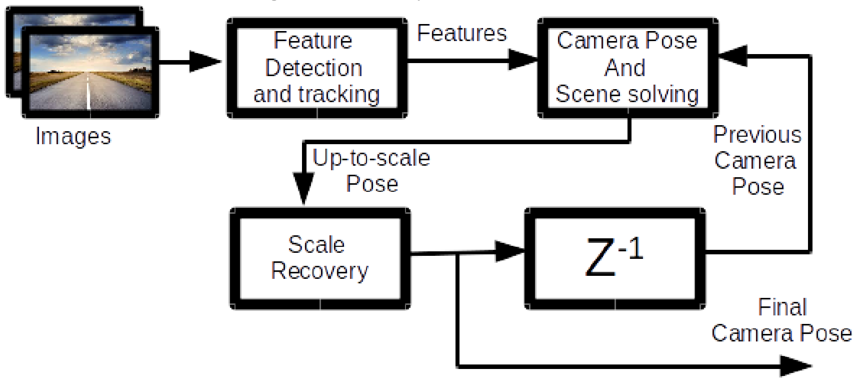
In order to verify the conceived algorithms performances, full MVO systems were assembled. A popular database was then used to measure the system output and enable comparison to other designs. This chapter describes the developed MVO solutions and presents the achieved results.

Two systems using a similar architecture have been developed during the course of this work. The first, proposed in (PEREIRA et al., 2017b) used three image frames and the cyclic estimator from section 8.1, and the second using two images as input and the estimator from section 8.2. The feature tracker and scale estimator present minor changes in both designs. Next section describes the common architecture.

9.1 Overall System Architecture

As stated in section 2.1, sparse visual navigation pipelines follow a resembling pattern composed of feature generation, geometric pose-scene estimation and scale recovery. Figure 9.1 shows the simplified system architecture used in this work.

Figure 9.1: System architecture



Source: Author

The Feature Detection and Tracking module receives as input the images from the camera and generates feature correspondences. The Camera Pose and Scene Solving module uses the tracked features and previously relative camera pose to calculate the current camera rotation and translation. The Scale Recovery module extracts and adjusts movement translation scale. The Z^{-1} symbol represents a memory used to provide a one frame delay in camera pose.

9.2 KITTI Dataset

Although images of reconstructed scenes and estimated paths give a subjective idea of a 3D algorithm precision, the use of a public dataset provides an objective way of measuring and comparing achieved results.

In this work the KITTI Dataset (GEIGER; LENZ; URTASUN, 2012a) was chosen as target application. The dataset for VO comprises a collection of 22 trajectories where a car equipped with cameras and a high precision GPS drives around cities and in highways. Trajectories are recorded as image sequences at 10 frames per second, and for each image the GPS corrected camera position and orientation is saved as ground truth.

For 11 of the trajectories the GPS corrected ground truth (camera position and orientation) is supplied, to be used for system calibration. For the other 11 trajectories the ground truth is not supplied. Trajectories are calculated using the provided images, and estimated results can be uploaded to the database developer's site (GEIGER; LENZ; URTASUN, 2012b). Rotation and translation errors are published and can be used for comparison to other solutions.

In the trajectories the car speed varies from 0 to around 90 Km/h, pedestrians, bicycles and moving cars, as well as steep turns and fast moving features make this a challenging application. The number of frames for the first 11 sequences is shown in table 9.1

Table 9.1: Trajectories Containing Ground Truth

sequence	00	01	02	03	04	05	06	07	08	09	10
num frames	4540	1100	4660	800	270	2760	1100	1100	4070	1590	1200

The metrics used to evaluate the system, proposed by the database developers are rotation error in degrees per meter and translation errors as a total displacement percentage. Trajectories are split in segments of 100 to 800 meters, and errors are averaged over all trajectory segments.

9.3 System Calibration

In the course of this work, several parameters have been adjusted, the most important ones will be discussed in this section. In the KITTI dataset (GEIGER; LENZ; URTASUN, 2012a) the GPS recorded camera pose is provided for the first 11 sequences for system calibration, and used to adjust the following parameters.

In feature detection and tracking the number of images in pyramid was set to 3, and windows sizes for feature detection is 3 by 3 pixels and for feature tracking 21 by 21 pixels. Decrease of pyramid levels led to poor tracking of features with high inter-frame displacement and increase led to increased number of outliers. Increase in window size of feature detector penalized system performance and showed no improvement in feature quality. Increase in tracker windows amplified tracking error.

Scale estimation needs parameter tuning in both ground feature selection and absolute scale estimation. Those are related to the ground zone, acceptable image gradient and tracking characteristics. Those parameters depend on the camera position and resolution. The camera recorded ground truths of sequences 0 to 11 were used to manually select good ground points and adjust those parameters.

In camera pose estimation step, the iterative estimation loop stops when the median projection error falls below a 0.1 pixels threshold. This value was calculated as the average median re-projection error after outlier removal in the test sequences.

The number of outliers removed in each backward selection iteration used in this work is 5% of the total number of features. A value of 2% showed similar accuracy with slower conversion rate (since more iterations are needed to remove outliers), and values higher then 10 % lead to unstable situations when larger proportion of outliers are tracked (moving cars and fast turns).

