Constrained Pose and Motion Estimation

Ronen Lerner
Constrained Pose and Motion Estimation

Research Thesis

In Partial Fulfillment of the
Requirements for the
Degree of Doctor of Philosophy

Ronen Lerner

Submitted to the Senate of the Technion - Israel Institute of Technology

SHVAT 5768        HAIFA        FEBRUARY 2008
The Research Thesis was Done Under the Supervision of Prof. Ehud Rivlin in the Faculty of Computer Science.

I wish to express my gratitude to my supervisor Prof. Ehud Rivlin for his wonderful guidance, support and help.

I would also like to thank my wife, my family and my friends for their support.

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED
## Contents

Abstract ............................................................................................................. 1  
List of symbols ................................................................................................... 3  

1 Background and Related Work ........................................................................... 6  

1.1 Pose Estimation ............................................................................................. 7  
1.2 Ego-motion Estimation and Structure-From-Motion Techniques ......................... 9  

1.2.1 Direct Methods for SFM and Ego-Motion Estimation ..................................... 10  
1.3 Vision Based Navigation Algorithms .................................................................. 10  

1.3.1 Vision Based Navigation using DTM .............................................................. 11  
1.3.2 Landmark Subset Selection ........................................................................... 14

2 Motivation and Research Approach ..................................................................... 16

3 Pose Recovery from Feature Correspondences and a Digital Terrain Map ............. 20  

3.1 The Two-View Constraint .............................................................................. 21  

3.1.1 Notation and Problem Definition ................................................................. 21  
3.1.2 Single-Frame Geometry .............................................................................. 23  
3.1.3 Two-Frame Geometry ................................................................................. 26  
3.1.4 Multiple Features ....................................................................................... 27  
3.1.5 Degenerate Scenarios ................................................................................ 29  
3.1.6 The Epipolar Constraint Connection ............................................................ 31  

3.2 Error analysis .................................................................................................. 34  

3.2.1 Pose and Motion Uncertainty Computation .................................................. 34  
3.2.2 Algorithm Sensitivity Study .................................................................... 38  

3.3 Algorithm Implementation .............................................................................. 45  

3.3.1 Internal vs. External Iterations .................................................................. 45


### 3.3.2 Dealing with Outliers

- 46

### 3.4 Experimental Results

- 47
  - 3.4.1 Simulation Results
    - 47
  - 3.4.2 Robustness of the Algorithm Against Large Errors in the Initial Guess
    - 49
  - 3.4.3 Lab Experiment Results
    - 51

### 3.5 A Comparison with SFM and Registration Algorithm

- 55
  - 3.5.1 The “SFM+ICP” Algorithm
    - 55
  - 3.5.2 Performance Comparison
    - 56

### 3.6 Extending The Algorithm For Omnidirectional Cameras

- 61
  - 3.6.1 The Omnidirectional Constraint
    - 62
  - 3.6.2 Experimental Results
    - 65

### 4 Direct Method for Video Based Navigation Using a Digital Terrain Map

- 72
  - 4.1 The Direct Constraint
    - 73
    - 4.1.1 The Two-Views + DTM Geometry
      - 74
    - 4.1.2 The Brightness Constancy Constraint
      - 75
  - 4.2 The Navigation Algorithm
    - 76
    - 4.2.1 Hierarchial Scheme for Large A Priori Errors
      - 77
    - 4.2.2 DTM Linearization using Z-Buffering
      - 77
    - 4.2.3 Performance Improvement by Internal and External Iterations
      - 81
    - 4.2.4 Stability Improvement using SVD
      - 82
    - 4.2.5 Gradient Based Pixel Selection
      - 85
    - 4.2.6 Outliers Effect Reduction using M-Estimator
      - 85
  - 4.3 Simulation Results
    - 86
  - 4.4 Flight Experiment Results
    - 91

### 5 Landmark Selection for Task-Oriented Navigation

- 97
  - 5.1 Landmark-Based Navigation
    - 99
    - 5.1.1 The Pose Covariance Matrix
      - 100
  - 5.2 Task-Oriented Grading of Landmark Subsets
    - 101
  - 5.3 Approximate Solution for the Subset Selection Problem
    - 104
    - 5.3.1 Proof of the Grade Function Convexity
      - 105
    - 5.3.2 Normalization Procedure for Improving Numerical Stability
      - 107
5.3.3 Posing the Relaxed Program as an SDP .................................................. 108
5.3.4 Generalization for non-identical distribution of measurements error ........... 111
5.4 $\Sigma_R$ Construction .................................................................................. 112
5.5 Results .......................................................................................................... 113
  5.5.1 Simulations .............................................................................................. 113
  5.5.2 Lab Experimentation ................................................................................ 115

6 Conclusions and Future Research Directions ..................................................... 122

Bibliography ........................................................................................................ 125
List of Figures

1.1 Pose and ego-motion definition as Euclidian transformations from the camera system to the
global reference system or to another camera system in different time instance. ....... 7

1.2 Clearance measurement (blue lines) together with a reconstructed motion path of the nav-
igating platform (black line) can be used to reconstruct a curve (red line) that can then be
registered to the DTM surface. Such procedure yields the platform pose. ............... 12

1.3 Two steps algorithm can be used for pose computation: first SFM algorithm is applied to
reconstruct a patch of the observed terrain. Next, a registration algorithm is activated in
order to find the transformation that connect between the DTM and the reconstructed patch.
This transformation is the sought pose. ......................................................... 13

3.1 The terrain feature, $Q_T$, is perspectively projected to the image plane point $q_1$ under the
true first camera frame (where $p_1$ represents its position and $R_1$ its orientation). Using this
projected point and the estimated pose of the camera ($p_{E_1}$ and $R_{E_1}$), the ray from $p_{E_1}$ in the
direction of $R_{E_1}q_1$ can be intersected with the DTM at $Q_E$. The DTM is linearized around
this point and $Q_T$ is assumed to lie on that tangent plane. .............................. 24

3.2 (a) The vector $v$ is being projected by $\mathcal{P}(u, n)$ onto the plane orthogonal to $n$ along the
direction of $u$. (b) In order to obtain the unknown vector $Q_T - Q_E$, the vector $p_1 - Q_E$
is being projected onto the linearization plane and along the $R_1q_1$ direction using the $\mathcal{P}$
projection operator. ................................................................. 26

3.3 Examples to constellations which lead to singularities of the algorithm. (a) features from
a surface which can be swept out by moving a planar curve along constant direction, (b)
features laying on a silhouette of arbitrary surface, (c) surface of revolution, (d) spiral ... 29
3.4 The examined scenario from the second camera frame’s \( (C_2) \) point of view. \( q_2 \) is the perspective projection of the terrain feature \( C_2Q_T \), and thus the two should coincide. Additionally, since \( q_1 \) is also a projection of the same feature in the \( C_1 \)-frame, the epipolar constraint requires that the two rays (one in the direction of \( q_2 \) and the other from \( p_{12} \) in the direction of \( R_{12}q_1 \)) will intersect.

3.5 Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the number of corresponding features. In both graphs, the left vertical axis measures the translational deviations (in meters) and corresponds to the solid graph-line, while the right vertical axis measures the rotational deviations (in radians) and corresponds to the dotted graph-line.

3.6 Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the image resolution.

3.7 Different DTM resolutions: (a) grid spacing = 190m, (b) grid spacing = 100m, (c) grid spacing = 30m.

3.8 Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the grid-spacing of the DTM.

3.9 Standard-deviation of the DTM’s height measurement with respect to the grid-spacing of the DTM.

3.10 DTM elevation differences: (a) 150m, (b) 300m, (c) 450m.

3.11 Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the height differences of the terrain.

3.12 Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the magnitude of the translational component of the ego-motion.

3.13 Outliers caused by terrain shape and DTM mismatch. \( C \) and \( C_E \) are true and estimated camera frames, respectively. \( Q_1E \) and \( Q_2E \) are outliers caused by terrain shape and by terrain/DTM mismatch, respectively.

3.14 (a) The virtual terrain, (b) The DTM constructed from this terrain (grid-spacing is coarser than in the experiment, for visualization purposes). Note that the building on the virtual terrain (box at the bottom) has been moved to the gray bump at the center of the DTM.
3.15 Translational (a) and rotational (b) errors of the calculated pose as a function of the number of iterations. The symbols I, II, III, and IV denote external-iterations. Each iteration contains 30 internal-iterations. The blue solid line - error-free scenario, red dotted line - a scenario with Gaussian error of 0.5 pixel S.D. Units are meters and radians.

3.16 The robustness simulations were conducted for 5 virtual cameras located in different poses. A DTM of Montana’s Rocky mountains was chosen to examine the algorithm on relatively rough terrain.

3.17 Percentage of success to converge to the true camera pose for different magnitudes of translational (a) and angular (b) errors in the algorithm’s initial guess. The red line corresponds to camera-1, green to camera-2, blue to camera-3, black to camera-4 and the dotted line to camera-5.

3.18 (a) A 3D terrain model of horizontal dimension $50 \times 77$ cm. (b) The DTM was constructed by using a laser-based 3D-scanner. The spatial grid was 1 cm (the one in the figure has a coarser grid for visualization purposes).

3.19 Two of the tested trajectories. Trajectory $a$ is mostly a translation while trajectory $b$ has significant changes in orientation.

3.20 (a) A frame taken from one of the camera’s trajectories. (b) The estimated correspondence of 400 features taken from an $20 \times 20$ regular grid over the image plane of this frame.

3.21 Experimental results for trajectories $a$ and $b$ (see Fig. 3.19). The diverging trajectories use the error model and no updates. The updated paths use the pose/motion algorithm to bound divergence.

3.22 Position errors (a) and orientation errors (b) of the drifted-path (dotted-line) and of the corrected-path (solid-line) of the second trajectory - $b$.

3.23 The synthetic terrain was scaled to obtain a variety of elevation variations: (a) 800 m, (b) 600 m, (c) 300 m. Different DTMs were obtained for terrain (b) by sampling the terrain under different spatial grids (resolutions): (d) 100 m, (e) 50 m, (f) 30 m.

3.24 Pose and ego-motion estimation accuracy using the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for different DTM resolutions. Resolution varies from 10 m to 100 m. (a) Position errors, in meters. (b) Orientation errors, in radians. (c) Motion translation errors, in meters. (d) Motion rotation errors, in radians.
3.25 Pose and ego-motion accuracies obtained by the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for terrains with varying elevation differences (from 300 m to 800 m). (a) Position error, in meters. (b) Orientation error, in radians. (c) Motion translation error, in meters. (d) Motion rotation error, in radians.  

3.26 Pose and ego-motion accuracy obtained by the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for different image resolutions (from $200 \times 200$ to $1000 \times 1000$). (a) Position error, in meters. (b) Orientation error, in radians. (c) Motion translation error, in meters. (d) Motion rotation error, in radians.  

3.27 Pose and ego-motion accuracy obtained by the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for different numbers of corresponding features pairs. The features were selected from a regular grid that was spanned over the image plane, where the grids resolutions varied from $4 \times 4$ (16 features) up to $20 \times 20$ (400 features). (a) Position error, in meters. (b) Orientation error, in radians. (c) Motion translation error, in meters. (d) Motion rotation error, in radians.  

3.28 When using an omnidirectional vision system a wide area of the terrain is visible (see the red area) even when the camera approaches a mountainside. When using a regular camera in similar scenario only small patch that is almost planar is observed (see the blue area).  

3.29 Geometrical description of expression (3.6.69) using the projection operator.  

3.30 (a) A 3D terrain model of horizontal dimension $115 \times 95$ cm. (b) The DTM was constructed by using a laser-based 3D-scanner. The spatial grid was 1mm (the one in the figure has a coarser grid for visualization purposes).  

3.31 Omnidirectional vision was obtained by a configuration of three cameras that were posed in different orientations.  

3.32 Two of the tested trajectories. Trajectory $a$ contains constant translational motion while trajectory $b$ has significant changes in orientation. The true paths are marked by black solid line, while the paths reconstructed by the algorithm are marked by red line. The black dotted lines represent the trajectories that would have been obtained in case the algorithm was not activated.  

3.33 Results for trajectories a (sub-figures (a) and (b)) and b (sub-figures (c) and (d)) when using the three cameras configuration. Position errors ((a) and (c)) and orientation errors ((b) and (d)) of the drifted path are marked with a black dashed line, and errors of the corrected path are marked with a red solid line.
3.34 (a) Translational errors of trajectory b when using 300 features coming from only one camera (blue dashed line) and when using 100 features from each of the three cameras (red solid line). (b) A frame captured by the single camera that was in use. 69

3.35 (a) The catadioptric system that was used for omnidirectional vision in the second experiment. (b) An example for optical-flow field that was extracted for the algorithm. Each small blue arrow shows a corresponding couple. 70

3.36 Results for trajectories a (sub-figures (a) and (b)) and b (sub-figures (c) and (d)) when using the catadioptric system. Position errors ((a) and (c)) and orientation errors ((b) and (d)) of the drifted path are marked with a black dashed line, and errors of the corrected path are marked with a red solid line. 71

4.1 The linearization plane $L_1$ with the normal $N_1$ is more proper choice in case we expect the $Q_T$ to be somewhere in the range $r_1$. However, if we have higher uncertainty and we can only expect $Q_T$ to be in range $r_2$, then we better use the linearization plane $L_2$ with the normal $N_2$. 79

4.2 The computed depth of the four adjacent pixels: $ql$, qr, qu and qd is used to construct four ground points adjacent to $Q_E$: $Q_l$, $Q_r$, $Q_u$ and $Q_d$. The normal can then be computed by taking the cross product of $V_1 = Q_r - Q_l$ and $V_2 = Q_u - Q_d$. 80

4.3 The virtual terrain is texture-mapped with an ortho-photo of a real terrain. A building is located in the middle of the landscape. A virtual camera was placed in two poses above the terrain (the first pose is marked by a dark red circle and the second by a dark green diamond). An error was added to these poses to produce the initial guess for the algorithm (the bright red circle and the bright green diamond). 86

4.4 (a) The rendered image from the first virtual camera pose. (b) the rendered image from the second pose. 87

4.5 The error of the translation parameters (a) in meters and of the rotation parameters (b) in radians along the algorithm iterations. Red line - error of the first virtual camera, green line - error of the second camera, black line - the end of a pyramid level processing. 88

4.6 (a) The virtual terrain without the building and the texture. (b) the rendered featureless image. 88

4.7 The error of the translation parameters (a) in meters and of the rotation parameters (b) in radians along the algorithm iterations, where featureless environment is faced. 89
4.8 (a) cropped part of the noised image. This part contains both the building that was omitted from the DTM and “salt and pepper” pixels. (b) The weights that were assigned by the M-estimator, varying from red (weight=0) to yellow (weight=1). The black pixels were not selected by the algorithm in the examined iteration.

4.9 The error of the translation parameters (a) in meters and of the rotation parameters (b) in radians along the algorithm iterations in the presence of outliers.

4.10 The terrain of the flight experiment in top-view (a) and perspective-view (b). The ground truth flight trajectory is marked by the green line. The north-east measurements are in kilometers and w.r.t the new Israeli grid.

4.11 The camera was attached to a device (the marked circle in the figure) that was carried by an ultra-light plane. In order to obtain a ground truth about the flight trajectory a D-GPS and inertial systems were also attached to the device.

4.12 One of the image couples that were selected for the algorithm. The internal parameters of the camera were used to remove the distortion and to normalize these images. The selection criteria guarantee that there will be a shift of 100 pixels between the principal point of one image to its corresponding point in the other image.

4.13 (a) The results of the whole trajectory, (b) The results of the trajectory beginning. The ground-truth trajectory is marked by the green line. The locations of the camera in which the navigation algorithm was activated are marked by green circles. The red line depicts the drifted trajectory that would have been obtained if no pose correction was performed. The blue line and the black line show the results of the offline and online trajectories respectively. The positions that were estimated by the algorithm are marked by blue ‘x’ marks. In the error-free scenario, the green circles and the corresponding blue ‘x’ marks should align.

4.14 The position error (in meters) along the beginning of the flight trajectory. The red line depicts the error that would have been obtained in case only the inertial navigation system was used. The black line shows the errors of the online trajectory, and the blue line show the error of the offline trajectory.

4.15 An image that was captured during one of the trajectory’s turns. The yellow lines depict the depth level sets. It can be seen that the viewed topography is almost flat, which prevent the computation of accurate pose.
5.1 Comparison between uncertainty ellipsoids in a 3D pose configuration space. It is clear that ellipsoids B and C are preferable to A, but the choice between these two is less obvious and should take into account the requirements of the specific navigation task.

5.2 Approximation factors that were obtained for different subset sizes. The solid line was obtained by the algorithm, the dotted line by taking the best of 10 uniformly selected subsets, and the dashed line by taking the best of 100 uniformly selected subsets.

5.3 Synthetic landmarks were placed on a plane parallel to the image plane z=30 (black dots). The obtained fractional $\alpha$ values (vertical lines) and the selected subset (circles) are presented. (a) A subset of 4 landmarks was selected when the navigation task required the camera position. (b) A subset of 10 landmarks was selected when only the roll angle of the camera was required.

5.4 Subset selection of 5 and 40 landmarks for different tasks. (a) and (d) show coplanar landmarks parallel to the image plane, (b) and (e) show coplanar landmarks in general position, (c) and (f) show landmarks placed on 3 orthogonal planes. In all six images the markers represent the selected landmarks according to different navigational tasks: an 'x' marker – a task of computing the X component of camera position, a diamond – a task of computing the Y component of camera position, a circle – a photographing task.

5.5 The weighted error obtained by different subset sizes for the scene presented in Fig. 5.4(c). The dashed line shows the mean error of the pose when using the uniformly selected subsets, the solid line shows the pose error when using the selected subset of the algorithm. In (a), the navigation task requires the X component of the camera position, while (b) shows the results for the photographing task.

5.6 The video camera mounted on 6 DOF robotic arm.

5.7 Subsets of 10 landmarks that were selected during the six experiments. In (a), (b) and (c) a path-following task was performed. In (d), (e) and (f) a photographing task was performed. (a) and (d) show the first scene, (b) and (e) the second scene, and (c) and (f) show the third scene. The stars mark the selected landmarks. The circles in the sub-figures of the photographing task show the location of the sight.

5.8 Mean and maximal millimetric errors obtained for different subset sizes when a path following task was performed. In (a) and (b) the first landmark constellation was used, in (c) and (d) the second constellation, and in (e) and (f) the third constellation. Solid line - the error obtained when using the selection algorithm, dashed line - the error obtained when selecting the subsets arbitrarily.
5.9 Mean and maximal projection errors (in pixels) obtained for different subset sizes when a photographing task was performed. In (a) and (b) the first landmark constellation was used, in (c) and (d) the second constellation, and in (e) and (f) the third constellation. Solid line - the error obtained when using the selection algorithm, dashed line - the error obtained when selecting the subsets arbitrarily.
Abstract

Developing totally autonomous navigation system for vehicles and UAVs is a challenging task. Most navigation systems today either integrate inertial/odometry measurements which yield an increasing drift along time, or use GPS measurements which might be unavailable under some circumstances. Vision based navigation algorithms compute the camera pose either using a set of landmarks or by ego-motion integration. For the first alternative, correspondences between the image features and the 3D landmarks need to be computed which is a challenging task. The ego-motion integration alternative suffers from the same drifts that appear in inertial navigation.

It can be stated that what unifies all pose estimation algorithms is that they utilize the available data – both the captured images and the information about the surrounding environment – in order to define a system of constraints on the navigation parameters. The accuracy and robustness of the different algorithms are a direct result of the constraint system quality. As we have more information at our disposal the conditionality of the constraint systems may improve which in turn yield stronger algorithms. However, there may be several possible methods to utilize the information for the navigation task. Some methods use this information in non-optimal manner while others fully exploit the available data. It is in the core of this research to examine proper utilization of the available data in order to achieve as accurate and robust navigation results as possible.

A novel algorithm for pose and motion estimation using image sequence and a Digital Terrain Map (DTM) is presented. Using a DTM as a global reference enables the recovery of the absolute position and orientation of a camera with respect to the external reference frame. Since the full structure of the observed terrain is encoded in the DTM no specific features need to be detected and matched (in contrast to the landmark based approach).

Two alternative methods are proposed for utilizing the available data to construct the constraints on the navigation parameters. One alternative starts by computing feature correspondences between two selected images and then use them, together with the 3D model, for the constraints construction. Another alternative is to combine in the constraints the well known brightness constancy constraint which directly relates to the brightness of the images pixels. This yields a "direct-method" scheme that spare us the need to compute the feature correspondence which might be very difficult in some scenarios.

Another question that is covered in this dissertation is how should one compare the results of two navigation algorithms. A task-oriented criterion for comparing the expected pose accuracies is developed. In the
spirit of this criterion, a task-oriented landmark selection algorithm is developed. This algorithm is capable of selecting a landmark subset that achieves the best task related errors.

The feasibility of the proposed algorithms is established through extensive experimentation, both under controlled simulated environments, in lab experiments using a scales 3D model, and even in an outdoor experiment using aerial images.


List of symbols

\( I_1, I_2 \) the two images that are used by the algorithm
\( \nabla I \) the gradient of an image \( I \)
\( t_1, t_2 \) the two time instances when the images were captured
\( \Delta x, \Delta y \) the horizontal and vertical grid spacing between adjacent pixels in the images
\( \vec{u} = (u, v) \) a location in an image
\( \vec{\Delta u} \) image displacement of a feature
\( \vec{\Delta u}_E \) estimated image displacement of a feature
\( \vec{\delta u} \) residual image displacement of a feature
\( \vec{\delta u}_N, \vec{\delta u}_T \) two components of \( \vec{\delta u} \) along the image gradient and its orthogonal complement
\( h \) the height function of the terrain
\( \hat{h} \) the biased height of a terrain point according to the DTM information
\( N \) the normal to the DTM at some location
\( Q_T, Q_E \) the true and estimated location of a 3D ground feature
\( \lambda_T, \lambda_E \) the true and estimated depth of a ground feature
\( q_1, q_2 \) the projection of a ground feature on the first and second image planes (in homogenous representation)
\( S_i \) in case there is no single center of projection, \( S_i \) marks the center of projection of the i’th feature
\( W \) the global (world’s) coordinate system
\( C(t_i) = C_i \) camera coordinate system at time \( t_i \)
\( \phi, \theta, \psi \) Euler rotation angles
\( R(t_i) = R_i \) rotation matrix of the camera orientation w.r.t \( W \) at time \( t_i \)
\( R_{iE} \) the a-priori estimated rotation matrix of the camera orientation w.r.t \( W \) at time \( t_i \)
\( p(t_i) = p_i \) the position of the camera w.r.t \( W \) at time \( t_i \)
\( p_{iE} \) the a-priori estimated position of the camera w.r.t \( W \) at time \( t_i \)
\( R_{12} \) the relative orientation of the camera between the \( t_1 \) and \( t_2 \) time instances
\( R_{12E} \) the a-priori estimate of the camera relative orientation between the \( t_1 \) and \( t_2 \) time instances
\( p_{12} \) the translation of the camera between the \( t_1 \) and \( t_2 \) time instances
\( p_{12E} \) the a-priori estimated translation of the camera between the \( t_1 \) and \( t_2 \) time instances
\( F_v \) a representation of a vector \( v \) under the coordinate system \( F \)
\( \mathcal{P}(u, n) \) a projection operator that projects vectors on \( n \)'s null space along the \( u \)-direction

\( \mathcal{L}(u, n) \) a projection operator that projects vectors on \( u \)'s direction along \( n \)'s null space

\( M^\dagger \) the pseudo-inverse matrix of \( M \)

\( v^\wedge \) the wedge operator that transforms a vector \( v \) into a skew-symmetric matrix

\( tr[M] \) the trace of a matrix \( M \)

\( \delta t, \delta \omega, \delta s \) infinitesimal change in the translation, rotation and scale of a similarity transformation

\( \Theta \) the parameters vector

\( \hat{\Theta} \) a perturbed parameters vector

\( f_i(\Theta) \) the residual function of the \( i \)'th constraint

\( F(\Theta) \) the concatenation of all the \( f_i \) functions

\( J_i \) the Jacobian matrix of the \( f_i \) function

\( J \) the Jacobian matrix of the \( F \) function

\( \Theta^k \) the estimate of \( \Theta \) at the \( k \)'th iteration

\( \Delta \Theta^k \) the parameters change at the \( k \)'th iteration

\( J^k \) the constraints system Jacobian at the \( k \)'th iteration

\( \sigma_I \) the standard deviation of the image noise

\( \Sigma_X \) the covariance matrix of \( X \)

\( \tilde{\nabla} I, \tilde{\Delta} I \) warped gradient and difference images

\( m_i \) the \( i \)'th eigenvector of a matrix

\( \lambda_i \) the \( i \)'th eigenvalue of a matrix

\( M_\Theta, \Lambda_\Theta \) the eigenvectors-eigenvalues decomposition of the covariance matrix \( \Sigma_\Theta \)

\( \Sigma_R \) the requirements matrix

\( \alpha_i \) the \( i \)'th landmark indicator variable

\( \alpha \) the concatenation vector of all \( \alpha_i \) variables

\( \tilde{N} \) normalization matrix

\( \tilde{J}_i, \tilde{\Sigma}_R, \tilde{\Sigma}_\Theta \) the Jacobian, requirements matrix and covariance matrix normalized according to \( \tilde{N} \)

\( P \succeq Q \) the matrix \( P - Q \) is positive semi-definite

\( y_i \) the \( i \)'th slack variable

\( Y \) a matrix containing all the slack variables

\( E_i \) an indicator matrix of the \( i \)'th slack variable

\( S(\Theta) \) the pose severity function

\( H_S \) the Hessian matrix of \( S \)
Chapter 1

Background and Related Work

Vision based navigation algorithms have been haunted both by the academy and by the industry for many years. The navigation problem includes the estimation of the camera pose, i.e. position and orientation, with respect to a global reference frame, and the camera ego-motion. In case a coordinate systems are attached both to the camera and to the global reference frame, the pose can be simply defined as the Euclidian transformation from the camera system to the reference system. The camera ego-motion in such a case is the Euclidian transformation that connects between the camera system at two consecutive time instances. See Fig. 1.1 for illustration of the above definitions.

Starting from the 19’Th century, mathematicians and the photogrammetry community have developed different solutions for the pose-estimation problem. With the development of the computer-vision field, these problems have become extremely important for a variety of tasks including camera calibration, object recognition and robot navigation. Large set of solutions have been suggested to these problems during the last decades. However, obtaining efficient, accurate and robust estimates for the navigation parameters has still remained a challenging task.

Vision based algorithms are not the only alternative for navigation system implementation. Nowadays, other alternative instruments exist e.g., GPS (Global Positioning System) receivers and inertial navigation systems. Due to the simplicity of their integration in navigation systems these instruments are commonly used. However, both GPS and inertial systems have their limitations.