9.4 A Note on Speed

During the sequences contained in the used dataset, there are moments in which the car speed reaches zero (the car stops). If there is no translation, triangulation can not be performed and such situations must be avoided.

Most of the works that use the database employ a key-frame based approach. In those cases a sub-set of images presenting a minimum inter-frame translation are chosen as reference avoiding low speed problems.

Since the simple used approach uses two consecutive frames, a method to detect stop condition must be used. In this work if the average feature displacement is below 2 pixels the car is considered to be stopped and the frame is rejected.

High speed and steep curves, are also challenging, and usually present slightly higher average error than normal driving. For the speed interval $(0\ to\ 90km/h)$ and dataset frame rate $(10\,frames/s)$ the error is tolerable and no special algorithm was used for those cases.

9.5 Results

The algorithms were implemented in Python, using SciPy (JONES et al., 2001–) and OpenCV (BRADSKI, 2000) for the low level functions. This section presents the achieved results.

Figure 9.2 shows the estimated trajectories for the first 11 sequences in meters. The figures give a qualitative idea of estimation precision for the monocular odometry system. Since estimation results for both evaluated pose estimation algorithms are very similar, only the trajectories estimated with algorithm 8.1 is showed. The ground truth is solid green and estimations in dashed blue.

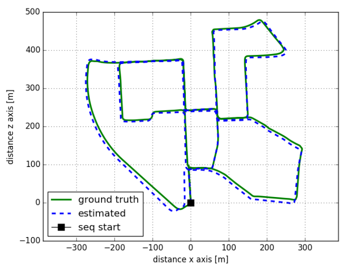
Although figure 9.2 show only horizontal vehicle displacement, the following results take into account the three axes for both camera rotation and translation.

Table 9.2 lists the results of both estimation methods for the first 11 routes, evaluated with the metrics proposed by the database developers. In the table translation error is measured as total percentage difference from estimated displacement to the ground truth in the three dimensions. Rotation errors are measured in degrees per traveled meter.

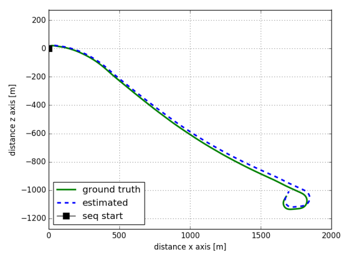
The table shows that the RI method reaches a slight improvement even using only two images for estimation. In both methods translation errors are in the order of 1% while rotation errors are close to 0.003 degrees per meter.