Inertial systems are equipped with a set of gyroscopes and accelerometers. The accelerometers constantly measure the different accelerations that the platform underdoes. By integrating these measurements along time the velocities of the platform (the ego-motion) can be computed and by integrating the velocities once again its trajectory can be recovered. This ego-motion integration approach for navigation is known by the name of dead-reckoning and it was first implemented by sailors few centuries ago, far before accelerometers were available. An inherent problem of the dead-reckoning approach is that small measurements errors
are integrated along time which lead to an increasing drifts. While most navigation system utilizes inertial navigation measurements, they all should integrate in their framework some pose estimation algorithm that is based on an external source and thus allows the correction the accumulated drift. Such framework can compute the navigation parameters with known and bounded error magnitude.

GPS measurements are commonly combined in such systems for drift prevention since they are based on external source of information i.e. the GPS’s satellites. However, the GPS receivers must have a clear line of sight with several transmitting satellites in order to obtain an accurate pose. Thus, when trying to navigate at indoors environment, at urban area surrounded by tall building, or even when fling in a narrow canyon, poor results can be expected. The aforementioned limitations gives the motivation to develop vision based navigation algorithm that can be used as GPS substitutes for drift elimination.

The following sections review the topics of vision based pose estimation, ego-motion and scene structure estimation, and their utilization for navigation systems. The relevant related works are elaborated throughout these sections.

1.1 Pose Estimation

Pose estimation is one of the most ancient problems in computer-vision. Due to its importance in the photogrammetry field different solutions have been developed for this problem starting from 1841 by the
German mathematician J.A.Grunert [36]. With the growth of the computer-vision field, this problem was haunted from different aspects since it is required for different vision tasks such as camera calibration, object recognition, and robot navigation. In the cases of camera calibration and object recognition we consider the camera’s frame (its coordinates system) as the reference frame and we are looking for the Euclidian transformation that connect this frame to the observed object frame. In the case of robot navigation which is the focus of this work, the observed scene’s frame is considered as the reference frame and we are seeking after the transformation to the camera. Obviously, these are two equivalent ways to look on the very same problem.

The traditional solutions for the pose estimation problem can be divided into two groups: closed-form solutions and iterative solutions. Most of the closed-form solutions are based on three points for which we know their 3-dimensional location under the object’s frame and their 2-dimensional projection on the image-plane. Since the solution of the pose transformation should take into account the non-linear rotational components and the non-linear perspective projection properties, all the closed-form solutions require finding the roots of a polynomial equation (usually fourth degree) and then selecting from the obtained possibilities the real and correct solution using a forth point. [19, 25, 28, 53, 62, 67] are all examples for solutions of this kind. See [36] for a thorough comparison between different closed-form solutions.

Different approach which allows further simplification of the problem is to model the projection of the camera as the so-called weak perspective projection. As described in [45], under this model assumption the camera pose can be uniquely derived up to a reflection. T.D.Alter has developed a closed-form solution under this projection by solving a biquadratic equation (see [2]).

In order to handle errors of the camera’s measurements, algorithms which use more points have been developed. For the case of N points correspondence, several algorithms which search for the least-square solution can be found. These algorithms search the solution of a non-linear optimization problem leading to iterative procedures which require initial guess of the transformation. Good example for such algorithm is presented in Haralick’s paper [35]. At any iteration of that algorithm the estimated pose from the previous iteration is being used to estimate the 3-dimensional location of the observed features and then the camera’s transformation is being adjusted to best match these reconstructed points with the 3D points which were supplied to the algorithm. Another famous algorithm that belongs to this category is POSIT [21]. At each of the algorithm iterations the camera’s pose is first derived under the weak perspective projection model. Then this estimated pose is being used to correct the perspective effects on the image features which will lead for better pose estimate in the next iteration. Other examples for iterative algorithms can be found in [40, 49, 54, 96].
Recent research in the pose estimation field has been focused on developing algorithms which does not require the correspondence between the 2D and 3D measurements. Instead, both optimal pose and optimal correspondence are simultaneously sought. Two examples for such algorithms are [55, 71]. However, these algorithms either tend to fail in different scenarios or not suitable for real-time application.

1.2 Ego-motion Estimation and Structure-From-Motion Techniques

During the past decades substantial effort has been devoted to the investigation of the geometry of multiple views and the translation of this understanding into algorithms that work in practice. Thorough survey can be found in [38]. In particular, several methods have been proposed for the estimation of the camera’s ego-motion from a sequence of images. Ego-motion estimation algorithms are usually being supplied with the location of corresponding features in the examined images or with the optical-flow field which approximates the image features’ velocities. Using properties which are taken from epipolar geometry, different constraints can be defined on the possible camera’s motion that has led to the obtained features displacement.

Another tightly connected problem is the Structure-From-Motion (SFM) problem. Here we are searching for the depths of the observed features for which we know only their 2-dimensional projection on the image-plane at two consecutive camera’s frames (optical-flow field). These depths can be derived as a second step after the ego-motion was already derived, or directly from the image features velocities. Nice example for an ego-motion estimation algorithm is presented in [70]. In that work, the directional error between the image features and the 3D features is being minimized for both frames. This minimization is being performed over the six ego-motion parameters in two steps. For each feature, an epipolar plane passing though the two camera positions was fixed and the directional-error was minimized for all possible 3D feature locations on that plane. In the second step, the result of the first minimization is being minimized over all choices of epipolar-plane. The structure is being derived later using optimal triangulation technique.

Several examples for other SFM algorithms can be found in [3, 6, 15, 17, 18, 39, 47, 68, 70, 77, 78, 79, 82, 84, 91, 92, 98, 99, 100], while the tutorials [69] and [95] discuss the advantages and disadvantages of such implementations.

All ego-motion estimation and SFM algorithms suffer from a common limitation which is caused by the ambiguity between depth and velocity. Under the epipolar geometry there is no way to distinguish between slow motion of close objects and fast motion of far objects. Thus, this family of algorithms will be able to derive the translational ego-motion and the observed feature’s depth only up to a scale-factor. Needless to say that since no external information is used, the camera pose with respect to the global reference cannot
be derived using these techniques alone and the scene structure is reconstructed with respect to the camera coordinates system.

1.2.1 Direct Methods for SFM and Ego-Motion Estimation

All the aforementioned algorithms can be categorized as feature based algorithms. A more recently studied approach for ego-motion estimation and scene structure recovery skips the features correspondence computation step and establish a direct connection between the gray-levels of the examined images and the sought navigation parameters. This family of algorithms is commonly referred to as direct methods. Most of the work in that field relies on the brightness constancy constraint that asserts the gray-level equality of corresponding locations in two consecutive frames of the video stream: \( I_2(u, v) = I_1(u + \Delta u, v + \Delta v) \). The image displacements \( \Delta u \) and \( \Delta v \) are then represented as functions of the camera motion parameters according to some motion model. In some works the displacements are modeled as a result of a two-dimensional image motion (e.g., [9, 48, 75, 89]). Other works assume a full 3D camera motion model (e.g., [32, 46, 51]). The 3D models represent the image displacements as a function of both the camera 3D motion parameters and the unknown depths of the scene. When plugging the expressions of \( \Delta u \) and \( \Delta v \) back into the brightness constancy constraint, we obtain a constraint about the camera motion and the scene depths. Such a constraint can be constructed for each pixel in the images which results in an over-determined system that can be solved to compute the optimal motion parameters and the scene structure. One important merit of the direct methods is their elegant treatment of the well known aperture problem. In some "featureless" environments it is very difficult to find good features to track when using the feature-based approaches. However, using direct methods, each pixel contributes to the constraints system according to the information it possesses. This property can be shown by linearizing \( I_1 \) in the derived constraint around \((u, v)\):

\[
I_2(u, v) = I_1(u, v) + \nabla I_1(u, v) \cdot (\Delta u, \Delta v)^T.
\]

Multiplying the displacement expression by \( \nabla I_1 \) guarantees that only its component along the image gradient (also known as the normal flow) is constrained.

1.3 Vision Based Navigation Algorithms

Pose and ego-motion estimation algorithms may serve for several application including camera calibration, object recognition, video stabilization, etc. One of the important utilizations of these algorithms, which is more relevant the the present dissertation, is autonomous navigation. Currently, there are two main approaches for utilizing the capabilities of vision algorithms for navigation system: landmark based navigation and ego-motion integration.
In the landmarks approach, several features in the landscape, for which we know their 3D location, are detected and located. Using the 2D and 3D data the camera’s pose can be derived by pose estimation algorithms as described in section 1.1. Once the landmarks were found, the pose derivation is simple and can achieve quite accurate estimates. The main difficulty is the detection of the features and their correct matching to the landmarks set. Several works that present such navigation-systems are [14, 23, 24, 44, 90]. In [44], full polyhedral model of the observed scene was used for the navigation task. 3-dimensional edges of buildings in the model were extracted and matched to 2-dimensional edges in the images. Such method is applicable for urban regions only where man-made structures exist and 3D edges can be extracted.

The second approach estimates the ego-motion of the camera (using one of the techniques that were reviewed in section 1.2) and then integrates these motions in a dead-reckoning style to drive the camera trajectory. One of the factors that make this approach attractive is that no specific features need to be detected, unlike the landmarks approach. Several navigation systems which use ego-motion integration can be found in [29, 43, 52, 58]. As was mentioned in the beginning of this chapter, the weakness of this approach comes from the fact that small errors of the ego-motion estimates are being accumulated during the integration process. Hence, the estimated camera’s path is drifted, and the pose estimation accuracy decrease along time. If such approach is being used, it would be desirable to reduce the drift by activating, once in a while, an additional algorithm that estimates the pose directly. Most available vision-based dead-reckoning algorithms activate some sort of landmarks detection to correct this drift, which brings back all the difficulties involved in such process.

1.3.1 Vision Based Navigation using DTM

The landmark approach can be reformulated by replacing the landmarks with a database describing the structure of the scene. The Digital Terrain Model (DTM) is a natural candidate to meet this requirement. In practice, a DTM is an ASCII or binary file that contains only spatial elevation data in a regular grid pattern in raster format for the whole or part of the earth. DTM’s are usually classified in levels according to their grid size. For instance, Level 0 denotes a 30 arc. sec (or approximately 1 km) grid map, while Level 5 denotes a fine, almost 1 m, grid map. Not all these maps are available to the general public, but 30 m are or will be available for most of the earth, while 10 m ones can be obtained for some regions (in particular, for the U.S.). For instance, the Shuttle Radar Topography Mission generated an elevation map with a 30 m grid and 16 m absolute vertical accuracy.

DTMs have already been used successfully as a positioning aid [5, 10]. In the case considered in the aforesaid references, relative altitude (clearance) information is used to compute the location of a flying
Figure 1.2: Clearance measurement (blue lines) together with a reconstructed motion path of the navigating platform (black line) can be used to reconstruct a curve (red line) that can then be registered to the DTM surface. Such procedure yields the platform pose.

platform. The procedure consists of storing a data record of terrain altitudes under the platform and then matching this data to the DTM database. See Fig. 1.2 for illustration of the clearance method. Good results for this class of algorithms have been reported in the literature for systems capable of independently measuring motion to a good accuracy; note that motion knowledge is required in order to transform the clearance measurements into a 3D curve that can later be registered into the 3D terrain for the pose computation.

One might think that 2D visual projections should compare favorably with the 1D altitude measurement, but the scaling ambiguity involved in the reconstruction makes this an involved procedure. In a straightforward translation of clearance-based to vision-based approach, an SFM algorithm could be used to recover motion and reconstruct an un-scaled patch of the scene. This patch will be reconstructed with respect to the camera coordinates system. Thereafter, a registration algorithm could be used for matching the patch to the DTM and thus computing the camera pose (see Fig. 1.3). Such 3D registration can be performed e.g., by the “Iterative Closest Point” (ICP) algorithm that was first presented by Chen and Medioni [16]. Note that since the reconstructed patch is an un-scaled reconstruction of the true terrain, a similarity transformation (Euclidian transformation + scale) need to be computed for the registration. Variations of these concepts have been used in [73, 80]. One of the difficulties of this approach is that it requires the intermediate step of reconstructing the 3D geometry of the scene, which is prone to errors in the face of inaccurate information.
Figure 1.3: Two steps algorithm can be used for pose computation: first SFM algorithm is applied to reconstruct a patch of the observed terrain. Next, a registration algorithm is activated in order to find the transformation that connect between the DTM and the reconstructed patch. This transformation is the sought pose.
1.3.2 Landmark Subset Selection

As mentioned in section 1.1, some pose estimation algorithms can integrate an arbitrary number of landmarks in the pose computation, leading to accurate and numerically stable results. However, due to performance limitations, many real-time navigation systems are restricted to the use of only a very small number (usually 4-10) of landmarks. This limitation arises from the large overhead of detecting and tracking these landmarks along the image sequence. In [14], for example, a navigation system is presented where only four landmarks are simultaneously tracked.

If the number of available landmarks is small as well, the system will use all the visible landmarks at hand. However, if the system is equipped with a large landmark database, a subset needs to be selected from the visible landmarks as the camera moves. An example of such a scenario is an unmanned aerial vehicle (UAV) that utilizes a digital map and an ortho-photo of the observed terrain. In this configuration, the 3D location of any point on the terrain is known, and any visually distinctive point can thus be used as a landmark. The number of potential landmarks in such a case is large, and a subset must be chosen. Another example is a Simultaneous Localization and Mapping (SLAM) system such as the ones in [20, 22, 37, 76]. These systems estimate the camera motion and simultaneously track new features along the path of the robot’s movement. The 3D locations of the tracked features are reconstructed and added to a landmark database. As a result, the database is progressively enlarged and after a while there will be more visible landmarks than it is possible to track.

While the navigating platform moves, new landmark subsets should be occasionally selected. The need for a new subset may arise, for example, when one of the landmarks leaves the camera field of view or after the camera has moved more than a certain distance since the last subset was chosen. Whenever a new subset is required, an initial guess of the camera pose can be utilized to filter the landmarks which are supposed to be visible at the moment and to predict their projection location on the image before actually detecting them. At this stage we face an important question: how do we choose the subset from the filtered landmarks wisely, in a manner that will lead to the best pose estimate?

In most previous works (e.g., [76, 93, 94]), the landmark selection problem was addressed from the image appearance standpoint, where the 3D location of the landmarks was disregarded and the selection criterion based solely on a measure of distinction of the 2D features in the captured image. In [74] the region from which each landmark is visible constituted the selection criteria in order to construct a small landmark subset that can be used from a variety of locations.

14
In [13, 14, 85] the 3D structure of the selected landmark constellation and its influence on the obtained accuracies was studied. In [13], the design of special positioning objects (i.e., fiducials) was considered. The relatively small dimensions of such objects (compared to their distance from the imaging device) enabled the authors to assume weak-perspective projection, which is inadequate for general landmark-based navigation. In [14, 85], a full-perspective projection model was used; however, the navigation problem was restricted to a two-dimensional world where only three pose parameters had to be estimated.
Chapter 2

Motivation and Research Approach

The geometrical problem of computing the pose (position and orientation) of an observer or a camera based on the visible scene is one of the most ancient and studied problems in the computer vision field. Due to its relevancy to photogrammetry, pioneering solutions for this problem were developed starting from the middle of the 19’th century. Nevertheless, obtaining an accurate and robust estimate of the camera pose in real-time and with limited resources is still considered challenging task and occupies both academy and industry researches.

In order to compute the camera pose with respect to its surrounding environment one needs to possess some information about the environment. This information can be used to construct a system of constraints on the pose parameters. As mentioned in chapter 1, in the absence of such information we are restricted to follow the dead-reckoning framework which unavoidably yields an increasing drift of the poses estimates along time. SLAM (Simultaneously Localization And Mapping) algorithms, for example, use only the video stream in order to autonomously reconstruct the 3D structure of the scene and simultaneously use this reconstruction for the navigation. As long as the camera maneuvers in the same area, observing the same parts of the environment again and again, this technique may obtain fair results. However, it is unsuitable for a platform that moves ahead for a long time, facing new parts of the scene along its trajectory. In such a case significant drifts will occur like any other dead-reckoning technique.

Traditionally, the source of information about the environment is supplied in the form of a landmark set. Several distinctive features needed to be selected in the environment and their 3D location is measured and stored in a database. Next, whenever the pose of the camera is required, a subset of the visible landmarks are detected in the image. Using the 3D landmarks and their corresponding 2D projections on the image the camera pose parameters can be constrained and estimated. Although accurate, robust and efficient algorithms exists for estimating the camera pose given such data recent research in the navigation field has redirected its focus toward other directions. Several reasons led to this move; First, the construction of
landmarks database is a complex procedure, whether the landmarks are selected from the natural features of the scene or they are a set of special markers that were manually placed around the scene. Indeed, most applications that use landmarks are for indoors navigation, where it is easier to find or place the landmarks and their location can be relatively easily measured. Additionally, once we have constructed the landmarks database, the navigation algorithm still needs to detect them in the image and to establish the correct correspondences between the 3D landmarks and the detected 2D features. This task is also involved and thus real-time applications are compelled to use very limited number of landmarks.

The utilization of Digital Terrain Maps (DTM) for navigation is relatively new. These models supply information about the structure of the environment in the form of elevation measurements of the terrain. Similarly to the landmarks, this source of information can be utilized for constraining the parameters of the pose. However, the construction of these maps is performed automatically and no specific features needed to be detected or matched. Instead, the structure of the whole surface is supplied and some sort of video-to-surface correlation procedure is performed in order to derive the poses.

From the bird’s eye view, it can be stated that what unifies all pose estimation algorithms is that they utilize the available data – both the captured images and the information about the surrounding environment – in order to define a system of constraints on the navigation parameters. Next, some sort of optimization procedure is activated to estimate the sought parameters. The accuracy and robustness of the different algorithms are a direct result of the constraint system quality. As we have more information at our disposal the conditionality of the constraint systems may improve which in turn yield stronger algorithms. However, there may be several possible methods to utilize the environmental information for the navigation task. Some methods use this information in non-optimal manner while others fully utilize the available data. It is in the core of this research to examine proper utilization of the available data, specifically the DTM information, in order to achieve as accurate and robust navigation results as possible.

A possible framework for pose estimation that makes use of a DTM information was already presented in section 1.3.1. Such an algorithm, in a nutshell, is composed of three main steps. First, feature correspondences between consecutive video frames are computed. Note that unlike the correspondence problem we had to face in the landmark based algorithms, here we only need to track very local image displacements which is by far an easier problem. In the second step of the algorithm, the corresponding pairs are fed into a Structure-From-Motion (SFM) algorithm in order to reconstruct a patch from the viewed terrain and to estimate the camera motion between the two time instances. The patch is reconstructed with respect to the camera coordinate system at this stage. Lastly, the transformation that matches the reconstructed patch to the DTM surface is computed using some registration procedure. This transformation together with the
estimated motion gives us the sought parameters of the two camera poses.

While such approach has the advantage of using well known and studied vision algorithms as building-blocks, it suffers from unnecessary amplified sensitivity due to the correspondence computation and the SFM steps. At the SFM step, the depth of $n$ sampled points is estimated, where $n$ usually varies from few hundreds to few tens of thousands. Note that our final goal is to compute only 6 or 12 parameters (6 for each camera pose or ego-motion), far less than the $n$ parameters sought in the SFM step. Moreover, at that challenging step, the additional DTM information is completely overlooked. It is claimed in this work that such scheme do not utilizes the DTM information in an optimal manner.

In chapter 3 I present a new algorithm that utilizes the features correspondence together with the DTM to directly compute the camera pose at the two frames. The algorithm is based on the following observation. Since the DTM supplies information about the structure of the observed terrain, each hypothesized pose of the camera dictates the depths of the visible features. Hence, given the pose at two frames, the features’ displacement can be uniquely determined. The objective of the algorithm is to find the pose at the two frames which lead to features’ displacement as close as possible to the given corresponding pairs. This new framework skips the intermediate SFM step and thus makes better utilization of the DTM data. The performance of this algorithm is thoroughly studied using an analytic error analysis, computer simulations, and lab experimentation. Its superiority on the three steps approach is also shown.

One should notice that the observations about the SFM step may be said about the first step, where features correspondence between the two selected images are computed. Once again the number of unknowns is much higher that the 12 sought pose parameters, once again the DTM information is overlooked during the step. In my dissertation I take the aforementioned concepts one step ahead. This time I skip the features correspondence computation step and establish a direct connection between the gray-levels of the two images, the DTM and the sought 12 parameters. Basically, I follow the same ideas of the direct methods that were elaborated in section 1.2.1. However, in all previous direct methods, as far as I know, no external information beside the images was used. Hence, these algorithms settled for estimating the camera ego-motion and a 3D reconstruction of the scene w.r.t to the camera coordinates. The key contribution of my new algorithm which differentiates it from its direct method counterparts is the integration of the DTM information which enables the computation of the camera poses. The required adaptations to incorporate the DTM information in the scheme are elaborated in chapter 4.

In analogy to the concepts of the features based approach, here I say that in the presence of the DTM information each hypothesized pose of the camera dictates the depth at each of the image pixels. Hence, given the pose at two frames, the image displacements can be uniquely determined and the first image can be
“warped” accordingly toward the second image. Therefore, the objective of the direct method algorithm will be to find the poses at the two frames which produces a warping transformation that brings the first image as close as possible (according to pixel-to-pixel comparison) to the second image. A constraint therefore can be established for each of the first image’s pixels that has a corresponding point in the interior of the second image, and the parameters of the two camera poses can then be computed by solving the constraints system. This scheme incorporates the DTM data along all the navigation parameters estimation process. As such, the developed algorithm utilizes the available DTM information to the fullest and therefore obtain better results.

Along this section I use the terms “better results” or “better algorithms”. An interesting issue that was studied as part of my research is what stands behind these vague terms. In case one estimates a scalar, the variance of the estimate may serve as a natural choice for comparing the two resulting accuracies and decide which one is better. Pose, on the other hand, is defined by six parameters: three for position and another three for the camera orientation. In [14, 13] the condition number of the pose covariance matrix was used as a criterion for pose estimate comparison. This criterion does not reflect the different severity of errors in the different pose parameters. For example, a unit error in the camera’s position (e.g., 1 cm) should not be considered equivalent to an angular unit error (e.g., 1 radian) in its orientation. Additionally, the purpose of the pose computation should not be overlooked. The navigation system usually supports a control system that uses the pose estimates to perform some predefined task. According to the requirements of the specific task, some of the pose parameters may be considered more essential than others. For example, if the platform needs to follow a predefined path, then accurately identifying its location along the path is not as important as identifying any drifts from the path. Another example is the task of landing an airplane on a landing track. Obviously, the set of relevant parameters and accuracies during landing differs from those that need to be controlled for maintaining straight and level flight. It would thus be desirable to define a criterion that put its emphasis on the pose parameters that are more important for the performed task.

In chapter 5 I present new task-oriented criterion for pose quality evaluation. The system designer can use a severity function to specify the adequacy of different poses for the specific navigation task. This function is used to construct a requirements matrix that reflects the importance of the different pose parameters for the task at hand. Next, I present a new algorithm for landmarks subset selection which integrates the task-oriented considerations. This algorithm select a subset of landmarks that yield the best pose accuracies with respect to the performed task, according to the defined requirements matrix. Experimental results on simulated and real data are presented in that chapter as well.
Chapter 3

Pose Recovery from Feature Correspondences and a Digital Terrain Map

This chapter deals with the problem of estimating the pose of a calibrated camera using corresponding features in two consecutive images and a Digital Terrain (or Elevation) model DTM/DEM. In addition to that, the location and orientation of the camera at any moment is approximately known. Such a-priori estimate can be obtained using dead-reckoning navigation algorithm (see description in chapter 1).

As mentioned in chapter 2, such a problem could be easily solved using a three steps algorithm: first compute the corresponding pairs between the two images, then use SFM algorithm to reconstruct a patch of the terrain and lastly compute the best transformation that align the reconstructed patch and the DTM surface using some registration procedure such as the Iterative Closest Point (ICP) algorithm. Variations of this very basic idea have been used in [73] and [80]. However, as was already pointed out, the intermediate SFM step is a weak link that deteriorates the performance of the whole algorithm. It struggles a far more complicated problem than the one we are addressing since it estimates the depth of each of the tracked features (that might range from few hundreds to few tens of thousands). This is much more challenging task than our final goal – to estimate the two poses of the camera (one for each image). Additionally, during this step we overlook the DTM valuable information.

In the following chapter I present a new algorithm that skips the SFM intermediate step and directly compute the two sought camera poses using the DTM data. Therefore, this new scheme makes better utilization of the DTM data which yields a better conditioned constraints system. The algorithm is based on the following observation. Since the DTM supplies information about the structure of the observed terrain, each hypothesized pose of the camera dictates the depths of the visible features. Hence, given the pose at the two frames, the features’ displacement can be uniquely determined. The objective of the algorithm is to find the pose at the two frames which lead to features’ displacement as close as possible to
the given corresponding pairs. The performance of this algorithm is thoroughly studied using an analytic error analysis, computer simulations, and lab experimentation. Its superiority on the three steps approach is also shown.

3.1 The Two-View Constraint

This section contains a derivation of the two-view constraint, relating the feature correspondences with the two poses of the camera at the two time instances in which the relevant images were captured. The section begins by introducing some notation and a formal presentation of the problem of interest. After that, a solution for this problem is built. This solution is then compared with the more classical SFM result. Some properties of the solution are also discussed.

3.1.1 Notation and Problem Definition

The problem definition requires two coordinate systems:

- \( C(t) \) - Camera-fixed coordinate system at time \( t \), such that:
  - The origin of \( C(t) \) is at the center of projection of the camera.
  - The \( Z \)-axis coincides with the optical axis.
  - The \( X \) and \( Y \) axis lie parallel to the horizontal and vertical axis of the image plane.

- \( W \) - Global (i.e., “World”) coordinate system.