Figure 9.2: Estimated Trajectories and Ground Truth

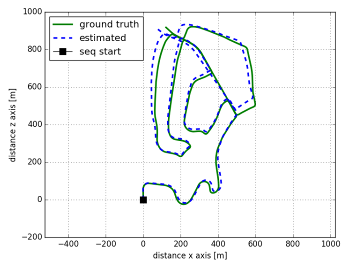
(a) traj 00



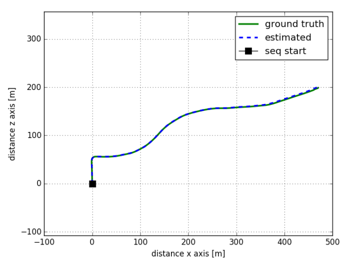
(b) traj 01



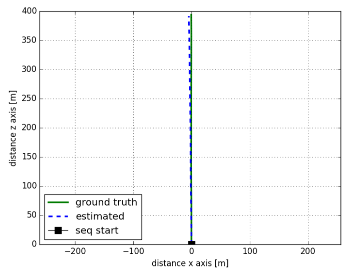
(c) traj 02



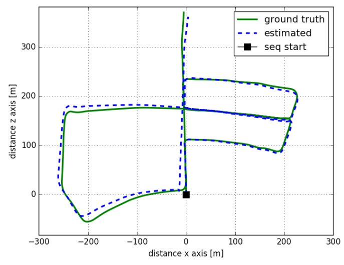
(d) traj 03



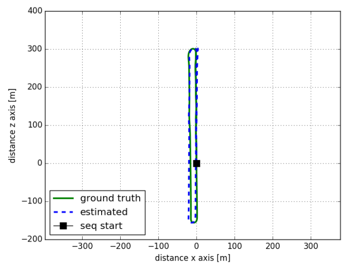
(e) traj 04



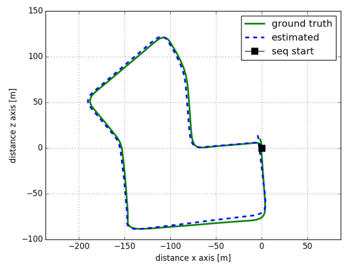
(f) traj 05



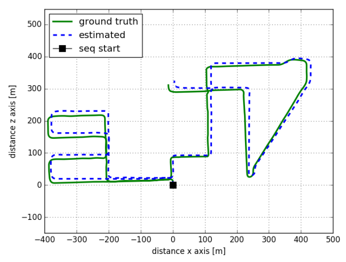
(g) traj 06



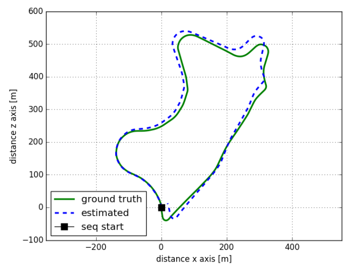
(h) traj 07



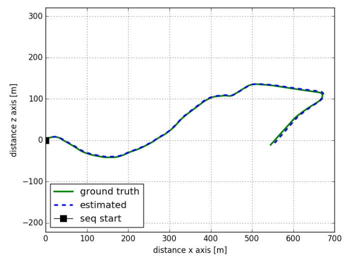
(i) traj 08



(j) traj 09



(k) traj 10



source: author

9.5.1 RI_MVO Convergence Rate

A problem of regular Resection-Intersection is the rate of optimization. In order to analyze the convergence rate, a standard RI version of the algorithm of section 8.2 was implemented suppressing the triangulation inside the loop (line 10 in algorithm 2). The image 73 of sequence 3 was used as an example, while the car performs a right turn (shown in figure 9.3), feature correspondences were tracked and outliers previously removed. The initial camera pose was set to pure forward

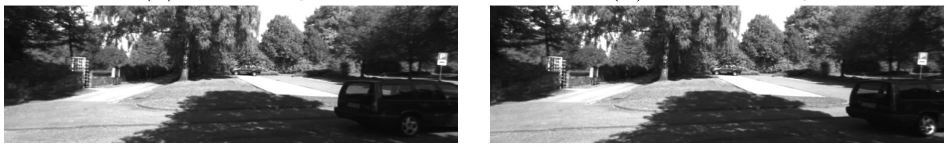
Table 9.2: Per trajectory translation error in [%] and rotation errors in [deg/m]

seq	translation [%]		rotation $[deg.10^{-3}/m]$	
	CYC_MVO	RI_MVO	CYC_MVO	RI_MVO
00	1.03	1.16	3.0	3.1
01	1.37	1.18	2.3	3.0
02	1.33	0.99	3.8	2.7
03	0.87	0.87	2.2	2.3
04	0.86	1.12	2.4	3.8
05	0.99	0.68	3.7	2.7
06	0.73	1.69	2.6	2.5
07	1.12	1.21	5.7	6.8
08	1.23	1.05	2.8	2.9
09	1.54	0.96	2.8	2.4
10	1.02	1.49	2.4	2.8
Avg	**1.23**	**1.02**	**2.8**	**2.8**

movement $\mathbf{w} = (0, 0, 0, 0, 0, 1)$ and fed to both the proposed and the standard RI.

Figure 9.3: Car Turn Example

(a) first image (b) second image



source: KITTI Dataset (GEIGER; LENZ; URTASUN, 2012a)

The proposed algorithm finds camera relative pose and point position that minimize the distance from located features to calculated projections. The median is regarded as a better error measure then the mean for being less affected by large errors due to tracking failure or non-static objects. Figure 9.4 shows the median projection error for both versions.

In the figure the vertical axis represents the median projection error in pixels and the horizontal axis is the iteration. The loop stops when the median errors falls below 0.1 pixels. In the proposed method the median projection error falls from 5.7 pixels to about 0.35 in the first iteration, and to about 0.05 in the second. In the standard version, 14 iterations were required for the projection error to drop to about 0.098 pixels.

The same test was repeated for all 4660 poses of sequence 02, the longest of the first 11 sequences. A histogram of the number of iterations needed for the median residual to reach 0.1 pixels is shown in figure 9.5. For the proposed method, up to three iterations were needed to reach the stop criterion in 93% of the cases.

Figure 9.4: Convergence Rate Comparison

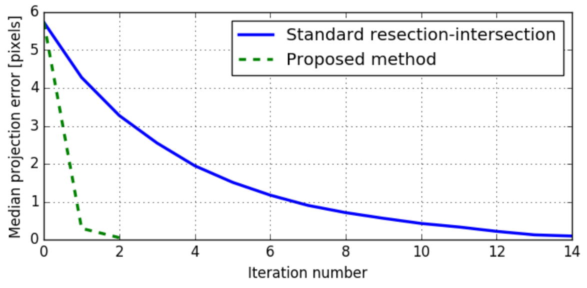
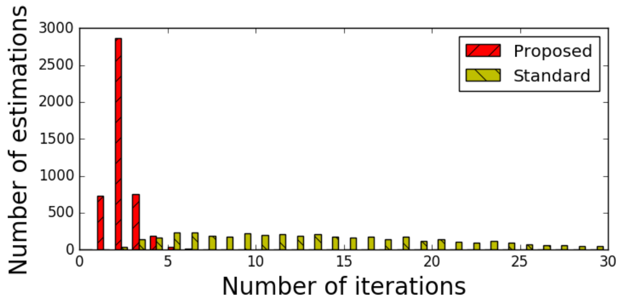