For discrete time instances \( t_i \), the somewhat simpler notation \( C_i = C(t_i) \) will often be used below. Moreover, the global coordinate system is chosen to facilitate a “flat earth” assumption\(^1\). The location and orientation of \( C(t) \) with respect to \( W \) will be denoted \( p(t) \) and \( R(t) \), respectively, with \( p(t) \in \mathbb{R}^3 \) and \( R(t) \) a rotation matrix. Throughout this paper, and whenever required for clarity, a left-superscript will be added to vectors to denote the coordinate-frame in which they are expressed (for example: \( ^Fv \) denotes the vector \( v \) in the \( F \) frame). Again for simplicity, the superscript will be dropped for vectors in the World coordinate system - see the definition of \( p(t) \) above. Also, \( R(t) \) will always denote the rotation from camera to world coordinates, so that, for instance:

\[
v = ^Wv = R(t)^Cv + p(t).
\]

\(^1\)For instance, \( W \) may be the local-level frame with the same origin as \( C(t_0) \) for some reference time \( t_0 \).
Consider now two consecutive time instances $t_1$ and $t_2$: the corresponding two frames of the camera will be referred to as $C_1$ and $C_2$, and likewise $p_i = p(t_i)$, $R_i = R(t_i)$ for $i = 1, 2$. The ego-motion transformation connecting the two frames is given by the translation vector $p_{12}$ (which is the position of the origin of the camera at $t_1$ under the $C_2$ frame) and the rotation matrix $R_{12} = R_2^T R_1$, such that

$$c_2 v = R_{12} c_1 v + p_{12}.$$  

The next ingredient in the formulation are the feature correspondence pairs. Given $Q_i \in \mathbb{R}^3$ a feature point, $\{\overrightarrow{u}_{ik}\} (i=1\ldots n, k=1,2)$, denote the perspective projections of the point on the image planes. More specifically, $\overrightarrow{u}_{i1} \in \mathbb{R}^2$ and $\overrightarrow{u}_{i2} \in \mathbb{R}^2$ represent the location in the image during the first and second frames, respectively. It is implicit in this notation that the correspondence problem has been solved between the projections in the two images. Assuming the focal distance has been conveniently normalized to unity, let $c_1 q_{i1}$ and $c_2 q_{i2}$ be $c_k q_{ik} = (\overrightarrow{u}_{ik}^T, 1)^T \in \mathbb{R}^3$. Although the $q_{ik}$ vectors are expressed in the corresponding camera frame, their left-superscript will be omitted throughout this paper for the sake of notation simplicity.

In this work the environmental information is given in the form of a DTM. For the discussion below, it is convenient to assume that the DTM is a function $h : \mathbb{R}^2 \rightarrow \mathbb{R}$ giving the altitude of the terrain, say, over the mean local sea level, for each geographical location $(U, V) \in \mathbb{R}^2$. Since $(U, V)$ are taken in $W$, this assumption involves extensive, database-dependent, albeit straightforward, manipulations with coordinate systems over the earth. When reporting numerical results in later chapters, the discrete and noisy nature of the data will be taken into account.

The constraint to be derived next assumes that the DTM can be linearized around a point. This in turn requires one to assume that a sufficiently good estimate of the pose of the camera at $t_1$ and its ego-motion between the two time instances are available. These estimates will be denoted by the subscript “$E$”, i.e., $p_{1E}, R_{1E}, p_{12E}$ and $R_{12E}$, to stress that these are a priori estimated quantities. Estimates can be obtained, for instance, from a dead-reckoning algorithm that uses inertial-system measurements.

With the notations introduced above, the **Pose and Motion from Correspondence and DTM** problem can be formulated as follows.

Given the following data:

1. A priori estimates for the camera pose and ego-motion $p_{1E}, R_{1E}, p_{12E}$ and $R_{12E}$.
2. Correspondence pairs $\{\overrightarrow{u}_{ik}\}$.
3. A DTM function $h : \mathbb{R}^2 \rightarrow \mathbb{R}$.

Find the true pose and motion $p_1$, $R_1$, $p_{12}$ and $R_{12}$ of the camera.
In practice, the presence of noise will not allow the computation of true pose and motion, and hence one should settle for \textit{a posteriori} estimates of these quantities. The reader should note that estimating the camera pose at the first time instance together with the ego-motion between the two time instances is completely equivalent to estimating the two poses of the camera at the two time instances. One formulation is obtain from the other by a simple transformations composition.

3.1.2 Single-Frame Geometry

To begin the discussion, a single feature point on the terrain, $Q_T$, will be considered in this and the next sections. Assuming a pinhole model for the calibrated camera, this feature is perspectively projected onto a point $q_1$ on the image-plane of the first camera frame $C_1$. The present section concentrates on the single-view geometry that will eventually lead to the two-view geometry discussed in the next section.

Using an initial guess of the camera pose at $t_1$, the line passing through $p_{E_1}$ along the direction of $q_1$ can be intersected with the DTM. A ray-tracing algorithm can be used for this purpose. The intersection point can be computed as

$$Q_E = p_{E_1} + \lambda_E R_{E_1} q_1,$$

for some $\lambda_E$ computed, e.g., by using ray-tracing. The subscript letter “$E$” again highlights the fact that this point is an estimated location. The true feature location $Q_T$ can similarly be expressed by:

$$Q_T = p_1 + \lambda_T R_1 q_1,$$

and in general, $Q_E \neq Q_T$. There are two main error sources that explain the difference between $Q_T$ and $Q_E$: the error in the a priori estimates for the pose and the errors in the determination of $Q_E$ caused by DTM discretization, as well as intrinsic errors. However, it is assumed that for reasonable a priori estimates and DTM-related errors, the two points are sufficiently close so that $Q_T$ can be approximated as belonging to a plane tangent to the DTM at the point $Q_E$; see Fig. 3.1. Specifically, if $N$ denotes the normal to the DTM at $Q_E$, then:

$$N^T (Q_T - Q_E) = 0.\quad (3.1.3)$$

The parameter $\lambda_T$ is the depth of the feature point and encodes the information about the structure of the scene. In order to avoid the structure reconstruction, the linearization assumption can be used to eliminate $\lambda_T$ from the expressions above. Indeed, from (3.1.2),

$$Q_T - Q_E = p_1 + \lambda_T R_1 q_1 - Q_E,$$

\textit{Technion - Computer Science Department - Ph.D. Thesis  PHD-2008-07 - 2008}
Figure 3.1: The terrain feature, $Q_T$, is perspectively projected to the image plane point $q_1$ under the true first camera frame (where $p_1$ represents its position and $R_1$ its orientation). Using this projected point and the estimated pose of the camera ($p_{E_1}$ and $R_{E_1}$), the ray from $p_{E_1}$ in the direction of $R_{E_1}q_1$ can be intersected with the DTM at $Q_E$. The DTM is linearized around this point and $Q_T$ is assumed to lie on that tangent plane.
and hence using (3.1.3) and after some re-ordering:

\[ N^T (p_1 - Q_E) + \lambda_T N^T R_1 q_1 = 0, \]

implying:

\[ \lambda_T = -\frac{N^T (p_1 - Q_E)}{N^T R_1 q_1}. \] (3.1.5)

The above expresses the depth of the scene point as a function of the camera pose and the (known) linearization plane parameters. It is this expression which enables us to avoid handling the depths of the scene points as unknowns. Instead, this expression can be replaced in our equations which will eventually result in a system of 12 unknowns (of the pose and motion parameters) instead of \( n + 12 \) unknowns for \( n \) tracked features.

Replacing (3.1.5) in (3.1.4) and grouping the different terms, one gets:

\[ Q_T - Q_E = \left( I - \frac{R_1 q_1 N^T}{N^T R_1 q_1} \right) (p_1 - Q_E). \] (3.1.6)

In order to further simplify the expression and facilitate its geometrical interpretation, the following projection operator is introduced:

\[ P(u, n) \triangleq \left( I - \frac{un^T}{n^T u} \right). \] (3.1.7)

This operator projects vectors onto the plane orthogonal to \( n \). Notice that the projection is not orthogonal but rather along the direction of \( u \). By using the above definition, it is straightforward to verify that \( n^T \cdot P(u, n) \equiv 0 \) and \( P(u, n) u \equiv 0 \). See Fig. 3.2(a) for a geometrical interpretation of \( P \).

Using the above operator, one can rewrite (3.1.6) as:

\[ Q_T - Q_E = P(R_1 q_1, N) (p_1 - Q_E), \] (3.1.8)

with the operator

\[ P(R_1 q_1, N) = \left( I - \frac{R_1 q_1 N^T}{N^T R_1 q_1} \right) \] (3.1.9)

projecting vectors onto the tangent plane to the DTM at \( Q_E \) along the direction of \( R_1 q_1 \).

Eqn. (3.1.8) has a nice geometric interpretation as shown in Fig. 3.2(b). The unknown vector \( Q_T - Q_E \) is the vector from \( Q_E \) to \( Q_T \) in the frame \( W \). It can be obtained by taking the vector from \( Q_E \) to \( p_1 \) and using the \( P \)-operator in order to project it onto the linearization plane orthogonal to \( N \) along the \( q_1 \) direction (\( R_1 q_1 \) will be used since the world’s frame representation of \( q_1 \) is required).
Figure 3.2: (a) The vector $v$ is being projected by $\mathcal{P}(u, n)$ onto the plane orthogonal to $n$ along the direction of $u$. (b) In order to obtain the unknown vector $Q_T - Q_E$, the vector $p_1 - Q_E$ is being projected onto the linearization plane and along the $R_1 q_1$ direction using the $\mathcal{P}$ projection operator.

### 3.1.3 Two-Frame Geometry

Suppose next that a second frame is now available. Then, the location of the feature point in the $C_2$ frame can be expressed as:

$$c_2 Q_T = p_{12} + R_{12} c_1 Q_T.$$  \hspace{1cm} (3.1.10)

Since

$$Q_T = p_1 + R_1 c_1 Q_T,$$

(3.1.10) can also be expressed as:

$$c_2 Q_T = p_{12} + R_{12} [R_1^T (Q_T - p_1)] = p_{12} + R_2^T (Q_T - p_1).$$  \hspace{1cm} (3.1.11)

Using a standard re-projection argument, one can claim that $c_2 Q_T$ can also be written using its projection onto the image plane:

$$c_2 Q_T = \lambda_{T2} q_2,$$  \hspace{1cm} (3.1.12)

where $\lambda_{T2}$ is the depth of the feature in $C_2$. To eliminate the dependence on the depth, use the equality:

$$\left( I - \frac{q_2 q_2^T}{q_2^T q_2} \right) q_2 = \mathcal{P} (q_2, q_2) q_2 = 0.$$  \hspace{1cm} (3.1.13)
For ease of notation, call
\[ \mathcal{P} (q_2) \doteq \mathcal{P} (q_2, q_2). \]

Using this in (3.1.11), one gets the constraint
\[ \mathcal{P} (q_2) [p_{12} + R_2^T (Q_T - p_1)] = 0. \tag{3.1.14} \]

The last step in getting a useful constraint and avoiding structure reconstruction is to replace \( Q_T - p_1 = (Q_T - Q_E) + (Q_E - p_1) \) in the equation above and use the one-view geometry constraint (3.1.8) to get:
\[ \mathcal{P} (q_2) [p_{12} + R_2^T (I - \mathcal{P} (R_1 q_1, N)) (Q_E - p_1)] = 0. \tag{3.1.15} \]

Using the definition of the projection operator (3.1.7):
\[ \mathcal{P} (q_2) \left[ p_{12} + R_2^T \frac{R_1 q_1 N^T}{N^T R_1 q_1} (Q_E - p_1) \right] = 0, \tag{3.1.16} \]
or:
\[ \mathcal{P} (q_2) \left[ p_{12} + \frac{R_1 q_1 N^T}{N^T R_1 q_1} (Q_E - p_1) \right] = 0. \tag{3.1.17} \]

This basic constraint involves all pose and ego-motion parameters defining the two frames of the camera and involves the measurements in the image plane and the estimated location for the feature point \( Q_E \). The pose and ego-motion parameters are, therefore, constrained to verify this equation.

**Remark:** Notice that (3.1.15) and its variants are trivially verified by multiplying by \( q_2^T \) on the left, so that this equation is equivalent to two – and not three – linearly independent equations.

### 3.1.4 Multiple Features

Suppose next that \( n \) feature points are tracked in two frames, so that the estimated locations \( Q_{Ei} \) and projections onto the image plane \( q_{1i} \) and \( q_{2i} \) are estimated and measured, respectively, for \( i = 1, \ldots, n \). Associated with each \( Q_{Ei} \) is the normal vector to the DTM at this point, namely \( N_i \). Taking this into account, one can re-write (3.1.17) in matrix form as:
\[
\begin{bmatrix}
-\mathcal{P} (q_{21}) & \mathcal{P} (q_{21}) \frac{R_{12} q_{11} N_{11}^T}{N_{11}^T R_{12} q_{11}} & p_{12} \\
-\mathcal{P} (q_{22}) & \mathcal{P} (q_{22}) \frac{R_{12} q_{12} N_{12}^T}{N_{12}^T R_{12} q_{12}} & p_{12} \\
& & \\
-\mathcal{P} (q_{2n}) & \mathcal{P} (q_{2n}) \frac{R_{12} q_{1n} N_{1n}^T}{N_{1n}^T R_{12} q_{1n}} & p_{12}
\end{bmatrix}
= \begin{bmatrix}
\mathcal{P} (q_{21}) \frac{R_{12} q_{11} N_{11}^T}{N_{11}^T R_{12} q_{11}} Q_{E1} \\
\mathcal{P} (q_{22}) \frac{R_{12} q_{12} N_{12}^T}{N_{12}^T R_{12} q_{12}} Q_{E2} \\
& & \\
\mathcal{P} (q_{2n}) \frac{R_{12} q_{1n} N_{1n}^T}{N_{1n}^T R_{12} q_{1n}} Q_{En}
\end{bmatrix}.	ag{3.1.18}
\]

Repeating this for each feature point:
\[
\begin{bmatrix}
-\mathcal{P} (q_{21}) & \mathcal{P} (q_{21}) \frac{R_{12} q_{11} N_{11}^T}{N_{11}^T R_{12} q_{11}} & p_{12} \\
-\mathcal{P} (q_{22}) & \mathcal{P} (q_{22}) \frac{R_{12} q_{12} N_{12}^T}{N_{12}^T R_{12} q_{12}} & p_{12} \\
& & \\
-\mathcal{P} (q_{2n}) & \mathcal{P} (q_{2n}) \frac{R_{12} q_{1n} N_{1n}^T}{N_{1n}^T R_{12} q_{1n}} & p_{12}
\end{bmatrix}
= \begin{bmatrix}
\mathcal{P} (q_{21}) \frac{R_{12} q_{11} N_{11}^T}{N_{11}^T R_{12} q_{11}} Q_{E1} \\
\mathcal{P} (q_{22}) \frac{R_{12} q_{12} N_{12}^T}{N_{12}^T R_{12} q_{12}} Q_{E2} \\
& & \\
\mathcal{P} (q_{2n}) \frac{R_{12} q_{1n} N_{1n}^T}{N_{1n}^T R_{12} q_{1n}} Q_{En}
\end{bmatrix}.	ag{3.1.19}
\]
In compact notation:

\[
\mathcal{A}_n \begin{bmatrix} p_{12} \\ p_1 \end{bmatrix} = \mathcal{B}_n. \tag{3.1.20}
\]

Note that \(\mathcal{A}_n\) and \(\mathcal{B}_n\) depend on known quantities: the estimated features, the normals of the DTM tangent planes, and the images of the features at the two time instances, together with the unknown orientation \(R_1\) and the relative rotation \(R_{12}\). At this point in our discussion, several remarks are in order.

**Remark 1:** The constraint (3.1.19) involves twelve “unknowns”, namely the pose and ego-motion of the camera. From the remark at the end of the previous section, the equation involves at most \(2n\) linearly independent constraints, so that at least six features at different locations \(Q_{Ti}\) are required to have a determinate system of equations. Usually, more vectors will be used in order to define an over-determined system, and hence reduce the effect of noise. Clearly, there are degenerate scenarios in which the obtained system is singular, no matter what is the number of available features. Examples for such scenarios include flying above completely planar or spherical terrain (see Section 3.1.5). However, in the general case where the terrain has “interesting” structure the system is non-singular and the twelve parameters can be obtained.

**Remark 2:** The constraint (3.1.19) is non-linear and, therefore, no analytic solution to it is readily available. Thus, an iterative scheme will be used in order to solve this system. A robust algorithm using Newton-iterations and M-estimator will be described in following sections.

**Remark 3:** Given Remark 2, one observes that the location and translation appear linearly in the constraint. Using the pseudo-inverse, these two vectors can be solved explicitly to give:

\[
\begin{bmatrix} p_{12} \\ p_1 \end{bmatrix} = \mathcal{A}_n^\dagger \mathcal{B}_n, \tag{3.1.21}
\]

so that, after resubstituting in (3.1.20):

\[
(I - \mathcal{A}_n \mathcal{A}_n^\dagger) \mathcal{B}_n = 0. \tag{3.1.22}
\]

This remark leads to two conclusions:

1. If the rotation is known to good accuracy and measurement noise is relatively low, then the position and translation can be determined by solving a linear equation. This fact may be relevant when “fusing” the procedure described here with other measurement, e.g., with inertial navigation.

2. Equation (3.1.22) shows that the estimation of rotation (both absolute and relative) can be separated from that of location/translation. This fact is also found when estimating pose from a set of visible
Figure 3.3: Examples to constellations which lead to singularities of the algorithm. (a) features from a surface which can be swept out by moving a planar curve along constant direction, (b) features laying on a silhouette of arbitrary surface, (c) surface of revolution, (d) spiral landmarks as shown in [56]. In that work, similarly to the present, the estimate is obtained by minimizing an objective function which measures the errors in the object-space rather than on the image plane (as in most other works). This property enables the decoupling of the estimation problem. Note however that [56] addresses only the pose rotation and translation decoupling while here the 6 parameters of absolute and relative rotations are separated from the 6 parameters of the camera location and translation.

3.1.5 Degenerate Scenarios

The proposed algorithm utilizes the information derived from the 2D movement of the tracked features on the image plane. It relies on the assumption that these movements dictate the ego-motion of the camera and the structure of the 3D features up to similarity. Next, the additional information supplied by the DTM is assumed to dictate the unknown similarity transformation by restricting the 3D features to lay on the terrain. However, in any case one of these assumptions does not hold a degenerate scenario arises and thus a singular system of constraints will be obtained.

Pure rotational ego-motion is a classic scenario where the first assumption does not hold. It is well established that under such motion the depth of the 3D feature has no influence on the projected features displacement. Collinear features are another example where the ego-motion cannot be determined.

Intuitive examples for scenarios where the second assumption does not hold include a planar or spherical terrain. Once the 3D structure of the features constellation was derived (from the image displacements) a whole manifold of solutions embedded in the similarities configuration space is adequate. In order to study the conditions under which the terrain surface yields a degenerate scenario I follow the Constraint Analysis
proposed by [81] and extend it from Euclidean transformations to similarities.

Assuming one is supplied with the true similarity (which registers the 3D features into the terrain) as an initial-guess for the algorithm, a degenerate scenario could be differentially characterized by the existence of infinitesimal perturbation of the similarity parameters such that the quality of the registration will not deteriorate. Let \( Q = C_2Q_T = C_2Q_E \) be a 3D feature laying on its corresponding tangent plane. By applying an infinitesimal translation of \( \delta t \in \mathbb{R}^3 \), scale of \( 1 + \delta s \in \mathbb{R} \) and rotation of \( \|\delta \omega\| \) around the \( \delta \omega \in \mathbb{R}^3 \) axis, this features is transformed to:

\[
Q' = (1 + \delta s)(Q + \delta \omega \times Q) + \delta t.
\] (3.1.23)

This equation is obtained from the first order approximation of Rodrigues formula. Therefore, a degenerate scenario arise when there are non-all-zero \( \delta t, \delta \omega, \delta s \) such that:

\[
C_2N_i^T \left[ Q'_i - Q_i \right] = C_2N_i^T \left[ \delta s \cdot Q_i + (1 + \delta s) (\delta \omega \times Q_i) + \delta t \right] = 0
\] (3.1.24)

for all tracked features (\( i = 1...n \)). The above constraint verifies that the vector of the 3D feature displacement induced by the similarity perturbation is parallel to the corresponding tangent plane and thus has no effect on the registration quality. One should notice that in case such perturbation is found, the scaled perturbation \( \lambda \cdot \delta t, \lambda \cdot \delta \omega, \lambda \cdot \delta s \) (for any \( \lambda \in \mathbb{R} \)) should also verify the constraint in order to create a whole subspace of adequate solutions in the similarities configuration space:

\[
C_2N_i^T \left[ \Delta Q_i(\lambda) \right] = 0,
\] (3.1.25)

where

\[
\Delta Q_i(\lambda) = \lambda \delta s \cdot Q_i + \lambda \delta \omega \times Q_i + \lambda^2 \delta s (\delta \omega \times Q_i) + \lambda \delta t.
\] (3.1.26)

Dividing (3.1.25) by \( \lambda \), subtracting the result from (3.1.24) and once again dividing by \( 1 - \lambda \) yields:

\[
\delta s (\delta \omega \times Q_i) = 0.
\] (3.1.27)

Since the 3D features are not collinear not all of them are parallel to \( \delta \omega \). Therefore, either \( \delta s = 0 \) or \( \delta \omega = 0 \).

In case \( \delta \omega = 0 \) equation (3.1.24) reduces to:

\[
C_2N_i^T \left[ \delta s \cdot Q_i + \delta t \right] = 0.
\] (3.1.28)

If \( \delta s = 0 \) then we remain with \( C_2N_i^T \delta t = 0 \). This means that \( \delta t \) is orthogonal to all the normal vectors - \( N_i \) which therefore must be coplanar. Surfaces with such characteristic are those which can be swept out by moving a planar curve along the \( \delta t \) direction (see Fig. 3.3 (a)). If \( \delta s \neq 0 \) then it can be assumed that \( \delta s = 1 \)
(since the scale of the perturbation is arbitrary). This leads to the constraint \( C_2 N_i^T (Q_i + \delta t) = 0 \) which means that after moving the camera by \( \delta t \) all 3D features belong to the surface’s silhouette (see Fig. 3.3 (b)).

In the presence of rotational perturbation (\( \delta \omega \neq 0 \)) there is no scale change. Hence, equation (3.1.24) reduces to:

\[
C_2 N_i^T [\delta \omega \times Q_i + \delta t] = 0
\] (3.1.29)

In the special case where \( \delta \omega \perp \delta t, \delta t \) can be expressed as a cross-product of \( \delta \omega \) and some \( d \in \mathbb{R}^3 \) which lead to the following representation of (3.1.29): \( C_2 N_i^T [\delta \omega \times (Q_i + d)] = 0 \). This equation shows that after translating the 3D features by \( d \) the rotation-axis \( \delta \omega \), the translated feature \( (Q_i + d) \) and the surface normal are coplanar. Such behavior is obtained from surfaces of revolution such as a sphere or a Gaussian hill (see Fig. 3.3 (c)). In the general case of arbitrary translation \( \delta t \) can be decomposed into two components: \( \delta t^\perp \) orthogonal to \( \delta \omega \) and \( \delta t^\parallel \) parallel to it. Therefore, one obtains \( C_2 N_i^T [\delta \omega \times (Q_i + d) + \delta t^\parallel] = 0 \). Surfaces which are consistent with that constraint include cylinder, spiral and others (see Fig. 3.3 (d)).

Remark: Since only infinitesimal perturbations are considered only small surface environment of each feature is significant for singularity conditions. Therefore, it is enough that the surface will satisfy the above conditions piecewise.

### 3.1.6 The Epipolar Constraint Connection

Before proceeding any further, it is interesting to look at (3.1.17) in the light of previous work in SFM and, in particular, epipolar geometry. In order to do this, it is worth deriving the basic constraint in the present framework and notation. Write:

\[
C_2 Q_T = \lambda_2 q_2 = p_{12} + \lambda_1 R_{12} q_1
\] (3.1.30)

for some scalars \( \lambda_1 \) and \( \lambda_2 \) (see Fig. 3.4). It follows that:

\[
p_{12} \times \lambda_2 q_2 = p_{12} \times \lambda_1 R_{12} q_1,
\] (3.1.31)

and hence:

\[
q_2^T (p_{12} \times R_{12} q_1) = 0.
\] (3.1.32)

For a vector \( x \in \mathbb{R}^3 \), let \( x^\wedge \) denote the skew-symmetric matrix:

\[
x^\wedge = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}
\]
Figure 3.4: The examined scenario from the second camera frame’s ($C_2$) point of view. $q_2$ is the perspective projection of the terrain feature $^C_2Q_T$, and thus the two should coincide. Additionally, since $q_1$ is also a projection of the same feature in the $C_1$-frame, the epipolar constraint requires that the two rays (one in the direction of $q_2$ and the other from $p_{12}$ in the direction of $R_{12}q_1$) will intersect.
Then, it is well known that the vector product between two vectors $x$ and $y$ can be expressed as:

$$x \times y = x^\wedge y.$$ 

Using this notation, the epipolar constraint (3.1.32) can be written as:

$$q_2^T (R_{12} q_1)^\wedge p_{12} = 0$$  \hspace{1cm} (3.1.33)

and symmetrically as:

$$q_1^T R_{12}^T q_2^\wedge p_{12} = 0$$  \hspace{1cm} (3.1.34)

The important observation here is that if the vector $p_{12}$ verifies the above constraint, then the vector $\kappa \cdot p_{12}$ also verifies the constraint, for any number $\kappa$. This is an expression of the ambiguity built into the SFM problem. On the other hand, the constraint (3.1.17) is non-homogeneous and hence does not suffer from the same ambiguity. In terms of the translation alone (and for only one feature point!), if $p_{12}$ verifies (3.1.17) for given $R_1$ and $R_{12}$, then also $p_{12} + \kappa q_2$ will verify the constraint, and hence the ego-motion translation is defined up to a one-dimensional vector. However, one has the following trivially:

$$q_1^T R_{12}^T q_2^\wedge q_2 = 0,$$  \hspace{1cm} (3.1.35)

and hence the epipolar constraint does not provide an additional equation that would allow us to solve for the translation in a unique manner. Moreover, observe that (3.1.17) can be written using a vector product instead of the projection operator as:

$$q_2^\wedge \left[ p_{12} + \frac{R_{12} q_1 N^T}{N^T R_1 q_1} (Q_E - p_1) \right] = 0.$$  \hspace{1cm} (3.1.36)

Taking into account the identity

$$(R_{12} q_1)^T q_2^\wedge R_{12} q_1 \equiv 0,$$  \hspace{1cm} (3.1.37)

it is possible to conclude that (3.1.36) $\longrightarrow$ (3.1.34), and hence the new constraint "contains" the classical epipolar geometry. Indeed, one could think of the constraint derived in (3.1.17) as strengthening the epipolar constraint by requiring not only that the two rays (in the directions of $q_1$ and $q_2$) should intersect, but, in addition, that this intersection point should lie on the DTM’s linearization plane. Observe, moreover, that taking more than one feature point would allow us to completely compute the translation (at least for the given rotation matrices).
3.2 Error analysis

The following section deals with the error-analysis of the proposed algorithm. By assuming appropriate characterization of the error sources, a closed form expression for the uncertainty of the estimated pose and motion is developed. Then, the influence of different factors is studied using extensive numerical simulations.

3.2.1 Pose and Motion Uncertainty Computation

Let \( R_1 \) be determined by the three Euler angles \((\phi_1, \theta_1, \psi_1)\), and similarly \( R_{12} \) by the Euler angles \((\phi_{12}, \theta_{12}, \psi_{12})\). In order to evaluate the algorithm’s performance, the objective-function of the minimization process needs to be defined first: For each of the \( n \) tracked features, the function \( f_i : \mathbb{R}^{12} \rightarrow \mathbb{R}^3 \) is defined as the left-hand side of the constraint described in (3.1.15):

\[
f_i(p_1, \phi_1, \theta_1, \psi_1, p_{12}, \phi_{12}, \theta_{12}, \psi_{12}) = \mathcal{P}(q_{2i}, q_{2i}) [p_{12} + R_{12} \mathcal{L}(R_1q_1i, N_i)(Q_{E_i} - p_1)] \tag{3.2.38}
\]

where \( \mathcal{L}(u, n) \) is defined as the complement of the \( \mathcal{P}(u, n) \) projection operator:

\[
\mathcal{L}(u, n) = \frac{uu^T}{n^Tu}, \tag{3.2.39}
\]

This operator projects vectors on \( u \)'s direction along a plane with normal \( n \). For the sake of notations simplicity \( \mathcal{L} \) will denote the expression: \( \mathcal{L}(R_1q_1i, N_i) \). In (3.2.38), \( R_{12} \) and \( \mathcal{L} \) are functions of \((\phi_{12}, \theta_{12}, \psi_{12})\) and \((\phi_1, \theta_1, \psi_1)\) respectively. Additionally, the function \( F : \mathbb{R}^{12} \rightarrow \mathbb{R}^{3n} \) will be defined as the concatenation of the \( f_i \) functions:

\[
F(p_1, \phi_1, \theta_1, \psi_1, p_{12}, \phi_{12}, \theta_{12}, \psi_{12}) = [f_1, \ldots, f_n]^T.
\]

According to these notations, the goal of the algorithm is to find the twelve parameters that minimize \( M(\theta, D) = \|F(\theta, D)\|^2 \), where \( \theta \) represents the 12-vector of the parameters to be estimated, and \( D \) is the concatenation of all the data obtain from the feature correspondences and the DTM. If \( D \) would have been free of errors, the true parameters were obtained. Since \( D \) contains some error perturbation, the estimated parameters are drifted to erroneous values. It has been shown in [33] that the connection between the uncertainty of the data and the uncertainty of the estimated parameters can be described by the following first-order approximation:

\[
\Sigma_{\theta} = \left( \frac{dg}{d\theta} \right)^{-1} \Sigma_D \left( \frac{dg}{dD} \right)^T \left( \frac{dg}{d\theta} \right)^{-1} \tag{3.2.40}
\]
Here, $\Sigma_\theta$ and $\Sigma_D$ represent the covariance matrices of the parameters and the data respectively. $g$ is defined as follows:

$$g(\theta, D) = \frac{d}{d\theta} M(\theta, D) = \frac{d}{d\theta} F^T F = 2J_\theta^T F$$ (3.2.41)

$J_\theta = dF/d\theta$ is the $(3n \times 12)$ Jacobian matrix of $F$ with respect to the twelve parameters. By ignoring second-order elements, the derivations of $g$ can be approximate by:

$$\frac{dg}{d\theta} \approx 2J_\theta^T J_\theta$$ (3.2.42)

$$\frac{dg}{dD} \approx 2J_\theta^T J_D$$ (3.2.43)

$J_D = dF/dD$ is defined in a similar way as the $(3n \times m)$ Jacobian matrix of $F$ with respect to the $m$ data components. Assigning (3.2.42) and (3.2.43) back into (3.2.40) yield the following expression:

$$\Sigma_\theta = (J_\theta^T J_\theta)^{-1} J_\theta^T J_D \Sigma_D J_D^T J_\theta (J_\theta^T J_\theta)^{-1}$$ (3.2.44)

The central component $J_D \Sigma_D J_D^T$ represents the uncertainties of $F$ while the pseudo-inverse matrix $(J_\theta^T J_\theta)^{-1} J_\theta^T$ transfers the uncertainties of $F$ to those of the twelve parameters. In the following subsections, $J_\theta$, $J_D$ and $\Sigma_D$ are explicitly derived.

**$J_\theta$ Calculation**

Simple derivations of $f_i$ which is presented in (3.2.38), yield the following results:

$$\frac{df}{dp_1} = -P(q_2, q_2) R_{12} \mathcal{L}$$ (3.2.45)

$$\frac{df}{d\alpha_1} = -P(q_2, q_2) R_{12} \mathcal{L} \left( \frac{d}{d\alpha_1} R_1 \right) \mathcal{L} (Q_E - p_1)$$ (3.2.46)

$$\frac{df}{dp_{12}} = P(q_2, q_2)$$ (3.2.47)

$$\frac{df}{d\alpha_{12}} = P(q_2, q_2) \left( \frac{d}{d\alpha_{12}} R_{12} \right) \mathcal{L} (Q_E - p_1)$$ (3.2.48)

In expressions (3.2.46) and (3.2.48): $\alpha_1 = \phi_1, \theta_1, \psi_1$ and: $\alpha_{12} = \phi_{12}, \theta_{12}, \psi_{12}$. The Jacobian $J_\theta$ is obtained by simple concatenation of the above derivations.
Before calculating $J_D$, the data vector $D$ must be explicitly defined. Two types of data are being used by the proposed navigation algorithm: data obtained from the feature correspondences and data obtained form the DTM. Each tracked feature is located at $q_1$ in the first image and at $q_2$ in the second image. One can consider $q_1$’s location as an arbitrary choice of some ground feature projection, while $q_2$ represent the new projection of the same feature on the second frame. Thus the feature correspondences error is realized through the $q_2$ locations.

The DTM errors influence the $Q_E$ and $N$ vectors in the constraint equation. As before, the DTM linearization assumption will be used. For simplicity the derived orientation of the terrain’s local linearization, as expressed by the normal, will be considered as correct while the height of this plane might be erroneous. The connection between the height error and the error of $Q_E$ will be derived in the next subsection. Resulting from the above, the $q_1$’s and the $N$’s can be omitted from the data vector $D$. It will be defined as the concatenation of all the $q_2$’s followed by concatenation of the $Q_E$’s.

The i’th feature’s data vectors: $q_2$, and $Q_E$, appears only in the i’th feature constraint, thus the obtained Jacobian matrix $J_D = [J_q, J_Q]$ is a concatenation of two block diagonal matrices: $J_q$ followed by $J_Q$. The i’th diagonal block element is the $3 \times 3$ matrix $df_i/dq_2$ and $df_i/dQ_E$ for $J_q$ and $J_Q$ respectively:

$$\frac{df}{dq_2} = -\frac{1}{\|q_2\|^2} \left[ (q_2^T \cdot c_2 Q) I + q_2 \cdot c_2 Q^T \right] P(q_2, q_2) \quad (3.2.49)$$

$$\frac{df}{dQ_E} = P(q_2, q_2) R_{12} L \quad (3.2.50)$$

$c_2 Q$ in expression (3.2.49) is the ground feature $Q$ under the second camera frame as defined in (3.1.11).

### $\Sigma_D$ Calculation

As mention above, the data-vector D is constructed from concatenation of all the $q_2$’s followed by concatenation of the $Q_E$’s. Thus $\Sigma_D$ should represent the uncertainty of these elements. Since the $q_2$’s and the $Q_E$’s are obtained from two different and uncorrelated processed the covariance relating them will be zero, which leads to a two block diagonal matrix:

$$\Sigma_D = \begin{bmatrix} \Sigma_q & 0 \\ 0 & \Sigma_Q \end{bmatrix} \quad (3.2.51)$$

In this work the errors of image locations and DTM height are assumed to be additive zero-mean Gaussian distributed with standard-deviation of $\sigma_I$ and $\sigma_h$ respectively. Each $q_2$ vector is a projection on the image
plane where a unit focal-length is assumes. Hence, there is no uncertainty about its \( z \)-component. Since a normal isotropic distribution was assumed for the sake of simplicity, the covariance matrix of the image measurements is defined to be:

\[
\Sigma_{q_i} = \sigma_i^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\]  
(3.2.52)

and \( \Sigma_q \) is the matrix with the \( \Sigma_{q_i} \)'s along its diagonal.

In [72] the accuracy of location’s height obtained by interpolation of the neighboring DTM grid points is studied. The dependence between this accuracy and the specific required location, for which height is being interpolated, was found to be negligible. Here, the above finding was adopted and a constant standard-deviation was set to all DTM heights measurements. Although there is a dependence between close \( Q \)'s uncertainties, this dependence will be ignored in the following derivations for the sake of simplicity. Thus, a block diagonal matrix is obtained for \( \Sigma_Q \) containing the \( 3 \times 3 \) covariance matrices \( \Sigma_{Q_i} \) along its diagonal which will be derived as follows: consider the ray sent from \( p_1 \) along the direction of \( R_1 q_1 \). This ray should have intersected the terrain at \( Q_E = p_1 + \lambda R_1 q_1 \) for some \( \lambda \), but due to the DTM height error the point \( \tilde{Q}_E = (\tilde{x}, \tilde{y}, \tilde{h})^T \) was obtained. Let \( h \) be the true height of the terrain above \( (\tilde{x}, \tilde{y}) \) and \( H = (\tilde{x}, \tilde{y}, h) \) be the 3D point on the terrain above that location.

Using that \( H \) belongs to the true terrain plane one obtains:

\[
N^T (Q_E - H) = N^T (p_1 + \lambda R_1 q_1 - H) = 0
\]  
(3.2.53)

Extracting \( \lambda \) from (3.2.53) and assigning it back to \( Q_E \)'s expression yields:

\[
Q_E = p_1 + R_1 L (H - p_1)
\]  
(3.2.54)

For \( Q_E \)'s uncertainty calculation the derivative of \( Q_E \) with respect to \( h \) should be found:

\[
\frac{dQ_E}{dh} = R_1 L \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T = \frac{R_1 q_1}{N^T R_1 q_1}
\]  
(3.2.55)

The above result was obtained using the fact that the \( z \)-component of \( N \) is 1: \( N = \begin{bmatrix} -\nabla DTM & 1 \end{bmatrix}^T \).

Finally, the uncertainty of \( Q_E \) is expressed by the following covariance-matrix:

\[
\Sigma_{Q_i} = \left( \frac{dQ_E}{dh} \right) \cdot \sigma_h^2 \left( \frac{dQ_E}{dh} \right)^T = \sigma_h^2 \cdot \frac{R_1 q_1 R_1^T}{(N^T R_1 q_1)^2}
\]  
(3.2.56)
\[ \Sigma_{C_2} \text{ Calculation} \]

The algorithm presented in this work estimates the pose of the first camera frame and the ego-motion. Usually, the most interesting parameters for navigation purpose will be the second camera frame since it reflect the most updated information about the platform location. The second pose can be obtained in a straightforward manner as the composition of the first frame pose together with the camera ego-motion:

\[ p_2 = p_1 - R_1 R_{12}^T p_{12} \]  
\[ R_2 = R_1 R_{12}^T \]  

(3.2.57)  
(3.2.58)

The uncertainty of the second pose estimates will be described by a \( 6 \times 6 \) covariance matrix that can be derived from the already obtained \( 12 \times 12 \) covariance matrix \( \Sigma_\theta \) by multiplication from both sides with \( J_{C_2} \). The last notation is the Jacobian of the six \( C_2 \) parameters with respect to the twelve parameters mentioned above. For this purpose, the three Euler angles \( \phi_2, \theta_2 \) and \( \psi_2 \) need to be extracted from (3.2.58) using the following equations:

\[ \phi_2 = \arctan \left( \frac{R_2(2,3)}{R_2(3,3)} \right) \]  
\[ \theta_2 = \arcsin (-R_2(1,3)) \]  
\[ \psi_2 = \arctan \left( \frac{R_2(1,2)}{R_2(1,1)} \right) \]  

(3.2.59)  
(3.2.60)  
(3.2.61)

Simple derivations and then concatenation of the above expressions yields the required Jacobian which is used to propagate the uncertainty from \( C_1 \) and the ego-motion to \( C_2 \).

### 3.2.2 Algorithm Sensitivity Study

The influence of different factors on the accuracy of the proposed algorithm is studied in this section. The closed form expression that was developed throughout the previous section is being used to determine the uncertainty of these estimates under a variety of simulated scenarios. Each tested scenario is characterized by the following parameters: the number of corresponding features being used by the algorithm, the image resolution, the grid spacing of the DTM (also referred as the DTM resolution), the amplitude of hills/mountains on the observed terrain, and the magnitude of the ego-motion components. At each simulation, all parameters except the examined one are set according to a predefined parameters set. In this default scenario, a camera with \( 400 \times 400 \) image resolution flies at altitude of 500m above the terrain. The terrain model dimensions are \( 3 \times 3 \) km with 300m elevation differences (Fig. 3.10(b)). A DTM of 30m grid spacing is being used to model the terrain (Fig. 3.7(c)). The DTM resolution leads to a standard-deviation of 2.34m
Figure 3.5: Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the number of corresponding features. In both graphs, the left vertical axis measures the translational deviations (in meters) and corresponds to the solid graph-line, while the right vertical axis measures the rotational deviations (in radians) and corresponds to the dotted graph-line for the height measurements. The default-scenario also defines the number of corresponding features to about 170, where an ego-motion of $\|p_{12}\| = 40m$ and $\|(\phi_{12}, \theta_{12}, \psi_{12})\| = 10^\circ$ differs the two images being used for the correspondence computation. Each of the simulations described below study the influence of different parameter. A variety of values are examined and 150 random tests are performed for each tested value. For each test the camera position and orientation were randomly selected, except the camera’s height that was dictated by the scenario’s parameters. Additionally, the direction of the ego-motion translation and rotation components were first chosen at random and then normalized to the require magnitude.

In Fig. 3.5, the first simulation results are presented. In this simulation the number of corresponding features that are used by the algorithm is varied and its influence on the obtained accuracy of the second pose and the ego-motion is studied. All parameters were set to their default values except for the features number. Fig. 3.5(a) presents the standard-deviations of the second frame of the camera while the deviations of the ego-motion are shown in Fig. 3.5(b). As expected, the accuracy improves as the number of features increases, although the improvement becomes negligible after the features’ number reaches about 150.
In the second simulation the influence of the image resolution was studied (Fig. 3.6). It was assumed that the image measurements contain uncertainty of half-pixel, where the size of the pixels is dictated by the image resolution. Obviously, the accuracy improves as image resolution increases since the quality of the features correspondence data is directly depends on this parameter.

The influence of DTM grid spacing is the objective of the next simulation. Different DTM resolutions were tested varying from 10m up to an extremely rough resolution of 190m between adjacent grid points (see Fig. 3.7). The readers attention is drawn to the fact that the obtained accuracy seems to decrease linearly with respect to the DTM grid-spacing (see Fig. 3.8). This phenomenon can be understood since, as was explained in the previous section, the DTM resolution does not affect the accuracy directly but rather it influences the height uncertainty which is involved in the accuracy calculation. As can be seen in Fig. 3.9, the standard-deviation of the DTM heights increases linearly with respect to the DTM grid spacing which is the reason for the obtained results.

Another simulation demonstrates the importance of the terrain structure to the estimates accuracy. In
the extreme scenario of flying above a planar terrain, the observed ground features do not contain the required information for the camera pose derivation, and a singular system is obtained as was discussed in section 3.1.5. As the height differences and the variability of the terrain increase, the features become more informative and a better estimates can be derived. For this simulation, the DTM elevation differences were scaled to vary from 50m to 450m (Fig. 3.10). It is emphasized that while the terrain structure plays a crucial role at the camera pose estimation together with the translational component of the ego-motion, it has no direct affect on the ego-motion rotational component. As the optical-flow is a composition of two vector fields - translation and rotation, the information for deriving the ego-motion rotation is embedded only in the rotational component of the flow-field. Since the features depths influence only the flow’s translational component it is expected that the varying height differences or any other structural change in the terrain will have no affect on the ego-motion rotation estimation. The above characteristics are well demonstrated in Fig. 3.11.

Since it is the translation component of the flow which holds the information required for the pose determination, it would be interesting to observe the effect of increasing the magnitude of this component. The last simulation presented in this section demonstrates the obtained pose accuracy when the ego-motion
Figure 3.8: Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the grid-spacing of the DTM.

Figure 3.9: Standard-deviation of the DTM’s height measurement with respect to the grid-spacing of the DTM.
Figure 3.10: DTM elevation differences: (a) 150m, (b) 300m, (c) 450m

Figure 3.11: Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the height differences of the terrain

43
Figure 3.12: Average standard-deviation of the second position and orientation (a), and the ego-motion’s translation and rotation (b) with respect to the magnitude of the translational component of the ego-motion. Translation component vary from 5m to 95m. Although it has no significant effect on the ego-motion accuracy, the uncertainty of the pose estimates decreases for a large magnitude of translations (see Fig. 3.12). As a conclusion from the above stated, the time gap between the two camera frames should be as long as the feature correspondence derivation algorithm can tolerate.
3.3 Algorithm Implementation

In this section I elaborate a possible implementation for the proposed algorithm. The derived constraint system is non-linear and hence needs to be solved by using a numerical procedure. In particular, a least-squares solution using the Newton-iterations scheme can be used.

3.3.1 Internal vs. External Iterations

In a practical implementation the approximation of the DTM by a plane is true only locally, and hence the Newton-iterations can only partially correct the errors in the initial a priori estimate. Notice that once the ray-tracing algorithm has located the estimated terrain features - the $Q_E$’s, these points, together with the tangent planes they determine, are kept fixed during the iterations. Since it is assumed that the $Q_T$ points lie somewhere on these planes, the Newton-iterations will not converge to the true pose and motion but rather to the best pose and motion for which the 3D features are on the required planes.

The limitation described above can be easily ameliorated by reactivating the ray-tracing algorithm between consecutive iterations. Namely, after each Newton-iteration the updated pose of the first camera frame could be used for the ray-tracing, leading to more accurate estimates of the $Q_E$’s and a refinement of the tangent-plane approximation. In the theoretical scenario of perfect DTM and image features location (infinite resolution and error-free), and when the initial guess of the camera pose is not too far from the true pose, such a scheme would converge to the true camera pose and ego-motion parameters as will be empirically shown in section 3.4. The resulting algorithm exacts, nevertheless, a high price in terms of computational cost. Indeed, in spite of having been a topic of continuous research in the computer-graphics community, ray-tracing algorithms are still considered to be involved and time consuming. Consequently, and taking into account real-time considerations, it is desirable to reduce the number of ray-tracing steps as much as possible. This observation leads to an alternative scheme based on internal- and external-iterations. The internal-iterations are the Newton-iterations discussed above. During these iterations, the $Q_E$’s and tangent-planes are kept constant and the algorithm proceeds until a convergence criterion has been met. When this occurs, an external-iteration is performed using the best available pose and motion data. During this iteration, ray-tracing is used to compute a new set of estimated locations and of tangent planes. The overall algorithm continues until the estimated locations converge.
3.3.2 Dealing with Outliers

In order to handle real data, a procedure for dealing with outliers must be included in the implementation. Three kinds of outliers should be considered:

1. Outliers present in the correspondence solution (i.e., “wrong matches”).
2. Outliers caused by the terrain shape, and
3. Outliers caused by relatively large errors between the DTM and the observed terrain.

The latter two kinds of outliers are illustrated in Fig. 3.13. The outliers caused by the terrain shape appear for terrain features located close to large depth variations. For example, consider two hills, one closer to the camera, the other farther away, and a terrain feature $Q$ located on the closer hill. The ray-tracing algorithm using the erroneous pose may “miss” the proximal hill and erroneously place the feature on the distal one. Needless to say, the error between the true and estimated locations is not covered by the linearization. To visualize the errors introduced by a relatively large DTM - actual terrain mismatch, suppose a building was present on the terrain when the DTM was acquired, but is no longer there when the experiment takes place. The ray-tracing algorithm will locate the feature on the building although the true terrain-feature belongs to a background that is now visible. As discussed above, the multi-feature constraint system is solved in a least-squares sense for the pose and motion variables. Given the sensitivity of least-squares to incorrect data, the inclusion of one or more outliers may result in the convergence to a wrong solution. A possible way to circumvent this difficulty is by using an M-estimator, in which the original solution is replaced by a weighted version. In this version, a small weight is automatically assigned to the constraints involving
outliers, thereby minimizing their effect on the solution. See [42] for further details about M-estimation techniques.

3.4 Experimental Results

Experiments were performed to verify the applicability, accuracy and robustness of the algorithm. Two types of experiments were conducted: one using synthetic data and the second using an experimental setup where data was obtained by a real camera focusing on a terrain model.

3.4.1 Simulation Results

In this experiment, a virtual-terrain of $300 \times 300$ meters was synthesized. The terrain contains patches of varying slopes representing hills of various heights, the tallest one being 60 meters high. The terrain also contains a $15 \times 15 \times 25$ meters "box" representing a manmade building. The terrain was then discretized to produce a model, i.e., the DTM, with a one-meter spatial grid. After computing the DTM, the synthetic terrain was modified by changing the location of the building so as to introduce a substantial terrain–DTM mismatch (see Fig. 3.14). Images from the terrain were obtained by using a virtual camera from various positions and orientations with respect to the terrain. A collection of 100 different correspondence pairs was analytically derived. The a priori estimate of the position and orientation of the camera was obtained by adding an error of approximately 17m and $3^\circ$.

Fig. 3.15 shows a typical example of the convergence of the algorithm. One can see that convergence is achieved after four external iterations. When the synthesized correspondence measurements were error-free the estimation process was able to completely remove the error from the pose initial guess. The outliers caused by the mislocated building did not deteriorate the estimate accuracy due to the utilization of the M-estimator. When i.i.d Gaussian noise of $\sigma = 0.001$ (roughly equivalent to 0.5 pixels for a $500 \times 500$ camera) was added to the correspondence measurements, less accurate estimate was obtained for the camera pose. However, as shown in Fig. 3.15 convergence speed was not significantly effected. During the tests, thirty internal iterations were performed for each external one, although it is clear that fewer iterations would have produced essentially the same result.
Figure 3.14: (a) The virtual terrain, (b) The DTM constructed from this terrain (grid-spacing is coarser than in the experiment, for visualization purposes). Note that the building on the virtual terrain (box at the bottom) has been moved to the gray bump at the center of the DTM.

Figure 3.15: Translational (a) and rotational (b) errors of the calculated pose as a function of the number of iterations. The symbols I, II, III, and IV denote external-iterations. Each iteration contains 30 internal-iterations. The blue solid line - error-free scenario, red dotted line - a scenario with Gaussian error of 0.5 pixel S.D. Units are meters and radians.
Figure 3.16: The robustness simulations were conducted for 5 virtual cameras located in different poses. A DTM of Montana’s Rocky mountains was chosen to examine the algorithm on relatively rough terrain.

3.4.2 Robustness of the Algorithm Against Large Errors in the Initial Guess

The optimization scheme used by the algorithm can only search for a local minimum. Since the problem is non-convex in the general case there may be scenarios in which the camera will converge to the wrong pose. These scenarios are characterized by large bias between the visible patch from the scene and the apparently visible patch taken from the DTM. The severity of this bias cannot be determined absolutely, but rather with respect to the “frequency” of the observed terrain: rough mountainous terrain (that contains high frequencies) will be more sensitive to small biases comparing to a terrain with soft and smooth hills (that only contains low frequencies).

Three factors should be considered for the characterization of such problematic scenarios: the magnitude of the initial guess error, the distance between the camera and the observed features and the roughness of the terrain. The second factor is very important when considering angular errors in the camera pose. In such case the bias magnitude of the scene features will be larger for distant features.

In order to check the robustness of the algorithm a series of simulations was conducted. A DTM of real terrain with 10m grid-spacing was used [97]. In light of the above observations, a rough terrain from Montana’s Rocky mountains was chosen to examine the algorithm in relatively difficult scenarios (see Fig. 3.16).

Five virtual cameras were placed in different locations and orientations, and about 225 feature correspondences were analytically derived for short baseline ego-motion of 20m along the camera Z direction. In each test of each virtual camera, a variety of positional errors (30m-300m) and angular errors (1°-10°) in the pose initial guess were randomly generated. The percentages of success under the tested error magnitudes are shown in Fig. 3.17 (a) and (b).
Figure 3.17: Percentage of success to converge to the true camera pose for different magnitudes of translational (a) and angular (b) errors in the algorithm’s initial guess. The red line corresponds to camera-1, green to camera-2, blue to camera-3, black to camera-4 and the dotted line to camera-5.

As can be seen, convergence to the true pose was obtained for any initial guess error smaller than 100m of the camera position and $4^\circ$ of its orientation. As expected, camera 5 is the most sensitive to angular error due to its large distance to the terrain. Camera 2 on the other hand is very sensitive to translational error. This may results from its low altitude which leads to a relatively small observable patch which is not very informative.

Performance using larger baselines was tested as well: 20m and 100m along the camera Z direction and along the camera X direction. However, similar results were obtained for all types and magnitudes of baselines. This align with the former argument regarding the factors which should influence the algorithm robustness. The baseline does not influence the above mentioned bias and thus should not influence the algorithm robustness.

Errors with magnitude of 100m and $4^\circ$ as mentioned above are considered huge for airborne vehicles. As was mentioned before, the accuracy of the pose computed by the proposed algorithm depends on different parameters: the number of corresponding features, the image and the DTM resolution, the structure of the terrain and the ego-motion baseline (see section 3.2). In most realistic scenarios, the average error is expected to be approximately 10m and 0.6$^\circ$ (compare for example to the errors in Fig. 3.15 and to the results in section 3.5). Therefore, in case it is desired to keep the errors under 15m and 1$^\circ$ for example, the vision-based algorithm should be activated in time intervals that prohibit the inertial navigation system to drift the pose by more than 5m and 0.4$^\circ$. In what follows the minimal rate for keeping the navigation error within the above mentioned margins is computed. An example for available consumer of-the-shelf navigation system is
Figure 3.18: (a) A 3D terrain model of horizontal dimension $50 \times 77$ cm. (b) The DTM was constructed by using a laser-based 3D-scanner. The spatial grid was 1 cm (the one in the figure has a coarser grid for visualization purposes).

The MIDG-II series IMU/INS system of Microbotics (see www.microbotics.com). Using this system for pure inertial navigation, the orientation solution diverges in $0.05^\circ/sec/\sqrt{Hz}$. This leads to accumulated angular error of $0.05 \cdot \sqrt{\Delta t}$ after $\Delta t$ seconds. As a result, an interval of no more than $(0.4/0.05)^2 = 64$ seconds should be kept for the desired orientation accuracy. As for the positional error, the inertial drift can be expressed by: $\Delta p = \Delta a \cdot \Delta t^2/2$, where $\Delta a$ is the acceleration error which is approximately $0.003m/sec^2$ in the MIDG-II system. Thus, an interval of $\sqrt{2 \cdot 5/0.003} \simeq 57$ seconds is required. To conclude, by activating the proposed algorithm at minimal rate of $1/57$ Hz the expected navigation errors will be kept far lower than its robustness breaking point. Therefore, the algorithm can be practically used for realistic navigation systems even when confronting rough terrains such as the Rockies mountains, and even when flying far away from the observed features.