Figure 9.5: Number of Iterations Until Convergence Histogram

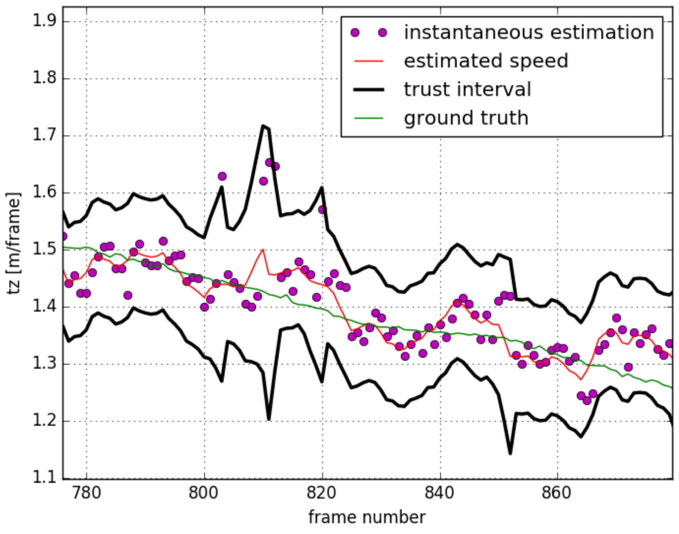


Source: Author

9.5.2 Scale Estimation

Scale recovery is independent of rotation estimation. Figure 9.6 show the scale recovery results. The green line is the true absolute per-frame z translation, and magenta points are instantaneous estimations. The red line shows the estimated speed.

Figure 9.6: Absolute Scale Estimation



Source: Author

In the figure an especially turbulent segment is presented. The car drives through irregular asphalt, and bounces up and down (pitch). The oscillations in the measurements are clear after frame 800.

In the figure very few bad points are selected. The multi-attribute ground point selector ensures that the magenta circles are close to the green line in figure 9.6.

9.5.3 Throughput

The single thread implementation was executed in an Intel i7-3770K CPU @ 3.5GHz. To meet real time requirement, the worst case allowable time per frame should be below 0.1 second for the used frame rate of 10 images per second. The average time per frame for the developed methods are shown in table 9.3.

Table 9.3: Seconds per Frame

Method	Seconds per Frame
CYC_MVO	0.17s
RI_MVO (opt triang)	0.12s
RI_MVO (fast triang)	0.07s

To highlight the importance of the fast triangulation of chapter 7 two versions of the RI systems are presented. One that uses the optimal method from OpenCV and the second that uses the optimal method by the proposed triangulation.

A profiling of the RI_MVO system is shown in table 9.4. In the table it is possible to note that image processing module is the most time consuming step, responsible for about half of the system computational load.

Table 9.4: Profiling Results RI_MVO

Operation	time [%]	avg time/frame [s]
Load Image	06.0	0.0045
Detection and tracking	50.2	0.0375
Resection Intersection	33.9	0.0250
Scale Estimation	08.9	0.0067
Other tasks	01.0	0.0007

9.6 Analysis and Comparison

This section provides a brief analysis of the results presented in the last section, and a comparison to other vehicular MVO solutions. The fast triangulation method is analyzed first, followed by camera estimation algorithm and a comparison of the MVO results to other implementations.

The proposed triangulation benefits from the projection invariance property, since the distance minimized is a projection distance and should not be confused with the midpoint method as explained in (HARTLEY; ZISSERMAN, 2004). Results from table 9.3 show a significant lower computational cost when compared to the optimal triangulation from (HARTLEY; STURM, 1997) and no increase in precision estimation was noticed. Triangulation methods like (KANATANI; SUGAYA; NIITSUMA, 2008) and (LINDSTROM, 2010) provide results similar to the optimal method, but still present a computational cost that is at least twice of the proposed triangulation.

Comparison of triangulation methods with synthetic data, in absence of errors shows the same result up to the sixth decimal digit. In the presence of errors (both in camera pose and feature tracking) the two methods generate very similar results. The main difference is that the optimal triangulation finds the feature 3D position that minimizes the error in both frames, while the proposed fast method considers the reference frame as being correct and finds a feature position that minimizes the projection error in the second frame. For that reason, the residuals (differences between calculated and traced feature positions) are larger in the second frame when using the fast triangulation method. Those residuals are the minimization objective in the proposed Gauss-Newton implementation.

In the in-depth revision of Bundle-Adjustment made in (TRIGGS et al., 2000) resection-intersection is classified as a low convergence-rate method. The assertion is verified by the results of figures 9.4 and 9.5. The analysis in (TRIGGS et al., 2000) is oriented towards large bundle-adjustment problems that concentrate on minimizing projection related residuals of several views simultaneously, while the proposed method uses only two images. When more than two views are used, triangulation can be solved either with algebraic solutions as in (KUKELOVA; PAJDLA; BUJNAK, 2013) or with non-linear optimization as in (HEDBORG; ROBINSON; FELSBERG, 2014) both with significantly higher computational cost.

A common problem with optimization based two-view pose estimation is convergence to local minima. Small errors in camera pose and feature position estimation may lead to stable configurations of scene structure and camera pose that do not correspond to the real situation, specially in the presence of outliers. Robustness to local minima is improved if more views are used in the estimation of either camera pose or scene structure.