3.4.3 Lab Experiment Results

Lab experimentation was performed using a real 3D model of a terrain and real images obtained by a camera. The dimensions of the model were $50 \times 77$ cm with elevation variations as high as 24 cm (see Fig. 3.18 (a)). A laser-based 3D-scanner was used to captured the terrain and build a DTM with a 1 cm spatial grid (see Fig. 3.18 (b)).

In each experiment the camera moved along a trajectory while attached to a robot manipulator. This configuration allowed moving of the camera in a controlled manner while also providing true measurements.
Figure 3.19: Two of the tested trajectories. Trajectory \( a \) is mostly a translation while trajectory \( b \) has significant changes in orientation.

for the pose of the camera at all time instances. Fig. 3.19 shows examples for two of the trajectories evaluated. The first trajectory (\( a \) in the figure) contains mostly translational camera motion with the orientation held essentially constant. For the second trajectory (respectively, \( b \) in the figure) position and orientation of the camera were changed in a significant manner. Although highly accurate “ground-truth” data for the trajectory of the camera was obtained from the robotic manipulator, this trajectory was corrupted using a simulated error model so that the “true” and the a priori trajectories drifted away with time. The error model was quite extreme: 7.4 mm/sec and \( 5^\circ/\text{sec} \), respectively. In order to compensate for this drift, the pose/motion estimation algorithm was called at 3/2 Hz rate. The two images used for the processing were the latest one available and a one-second old frame. The a priori information was derived from the available drifted pose at these two frames. When used for real navigation-system, it might be preferable to use an adaptive time-gap for the two frames, which takes into account the estimated velocities and the already reconstructed trajectory. As was mentioned in section 3.2, the magnitude of translation between the two frames is important to the accuracy and stability of the algorithm.

During the experiments, gray-scale images of \( 1024 \times 768 \) were obtained using a Dragonfly video camera at a rate of 15 frames per second. Correspondence between about 400 features was derived using the Lucas-Kanade tracking method ([57], [11]). Features were not selected using an image-dependent algorithm, but rather, by using a regular grid spanned over the image-plane. A typical frame and its features correspondence are shown in Fig. 3.20.
Figure 3.20: (a) A frame taken from one of the camera’s trajectories. (b) The estimated correspondence of 400 features taken from an $20 \times 20$ regular grid over the image plane of this frame.

As shown in Figures 3.21(a) and 3.21(b), the algorithm converged to reasonable estimates for the navigation parameters along the two trajectories described above. The figures show the ground-truth together with two trajectories computed using the error model: the first contains no updates while the second was updated periodically by using the pose/motion algorithm, at a 3/2 Hz rate. The figures clearly show that the corrected-path remains close to the true-path along the whole trajectory.

Fig. 3.22(a) shows the position errors of the drifted and corrected paths for the experiment (b). It can be seen that the errors of the corrected-path are kept small while the errors in the uncompensated path increase gradually. Fig. 3.22(b) shows the orientation errors for the two computed paths. The saw-tooth shaped graph of the corrected-path is characteristic: the orientation errors accumulate between updates but are strongly reduced each time the algorithm is used.
Figure 3.21: Experimental results for trajectories a and b (see Fig. 3.19). The diverging trajectories use the error model and no updates. The updated paths use the pose/motion algorithm to bound divergence.

Figure 3.22: Position errors (a) and orientation errors (b) of the drifted-path (dotted-line) and of the corrected-path (solid-line) of the second trajectory - b.
3.5 A Comparison with SFM and Registration Algorithm

As mentioned in chapter 2 and in the beginning of this chapter, the introduced algorithm is not the only possible approach to the problem at hand. An alternative is to divide the problem into sub-problems, and use existing algorithms as building-blocks for a solution. For instance, one can formulate a three-step approach, by first establish a feature correspondences between the two images, then estimating the motion and structure using an SFM algorithm, and lastly finding the pose by matching the reconstructed structure to the DTM. The purpose of this section is to present the implementation details for an algorithm as such, and then compare its performance with the new algorithm. As the experiments confirm, the fact that the novel algorithm uses the DTM to constrain simultaneously pose and motion computation is advantageous over the three-step alternative.

3.5.1 The “SFM+ICP” Algorithm

In this subsection the implementation details of the three-step algorithm are presented. Starting from correspondence pairs in two frames, numerous algorithms have been developed and studied for estimating ego-motion and reconstructing the scene. The algorithm presented in [70] was selected for the SFM step. In this work, the camera ego-motion was first derived and the structure of the scene was later reconstructed using the corresponding pairs and the estimated motion. Being visual-based, this algorithm suffers from the velocity vs. structure-scale ambiguity discussed in chapter 1. Additionally, the algorithm makes no use of the DTM information, and hence can only estimate camera motion.

Once the structure has been recovered, the “Iterative Closest Point” algorithm (ICP) can be used to estimate pose. By using the ICP algorithm presented by Chen and Medioni [16], the Euclidean transformation that best matches a set of points to a given surface can be estimated. In the present context, the points of the reconstructed structure given in the coordinates frame of the camera can be fed into the ICP algorithm to find the transformation, giving the best matching with the actual terrain surface as encoded by the DTM. Given that the SFM algorithm yields the scene structure only up to an unknown scale-factor, a slightly modified version of ICP is required, in which a similarity transformation is optimized instead of the more usual Euclidean one. The camera pose and the scale factor can be extracted easily from the estimated similarity transformation, and the scale factor ambiguity can be removed from the translational component of the ego-motion.
Figure 3.23: The synthetic terrain was scaled to obtain a variety of elevation variations: (a) 800 m, (b) 600 m, (c) 300 m. Different DTMs were obtained for terrain (b) by sampling the terrain under different spatial grids (resolutions): (d) 100 m, (e) 50 m, (f) 30 m.

### 3.5.2 Performance Comparison

The performance of the new presented algorithm was compared to the three-step approach discussed above by performing a large number of numerical experiments. In order to have a completely controlled environment, a $3 \times 3$ kilometer synthetic terrain was created, similar to the one used in the previous section (see Fig. 3.23(b)). Several different views were obtained using a virtual camera constrained to 600 meters above the terrain. A pure translation was selected as the virtual ego-motion, with a relatively large baseline of $\|p_{12}\| = 150m$. Observe that the length of the baseline should have a similar effect on both approaches to the problem.

Performance was studied under different scenarios in a similar framework to the one followed in section 3.2.2. Each scenario was characterized by the following parameters: the grid spacing of the DTM (also referred to as the DTM resolution), the altitude variations on the observed terrain, the resolution of images obtained by the virtual camera, and the number of corresponding pairs being used by the algorithm.

At each simulation, all parameters except for the one being tested were kept at predefined values. For example, in the default scenario, the terrain was scaled to contain 600 m elevation differences (Fig. 3.23(b)), and a DTM with a 50 m spatial grid was used as a model of the terrain (Fig. 3.23(e)). The camera is assumed to consist of $500 \times 500$ pixels and a maximum of 400 corresponding pairs were analytically derived prior to the calculations.
Figure 3.24: Pose and ego-motion estimation accuracy using the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for different DTM resolutions. Resolution varies from 10 m to 100 m. (a) Position errors, in meters. (b) Orientation errors, in radians. (c) Motion translation errors, in meters. (d) Motion rotation errors, in radians.

Each of the simulations described below studies the influence of a different parameter. A variety of values were examined and 150 random tests were performed for each tested value. For each test the camera position and orientation were randomly selected (except for height over the terrain). Additionally, the direction of the ego-motion translation was chosen at random.

Fig. 3.24 shows the results of the first simulation where the resolution of the DTM was varied and its influence on the accuracy was studied. All parameters were set to their default values except for the DTM resolution, which was varied from 10 m up to a worst case of 100 m between adjacent grid points (see Fig. 3.23 (d)-(f)). Fig. 3.24 (a) and (b) show that better estimates for the camera position and orientation were obtained by using the new algorithm, for all tested resolutions. Better estimates were obtained for most ego-motion parameters as well, although this advantage becomes marginal as the DTM grid spacing increases (see Fig. 3.24 (c) and (d)). This behavior was expected since the advantage of the new algorithm stems from the utilization of the DTM data for the ego-motion computation. Notice that the new algorithm strongly outperforms the three-step procedure when the grid-spacing is 40 m and better – a level compatible with modern DTM databases.
The next simulation demonstrates the relative importance of different terrain structures on the achievable accuracy. According to the discussion in section 3.1.5, in the extreme scenario of flying above a planar terrain, the observed ground features do not contain the required information for the camera pose derivation, and the system of equations becomes singular. As the slope and the variability of the terrain increases, the features become more informative and better estimates can be derived. For this simulation, the virtual terrain elevation differences were scaled to vary from 300 m to 800 m (Fig. 3.23 (a)-(c)). As can be seen in Fig. 3.25, better estimates for the camera pose and motion were obtained by using the new algorithm, when elevation differences were greater than 350 m. However, as the terrain flattens, the advantage of the new algorithm can easily change to a disadvantage. Motion estimation is not directly influenced by the structure of the terrain when using the SFM algorithm. The new algorithm, on the other hand, estimates the pose and motion simultaneously. Hence, in a non-informative scenario of relatively flat terrain, pose and motion are drifting simultaneously, leading to an overall larger drift. As a demonstration of the above property, one can see how the gap between the two algorithms becomes small and even favors the SFM+ICP for low elevation differences.
Figure 3.26: Pose and ego-motion accuracy obtained by the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for different image resolutions (from $200 \times 200$ to $1000 \times 1000$). (a) Position error, in meters. (b) Orientation error, in radians. (c) Motion translation error, in meters. (d) Motion rotation error, in radians.

As could be expected, performance is improved for both algorithms as image resolution increases. In the third set of simulations, the image resolution was varied from a low resolution of $200 \times 200$ to a high resolution of $1000 \times 1000$. Fig. 3.26, shows that the new algorithm achieves better pose accuracies for all resolutions. However, the gap between the two algorithms becomes small for very high resolutions, and a small difference actually favors the SFM+ICP algorithm in ego-motion accuracies, as can be seen in Fig. 3.26 (c) and (d). This characteristic could be expected since the “fusion” of noisy DTM information in the motion computation can improve or damage the obtained accuracy compared with the SFM, which ignores this information. In the theoretical scenario of infinite image resolution, it is clear that a perfect motion estimate can be obtained using SFM (excluding translation scale), while the new algorithm will still diverge due to errors encoded in the DTM.

The final simulation compares the two algorithms for different numbers of corresponding features. The features were not selected using an image-dependent selection algorithm but rather from a regular grid that was spanned over the image plane, where the resolution of the grid varies from $4 \times 4$ (16 features) up to $20 \times 20$ (400 features); see Fig. 3.18 (b) for an illustration of this grid. Fig. 3.27 shows that the new algorithm achieves better estimates for the pose and motion parameters when at least 64 corresponding features were
Figure 3.27: Pose and ego-motion accuracy obtained by the new algorithm (solid line) and the SFM+ICP algorithm (dotted line) for different numbers of corresponding features pairs. The features were selected from a regular grid that was spanned over the image plane, where the grids resolutions varied from $4 \times 4$ (16 features) up to $20 \times 20$ (400 features). (a) Position error, in meters. (b) Orientation error, in radians. (c) Motion translation error, in meters. (d) Motion rotation error, in radians.

available. However, the gap between the two algorithms converges for large numbers of features. This result is due to the Gaussian error assumption on the image measurements that leads to improving the estimate of the navigation parameters as the number of features increases.
3.6 Extending The Algorithm For Omnidirectional Cameras

Recently, an increasing interest in omnidirectional vision for applications in robotics could be noted. Technically, omnidirectional vision, sometimes also called panoramic vision, can be achieved in various ways. Examples include camera with extreme wide angle lenses (“fish-eye”), cameras with hyperbolic or parabolic mirrors mounted in front of a standard lens (catadioptric imaging), sets of cameras mounted in a ring-like or sphere-like configuration (polydioptric imaging), or an ordinary camera that rotates around an axis and takes a sequence of images that covers a field of view of 360 degrees [30, 31, 50, 60, 63, 64, 65, 86, 87, 88].

Omnidirectional vision provides a very large field of view, which has some useful properties. For instance, it enables the tracking of objects which are placed in different directions in the surrounding scene. It is well established that such variety of features facilitates the obtainment of a robust and accurate estimate of the camera pose. On the other hand, vision algorithms have to account for the specific properties of the particular omnidirectional imaging sensor setup in use. This may comprise theoretical and methodological challenges, as is the case for catadioptric vision. Here, the extreme geometrical distortions of the images caused by the parabolic or hyperbolic mirror require a suitable adaptation of image interpretation methods.

The projection induced by an omnidirectional camera is the transformation from the 3D space to the image(s) plane. The least restrictive assumption that can be made about any camera model is that the inverse image of a point is a line in space. For many omnidirectional cameras, all such lines do not necessarily intersect in a single point. Their envelope is called a dia-caustic and represents a locus of viewpoints. If all the lines intersect in a single point, then the system has a single effective viewpoint and it is a central projection. In [4] a theorem is presented stating that a catadioptric camera has a single effective viewpoint if and only if the mirrors cross-section is a conic section. In any other case, including multiple cameras configurations, rotating camera systems and other shapes of mirrors, there is no single center of projection. The data acquired by such omnidirectional systems cannot be processed by vision algorithms that were developed under the single effective viewpoint assumption.

In this section the navigation algorithm that was presented is extended to handle omnidirectional data. The most general case of non-central projection (“multi-optical center”) is analyzed. The single center of projection case that was previously analyzed becomes a particular case of this general formulation when all optical centers are located in a single point. As was shown in section 3.2, one of the most important factors that influence the robustness and the accuracy of the navigation algorithm is the complexity of the observed terrain. The extreme case, where only a planar segment of the terrain is visible, results in an ill-conditioned system which may lead to the failure of the algorithm. Whenever the navigating platform comes
Figure 3.28: When using an omnidirectional vision system a wide area of the terrain is visible (see the red area) even when the camera approaches a mountainside. When using a regular camera in similar scenario only small patch that is almost planar is observed (see the blue area).

close to a mountainside in the terrain, such an ill-conditioned scenario might arise if a regular camera (not omnidirectional one) is used. However, when using an omnidirectional vision system, the rest of the terrain will still be visible even if the platform approaches one of the mountainsides (see Fig. 3.28). Therefore, more robust and accurate results can be achieved when using omnidirectional vision.

3.6.1 The Omnidirectional Constraint

In the following section I repeat the derivations that were elaborated in section 3.1 with the necessary modifications for the non-central projection case of the omnidirectional data. No special assumption is made on the omnidirectional acquisition system. It is assumed, however, that the system was fully calibrated. As a result, for each visible feature it is possible to compute its line of sight with respect to the camera system \( -C \), which can be defined by a source point \( -C S_1 \) and a unit-vector \( -C q_i \), oriented from the source point to the observed feature. For the sake of notation simplicity the left superscript of the \( q_i \) vectors will be omitted.

Let \( Q_T \in \mathbb{R}^3 \) be a location of a ground feature point in the 3D world. At two different time instances \( t_1 \) and \( t_2 \), this feature point is detected in the omnidirectional images and its lines of sight \( \{ -C S_1, q_1 \} \) and \( \{ -C S_2, q_2 \} \) – are computed. Using an initial-guess of the pose of the camera at \( t_1 \), the line passing through \( -C S_1 \) in direction of \( q_1 \) can be intersected with the DTM using any ray-tracing algorithm. As before the location of this intersection is denoted as \( Q_E \). Denoting by \( N \) the normal of the plane tangent to the DTM at the point \( Q_E \), one can write:

\[
N^T (Q_T - Q_E) \approx 0. \tag{3.6.62}
\]

The true ground feature \( Q_T \) can be described using true pose parameters:

\[
Q_T = ^wS_1 + R_1 \cdot q_1 \cdot \lambda = R_1 \cdot (-C S_1 + q_1 \cdot \lambda) + p_1. \tag{3.6.63}
\]
Here, $\lambda$ denotes the distance between $wS_1$ and the feature point $Q_T$. In the aforementioned equation I use the feature’s transformed source point:

$$wS_1 = R_1 C_1 S_1 + p_1.$$  \hspace{1cm} (3.6.64)

Replacing (3.6.63) in (3.6.62) we get:

$$N^T [R_1 \cdot (C_1 S_1 + q_1 \cdot \lambda) + p_1 - Q_E] = 0.$$  \hspace{1cm} (3.6.65)

From this expression, the distance of the true feature can be computed using the estimated feature location:

$$\lambda = \frac{N^T Q_E - N^T wS_1}{N^T R_1 q_1}.$$  \hspace{1cm} (3.6.66)

Equation (3.6.66) can be assigned into (3.6.63) and after reorganization we get:

$$Q_T = \frac{R_1 q_1 N^T}{N^T R_1 q_1} Q_E - \frac{R_1 q_1 N^T}{N^T R_1 q_1} wS_1 + wS_1.$$  \hspace{1cm} (3.6.67)

By adding and subtracting $Q_E$ to (3.6.67), and after reordering:

$$Q_T = Q_E + \left[1 - \frac{R_1 q_1 N^T}{N^T R_1 q_1}\right] wS_1 - \left[1 - \frac{R_1 q_1 N^T}{N^T R_1 q_1}\right] Q_E.$$  \hspace{1cm} (3.6.68)

Using the projection operator that was introduced in (3.1.7), (3.6.68) becomes:

$$Q_T = Q_E + P(R_1 q_1, N) (wS_1 - Q_E).$$  \hspace{1cm} (3.6.69)

Similarly to the corresponding expression (3.1.8) from section 3.1, the above expression has a clear geometric interpretation (see Fig.3.29). The vector from $Q_E$ to $wS_1$ is being projected onto the tangent plane. The projection is along the direction $R_1 q_1$.

Our next step will be transferring $Q_T$ from the global coordinates frame - $W$ into the first camera’s frame $C_1$ and then to the second camera’s frame $C_2$. Since $p_1$ and $R_1$ describe the transformation from $C_1$ into $W$, I will use the inverse transformation:

$$^{c_2}Q_T = R_{12} R_1^T (Q_T - p_1) + p_{12}.$$  \hspace{1cm} (3.6.70)

Assigning (3.6.69) into (3.6.70) gives:

$$^{c_2}Q_T = R_{12} \cdot ^{c_1}S_1 + p_{12} + R_{12} \mathcal{L} (Q_E - wS_1).$$  \hspace{1cm} (3.6.71)

$L$ in the above expression represents:

$$L = \frac{q_1 N^T}{N^T R_1 q_1}.$$  \hspace{1cm} (3.6.72)
$q_2$ is a unit-vector pointing to the true ground-feature $Q_T$. Thus, the vectors $q_2$ and $(c_2^T Q_T - c_2^T S_2)$ should coincide. This observation can be expressed mathematically by projecting $(c_2^T Q_T - c_2^T S_2)$ on the ray continuation of $q_2$:

$$c_2^T Q_T - c_2^T S_2 = q_2 \cdot (q_2^T \cdot (c_2^T Q_T - c_2^T S_2)) \quad (3.6.73)$$

In expression (3.6.73), $q_2^T \cdot (c_2^T Q_T - c_2^T S_2)$ is the magnitude of $(c_2^T Q_T - c_2^T S_2)$’s projection on $q_2$. By reorganizing (3.6.73) and using the projection operator, we obtain:

$$\mathcal{P}(q_2, q_2) \cdot (c_2^T Q_T - c_2^T S_2) = 0 \quad (3.6.74)$$

where:

$$\mathcal{P}(q_2, q_2) = [I - q_2 \cdot q_2^T] \quad (3.6.75)$$

$(c_2^T Q_T - c_2^T S_2)$ is being projected on the orthogonal complement of $q_2$. Since $(c_2^T Q_T - c_2^T S_2)$ and $q_2$ should coincide, this projection should yield the zero-vector. Plugging (3.6.71) into (3.6.74) yields our final constraint:

$$\mathcal{P}(q_2, q_2) [R_{12} \cdot c_1 S_1 + p_{12} + R_{12} L (Q_E - w S_1) - c_2^T S_2] = 0 \quad (3.6.76)$$
Figure 3.30: (a) A 3D terrain model of horizontal dimension $115 \times 95$ cm. (b) The DTM was constructed by using a laser-based 3D-scanner. The spatial grid was 1mm (the one in the figure has a coarser grid for visualization purposes).

Note that in the single center of projection case, one should assign zero to $C_1 S_1$ and to $C_2 S_2$, and to assign $p_1$ to $w_1 S_1$. In such a case equation (3.6.76) reduces to:

$$P(q_2, q_2) [p_{12} + R_{12} \mathcal{L} (Q_E - p_1)] = 0$$

which is identical to (3.1.17) that was derived in section 3.1.

3.6.2 Experimental Results

Lab experimentation was performed using a real 3D model of a terrain and images from an omnidirectional acquisition system. The dimensions of the model were $115 \times 95$ cm with elevation variations as high as 32cm (see Fig.3.30(a)). A laser-based 3D-scanner was used to capture the terrain and build a DTM with a 1mm spatial grid (see Fig.3.30(b)).

Two types of omnidirectional acquisition systems were tested: a configuration of three regular cameras heading to different directions, and a catadioptric system with a parabolic mirror.

Three Cameras Configuration

Three cameras with a wide field of view ($80^\circ$ each) were firmly attached to a robotic arm. Each camera was posed in a different orientation (see Fig. 3.31). Their internal parameters and relative pose parameters were accurately estimated as part of the system calibration phase. In each experiment the cameras configuration was moved along a different trajectory. The robotic arm allowed moving of the cameras in a controlled
manner while also providing true measurements for the pose of the cameras at all time instances. Fig. 3.32 shows examples of two of the trajectories evaluated. The first trajectory (a in the figure) contains constant translational motion with the orientation held constant. In the second trajectory (b in the figure) position and orientation of the cameras were changed significantly. Although highly accurate “ground-truth” data for the trajectory of the cameras was obtained from the robotic manipulator, this trajectory was corrupted using a simulated error model so that the “true” and the a priori trajectories drifted away with time. The error model drifted the trajectory position and orientation by 1 mm/sec and 0.7°/sec, respectively. In order to compensate for this drift, the proposed algorithm was called at 1 Hz rate. Whenever activated, this algorithm was supplied with the latest 3 images (one from each camera) and a previous image triplet that was captured 20mm away. The a priori information was derived from the available drifted pose at these two frames. Since 20mm baseline was desired, the algorithm was activated for the first time only after 3 seconds of movement. Later, it was periodically activated in 1 second gaps.

During the experiments, gray-level images of 640 × 480 pixels were obtained from each of the three cameras. Correspondence between about 100 features per camera (300 features all together) was derived using the Lucas-Kanade tracking method [11, 57]. Features were not selected using an image-dependent algorithm, but rather, by using a regular grid spanned over the image-plane.
Figure 3.32: Two of the tested trajectories. Trajectory \textit{a} contains constant translational motion while trajectory \textit{b} has significant changes in orientation. The true paths are marked by black solid line, while the paths reconstructed by the algorithm are marked by red line. The black dotted lines represent the trajectories that would have been obtained in case the algorithm was not activated.

As shown in Figure 3.32, the algorithm converged to reasonable estimates for the navigation parameters along the two trajectories described above. The figure shows the “ground-truth” together with two trajectories computed using the error model: the first contains no updates while the second was updated periodically by using the proposed algorithm, at a 1 Hz rate. The figure clearly show that the corrected-path remains close to the true-path along the whole trajectory.

Figure 3.33 shows the position and orientation errors of the drifted and corrected paths for the two trajectories. It can be seen that the errors of the corrected path are kept small while the errors in the uncompensated path increase gradually.

In order to demonstrate the importance of the omnidirectional vision usage, the two trajectories were also reconstructed using 300 features coming from only one of the cameras, while the data from the other two cameras were ignored. Fig. 3.34(a) compares the translational accuracies that were obtained when using one vs. three cameras while reconstructing trajectory \textit{b}. A clear advantage can be observed for the utilization of the omnidirectional configuration. In section 3.2, the sensitivities of the proposed algorithm were studied. It was found that the obtained accuracy is highly related to the complexity of the observed terrain. The extreme case, where only a planar segment of the terrain is visible, results in an ill-conditioned
Figure 3.33: Results for trajectories a (sub-figures (a) and (b)) and b (sub-figures (c) and (d)) when using the three cameras configuration. Position errors ((a) and (c)) and orientation errors ((b) and (d)) of the drifted path are marked with a black dashed line, and errors of the corrected path are marked with a red solid line.
Figure 3.34: (a) Translational errors of trajectory b when using 300 features coming from only one camera (blue dashed line) and when using 100 features from each of the three cameras (red solid line). (b) A frame captured by the single camera that was in use.

system which leads to the failure of the algorithm. Whenever the navigating platform comes close to one of the mountainsides of the terrain, such an ill-conditioned scenario might happen if a regular camera (not omnidirectional one) is used. However, if using an omnidirectional vision system, then the rest of the terrain will still be visible even when approaching one of the mountainsides. Therefore, more robust and accurate results can be expected when using omnidirectional vision, as confirmed by Fig. 3.34(a). Note the blue dot in this figure. At that time instance, the algorithm performance was relatively poor for the single camera scenario since only small segment of the terrain was visible to that camera - Fig. 3.34(b).

**Catadioptric System**

In the second experiment the three regular cameras were replaced by a single catadioptric system which is constructed of a parabolic mirror mounted in front of an orthographic camera (see Fig. 3.35(a)). Images of $1024 \times 768$ pixels were captured by this camera and 300 feature correspondences between two consecutive images were computed for the algorithm using the Lucas-Kanade method (see Fig. 3.35(b)). It should be noted that this tracking method is not optimal for catadioptric images due to the nature of the distortion of this kind of images. However, since the catadioptric system was first calibrated, these distortions can be computed and then cancelled. For each feature, a warped images can be rendered from the original images such that the local area of the feature appears as if it would be in a regular perspective camera. Next the Lucas-Kanade tracking method can be activated on these warped images with no special difficulty.
Figure 3.35: (a) The catadioptric system that was used for omnidirectional vision in the second experiment. (b) An example for optical-flow field that was extracted for the algorithm. Each small blue arrow shows a corresponding couple.

The translational and angular accuracies that were obtained during the two examined trajectories are presented in Figure 3.36. The slight deterioration in the algorithm performance (compared to its performance with the three cameras configuration) is probably due to the low resolution at the periphery of catadioptric images and due to the usage of the Lucas-Kanade tracking method directly on the distorted images.
Figure 3.36: Results for trajectories a (sub-figures (a) and (b)) and b (sub-figures (c) and (d)) when using the catadioptric system. Position errors ((a) and (c)) and orientation errors ((b) and (d)) of the drifted path are marked with a black dashed line, and errors of the corrected path are marked with a red solid line.
Chapter 4

Direct Method for Video Based Navigation Using a Digital Terrain Map

This chapter discusses a novel approach for estimating the position and orientation of a moving platform equipped with a calibrated camera using the captured images and data obtained from a DTM.