Among the monocular solutions present on the KITTI benchmark, the works (PEREIRA et al., 2017a), (FANANI et al., 2016) and this work use only two views for pose estimation. The results from (PEREIRA et al., 2017a) are not comparable, since the images were processed in reversed order, and the system benefits from robustness from backward movement. In (FANANI et al., 2016) the intensity differences of pixels in a patch around the feature are minimized, instead of projection distance. In that solution coarse camera pose estimation and outlier removal are both carried out before the optimization process, which also increases the system robustness. The methods used in this work use either 3 frames or only the last two images for pose estimation. The last previously calculated camera pose is enough for optimization initialization given the constrained movement of a camera equipped vehicle.

The fast convergence rate of the proposed RI_MVO allows for bad correspondences to be sequentially removed, provided there are less outliers than inliers or that the outliers are not correlated. For the 22 sequences of the KITTI dataset, up to three outlier removal iterations were needed in 65% of the cases.

The ground truth (GPS detected camera poses) is not supplied for trajectories 11 to 21, camera pose estimations can be uploaded to the dataset website, and average results are published. The works presented in this thesis are labeled RI_MVO and FVO in the benchmark (GEIGER; LENZ; URTASUN, 2012b). Table 9.5 use information from the benchmark to compare the achieved estimation to other monocular published works.

From the comparison of table 9.5, it is possible to note that the achieved precision in both implementations are very close to each other, which is expected since both algorithms minimize the projection difference, and the feature tracking is the same. The RI method achieves lower execution time in the same hardware platform while using only two views, no tracking cross check and no RANSAC after initialization.

Table 9.5: Results Comparison for Trajectories 11-21

Work	Transl [%]	Rot [deg/m]	Runtime	Environment
VISO2-M	11.9	0.0234	0.1	1 core @ 2.5Ghz (C/C++)
MLM-SFM	2.54	0.0057	0.03	5 cores @ 2.5Ghz (C/C++)
FTMVO	2.24	0.0049	0.11	1 core @ 2.5Ghz (C/C++)
PMO / PbT-M2	2.05	0.0051	1.0	1 core @ 2.5Ghz (Python + C/C++)
CYC_MVO	1.29	0.0031	0.2	1 core @ 3.5Ghz (Python + C/C++)
RI_MVO	1.13	0.0032	0.07	1 core @ 3.5Ghz (Python + C/C++)

The work named VISO2-M can be found in (GEIGER; ZIEGLER; STILLER, 2011), MLM-SFM is in (SONG; CHANDRAKER; GUEST, 2016), FTMVO is in (Mirabdollah M., Mertshching B. (GET Lab, 2015) and PMO / PbT-M2 is in (FAN; WANG; WU, 2016).

An accurate computational complexity analysis is not trivial since different solutions are developed in different environments and executed in different hardware. The work from Song and Chandraker (SONG; CHANDRAKER; GUEST, 2016) is a C/C++ multi-thread system that presents high throughput. Tracking is based in ORB feature descriptor (RUBLEE et al., 2011) which is faster than the KLT tracker (HARRIS; STEPHENS, 1988) (BOUGUET, 2000) used by (PEREIRA et al., 2017b) but in the experiments performed in the course of this work ORB features presented a lower tracking accuracy.

The combination of scale transmission and absolute scale correction as explained in 8.2, slightly improved the translation estimation when compared to FVO (PEREIRA et al., 2017b). The rotation accuracy however was very similar. It is important no note that scale estimation is important for translation errors, but has nearly no impact on orientation precision.

9.7 Chapter Conclusion

This chapter presented two variations of a monocular visual odometry system for camera equipped vehicles. The system used concepts discussed in last chapters to improve pose estimation.

A cyclic PnP algorithm (CYC_MVO), with variable reprojection threshold was used to calculate camera pose with the three last images processed in reversed order or with a resection-intersection two view process (RI_MVO). Scale was recovered using ground points selected with the multi-attribute ground feature selector. Fast triangulation was used for depth estimation. And feature tracking was enhanced with perspective image transformation.

Python and OpenCV implementations tested on the KITTI dataset, achieved rotation and translation precision superior to all other published works, using only the last two or three image frames as input.

10 CONCLUSION

"The true sign of intelligence is not knowledge but imagination."

Albert Einstein

This thesis presented a new monocular navigation system for camera equipped vehicles. The system follows the standard sparse location and mapping pipeline, incorporating significant advances in feature tracking, scale detection and mainly camera relative pose estimation.

This work started with an in depth analysis of the characteristics of projection error in movement along the optical axis. Forward and backward camera movement are compared and the conclusions of that study point to a cyclic three-view pose estimation algorithm that outperformed published works of the time.

The cyclic pose estimator is further improved in a two-view Resection Intersection (RI) based camera pose estimator that present robustness and convergence rate superior to other implementations. The conceived method is based on non-linear optimization for projection residual minimization. The key idea in the formulated algorithm is to take into account the estimated depth deviation when applying an incremental camera pose variation.

As the developed RI pose estimator demands multiple triangulations for each camera pose update iteration, a fast inexact triangulation algorithm is proposed. The method estimates feature depth based on Euclidean distance minimization from projection to traced feature in a single image.

The developed two-view camera pose estimation method combines simplicity, performance and robustness like no other two-view bundle adjustment algorithm used in MVO solutions on the KITTI dataset. It is therefore an important advancement that is likely to present valuable results in other computer vision problems such as sparse 3D reconstruction, stereo vision navigation or augmented reality applications. To that end, it is intended to translate the algorithm to C++ and integrate it into upcoming versions of the OpenCV (BRADSKI, 2000) library.

The feature tracker precision for ground points is improved using Lucas-Kanade algorithm over a perspective warped version of the original target image.

The inverse warping map is used to translate the found feature positions to their real locations in the original image.

Camera nodding detection and compensation enhances scale estimation based on fixed known camera height. Camera scale transmission based on close points helps smoothing the predicted movement.

The developments are combined in a monocular odometry system and tested in a popular vehicle visual navigation database. Achieved precision in camera pose estimation outperforms the state of the art published works in visual odometry, proving the efficiency of created algorithms.

Besides the technical contributions of the thesis proposal, the present work is expected to serve as a seminal work for future academic thesis and dissertations in Computer Vision and microelectronics. Both extending the camera pose estimation and scene structure aproximation algorithms to other CV related problems and proposing parallel implementations suitable for hardware accelerated real-time applications.

Dealing with limited visibility scenarios due to fog, snow or bright sun remain as a future effort. Other promising continuation directions include tests in different datasets aimed toward unmanned aerial vehicles, 3D point cloud generation or multi-camera SLAM systems.

REFERENCES

ALIAKBARPOUR, H.; PALANIAPPAN, K.; SEETHARAMAN, G. Parallax-tolerant aerial image georegistration and efficient camera pose refinement–without piecewise homographies. **IEEE Transactions on Geoscience and Remote Sensing**, PP, n. 99, p. 1–20, 2017. ISSN 0196-2892.

BAKER, S.; MATTHEWS, I. Lucas-Kanade 20 Years On : A Unifying Framework : Part 1 2 Background : Lucas-Kanade. **International Journal of Computer Vision**, v. 56, n. 3, p. 221–255, 2004. ISSN 09205691. Available from Internet: <http://www.springerlink.com/openurl.asp?id=doi:10.1023/B:VISI.0000011205.11775.fd>.

BANCHOFF, T. **Beyond the Third Dimension: Geometry, Computer Graphics, and Higher Dimensions**. New York, NY, USA: W. H. Freeman & Co., 1990. ISBN 0716750252.

BJORCK, A. **Numerical Methods for Least Squares Problems**. [S.l.]: Siam Philadelphia, 1996.

BOUGUET, J. yves. Pyramidal implementation of the lucas kanade feature tracker. **Intel Corporation, Microprocessor Research Labs**, 2000.

BOYKOV, Y.; FUNKA-LEA, G. Graph cuts and efficient n-d image segmentation. **International Journal of Computer Vision**, v. 70, n. 2, p. 109–131, Nov 2006. ISSN 1573-1405. Available from Internet: <http://dx.doi.org/10.1007/s11263-006-7934-5>.

BRADSKI, G. Opencv. **Dr. Dobb's Journal of Software Tools**, 2000.

DANIILIDIS, K.; MARAGOS, P.; PARAGIOS, N. (Ed.). **BRIEF: Binary Robust Independent Elementary Features**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. 778–792 p. ISBN 978-3-642-15561-1.

CHATILA, R.; LAUMOND, J. Position referencing and consistent world modeling for mobile robots. In: **Proceedings. 1985 IEEE International Conference on Robotics and Automation**. [S.l.: s.n.], 1985. v. 2, p. 138–145.

CHEKHLOV, D. et al. Real-time and robust monocular slam using predictive multi-resolution descriptors. In: **In 2nd International Symposium on Visual Computing**. [S.l.: s.n.], 2006.

CHENG, Y.; MAIMONE, M.; MATTHIES, L. Visual odometry on the mars exploration rovers. In: **2005 IEEE International Conference on Systems, Man and Cybernetics**. [S.l.: s.n.], 2005. v. 1, p. 903–910 Vol. 1. ISSN 1062-922X.

CONN, A. R.; GOULD, N. I. M.; TOINT, P. L. **Trust-region Methods**. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. ISBN 0-89871-460-5.

DAVISON, A. J. et al. MonoSLAM: Real-time single camera SLAM. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 29, n. 6, p. 1052–1067, 2007. ISSN 01628828.

DAWSON-HOWE, K. **A Practical Introduction to Computer Vision with OpenCV**. 1st. ed. [S.l.]: Wiley Publishing, 2014. ISBN 1118848454, 9781118848456.

ENGEL, J.; KOLTUN, V.; CREMERS, D. Direct sparse odometry. In: **arXiv:1607.02565**. [S.l.: s.n.], 2016.

ENGEL, J.; STURM, J.; CREMERS, D. Semi-dense visual odometry for a monocular camera. **Proceedings of the IEEE International Conference on Computer Vision**, p. 1449–1456, 2013. ISSN 1550-5499.