In the beginning of the previous chapter a straightforward method for utilizing the DTM which is composed by three steps was mentioned. In this basic algorithm features correspondence is first established between two consecutive frames in the video. Next, a patch from the ground is reconstructed (up to an unknown scale) using structure-from-motion (SFM) algorithm. Finally, the reconstructed patch is registered to the DTM surface by a similarity transformation (Euclidian transformation + scale). This scheme, as noted already, suffers from unnecessary amplified sensitivity due to the SFM step. At that step, the depth of $n$ sampled points is estimated, where $n$ usually varies from few hundreds to few tens of thousands. Note that our final goal is to compute only 12 parameters (6 for each of the poses), far less than the $n$ parameters sought in the SFM step. Moreover, at that challenging step, the additional DTM information is completely overlooked.

In chapter 3 this problematic observation was handled by skipping the intermediate SFM step. A new algorithm was presented that utilizes the features correspondence together with the DTM to directly compute the camera pose at the two frames. One should notice that the very same observations that were made about the SFM step may be said about the first step, where features correspondence between the two selected images are computed. Once again the number of unknowns is much higher that the 12 sought pose parameters, once again the DTM information is overlooked during the step. In the present chapter I take the concepts that were first introduced in chapter 3 one step ahead. The presented algorithm skips the features correspondence computation step and establish a direct connection between the gray-levels of the two images, the DTM and the sought 12 parameters. Basically, it follows the same ideas that were successfully applied by a
family of algorithms commonly referred to as *direct methods*. See section 1.2.1 for some background about this field.

In all previous direct-method algorithms, as far as I know, no external information beside the images was used. Hence, these algorithms settled for estimating the camera ego-motion and a 3D reconstruction of the scene w.r.t to the camera coordinates. The key contribution of the present work which differentiates it from its direct method counterparts is the integration of the DTM information which enables the computation of the camera pose. The required adaptations to incorporate the DTM information in the scheme are elaborated throughout this chapter.

In analogy to the concepts of the features based approach, here I say that in the presence of the DTM information each hypothesized pose of the camera dictates the depth at each of the image pixels. Hence, given the pose at two frames, the image displacements can be uniquely determined and the first image can be “warped” accordingly toward the second image. Therefore, the objective of the direct method algorithm will be to find the poses at the two frames which produces a warping transformation that brings the first image as close as possible (according to pixel-to-pixel comparison) to the original second image. A constraint therefore can be established for each of the first image’s pixels that has a corresponding point in the interior of the second image, and the parameters of the two camera poses can then be computed by solving a system of these constraints. Note that our constraint relies on the assumption that corresponding points at the two images has the same gray-level (known as the brightness constancy constraint). Since I use two sequential images of a video stream and since most realistic terrain surfaces has Lambertian properties (excluding water and snow) this assumption should hold in most cases.

### 4.1 The Direct Constraint

Given the DTM and a couple of images that were captured by the camera from two different positions and possibly different orientations, the following section details the derivation of a constraint system on the navigation parameters.

In the following derivations it will be more convenient to represent the unknown parameters in terms of the first and second poses: $p_1$, $R_1$, $p_2$ and $R_2$. As before, I will assume that an a-priori estimate ($p_{1E}$, $R_{1E}$, $p_{2E}$ and $R_{2E}$) is available from some dead-reckoning navigation algorithm or any other source. Note that this new representation is completely equivalent to the pose+motion representation that was used so far. Indeed, given the two poses of the camera its motion can be computed as:

$$R_{12} = R_2^T R_1,$$  \hspace{1cm} (4.1.1)
\[ p_{12} = R_2^T(p_1 - p_2). \] (4.1.2)

### 4.1.1 The Two-Views + DTM Geometry

The first steps of the new constraint derivation are similar to those presented in chapter 3. Using the DTM linearization at \( Q_E \) we can express the unknown \( Q_T \) as a function of the first pose parameters (and other known data):

\[ Q_T = Q_E + \mathcal{P}(R_1q_1, N)(p_1 - Q_E). \] (4.1.3)

Next, the representation of this ground point under the second camera coordinate system is:

\[ c_2Q_T = R_2^T(Q - p_2). \] (4.1.4)

Let \( c_2Q_T = [X_2, Y_2, Z_2]^T \) and \( R_2 = [r_{2x}, r_{2y}, r_{2z}] \) where \( r_{2x}, r_{2y} \) and \( r_{2z} \) are the columns of \( R_2 \). The three components of (4.1.4) can be rewrite explicitly:

\[ c_2Q_T = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} r_{2x}^T(Q - p_2) \\ r_{2y}^T(Q - p_2) \\ r_{2z}^T(Q - p_2) \end{bmatrix}. \] (4.1.5)

The projection of the ground point \( Q_T \) on the image-plane of the camera at the first and second time instances can be expressed by \( c_1q_1 = (u_1, v_1, 1)^T \) and \( c_2q_2 = (u_2, v_2, 1)^T \) respectively, where a unit focal-length is arbitrarily chosen for the normalized image plane. By perspectively projecting \( c_2Q_T \), as appears in (4.1.5), on the second image plane one obtains:

\[ \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 + \Delta u \\ v_1 + \Delta v \end{bmatrix} = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix}, \] (4.1.6)

where \( \Delta u \) and \( \Delta v \) are the horizontal and vertical image displacements of the \((u_1, v_1)\) feature. Combining (4.1.5) with (4.1.6) we can express the image displacement of a ground feature as a function of the two poses together with some known data (the feature location on the first image and its corresponding DTM tangent plan):

\[ \Delta u = \frac{r_{2x}^T(Q - p_2)}{r_{2z}^T(Q - p_2)} - u_1, \] (4.1.7)

\[ \Delta v = \frac{r_{2y}^T(Q - p_2)}{r_{2z}^T(Q - p_2)} - v_1, \] (4.1.8)

where \( Q_T \) was expressed in (4.1.3) as a function of \( p_1 \) and \( R_1 \) (and the known tangent plane data).
4.1.2 The Brightness Constancy Constraint

If one would track some image features along the image sequence, then (4.1.7) and (4.1.8) could have serve directly as constraints on the parameters of the two poses (given the measured feature displacement between the two relevant images). Here, on the other hand, I would like to avoid feature tracking as was already discussed. Therefore, instead of comparing (4.1.7) and (4.1.8) to some measured displacements, here I “plug” these expressions to the well known Brightness Constancy Constraint. Under the assumption of a stationary and Lambertian scene, this constraint simply states that the brightness (or gray-level) of corresponding points at the two images should be equal, namely:

\[ I_1(u_1, v_1) = I_2(u_1 + \Delta u, v_1 + \Delta v), \]  

where \( I_1 \) and \( I_2 \) are the two images that were captured by a moving camera at two time instances \( t_1 \) and \( t_2 \). Let \( \vec{u}_i = (u_i, v_i)^T \) and \( \vec{\Delta}u = (\Delta u, \Delta v)^T \), such that

\[ \vec{u}_2 = \vec{u}_1 + \vec{\Delta}u. \]  

Eqn. (4.1.9) can be rewritten as:

\[ I_1(\vec{u}_1) = I_2(\vec{u}_1 + \vec{\Delta}u). \]  

At this point, in most previous direct-method algorithm, it was assumed that \( \vec{\Delta}u \) is relatively small and thus \( I_2 \) may be linearized at \( \vec{u}_1 \), to obtain:

\[ I_2(\vec{u}_1 + \vec{\Delta}u) = I_2(\vec{u}_1) + \nabla I_2(\vec{u}_1) \vec{\Delta}u. \]  

where \( \nabla I_2 \) denote the gradient of \( I_2 \). Putting together (4.1.11) and (4.1.12) we arrive to the condition:

\[ \nabla I_2(\vec{u}_1) \vec{\Delta}u + \Delta I(\vec{u}_1) = 0 \]  

where \( \Delta I(\vec{u}_1) = I_2(\vec{u}_1) - I_1(\vec{u}_1) \).

Plugging (4.1.7) and (4.1.8) in the above equation would yield the desirable constraint on the poses parameters. However, as was aforementioned, (4.1.12) is based on the assumption that \( \vec{\Delta}u \) is relatively small, say about a pixel size or less, which is usually not the case. On the contrary, it was shown in section 3.2 that large magnitude baseline between the two images is important for the accuracy of the estimated navigation parameters. Therefore, in the present work it is assumed that \( \vec{\Delta}u \) may be arbitrarily large. However, it is also assumed that \( \vec{\Delta}u \) can be estimated such that the residual displacement (the difference between the true \( \vec{\Delta}u \) and its estimate) is in order of one pixel size. Such an estimate is obtained by assigning the best available
estimates of the poses in (4.1.7) and (4.1.8). At the beginning, the a priori estimates of the poses will be used, and while the iterative algorithm proceeds, they will be replaced with the most current estimates.

Let $\vec{\Delta u}$ and $\vec{\delta u}$ be the estimated displacement and the residual displacement respectively, such that:

$$\vec{\Delta u} = \vec{\Delta u}_E + \vec{\delta u}. \quad (4.1.14)$$

Therefore, it is possible to linearize $I_2$ at $\vec{u}^*_1 + \vec{\Delta u}_E$:

$$I_2(\vec{u}^*_1 + \vec{\Delta u}_E + \vec{\delta u}) = I_2(\vec{u}^*_1 + \vec{\Delta u}_E) + \nabla I_2(\vec{u}^*_1 + \vec{\Delta u}_E) \cdot \vec{\delta u}. \quad (4.1.15)$$

Plugging (4.1.15) back into (4.1.11) we get the following constraint:

$$I_2(\vec{u}^*_1 + \vec{\Delta u}_E) - I_1(\vec{u}^*_1) + \nabla I_2(\vec{u}^*_1 + \vec{\Delta u}_E) \cdot \vec{\delta u} = 0. \quad (4.1.16)$$

Using (4.1.7) and (4.1.8) $\vec{\delta u}$ can be expressed as a function of the sought navigation parameters:

$$\vec{\delta u} = \vec{\Delta u} - \vec{\Delta u}_E = \begin{bmatrix} \frac{v_{2x}^T(Q_T - p_2)}{v_{2x}^T(Q_T - p_2)} - u_1 - \Delta u_E \\ \frac{v_{2x}^T(Q_T - p_2)}{v_{2x}^T(Q_T - p_2)} - v_1 - \Delta v_E \\ \frac{v_{2x}^T(Q_T - p_2)}{v_{2x}^T(Q_T - p_2)} - \Delta v_E \end{bmatrix}, \quad (4.1.17)$$

($Q_T$ is expressed as a function of $p_1$ and $R_1$ in (4.1.3)) while all other components of (4.1.16) are known quantities. Throughout this paper, the constraint (4.1.16) will be referred to as the direct constraint. This constraint can be written for each pixel in the image: $\vec{u}^*_1$ is chosen to be the center of the pixel and $\vec{\Delta u}_E$ is its computed displacement based on the most current poses estimate. Note that while $I_1$ is sampled at its pixels centers ($I_1(\vec{u}^*_1)$ is the gray-level of the corresponding pixel), $I_2$ and $\nabla I_2$ are sampled at $\vec{u}^*_1 + \vec{\Delta u}_E$ which is, in general, not a center of a pixel. The values of $I_2$ and $\nabla I_2$ at that image location is calculated using bilinear interpolation on the values of the four adjacent pixels.

### 4.2 The Navigation Algorithm

In section 4.1 the mathematical foundations of the proposed algorithm were laid and the direct constraint was derived. This constraint could be defined for each pixel in the image to construct an over-determined system in terms of the twelve sought parameters (of the two poses). Next, the system can be solved by any non-linear optimization procedure such as Newton-Raphson or Levenberg-Marquate algorithms. However, in order to obtain a stable, robust and computationally feasible algorithm, several implementation related issues should be addressed.
4.2.1 Hierarchical Scheme for Large A Priori Errors

The direct constraint (4.1.16) is based on the assumption that the estimated image displacements $\Delta u_E$ (that are computed by assigning the best available pose estimates to (4.1.7) and (4.1.8)) are relatively accurate, and that the residual displacements $\delta u$ are in order of one pixel or less. This assumption enabled us to linearize the $I_2$’s gray-levels around $\vec{u}_1 + \Delta u_E$. However, at the beginning of the optimization procedure, the $\Delta u_E$ displacements are computed from the a priori estimates of the two poses: $p_{1E}$, $R_{1E}$, $p_{2E}$ and $R_{2E}$. These quantities were assumed to be very rough estimates of the poses and therefore the resulting displacement estimates may have rather large error, particularly more than one pixel.

In order to circumvent this problem, an hierarchial (“coarse-to-fine”) approach is followed using Gaussian pyramids of the two images. A displacement error of several pixels at the original images translates to a sub-pixel error at coarser resolution images coming from higher levels of the pyramids. For the first examined level of the pyramid (with the lowest resolution) the a priori estimates of the poses will be used. Although these estimates are not assumed to be very accurate, for these coarse resolution images they should be adequate. Using the images from this level of the pyramids the algorithm will be able to refine the pose estimates, and thus also to refine the displacements estimates. Therefore, when continuing to the next pyramids level (where the resolution of the images is doubled), the displacements’ residuals can be assumed to be in order of one pixel size once again. The algorithm continues in this manner stage by stage, level by level, until the base of the pyramids where the original images are processed to perform the final refinement of the pose estimates.

4.2.2 DTM Linearization using Z-Buffering

In subsection 4.1.1 I have used a linearization of the DTM. Therefore, based on the current estimate of the first camera pose, I need to compute the ground point that is shown in each pixel of the image and the 3D terrain normal of that location. In chapter 3 this data was required only for a very spars set of image features. Thus, ray-tracing algorithm was chosen to detect the ground point that correspond to each of the tracked features. Here, on the other hand, since this data is required for potentially all the pixels in the image, ray-tracing would not be a preferable choice. Instead, Z-buffering algorithms [26] may serve as an adequate solution to the present problem. These algorithms are implemented by hardware (any common graphical card) and thus can achieve very high performance even when the resolutions of the image and of the DTM are relatively high. Most real-time graphical application, e.g. virtual reality and flight simulators, are based on this technology. In order to use Z-buffering algorithm, first the DTM need to be triangulated. This task
is straightforward since the DTM is naturally kept as a regular grid of heights. Hence, every four adjacent
grid point construct two triangles. The Z-buffer is an image of depths: for each pixel it holds the depth (with
respect to the camera) of the projected ground point. At the beginning these depths are initialized to infinity.
Next, the set of the DTM’s triangles is traversed in arbitrary order, each triangle is transformed to the $C_1$
coordinate system and then projected on the image plane. Finally, the depths of the pixels that are covered
by the projected triangle are updated (unless they were already updated with smaller depth).

At the end of the Z-buffering algorithm, we have for each pixel center $q_i = (u_i, v_i, 1)$ its corresponding
estimated depth $- \lambda_{E_i}$. The estimated 3D ground point is then obtained by the multiplication:

$$c_1 Q_E = \lambda_{E_i} \cdot q_i = \begin{bmatrix} u_i \cdot \lambda_{E_i} \\
v_i \cdot \lambda_{E_i} \\
\lambda_{E_i} \end{bmatrix}. \quad (4.2.18)$$

As for the terrain normal computation, there are two alternatives. In some graphics cards it is possible to
obtain, in addition to the depth image, an indices image in which each pixel holds the index of the last
triangle that updated its depth. If such hardware is used, a possible solution is to take the triangle’s normal
as an approximation to the terrain’s normal at that point.

Although this solution is very simple and fast, it has a serious problem with respect to our algorithm
convergence and stability. Consider the 1D terrain shown in Fig. 4.1. $N_1$ is the normal of the terrain at
the point $Q_E$. In case an estimate of the camera’s first pose was already refined or in case there is a short
distance between the camera and the terrain, the true ground point $Q_T$ is expected to be in a very short range
from $Q_E$ (e.g. in range $r_1$). In such a case, the tangent plane – $L_1$ which is orthogonal to $N_1$ is an adequate
linearization of the terrain. On the other hand, if the location of $Q_T$ is expected to be in a longer range
from $Q_E$ (e.g. $r_2$), either due to larger uncertainty of the camera pose or due to larger distance between the
camera and the terrain, then $L_1$ is not a proper approximation of the terrain. It is clear that in such a case $L_2$
with the normal $N_2$ would be preferable linearization.

Selecting triangles’ normal as suggested above will always return us $N_1$ in this scenario since it bases
the computation of the normal only on a single triangle that represents a small patch of the terrain, totally
disregarding the expected range of $Q_T$. A possible enhancement that may solve this problem is to compute
for each triangle in the DTM a series of normals, each of them is the normal of a smoothed version of
the terrain where different smoothing kernels are used to produce each of the normals in the series. This
task is performed as part of a preprocess phase and thus does not damage the performance of the real-
time algorithm. During the navigation algorithm, whenever a normal is computed, the index of the triangle
is determined by the Z-buffering hardware and then, based on the expected uncertainty of $Q_T$, the most

78
Figure 4.1: The linearization plane $L_1$ with the normal $N_1$ is more proper choice in case we expect the $Q_T$ to be somewhere in the range $r_1$. However, if we have higher uncertainty and we can only expect $Q_T$ to be in range $r_2$, then we better use the linearization plane $L_2$ with the normal $N_2$.

Proper normal will be selected from the triangle’s normals series. This requires that the pose estimation algorithm we use will be capable to supply at the end of each iteration not only the poses estimate but also the covariance of this estimate (Kalman filter alike). This covariance can be propagated to the uncertainty of $Q_T$ taking into account the distance between the camera and the terrain.

A different alternative which doesn’t require the triangles’ indices or such covariance estimates is to approximate the terrain’s normals directly from the depth image. Let $q_0 = (x_0, y_0, 1)$ be a location of a pixel in the image, and let $q_l = q_0 - \Delta x \cdot e_1$, $q_r = q_0 + \Delta x \cdot e_1$, $q_u = q_0 - \Delta y \cdot e_2$ and $q_d = q_0 + \Delta y \cdot e_2$ be the location of its four adjacent pixels, where $\Delta x$ and $\Delta y$ are the image distance between adjacent pixels along the rows and columns respectively, and $\{e_1, e_2, e_3\}$ are the standard basis vectors of $\mathbb{R}^3$ (see Fig.4.2).

Assigning the computed depths of the four points in (4.2.18) yields their corresponding ground points:

\[

c_1 Q_l = \lambda_{E_l} \cdot (q_0 - \Delta x \cdot e_1), \\
c_1 Q_r = \lambda_{E_r} \cdot (q_0 + \Delta x \cdot e_1), \\
c_1 Q_u = \lambda_{E_u} \cdot (q_0 - \Delta y \cdot e_2), \\
c_1 Q_d = \lambda_{E_d} \cdot (q_0 + \Delta y \cdot e_2). \\
\]

Next, I define the two vectors:

\[
V_1 = c_1 Q_r - c_1 Q_l = D_h \cdot q_0 + S_h \Delta x \cdot e_1, \\
V_2 = c_1 Q_u - c_1 Q_d = D_v \cdot q_0 - S_v \Delta y \cdot e_2,
\]
Figure 4.2: The computed depth of the four adjacent pixels: \( q_l, q_r, q_u \) and \( q_d \) is used to construct four ground points adjacent to \( Q_e: Q_l, Q_r, Q_u \) and \( Q_d \). The normal can then be computed by taking the cross product of \( V_1 = Q_r - Q_l \) and \( V_2 = Q_u - Q_d \).

where \( D_h = \lambda_{Er} - \lambda_{E_l}, S_h = \lambda_{Er} + \lambda_{E_l}, D_v = \lambda_{E_u} - \lambda_{E_d} \) and \( S_v = \lambda_{E_u} + \lambda_{E_d} \). \( V_1 \) and \( V_2 \) should span the terrain’s tangent plan, and thus the normal can be computed from their cross-product:

\[
C_1N = V_1 \times V_2 = D_h S_v \Delta y (e_2 \times q_0) + S_h D_v \Delta x (e_1 \times q_0) - S_h S_v \Delta x \Delta y (e_1 \times e_2).
\] (4.2.22)

Assigning \( e_1 \times q_0 = (0, -1, y_0)^T, e_2 \times q_0 = (1, 0, -x_0)^T \) and \( e_1 \times e_2 = e_3 \) in (4.2.22) we arrive to a simple formula of the terrain normal which correspond to the \( q_0 \) pixel:

\[
C_1N = \begin{bmatrix}
D_h S_v \Delta y \\
-S_h D_v \Delta x \\
-x_0 D_h S_v \Delta y + y_0 S_h D_v \Delta x - S_h S_v \Delta x \Delta y
\end{bmatrix}
\] (4.2.23)

Lastly, the estimated orientation of the camera is used to transform the normals to the global coordinate system \( W \). One should note that the obtained normal vectors are not unit-vectors. If one wishes, he can normalizes them as a last step. However, after reviewing the role of the normal in our constraint, we see that it multiplies both the numerator and the denominator of a fraction and thus the normalization is unnecessary.

Beside not requiring that triangles’ indices will be returned by the hardware, the depth based computation of the normals implicitly handles the \( QT \) uncertainty issue that was already mentioned. The distance
between the four ground points – $Q_{E_l}$, $Q_{E_r}$, $Q_{E_u}$ and $Q_{E_d}$ depends on the distance between their corresponding image points – $q_l$, $q_r$, $q_u$ and $q_d$ which depends on the pyramid level I use at that moment. Therefore, at the beginning of the pose refinement process, when the pose uncertainty is rather large, the utilization of the low resolution images will yield linearizations that approximate wider areas of the terrain. After several iterations, when the pose is refined and its uncertainty reduces, I start to use images with higher resolutions that produce more local linearizations. The distance between the four ground points which participates in the normal computation also depends on the distance between the camera and the terrain, such that distant locations will be “automatically” approximated based of a wider area than locations in the proximity of the camera.

It is important to stress that while the depth based normals has the nice aforementioned property of being computed based on “adaptive areas”, this computation doesn’t really involve all the terrain data of that area, but only the four sampled points. If possible, as far as time considerations are concerned, one may ease this problem by always computing the depth image in the full resolution or even higher resolution, and then reduce it to the required resolution in a similar manner to the reduction operations of Gaussian pyramids. Then, each of the four sample points we use will not be just a local samples of the terrain depth but rather a Gaussian weighted average of depths in their neighborhood. Thus, when computing the normal based on these four ground points we will actually involve much more terrain data of the relevant area. This improvement might be time consuming and in all realistic scenarios I have tested, the algorithm performed very well without it.

4.2.3 Performance Improvement by Internal and External Iterations

The derivation of our direct constraint involves two types of linearizations: the DTM is linearized by tangent planes and the gray-levels of the second image are linearized in (4.1.15). Both linearizations are only an approximation of the non-linear reality. Therefore, this linearization should be reestablished from time to time using the updated and refine data in order to converge to the true optimum of the problem.

The reestablishment of the linearizations could have been done at the beginning of each iteration, but it would be very computational demanding to perform in our case. For the DTM linearization, the depth and terrain’s normal should be recomputed for each pixel as elaborated in section 4.2.2. For the linearization of $I_2$, the $\overrightarrow{\Delta u_E}$ displacements of each pixel should be computed using the pose estimates, and the values of $I_2$ and its gradients should be computed at the $\overrightarrow{u_1} + \overrightarrow{\Delta u_E}$ locations using bilinear interpolation. These two tasks are relatively “heavy” and I would like to execute them as rarely as possible. Therefore, I use similar technique to the one presented in chapter 3 and define the algorithm’s scheme as a sequence of
Each external iteration starts by performing the linearization related tasks, and then it runs several internal iterations which can be implemented by any optimization procedure such as the commonly used Newton’s iterations. After the internal iterations meet some stopping criteria, the linearization data is considered to be utilized to the fullest, and the next external iteration begins.

To facilitate the reutilization of the bilinear interpolated values, the interpolation results should be stored in additional images $\tilde{I}_2$ and $\nabla \tilde{I}_2$. These images are commonly referred to as the warped versions of the original images $I_2$ and $\nabla I_2$. In the warped images, the $\tilde{u}_1$ entry contains the interpolation result of the $\tilde{u}_1 + \tilde{\Delta} u_E$ location:

$$\tilde{I}_2(\tilde{u}_1) \equiv I_2(\tilde{u}_1 + \tilde{\Delta} u_E),$$ (4.2.24)

$$\nabla \tilde{I}_2(\tilde{u}_1) \equiv \nabla I_2(\tilde{u}_1 + \tilde{\Delta} u_E).$$ (4.2.25)

In case the estimated displacements $\tilde{\Delta} u_E$ are accurate, the warped image $\tilde{I}_2$ is aligned with $I_1$. Hence, it is sometimes said that $I_2$ was warped toward $I_1$. The reader should note that the warping is non-linear operation. Particularly, the gradients of $\tilde{I}_2$ (that probably would have been denoted by $\nabla (\tilde{I}_2)$) are totally different from $\nabla \tilde{I}_2$. It is important that the gradients will be computed first on the non-warped image $I_2$ and only then the two images of the partial derivatives will be warped to obtain $\nabla \tilde{I}_2$. Using the warped images, the direct constraint (4.1.16) takes the simpler form:

$$\tilde{\Delta}I(\tilde{u}_1) + \nabla \tilde{I}_2(\tilde{u}_1)\delta u = 0,$$ (4.2.26)

where $\tilde{\Delta}I(\tilde{u}_1) = \tilde{I}_2(\tilde{u}_1) - I_1(\tilde{u}_1)$. The $\tilde{\Delta}I$ and $\nabla \tilde{I}_2$ images are constructed at the beginning of each external iteration and their values can be reused in (4.2.26) during the consecutive internal iterations.

The stopping criterion of the internal iterations should take into account three factors. First, it should account for convergence by comparing the pose results of the few last iterations. In addition, since I reuse the terrain linearization the internal iterations should be stopped after the first camera pose was significantly changed. This change could be measured based on the position and orientation parameters, but preferably based on the maximal distance between the $Q_E$ points and their corresponding computed $Q_T$ points. Lastly, the magnitude of the image displacements’ residuals $\delta u$ should be bounded as well due to the reutilization of the warped images $\tilde{I}_2$ and $\nabla \tilde{I}_2$ during the internal iterations.