FAN, B.; WANG, Z.; WU, F. **Local Image Descriptor: Modern Approaches**. Springer Berlin Heidelberg, 2016. (SpringerBriefs in Computer Science). ISBN 9783662491737. Available from Internet: <https://books.google.com.br/books?id=ugJTCwAAQBAJ>.

FANANI, N. et al. Keypoint trajectory estimation using propagation based tracking. **2016 IEEE Intelligent Vehicles Symposium (IV)**, p. 933–939, 2016.

FERNANDES, L. C. et al. Carina intelligent robotic car: Architectural design and applications. **Journal of Systems Architecture - Embedded Systems Design**, p. 372–392, 2014.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, ACM, New York, NY, USA, v. 24, n. 6, p. 381–395, jun. 1981. ISSN 0001-0782. Available from Internet: <http://doi.acm.org/10.1145/358669.358692>.

FORSTER, C. et al. Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. **2015 IEEE International Conference on Robotics and Automation (ICRA)**, p. 111–118, May 2015. ISSN 1050-4729.

FORSTER, C.; PIZZOLI, M.; SCARAMUZZA, D. Svo: Fast semi-direct monocular visual odometry. **2014 IEEE International Conference on Robotics and Automation (ICRA)**, p. 15–22, May 2014. ISSN 1050-4729.

GAO, X.-S. et al. Complete solution classification for the perspective-three-point problem. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 25, n. 8, p. 930–943, Aug 2003. ISSN 0162-8828.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: **Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on**. [S.l.: s.n.], 2012. p. 3354–3361. ISSN 1063-6919.

GEIGER, A.; LENZ, P.; URTASUN, R. **kitti web site**. 2012. [Online; accessed 2016-07-11]. Available from Internet: <http://www.cvlibs.net/datasets/kitti/eval_odometry>.

GEIGER, A.; ZIEGLER, J.; STILLER, C. Stereoscan: Dense 3d reconstruction in real-time. In: **2011 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.: s.n.], 2011. p. 963–968. ISSN 1931-0587.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **In Proc. of Fourth Alvey Vision Conference**. [S.l.: s.n.], 1988. p. 147–151.

HARTLEY, R. I.; STURM, P. Triangulation. **Computer Vision and Image Understanding**, v. 68, n. 2, p. 146–157, 1997. ISSN 10773142. Available from Internet: <http://linkinghub.elsevier.com/retrieve/pii/S1077314297905476>.

HARTLEY, R. I.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. Second. [S.l.]: Cambridge University Press, ISBN: 0521540518, 2004.

HEDBORG, J.; ROBINSON, A.; FELSBERG, M. Robust three-view triangulation done fast. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops**, p. 152–157, 2014. ISSN 21607516.

IRANI, M.; ROUSSO, B.; PELEG, S. Recovery of ego-motion using image stabilization. In: **1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 1994. p. 454–460. ISSN 1063-6919.

JONES, E. et al. **SciPy: Open source scientific tools for Python**. 2001–. [Online; accessed 2016-07-11]. Available from Internet: <http://www.scipy.org/>.

KANATANI, K.; SUGAYA, Y.; NIITSUMA, H. Triangulation from Two Views Revisited: Hartley-Sturm vs. Optimal Correction. **Procedings of the British Machine Vision Conference 2008**, p. 18.1–18.10, 2008. Available from Internet: <http://www.bmva.org/bmvc/2008/papers/55.html>.

KITT, B. et al. Monocular visual odometry using a planar road model to solve scale ambiguity. In: LILIENTHAL, A. J. (Ed.). [S.l.]: Learning Systems Lab, AASS, Örebro University, 2011. p. 43–48.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small ar workspaces. In: **Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on**. [S.l.: s.n.], 2007. p. 225–234.

KUKELOVA, Z.; PAJDLA, T.; BUJNAK, M. Fast and stable algebraic solution to l2 three-view triangulation. In: **2013 International Conference on 3D Vision - 3DV 2013**. [S.l.: s.n.], 2013. p. 326–333. ISSN 1550-6185.

LEPETIT, V.; MORENO-NOGUER, F.; FUA, P. Epnp: An accurate o(n) solution to the pnp problem. **International Journal of Computer Vision**, v. 81, p. 155–166, 2008.

LINDEBERG, T. Feature detection with automatic scale selection. **Int. J. Comput. Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 30, n. 2, p. 79–116, nov. 1998. ISSN 0920-5691. Available from Internet: <http://dx.doi.org/10.1023/A:1008045108935>.

LINDSTROM, P. Triangulation made easy. In: **Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on**. [S.l.: s.n.], 2010. p. 1554–1561. ISSN 1063-6919.

LOURAKIS, M. I. A.; ARGYROS, A. A. Sba: A software package for generic sparse bundle adjustment. **ACM Trans. Math. Softw.**, ACM, New York, NY, USA, v. 36, n. 1, p. 2:1–2:30, mar. 2009. ISSN 0098-3500. Available from Internet: <http://doi.acm.org/10.1145/1486525.1486527>.

LOWE, D. G. Object recognition from local scale-invariant features. In: **Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on**. [S.l.: s.n.], 1999. v. 2, p. 1150–1157 vol.2.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: **Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. (IJCAI'81), p. 674–679. Available from Internet: <http://dl.acm.org/citation.cfm?id=1623264.1623280>.

MENG, X.; HONG, F.; CHEN, Y. Monocular simultaneous localization and mapping with a modified covariance extended kalman filter. In: **2009 IEEE International Conference on Intelligent Computing and Intelligent Systems**. [S.l.: s.n.], 2009. v. 2, p. 900–903.

MERRELL, P. et al. Interactive furniture layout using interior design guidelines. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 30, n. 4, p. 87:1–87:10, jul. 2011. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2010324.1964982>.

Mirabdollah M., Mertshching B. (GET Lab, U. o. P. Fast Techniques for Monocular Visual Odometry. In: **GCPR 2015**. [S.l.: s.n.], 2015. p. 297–307.

MORAVEC, H. P. **Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover**. Thesis (PhD) — Stanford University, Stanford, CA, USA, 1980. AAI8024717.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDóS, J. D. Orb-slam: A versatile and accurate monocular slam system. **IEEE Transactions on Robotics**, v. 31, n. 5, p. 1147–1163, Oct 2015. ISSN 1552-3098.

NISTER, D. An efficient solution to the five-point relative pose problem. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 26, n. 6, p. 756–770, June 2004. ISSN 0162-8828.

PAHLAVAN, K.; BAO, G.; LIANG, M. Body-slam: Simultaneous localization and mapping inside the human body. In: **Keynote speech, 8th International Conference on Body Area Networks (BodyNets), Boston, MA, September 30-October**. [S.l.: s.n.], 2013. v. 2.

PEREIRA, F. et al. Backward motion for estimation enhancement in sparse visual odometry. In: **2017 Workshop of Computer Vision (WVC)**. [S.l.: s.n.], 2017. p. 61–66.

PEREIRA, F. I. et al. Monocular visual odometry with cyclic estimation. In: **2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.: s.n.], 2017. p. 1–6.

PEREIRA, F. I. et al. A novel resection-intersection algorithm with fast triangulation applied to monocular visual odometry. In: **IEEE Transactions on Intelligent Transportation Systems**. [S.l.: s.n.], 2018. p. 1–10.

PERSSON, M. et al. Robust stereo visual odometry from monocular techniques. **IEEE Intelligent Vehicles Symposium, Proceedings**, n. Iv, p. 686–691, 2015.

RADKE, R. J. **Computer Vision for Visual Effects**. New York, NY, USA: Cambridge University Press, 2012. ISBN 0521766877, 9780521766876.

ROSTEN, E.; DRUMMOND, T. Fusing points and lines for high performance tracking. In: **Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2**. Washington, DC, USA: IEEE Computer Society, 2005. (ICCV '05), p. 1508–1515. ISBN 0-7695-2334-X-02. Available from Internet: <http://dx.doi.org/10.1109/ICCV.2005.104>.

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: **Proceedings of the 9th European Conference on Computer Vision - Volume Part I**. Berlin, Heidelberg: Springer-Verlag, 2006. (ECCV'06), p. 430–443. ISBN 3-540-33832-2, 978-3-540-33832-1. Available from Internet: <http://dx.doi.org/10.1007/11744023_34>.

RUBLEE, E. et al. Orb: An efficient alternative to sift or surf. In: **2011 International Conference on Computer Vision**. [S.l.: s.n.], 2011. p. 2564–2571. ISSN 1550-5499.

SCARAMUZZA, D. et al. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In: **2009 IEEE 12th International Conference on Computer Vision**. [S.l.: s.n.], 2009. p. 1413–1419. ISSN 1550-5499.

SE, S.; LOWE, D. G.; LITTLE, J. J. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. **I. J. Robotics Res.**, v. 21, p. 735–760, 2002.

SHI, J.; TOMASI, C. Good features to track. In: **Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on**. [S.l.: s.n.], 1994. p. 593–600. ISSN 1063-6919.

SONG, S.; CHANDRAKER, M.; GUEST, C. C. Parallel, real-time monocular visual odometry. **Proceedings - IEEE International Conference on Robotics and Automation**, p. 4698–4705, 2013. ISSN 10504729.

SONG, S.; CHANDRAKER, M.; GUEST, C. C. High Accuracy Monocular SFM and Scale Correction for Autonomous Driving. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 38, n. 4, p. 730–743, 2016. ISSN 01628828.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1st. ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN 1848829345, 9781848829343.

TRIGGS, B. et al. Bundle adjustment - a modern synthesis. In: **Proceedings of the International Workshop on Vision Algorithms: Theory and Practice**. London, UK, UK: Springer-Verlag, 2000. (ICCV '99), p. 298–372. ISBN 3-540-67973-1. Available from Internet: <http://dl.acm.org/citation.cfm?id=646271.685629>.

WOHLER, C. **3D Computer Vision: Efficient Methods and Applications**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 3642017312, 9783642017315.

ZHOU, D.; DAI, Y.; LI, H. Reliable scale estimation and correction for monocular visual odometry. In: **2016 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.: s.n.], 2016. p. 490–495.