### 4.2.4 Stability Improvement using SVD

As a starting point, let us assume that we are using Newton iterations (see e.g. [38]) for the internal iterations of our algorithm. Let $\theta_j$ ($j=1...12$) be the twelve parameters to be estimated and let $\Theta \in \mathbb{R}^{12}$ be their vectorial...
representation. Rearranging (4.2.26) we have a system of \( n \) constraints (\( n \) being the number of used pixels) of the form:

\[
f_i(\Theta) = \nabla I_2(u_i^1) \tilde{\delta} u_i(\Theta) = -\Delta I(u_i^1),
\]

where \( i=1...n \). Note that only \( \tilde{\delta} u_i \) is a function of the poses parameters and the rest are known quantities. Let \( f \in \mathbb{R}^n \) be the vector concatenation of the \( f_i \) functions. Using Newton’s algorithm the k’th iteration starts by computing the Jacobian matrix at \( \Theta^{k-1} \) (the last poses estimate we’ve computed):

\[
J^{k-1} \equiv J(\Theta^{k-1}) \equiv \left[ \frac{\partial f_i}{\partial \theta_j} \right]_{i,j}.
\]

Next, a refinement of \( \Theta^{k-1} \) is computed by the pseudo-inverse matrix of the Jacobian:

\[
\Theta^k = \Theta^{k-1} - \Delta \Theta^{k-1},
\]

where

\[
\Delta \Theta^{k-1} = \left( J^{k-1T} \cdot J^{k-1} \right)^{-1} J^{k-1T} \cdot \left( \tilde{\Delta} I + f(\Theta^{k-1}) \right),
\]

and \( \tilde{\Delta} I \in \mathbb{R}^n \) is the vector concatenation of the \( \tilde{\Delta} I(u_i^1) \) values.

There are some degenerated cases however, in which only part of the twelve parameters can be computed. Simple and intuitive example is the scenario in which we are flying above a planar surface. It is clear that in such a case the position of the camera cannot be fully determined but only its height w.r.t the plane can be recovered. See section 3.1.5 for a detailed discussion and analysis of different degenerate scenarios. Even in non-degenerate cases, the distinction between certain translations and rotations in the camera ego-motion (i.e. in the relative pose of the camera at \( C_2 \) w.r.t the \( C_1 \) system) might be ill-conditioned, especially when using the low resolution images of the Gaussian pyramids. For example, translating the camera to the right (with no change in its orientation) might produce image displacements that are very similar to those produced by rotating the camera to the right. The difference between the two displacements will be observable in the high resolution images. However, at the beginning of the process, when the low resolution images are utilized, both types of ego-motion will produce essentially the same gray-level changes and therefore the algorithm will face an ambiguous scenario. If Newton iterations are used to optimize the constraints system, then erroneous and arbitrarily large steps might be applied to the estimated \( \Theta \) along the ill-conditioned directions.

In order to prevent such arbitrary steps I would like to use a scheme in which the pose parameters are being refined only if the suggested refinement has strong support by the available data. A well known technique which is based on singular values decomposition (see e.g. [38]) can be an adequate solution.
The j’th column of $J$ reflects the gray-level change of each pixel in $\tilde{I}_2$ in case the $\theta_j$ parameter would be increased by one. Therefore, the eigenvector of $J^TJ$ which corresponds to the largest eigenvalue is the parameters change direction along which the largest brightness change occurs. Similarly, changing the parameters along the direction of the eigenvector with the smallest eigenvalue will cause the smallest brightness change to the warped image. Therefore, a singular value decomposition is applied on $J^TJ$. Due to its symmetry this decomposition actually yields its eigenvalues and eigenvectors. Next, I decompose $\Delta \Theta^{k-1}$ to its components along the computed eigenvectors and I sort them according to the magnitude of their corresponding eigenvalues, from largest to smallest. Then, these components are added to $\Theta^{k-1}$, one by one, until the corresponding eigenvalue drops below some predefined threshold, or until the maximal step size was exceeded. The eigenvalues threshold depicts the minimal ratio between the gray-levels change and the parameters change we trust. As a result, at the beginning of the optimization process, only the first few components of $\Delta \Theta^{k-1}$ will be used to update the poses. Thus only the obvious errors of the poses estimate will be corrected. During the algorithm, when higher resolutions are used, the eigenvalues increase, and more components (up to the whole 12) will be taken to correct the rest of the errors (that were considered ambiguous earlier).

When using such a scheme, there is a scale or units issue that should be addressed. Note that the first eigenvector (that corresponds to the largest eigenvalue) is the direction in which biasing the poses parameters by one yields the largest gray-level change in the warped image. However, this direction involves both metric and angular parameters and different choice of units will lead to different eigenvectors. For example, if we use meters for the positions parameters and radians for the orientation parameters, it is clear that larger gray-level changes are obtained when changing the camera orientation by one radian, than when shifting the camera position in one meter (in case the flight altitude is e.g. 1km above the terrain). On the other hand, if we are using kilometers for translation and degrees for rotation, then the exact opposite is true: 1km change in the camera position will yield far larger brightness change to the image pixels, compared to a small orientation change of $1^\circ$.

It is therefore necessary to determine the balance between the rotation and translation units of the parameters. I have set arbitrarily the rotation unit to one milliradian, and I set the translation unit accordingly, such that a unit change of rotation and of translation will yield the same average gray-levels change in the image. This is done by first dividing the $n \times 12$ Jacobian into two $n \times 6$ matrices - one with the rotation related columns and the other with the translation related columns. Then, the entries of each matrix are averaged and the translation related columns are scaled by the ratio between the two computed averages.
4.2.5 Gradient Based Pixel Selection

A nice property of the direct constraint is that it naturally handles the well-known aperture problem which states that one can only determines the displacement of a pixel along the direction of its gradient (usually referred to as normal-flow). Reviewing our direct constraint (4.2.26) we can see that indeed the \( \vec{\delta u} \) (which can be in arbitrary direction) is multiplied by \( \vec{\nabla I}_2 \) and thus only the component along this gradient is relevant for the constraint. Namely, let \( \vec{\delta u} \) be the summation of two components: \( \vec{\delta u}_N \) along the image gradient and \( \vec{\delta u}_T \) – the orthogonal complement such that \( \vec{\delta u} = \vec{\delta u}_N + \vec{\delta u}_T \) and \( \vec{\nabla I}_2 \cdot \vec{\delta u}_T = 0 \). Thus (4.2.26) can be rewritten as:

\[
\tilde{\Delta} I(u_i) + \vec{\nabla I}_2(u_i) \vec{\delta u}_N = 0. \tag{4.2.31}
\]

Moreover, the magnitude of the gradients actually weight the constraints such that pixels with strong gradients are more dominant in the optimization compared to other pixels with smaller gradients. This means that each pixel contributes to the constraints system according to the information it “possesses”.

In order to speed-up the algorithm it is desirable to use only a subset of the image pixel instead of the whole image. Based on the aforementioned observations, \( k \) pixels with large gradients are selected, where \( k \) depends on speed vs. accuracy considerations (\( k=10000 \) in our application).

It is shown in chapter 5 that wide distribution of the utilized features over the image plane is very important to compute the pose accurately. Thus, the selected \( k \) pixels are not necessarily the pixels with the maximal gradient (which might all be located in a very narrow area in the image). Instead, low gradient pixels are pruned in a first stage, and then a random subset of \( k \) pixels is selected from the remained image.

4.2.6 Outliers Effect Reduction using M-Estimator

The least-squares solution, although optimal under i.i.d normal noise, is known to be very sensitive to outliers datums. Beside the images noise there are several possible sources of outliers. First, I assume that the observed scene is stationary and Lambertian. If one of these assumptions is broken, e.g. if a moving car is present in the scene or if a window’s glass create a strong specular effect in one of the captured images, then an outlying constraint is produced. Moreover, outliers can be caused around depth discontinuities of the scene or in case there is a mismatch between the DTM and the actual terrain (see section 3.3.2). A possible way to improve the algorithm’s robustness against such outliers is by using an M-estimator, in which the original least-squares solution is replaced by a weighted version. In this version, smaller weights are assigned to constraints with larger residuals. In our implementation Huber’s M-estimator was used. See [42] for further details about M-estimation techniques.
Figure 4.3: The virtual terrain is texture-mapped with an ortho-photo of a real terrain. A building is located in the middle of the landscape. A virtual camera was placed in two poses above the terrain (the first pose is marked by a dark red circle and the second by a dark green diamond). An error was added to these poses to produce the initial guess for the algorithm (the bright red circle and the bright green diamond).

4.3 Simulation Results

Before testing the presented algorithm on a real data of aerial images, a series of simulations was executed under a synthetic and controlled environment. A virtual terrain of \(3000 \times 3000\) meters with hills up to 300 meters high was synthesized. An ortho-photo of real terrain was texture-mapped to this terrain in order to make it more realistic. Additionally, a building was placed in the middle of this virtual landscape, see Fig. 4.3. The DTM of the terrain was produced by sampling it in regular grid with 10m spacing. Next, a virtual camera was placed in two different positions and orientations, about 1km high and with 200m baseline between them, and the scene was rendered to produce a \(200 \times 200\) image for each camera pose. Fig. 4.4 shows the two rendered images. Although the exact poses of the virtual camera are known, the navigation algorithm was supplied with a biased estimate to simulate a drift in the navigation parameters: error of about 170m was added to each camera positions and \(7^\circ\) to its orientations (Fig. 4.3 depicts the true and biased poses of the camera). Such a bias in the navigation parameters is considered huge and it will take a considerable duration until most inertial based navigation systems will drift beyond this error.

Starting from the biased initial guess, the algorithm was activated using the two rendered images and
the DTM. Fig. 4.5 shows a typical result of the algorithm convergence. One can see that almost all the a priori error of the two poses was corrected by the algorithm. During the first set of iterations where the lowest resolution images were processed, only part of the error was corrected. Due to the SVD mechanism only pose corrections that has high evidence in the processed images is adopted. When the next level of the pyramids was processed and more ground feature could be observed, further error could be corrected and so on. Note that even in this error-free environment the estimated pose still contains a small error, about 1 meter in position and 0.05° in rotation. This is due to the finit resolution both of the images and of the DTM.

One of the advantages of the direct method algorithm compared to its feature-based alternative is that it can handle a completely featureless scene. To examine this behavior, both the building and the mapped ortho-photo were removed to produce a smooth and featureless surface, see Fig. 4.6. The convergence graphs are shown in Fig. 4.7. We see that although the processed images contain no features at all (which would have lead to the failure of a feature-based navigation algorithm) the direct method algorithm still succeeds to remove most of the poses errors, obtaining an estimate with only 4m error in the positions and error of 0.26° in the orientations.

Another interesting simulation test demonstrates the robustness of the algorithm in the presence of outliers. One possible source of outliers is a strong noise in the image that cause a mismatch between the two
Figure 4.5: The error of the translation parameters (a) in meters and of the rotation parameters (b) in radians along the algorithm iterations. Red line - error of the first virtual camera, green line - error of the second camera, black line - the end of a pyramid level processing.

Figure 4.6: (a) The virtual terrain without the building and the texture. (b) the rendered featureless image.
images (even after the warping of one toward the other). Other outliers may be caused by a mismatch between the DTM and the real terrain. In order to simulate a scenario that includes outliers a “salt and pepper” noise was applied to 3% of the second image’s pixels. In addition, while the images were rendered using the terrain with the building, the algorithm was provided with the building-free version of the DTM (see Fig. 4.6(a)). Fig. 4.8(a) shows the area of the building in the noised image. Fig. 4.8(b) presents the pixels that were selected by the algorithm in one of its last iterations and the weights that were assigned to these pixels by the M-estimator. One can see that most of the outlying pixels got very low weights which diminish their effect on the obtained result. Note that the vertical edges of the building were detected as outliers while the horizontal edges did not. This result was obtained since the camera ego-motion was horizontal in that test. As a result, the horizontal edges did not cause any images mismatch (the famous aperture problem) and thus they were not considered outliers. Fig. 4.9 shows the obtained errors along the algorithm iterations. Once again, most of the navigation error was corrected in spite of the “contaminated” scene, obtaining poses estimate with accuracy of 18m in the positions parameters and $0.37^\circ$ in its orientation parameters.
Figure 4.8: (a) cropped part of the noised image. This part contains both the building that was omitted from the DTM and “salt and pepper” pixels. (b) The weights that were assigned by the M-estimator, varying from red (weight=0) to yellow (weight=1). The black pixels were not selected by the algorithm in the examined iteration.

Figure 4.9: The error of the translation parameters (a) in meters and of the rotation parameters (b) in radians along the algorithm iterations in the presence of outliers.
4.4 Flight Experiment Results

Additionally to the simulations, the performance of the proposed algorithm was examined on real aerial images and a DTM of a real terrain. The experiment was performed in a hilly area of the Israeli Galilee containing elevation variations as high as 300m (see Fig. 4.10). A 9km × 5km DTM of the relevant region was obtained from The Survey of Israel organization (MAPI) [59]. The DTM has 4m resolution and its elevation measurements contain up to 2m errors.

During the experiments, images of 1024 × 768 pixels were obtained by a Dragonfly video camera at a rate of 7.5 frames per second (see Fig. 4.12). The camera was carried by an ultra-light plane (see Fig. 4.11) that flew above the terrain. Before the flight, this camera was first calibrated using [12] to obtain all its internal parameters. In order to have a ground-truth data of the flight trajectory, the ultra-light was also equipped with a differential GPS system that produces position estimates at 1Hz with up to 0.5m error. In addition, a hybrid navigation system that combines inertial navigation with GPS data was firmly attached to the camera’s body. This system is capable of estimating both the plane’s position with accuracy of 5m and its orientation with accuracy of 1.5° at a rate of 10Hz. The two trajectories were merged to produce both accurate and densely sampled ground-truth trajectory which is depicted in Fig. 4.10.

Although the true trajectory was known due to the availability of GPS measurements, it was not supplied to the examined vision based navigation algorithm. Instead, the camera ego-motion was deliberately biased in a rate of 2m per second in its translation parameters and in 0.05° per second in its rotation. Such bias simulates a realistic navigation drift that might occurs when using dead-reckoning navigation based on the inertial measurements alone. The biased ego-motion measurements were integrated along time to obtain the drifted trajectory which is depicted in Fig. 4.13. It is this trajectory that would have been obtained if no global pose correction (either GPS or vision based) was incorporated in the navigation process.

In order to keep the trajectory drift bounded below reasonable magnitude, the proposed algorithm was activated in 0.1Hz. On each activation the most recently captured image together with one of the previously captured images were selected to be used by the algorithm. The gap between the image couple was not constant. On one hand we would like to select an image couple with baseline as large as possible as it significantly improves the obtained accuracy. On the other hand, by defining a constant gap in terms of time or camera shift the selected image couple might capture different terrain areas with small or even no overlap. In such a case the algorithm won’t converge to the desired poses. Thus, the criterion for selecting the image couple was based on the movement of the camera field of view. The principal-point of the second image was defined as an anchor point and it was tracked back in the images sequence until an image movement of 100

91
Figure 4.10: The terrain of the flight experiment in top-view (a) and perspective-view (b). The ground truth flight trajectory is marked by the green line. The north-east measurements are in kilometers and w.r.t the new Israeli grid.
Figure 4.11: The camera was attached to a device (the marked circle in the figure) that was carried by an ultra-light plane. In order to obtain a ground truth about the flight trajectory a D-GPS and inertial systems were also attached to the device.

pixels was reached. Fig. 4.12 shows one of the selected image couple. Once an image couple was selected, an initial guess of their camera pose was computed by integrating the biased ego-motions measurements, starting from the last computed pose (that was obtained in the previous activation of the algorithm). The image couple together with the a-priori pose estimates were supplied to the algorithm and the camera pose was refined. By comparing the algorithm results to the ground-truth it was found that the mean error of the algorithm’s estimates is 29.3m in the camera position and 1.39° in its orientation. Such error magnitude is reasonable and allows a proper navigation of the platform when considered with respect to the flight elevation (300m-1000m). Note that such error would have been accumulated after about 15-30 seconds of flight in case the algorithm was not activated, and unlike our bounded error case, it would further increase along time.

The obtained pose estimates could be “fused” together with the drifted ego-motion estimates to construct the full flight trajectory. Fig. 4.13 contains two types of such trajectories. The first is the trajectory of the “online” results. For each moment this line represents the best estimate that could be produced by navigation system on that moment. It can be observed that between the algorithm activations there is an accumulated drift. However, when the algorithm is reactivated most of this drift is corrected. Moreover, since the algorithm error doesn’t depend on the accuracy of the initial guess, the navigation error is kept bounded along the whole trajectory (see also Fig. 4.14). Another trajectory that is presented in Fig. 4.13 is the one of the “offline” results. Let $t_1$ and $t_2$ be two time instances at which the algorithm was activated. Given the pose estimate at $t_2$, a-posteriori estimates can be retroactively computed to all the camera poses in the time interval $(t_1, t_2)$ in order to correct the drift. Such a procedure would yield much smoother
Figure 4.12: One of the image couples that were selected for the algorithm. The internal parameters of the camera were used to remove the distortion and to normalize these images. The selection criteria guarantee that there will be a shift of 100 pixels between the principal point of one image to its corresponding point in the other image.

and accurate trajectory, as can be seen in the figure. Even smoother trajectory could have been produced by feeding both the inertial navigation measurements and the computed poses to a Kalman-filter. Such a procedure, however, will not be elaborated here.

The reader might notice that no algorithm activations appear along the trajectory turns (at the most eastern and western parts of the trajectory). At those parts, there were two factors that prevented the success of the algorithm. First, the rotation of the camera at these parts is very fast. Thus, in order to select an image couple with sufficient overlap we are compelled to use very short baseline which significantly deteriorate the accuracy of the algorithm. This problem could be solved by using an omni-directional camera. Even in cases of sharp turns there will be large overlap between the captured panoramas and thus large baseline can still be maintained. The utilization of such camera was already shown to be useful for the feature-based version of this algorithm (see section 3.6). Another problem that prevent the success of the algorithm at the turning parts of the trajectory is that the observable terrain at that area was almost flat, see Fig. 4.15. It was shown in chapter 3 that the accuracy of the algorithm tightly depends on the complexity of the viewed topography. An almost flat terrain reduce the accuracy of the algorithm and even might yield an ill-conditioned constraints system. Due to the aforementioned difficulties it was decided to turn off the vision based algorithm during the turns, and to reactivate it as soon as the turn was over and the viewed terrain became hilly again.
Figure 4.13: (a) The results of the whole trajectory, (b) The results of the trajectory beginning. The ground-truth trajectory is marked by the green line. The locations of the camera in which the navigation algorithm was activated are marked by green circles. The red line depicts the drifted trajectory that would have been obtained if no pose correction was performed. The blue line and the black line show the results of the offline and online trajectories respectively. The positions that were estimated by the algorithm are marked by blue ‘x’ marks. In the error-free scenario, the green circles and the corresponding blue ‘x’ marks should align.
Figure 4.14: The position error (in meters) along the beginning of the flight trajectory. The red line depicts the error that would have been obtained in case only the inertial navigation system was used. The black line shows the errors of the online trajectory, and the blue line show the error of the offline trajectory.

Figure 4.15: An image that was captured during one of the trajectory’s turns. The yellow lines depict the depth level sets. It can be seen that the viewed topography is almost flat, which prevent the computation of accurate pose.
In this chapter the problem of *landmark-based navigation* is examined. Landmarks are distinctive features in the surrounding scene for which the 3D location is known with respect to some global coordinate system. Consider an autonomous vehicle equipped with a camera. In order to perform vision-based navigation, a set of predefined landmarks is supplied and the 2D projections on the camera’s image-plane are identified and tracked during the vehicle’s movement. Given the 3D and 2D data, the navigation problem is defined as the estimation of the camera *pose* (position and orientation) with respect to the global reference frame.

During the last two decades robust pose estimation algorithms have been developed by the computer vision community. These algorithms can integrate an arbitrary number of landmarks in the pose computation, leading to accurate and numerically stable results (e.g., [21, 19, 71, 35, 40, 54]). However, due to performance limitations, many real-time navigation systems are restricted to the use of only a very small number (usually 4-10) of landmarks. This limitation arises from the large overhead of detecting and tracking these landmarks along the image sequence. In [14], for example, a navigation system is presented where only four landmarks are simultaneously tracked.

If the number of available landmarks is small as well, the system will use all the visible landmarks at hand. However, if the system is equipped with a large landmark database, a subset needs to be selected from the visible landmarks as the camera moves. An example of such a scenario is an unmanned aerial vehicle (UAV) that utilizes a digital map and an ortho-photo of the observed terrain. In this configuration, the 3D location of any point on the terrain is known, and any visually distinctive point can thus be used as a landmark. The number of potential landmarks in such a case is large, and a subset must be chosen. Another example is a Simultaneous Localization and Mapping (SLAM) system such as the ones in [76, 37, 20, 22]. These systems estimate the camera motion and simultaneously track new features along the path of the
robot’s movement. The 3D locations of the tracked features are reconstructed and added to a landmark database. As a result, the database is progressively enlarged and after a while there will be more visible landmarks than it is possible to track.

While the navigating platform moves, new landmark subsets should be occasionally selected. The need for a new subset may arise, for example, when one of the landmarks leaves the camera field of view or after the camera has moved more than a certain distance since the last subset was chosen. Whenever a new subset is required, an initial guess of the camera pose can be utilized to filter the landmarks which are supposed to be visible at the moment and to predict their projection location on the image before actually detecting them. At this stage we face an important question: how do we choose the subset from the filtered landmarks wisely, in a manner that will lead to the best pose estimate according to the requirements of the specific navigation task? This task-oriented landmark selection problem stands at the center of the present chapter.

In most previous works (e.g., [76], [94] and [93]), the landmark selection problem was addressed from the image appearance standpoint, where the 3D location of the landmarks was disregarded and the selection criterion based solely on a measure of distinction of the 2D features in the captured image. In [74] the region from which each landmark is visible constituted the selection criteria in order to construct a small landmark subset that can be used from a variety of locations.

In [13], [14] and [85], as in the present work, the 3D structure of the selected landmark constellation and its influence on the obtained accuracies was studied. In [13], the design of special positioning objects (i.e., fiducials) was considered. The relatively small dimensions of such objects (compared to their distance from the imaging device) enabled the authors to assume weak-perspective projection, which is inadequate for general landmark-based navigation. In [14] and [85], a full-perspective projection model was used; however, the navigation problem was restricted to a two-dimensional world where only three pose parameters had to be estimated.

None of the aforementioned works addressed task-oriented considerations when selecting the landmark subset. Both [13] and [14] used the condition number of the pose covariance matrix as the landmark selection criterion. This criterion does not reflect the different severity of errors in the different pose parameters. For example, a unit error in the camera’s position (e.g., 1 cm) should not be considered equivalent to an angular unit error (e.g., 1 radian) in its orientation. Additionally, the purpose of the pose computation should not be overlooked. The navigation system usually supports a control system that uses the pose estimates to perform some predefined task. According to the requirements of the specific task, some of the pose parameters may be considered more essential than others. For example, if the platform needs to follow a predefined path, then accurately identifying its location along the path is not as important as identifying any drifts from the
Another example is the task of landing an airplane on a landing track. Obviously, the set of relevant parameters and accuracies during landing differs from those that need to be controlled for maintaining straight and level flight. It would thus be desirable to select a subset of landmarks that minimizes the error in some of the pose parameters even at the expense of larger errors in the other parameters.

In this chapter I present a new criterion for task-oriented landmark selection. The system designer can use a severity function to specify the adequacy of different poses for the specific navigation task. This function can be used to construct a requirements matrix that reflects the importance of the different pose parameters for the task at hand. Next, I show that the landmark subset selection problem can be approximated by solving a Semi-Definite Programming problem. A solution for this class of optimizations can be found easily and rapidly, qualifying the proposed algorithm for real-time navigation systems. Although this chapter shows how Semi-Definite Programming can be utilized for landmark selection, it can also be applied to variety of problems from the computer-vision and robotics fields.

5.1 Landmark-Based Navigation

Before considering the task-oriented landmark selection problem, I briefly summarize the landmark-based navigation problem. Let \( Q_i \in \mathbb{R}^3 \) \((i = 1, ..., n)\) be a set of available landmarks. The 3D location of these points is assumed to be known with respect to some reference coordinate system \( W \). In order for an autonomous vehicle to navigate in this scene, it is equipped with a calibrated camera, to which another Cartesian coordinate system, denoted \( C \), is attached. Traditionally, the origin of this system coincides with the camera’s center of projection and the Z-axis is oriented along the optical axis. The pose of the camera with respect to \( W \) can be represented by an orthonormal rotation matrix, \( R \in SO(3) \), and by the camera position vector \( p \in \mathbb{R}^3 \) such that

\[
^C Q_i = R^T (Q_i - p), \tag{5.1.1}
\]

where \(^C Q_i\) is the representation of \( Q_i \) in the camera’s system \( C \). Due to the orthonormality of \( R \), the camera’s orientation has only three degrees of freedom, usually represented by the Euler-angles \( \phi, \theta, \) and \( \psi \), which reflect the rotation around the \( X, Y, \) and \( Z \) axes respectively. Thus, the camera pose is fully defined by a 6D parameter vector, \( \Theta = (\phi, \theta, \psi, p^T)^T \).

In the camera frame, the 3D landmarks are perspectively projected to their 2D location in the image-plane:

\[
(u_i, v_i) = \left( \frac{^C Q_{ix}}{^C Q_{iz}}, \frac{^C Q_{iy}}{^C Q_{iz}} \right). \tag{5.1.2}
\]
Given the 3D landmarks and their corresponding 2D camera measurements \((\hat{u}_i, \hat{v}_i)^T\), the navigation problem is to accurately estimate the camera pose parameters \(\hat{\Theta}\). These parameters can be estimated by a non-linear optimization procedure that minimizes the squared error between the camera’s 2D measurements and the landmark projections (which are calculated using the pose hypothesis):

\[
\hat{\Theta} = \arg \min_{\Theta} \sum_{i=1}^{n} \left( (u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2 \right).
\] (5.1.3)

This least-squares solution is known to be the optimal solution when independently and identically distributed (i.i.d) isotropic errors are assumed. However, this solution is also known to be extremely sensitive to outliers. In order to estimate the pose parameters in a more robust manner, standard robust techniques, such as M-estimation (that searches for a re-weighted least-squares solution), should be used in practice. In the present work it is assumed that the pose parameters are periodically estimated as part of the control-loop. As a result, a relatively accurate initial guess of these parameters is always available. Hence, any outlier will yield a large residual from the very beginning and therefore it will be easily suppressed using the aforementioned robust techniques.

### 5.1.1 The Pose Covariance Matrix

The 2D measurements obtained from the camera are not error-free. These errors result from errors in the feature detection procedure and are commonly modelled as i.i.d Gaussian additive errors. Let \(\sigma_I\) be the standard deviation of this isotropic Gaussian distribution. In the absence of these errors the exact pose \(\Theta\) would have been obtained; however, in realistic scenarios these errors propagate through the optimization process and lead to the perturbed estimate of the pose \(\hat{\Theta}\).

Let \(f_i : \mathbb{R}^6 \mapsto \mathbb{R}^2\) be the perspective projection function of the \(i\)th landmark:

\[
f_i(\Theta) = (u_i, v_i)^T.
\] (5.1.4)

Then the Jacobian \(J_i\) of the \(i\)th landmark is the \(2 \times 6\) matrix containing all \(f_i\)’s partial derivatives, and the Jacobian matrix of all the landmarks which participate in the pose computation is defined as the concatenation of all the respective \(J_i\)s:

\[
J = [J_1^T, \ldots, J_n^T]^T.
\] (5.1.5)

Following the derivations in [34], a first-order approximation of the error propagation from image measurements to the pose parameters is given by the covariance matrix of \(\Theta\):

\[
\Sigma_\Theta = (J^T J)^{-1} J^T \Sigma_I J (J^T J)^{-1},
\] (5.1.6)

100
where $\Sigma_I$ in the above expression is the covariance matrix of the image measurements, which reflects the errors in the 2D measurements. Since it was assumed that these errors are i.i.d. and isotropic, $\Sigma_I$ takes the special form of a diagonal matrix with constant value $\sigma_I^2$ along the diagonal. Hence, the expression for the pose covariance matrix may be simplified as follows:

$$
\Sigma_\Theta = (J^T J)^{-1} J^T (\sigma_I^2 I) J (J^T J)^{-1}
$$

$$
= \sigma_I^2 (J^T J)^{-1} (J^T J) (J^T J)^{-1}
$$

$$
= \sigma_I^2 (J^T J)^{-1} = \sigma_I^2 \left( \sum_{i=1}^{n} J_i^T J_i \right)^{-1}
$$

(5.1.7)

The diagonal of the pose covariance matrix contains the variances of the six pose parameters, while the off-diagonal elements represent the dependencies between these parameters. This symmetric matrix also represents a 6D ellipsoid (usually known as the uncertainty ellipsoid) in the pose configuration space. The main axes of this ellipsoid are in the direction of $\Sigma_\Theta$’s eigenvectors and their lengths correspond to the square-roots of $\Sigma_\Theta$’s eigenvalues. One can think of this ellipsoid as an approximation of the volume in which the real pose is located up to some certainty. For accurate pose estimates this volume will be relatively small.

### 5.2 Task-Oriented Grading of Landmark Subsets

Consider a scenario where 100 landmarks are available for robot navigation. Selecting all these landmarks for the camera pose computation will probably yield a very accurate estimate. However, for each selected landmark, its 2D measurements need to be extracted from each image along the robot’s trajectory. A feature-extraction algorithm may be used in the first frame in order to identify the landmarks in the image, and some tracking algorithm will be used in the consecutive frames to obtain the 2D feature displacement. Since navigation is a real-time task, the overhead of the tracking procedure must be considered. According to the specific system capabilities and performance, the maximal number of features that can be tracked is usually bounded to be very small (e.g., 10 or even 4, as was proposed in [14]). It is clear that different selections of landmark subsets will lead to different pose accuracies. As an illustrative example, consider the choice of a subset containing landmarks with very small distances between them. Their projection rays will form a very narrow bundle, which will in turn lead to a very inaccurate pose estimate as compared to a subset of landmarks that are far away from each other. The landmark selection problem is simply defined as the problem of finding the best subset – the one that will lead to the most accurate pose.
Figure 5.1: Comparison between uncertainty ellipsoids in a 3D pose configuration space. It is clear that ellipsoids B and C are preferable to A, but the choice between these two is less obvious and should take into account the requirements of the specific navigation task.

As was already shown in section 5.1.1, the pose accuracy is not represented by a scalar but rather by a $6 \times 6$ covariance matrix. Any landmark subset will lead to a different covariance matrix. This leads to a fundamental question: given two covariance matrices, which one is “better”? Each covariance matrix reflects an uncertainty ellipsoid. If one of the ellipsoids contains the other, then it is clear that the smaller one should be preferred. For example, one can see that ellipsoids B and C in Fig. 5.1 are preferable to ellipsoid A. However, the choice between ellipsoids B and C is less obvious and should take into account the requirements of the specific navigation task. For example, if for some reason the x-parameter is much more important than the y-parameter to our navigation task, it may be preferable to choose ellipsoid C over ellipsoid B although it has higher uncertainty along the (less important) y direction.

In [13] and [14] each landmark subset was graded according to the condition number of its covariance matrix, which is defined as the ratio between the largest and smallest eigenvalues. In terms of uncertainty ellipsoids, the condition number is the squared lengths ratio of the longest and shortest main-axes, thus measuring the “roundness” of the ellipsoid. Using this criterion will bring us to choose the landmark subset with the most spherical uncertainty ellipsoid. Note that in our 3D example, ellipsoid A would have been chosen according to the condition number. Another problem with this criterion is that it perceives the configuration space as a Euclidian space, and thus an error of one angular unit (e.g., radian) in the camera orientation would be considered equivalent to an error of one metric unit (e.g., centimeter) in the pose...
translation vector. This behavior seems arbitrary and does not reflect the real severity of such errors.

In order to address the aforementioned issues, a new grading criterion for landmark subsets is proposed. First, instead of grading according to the uncertainty ellipsoid’s roundness, we would like to use a criterion that reflects its size. Two straightforward alternatives are the summation and the multiplication of the covariance matrix eigenvalues. These quantities can be easily obtained as the covariance matrix’s trace and determinant respectively. At first glance, it seems that the product of the eigenvalues would be a better choice since it is proportional to the squared volume of the uncertainty ellipsoid. However, such a criterion might prefer an ellipsoid with very long axis when the rest of the axes are very short and hence compensate for the long one. When summing the eigenvalues, on the other hand, the squared lengths of the axes are summed and hence will be relatively large even if only one of the axes is long. Additionally, a requirements matrix should be supplied by the system designer, who is familiar with the requirements of the specific navigation task. The requirements matrix, denoted $\Sigma_R$, should be symmetric, positive semi-definite, and reflect the importance of the different pose parameters (particularly the correct balance between angular and translational errors). The length of the uncertainty ellipsoid’s main axes will not be measured using the Euclidian norm but rather by the Mahalanobis norm induced by this requirements matrix. In section 5.4 a systematic method for constructing such a matrix is described.

Given a covariance matrix $\Sigma_\Theta$ that was obtained from a landmark subset, the grading criterion is developed as follows. Let $\Sigma_\Theta = M_\Theta \Lambda_\Theta M_\Theta^T$ be the eigenvectors-eigenvalues decomposition of $\Sigma_\Theta$. $M_\Theta = [\hat{m}_1, \cdots, \hat{m}_6]$ is an orthonormal matrix in which the eigenvectors of the covariance matrix are its columns, and $\Lambda_\Theta$ is a diagonal matrix containing the eigenvalues - $\lambda_i$. Therefore, the grade of the covariance matrix, which is defined to be the sum of squared Mahalanobis lengths of the ellipsoid’s main axes, is:

$$ grade = \sum_{i=1}^{6} \left\| \sqrt{\lambda_i} \cdot \hat{m}_i \right\|_R^2 = \sum_{i=1}^{6} \left( \sqrt{\lambda_i} \cdot \hat{m}_i \right)^T \Sigma_R \left( \sqrt{\lambda_i} \cdot \hat{m}_i \right) = \sum_{i=1}^{6} \lambda_i \cdot \hat{m}_i^T \Sigma_R \hat{m}_i, \quad (5.2.8) $$

where $\| \cdot \|_R$ denotes the Mahalanobis norm. In contrast to a simple summation of $\Sigma_\Theta$’s eigenvalues, here I obtained their weighted sum. The weights $\hat{m}_i^T \Sigma_R \hat{m}_i$ represent the severity of the pose errors in the $\hat{m}_i$ direction.

During the optimization process, where the landmark subset with the minimal grade is sought, the grade function is evaluated many times for different subsets. Therefore, in order to reduce the overhead of the optimization, it would be desirable to avoid the eigenvectors-eigenvalues decomposition of $\Sigma_\Theta$. Since it is only the sum of squared lengths that we need for the grade computation, this function takes a much simpler form:

$$ grade = tr [\Sigma_R \Sigma_\Theta], \quad (5.2.9) $$

103
where \( \text{tr} \) represents the matrix trace. The two grade definitions (5.2.8) and (5.2.9) are equivalent since:

\[
\text{tr} [\Sigma_R \Sigma_{\Theta}] \quad = \quad \text{tr} [\Sigma_R M_{\Theta} \Lambda_{\Theta} M_{\Theta}^T] \\
= \quad \text{tr} \left[ \sqrt{\Lambda_{\Theta}} M_{\Theta}^T \Sigma_R M_{\Theta} \sqrt{\Lambda_{\Theta}} \right] \\
= \quad \text{tr} \left[ \left[ \sqrt{\lambda_i \lambda_j \hat{m}_i^T \Sigma_R \hat{m}_j \Sigma_R \hat{m}_j} \right]_{i,j=1,...,6} \right] \\
= \quad \sum_{i=1}^{6} \lambda_i \hat{m}_i^T \Sigma_R \hat{m}_i,
\]

where \( \sqrt{\Lambda_{\Theta}} \) is the diagonal matrix containing the square roots of the six eigenvalues \(-\sqrt{\lambda_i}\). In the above manipulation I used a cyclic permutation of the matrices in the trace. Such a permutation is known to preserve the trace.

To conclude, given the requirements matrix of the navigation task, any landmark subset can be evaluated in a task-oriented manner by computing its corresponding covariance matrix, and then grading it according to the trace of the multiplication of these two matrices.

**5.3 Approximate Solution for the Subset Selection Problem**

Equipped with the task-oriented grading criterion, one can address the central problem of this chapter: the task-oriented landmark selection problem. Given a set of \( n \) available and visible landmarks, we would like to obtain the best landmark subset of some predefined size \( k \) \((k < n)\). This problem can be posed as an integer programming optimization problem by introducing \( n \) indicator variables, \( \alpha_i \in \{0, 1\} \) \((i = 1, \ldots, n)\), each indicating whether the corresponding landmark was selected to the subset. Let \( \alpha = (\alpha_1, \ldots, \alpha_n) \) be the vector concatenation of these variables. Stipulating the participation of each \( J_i \) in (5.1.7) according to its corresponding \( \alpha_i \) yields the subset’s covariance matrix:

\[
\Sigma_{\Theta}(\alpha) = \sigma_I^2 \left( \sum_{i=1}^{n} \alpha_i \cdot J_i^T J_i \right)^{-1}.
\]  

(5.3.10)

By substituting (5.3.10) into (5.2.9) and ignoring the constant factor \( \sigma_I^2 \), the integer-program becomes:

\[
\arg \min_{\alpha} \text{tr} \left[ \Sigma_R \left( \sum_{i=1}^{n} \alpha_i \cdot J_i^T J_i \right)^{-1} \right] \\
\text{s.t :} \\
\sum_{i=1}^{n} \alpha_i = k \\
\alpha_i \in \{0, 1\}.
\]

(5.3.11)
The first constraint guarantees that the obtained subset size will be as required, while the second constraint enforces the Boolean behavior of the indicators.

Computing the exact solution for this program is NP-Hard. However, a very good approximation can be obtained by solving the problem relaxation, where the Boolean restriction of the \( \alpha_i \) variables is replaced by the relaxed constraint \( 0 \leq \alpha_i \leq 1 \). The objective function of this program is convex as shown in the following subsection 5.3.1. Therefore, it can be solved using any non-linear optimization toolbox (e.g., [61]) to obtain the fractional solution. In order to decide which of the landmarks should be selected, a rounding heuristic should be applied to the obtained fractional \( \alpha_i \) variables. A well-known rounding method [7] proceeds as follows: each fractional \( \alpha_i \) is perceived as the probability that the corresponding landmark will be selected to the subset. Hence, several subsets are randomly constructed according to these probabilities; next, the grade of each subset is evaluated according to (5.2.9), and the subset with the minimal grade is chosen. Note that although the expected size of the subsets is \( k \) as desired (due to the subset size constraint), the actual size of the randomly generated subsets may be slightly different. Therefore, the random subsets should be corrected by adding or discarding landmarks in order to reach the necessary size, where the choice of which landmarks to add/discard is in accordance with the value of each fractional \( \alpha_i \): for subsets that are too large I discard the landmarks with the lowest \( \alpha_i \), and for subsets that are too small I add the landmarks with the largest \( \alpha_i \). Experimental results, which are presented in section 5.5, demonstrate that this scheme can obtain a very good approximation for the optimal solution.

### 5.3.1 Proof of the Grade Function Convexity

In this subsection I present a convexity proof for the objective function of the relaxed selection problem:

\[
\arg\min_{\alpha} \quad \text{grade}(\alpha) \\
\text{s.t.:} \\
\sum_{i=1}^{n} \alpha_i = k \\
0 \leq \alpha_i \leq 1,
\]

where

\[
\text{grade}(\alpha) = \text{tr} \left[ \Sigma_R \left( \sum_{i=1}^{n} \alpha_i \cdot J_i^T J_i \right)^{-1} \right].
\]

It is shown that the objective function \( \text{grade} \) is convex on \( \mathbb{R}_+^n \), and thus maintains its convexity on any convex subset of \( \mathbb{R}_+^n \). Since each of the constraints in the relaxed problem defines a convex set (half-space...
or hyperplane), so does their intersection, and therefore grade is convex on the problem’s feasible set. As a result, a search for a local minimum will surely lead to the global minimum as well.

**Proof.** There are several methods for proving function convexity. The *line restriction method*, which is used in this proof, relies on the following lemma.

**Lemma 1.** Let \( g : \mathbb{R} \to \mathbb{R} \) be defined as: \( g(t) = \text{grade}(\alpha + tv) \), where the domain of \( g \) is \( \{ t \mid \alpha + tv \text{ is feasible} \} \). Then, \( \text{grade} : \mathbb{R}^n_+ \to \mathbb{R} \) is convex iff \( g \) is convex (in \( t \)) for any feasible \( \alpha \) and \( v \in \mathbb{R}^n \).

Substituting (5.3.13) in the above \( g \) definition yields:

\[
g(t) = \text{tr} \left[ \Sigma_R \cdot A^{-1} \right], \tag{5.3.14}
\]

where

\[
A = \sum_{i=1}^{n} (\alpha_i + tv_i) J_i^T J_i. \tag{5.3.15}
\]

In order to show that \( g \) is convex, it is proved that its second derivative is non-negative. The following two equations are required for the derivations:

\[
\frac{\partial}{\partial t} \{ \text{tr} [X] \} = \text{tr} \left[ \frac{\partial}{\partial t} \{ X \} \right], \tag{5.3.16}
\]

\[
\frac{\partial}{\partial t} \{ X^{-1} \} = -X^{-1} \cdot \frac{\partial}{\partial t} \{ X \} \cdot X^{-1}, \tag{5.3.17}
\]

for any matrix \( X \) and scalar \( t \). Therefore, the first derivative of \( g \) is:

\[
g' = \text{tr} \left[ \Sigma_R \cdot \frac{\partial}{\partial t} \{ A^{-1} \} \right] = -\text{tr} \left[ \Sigma_R \cdot A^{-1} B A^{-1} \right], \tag{5.3.18}
\]

where the matrix \( B \) is defined as:

\[
B = \frac{\partial}{\partial t} \{ A \} = \sum_{i=1}^{n} v_i J_i^T J_i. \tag{5.3.19}
\]

Next, the second derivative of \( g \) is obtained by:

\[
g'' = -\text{tr} \left[ \Sigma_R \cdot \left( \frac{\partial}{\partial t} \{ A^{-1} \} \right) \cdot B A^{-1} + A^{-1} B \frac{\partial}{\partial t} \{ A^{-1} \} \right]
= -\text{tr} \left[ \Sigma_R \cdot \left( -A^{-1} B A^{-1} A^{-1} \right) \cdot B A^{-1} \right]
= 2\text{tr} \left[ \Sigma_R \cdot A^{-1} B A^{-1} A^{-1} \right]
= 2\text{tr} \left[ B A^{-1} \cdot \Sigma_R \cdot A^{-1} B A^{-1} \right]. \tag{5.3.20}
\]
In the last step of the above derivations I used a circular permutation of the matrices inside the trace. Such a permutation keeps the trace value unchanged. Since \((\alpha + tv)\) is assumed to be feasible, \(A\) is a symmetric positive semi-definite matrix and so is its inverse \(A^{-1}\). Therefore, \(A^{-1}\) can be decomposed using the Cholesky decomposition, as \(A^{-1} = C \cdot C^T\), where \(C\) is a real matrix. Once again the circular permutation of the matrices inside the trace will be used:

\[
g'' = 2 \text{tr} \left[ \left( C^T BA^{-1} \right) \cdot \Sigma_R \cdot \left( A^{-1} BC \right) \right].
\]

(5.3.21)

Because \(A^{-1}\) and \(B\) are symmetric matrices, it is clear that \((C^T BA^{-1}) = (A^{-1} BC)^T\). Taking into account that \(\Sigma_R \geq 0\), the multiplication of this matrix by \((C^T BA^{-1})\) and its transpose from each side does not affect its positiveness. Thus, the trace is activated on a positive semi-definite matrix, which yields a non-negative value, as desired.

\[
\square
\]

5.3.2 Normalization Procedure for Improving Numerical Stability

Computing the objective function of (5.3.11) requires the inversion of

\[
\sum_{i=1}^{n} \alpha_i \cdot J_i^T J_i.
\]

When all the selected landmarks are far away from the camera, the translational elements of the Jacobian become extremely small compared to the rotational counterparts, which are not affected by the landmark’s range. Therefore, an ill-conditioned matrix is obtained and the inversion procedure might be prone to numerical instability. In order to prevent such a problem, the Jacobian elements can be scaled by multiplication with a diagonal “normalization” matrix \(\tilde{N}\):

\[
\tilde{J}_i = J_i \cdot \tilde{N}.
\]

(5.3.22)

Each element on \(\tilde{N}\)’s diagonal can be chosen such that the average of the elements in the corresponding column of \(\tilde{J}\) will be 1. Next, one can define the normalized covariance matrix of a subset as:

\[
\tilde{\Sigma}_\Theta(\alpha) = \left( \sum_{i=1}^{n} \alpha_i \cdot \tilde{J}_i^T \tilde{J}_i \right)^{-1},
\]

(5.3.23)

where the constant factor \(\sigma_I^2\) was dropped as in (5.3.11). The computation of the normalized matrix should be numerically stable. However, plugging (5.3.23) back into (5.2.9) yields a different objective function that may obtain its minimum at a different \(\alpha\). In order to circumvent the aforementioned obstacle, the same
normalization should be applied to the requirements matrix:

\[ \tilde{\Sigma}_R = \tilde{N} \cdot \Sigma_R \cdot \tilde{N}. \]  (5.3.24)

As a result,

\[
tr \left[ \tilde{\Sigma}_R \tilde{\Sigma}_\Theta (\alpha) \right] = \left[ \tilde{N} \Sigma_R \tilde{N} \cdot \left( \sum_{i=1}^{n} \alpha_i \cdot \tilde{N} J_i^T J_i \tilde{N} \right)^{-1} \right] = \left[ \tilde{N}^{-1} \tilde{N} \Sigma_R \tilde{N} \cdot \left( \sum_{i=1}^{n} \alpha_i \cdot J_i^T J_i \right)^{-1} \right] = \left[ \Sigma_R \Sigma_\Theta (\alpha) \right].
\]

### 5.3.3 Posing the Relaxed Program as an SDP

The relaxed problem is a constrained non-linear optimization problem. Although it can be solved using general optimization toolboxes (e.g., [61]), the overhead of converging to an accurate solution might be large, thus disqualifying the proposed method for real-time navigation systems. However, this problem can be easily converted to a *Semi-Definite Programming (SDP)* problem for which powerful and very efficient algorithms exist [1, 7, 41, 66]. One can think of SDP as an extension of the well-known linear programming, in which the linear inequality constraints are extended by the so-called *Linear Matrix Inequality (LMI)* constraint. Such an LMI constraint on the \( \alpha \) variables should be in the form:

\[
\sum_{i=1}^{n} A_i \cdot \alpha_i + C \succeq 0,
\]  (5.3.25)

where \( A_i \) and \( C \) are symmetric matrices and the notation \( P \succeq Q \) reflects that \( P - Q \) should be positive semi-definite. Despite its name, one can see that such a constraint can express non-linear behavior (through the requirement that the matrix will be positive semi-definite).

Note that in the SDP formulation the objective function is still required to be linear in the problem’s variables. Recall the original relaxed problem:

\[
\begin{align*}
\arg \min_{\alpha} & \quad tr \left[ \tilde{\Sigma}_R \tilde{\Sigma}_\Theta (\alpha) \right] \\
\text{s.t.} : & \quad \sum_{i=1}^{n} \alpha_i = k \\
& \quad 0 \leq \alpha_i \leq 1.
\end{align*}
\]  (5.3.26)
In order to transfer the non-linearity of $\tilde{\Sigma}_\Theta(\alpha)$ from the objective function to the problem constraints (where non-linearity can be handled), I introduce 21 additional slack variables arranged in a $6 \times 6$ symmetric matrix $Y$:

$$
Y = 
\begin{bmatrix}
    y_1 & y_2 & y_4 & y_7 & y_{11} & y_{16} \\
    y_2 & y_3 & y_5 & y_8 & y_{12} & y_{17} \\
    y_4 & y_5 & y_6 & y_9 & y_{13} & y_{18} \\
    y_7 & y_8 & y_9 & y_{10} & y_{14} & y_{19} \\
    y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & y_{20} \\
    y_{16} & y_{17} & y_{18} & y_{19} & y_{20} & y_{21}
\end{bmatrix}.
$$

(5.3.27)

With these new variables, the problem can be redefined as:

$$
\begin{align*}
\text{arg min}_{\alpha,Y} & \quad \text{tr} \left[ \tilde{\Sigma}_R Y \right] \\
\text{s.t.:} & \quad Y \succeq \tilde{\Sigma}_\Theta(\alpha) \\
& \quad \sum_{i=1}^n \alpha_i = k \\
& \quad 0 \leq \alpha_i \leq 1.
\end{align*}
$$

(5.3.28)

In order to verify that (5.3.26) and (5.3.28) are equivalent, one needs to show that the following two conditions hold:

- Every feasible solution of (5.3.26) can be extended to a feasible solution of (5.3.28) by setting some values to $Y$ such that the two objective functions coincide (which implies that the optimum of (5.3.26) $\geq$ the optimum of (5.3.28)).

- The objective function of (5.3.26) is a lower bound of the objective function in (5.3.28) for any given $\alpha$ and $Y$ (implying that the optimum of (5.3.26) $\leq$ the optimum of (5.3.28)).

The first condition is easily verified by letting $Y$ be equal to $\tilde{\Sigma}_\Theta(\alpha)$. The second condition is proved in the following lemma:

**Lemma 2.** Let $Y \succeq \tilde{\Sigma}_\Theta(\alpha)$ and $\tilde{\Sigma}_R \succeq 0$ be defined as before. Then:

$$
\text{tr} \left[ \tilde{\Sigma}_R Y \right] \geq \text{tr} \left[ \Sigma_R \tilde{\Sigma}_\Theta(\alpha) \right].
$$
Proof.

\[
\begin{align*}
\text{tr} \left[ \tilde{\Sigma}_R Y \right] - \text{tr} \left[ \tilde{\Sigma}_R \tilde{\Sigma}_\Theta (\alpha) \right] &= \text{tr} \left[ \tilde{\Sigma}_R \left( Y - \tilde{\Sigma}_\Theta (\alpha) \right) \right] \\
&= \text{tr} \left[ U U^T \left( Y - \tilde{\Sigma}_\Theta (\alpha) \right) \right] \\
&= \text{tr} \left[ U^T \left( Y - \tilde{\Sigma}_\Theta (\alpha) \right) U \right] \geq 0.
\end{align*}
\]

In the above derivation, the Cholesky decomposition was used to decompose the requirements matrix into \( \tilde{\Sigma}_R = U U^T \). The next step is based on the well-known property that multiplying positive semi-definite matrix \( A \) from both sides by any matrix, \( U^T A U \), will not effect its positiveness. The last inequality results from the fact that the matrix trace is equal to the sum of its eigenvalues, which are all non-negative for positive semi-definite matrices.

In order to represent the non-linear constraint in (5.3.28) as an LMI, the Schur complement lemma will be used:

**Lemma 3** (Schur complement lemma). Let 

\[
A = \begin{bmatrix} B & C^T \\ C & D \end{bmatrix}
\]

be a symmetric matrix where \( B \) is positive definite. Then, \( A \) is positive semi-definite iff \( D - CB^{-1}C^T \) is positive semi-definite.

See [7] for a proof of this lemma. Thus, the constraint \( Y \succeq \tilde{\Sigma}_\Theta (\alpha) \) can be replaced by:

\[
\begin{bmatrix} \tilde{\Sigma}_\Theta^{-1} (\alpha) & I \\ I & Y \end{bmatrix} \succeq \begin{bmatrix} \sum_{i=1}^{n} \alpha_i \tilde{J}_i^T \tilde{J}_i & I \\ I & Y \end{bmatrix} \succeq 0. \tag{5.3.29}
\]

Finally, the LMI representation of our constraint is:

\[
\begin{bmatrix} \tilde{J}_1^T \tilde{J}_1 & 0 \\ 0 & 0 \end{bmatrix} \alpha_1 + \ldots + \begin{bmatrix} \tilde{J}_n^T \tilde{J}_n & 0 \\ 0 & 0 \end{bmatrix} \alpha_n + \begin{bmatrix} 0 & 0 \\ 0 & E_1 \end{bmatrix} y_1 + \ldots + \begin{bmatrix} 0 & 0 \\ 0 & E_{21} \end{bmatrix} y_{21} + \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \succeq 0, \tag{5.3.30}
\]
where $E_i (i = 1, \ldots, 21)$ are $6 \times 6$ matrices with all elements equal to zero except the entries of the corresponding $y_i$ in $Y$ (as was defined in (5.3.27)), which are set to one. For example:

$$E_2 \equiv \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

(5.3.31)

The relaxed problem in its new formulation can be fed into an SDP toolbox such as [8, 27, 83] to rapidly obtain its solution. Such toolboxes solve semi-definite problems using interior point algorithms, which simultaneously optimize two problems: the original minimization problem – known as the primal problem – and its dual maximization problem. As in the linear programming scenario, the optimum solutions of the two problems coincide. Monitoring the decreasing gap between the two solutions thus gives us a very simple stopping criterion for the optimization process. The convergence speed of interior point algorithms is known to be exponential. Together with the convexity of the problem, this implies that a correct solution with the desired accuracy can be obtained in almost a fixed number of iterations regardless of the quality of the initial guess. On a Pentium 4 machine, full convergence is reached after about 0.1 seconds for 10 available landmarks and about 0.5 seconds in the case of 100 available landmarks. Much faster results can be obtained with a small compromise on the obtained accuracy. Since the selection procedure should be activated only once in a while, its time consumption is not too high for reasonable problem sizes. Thus, the algorithm can be integrated into real-time navigation and control systems of autonomous robots.

5.3.4 Generalization for non-identical distribution of measurements error

There might be cases where some measure of stability or accuracy is associated with each of the available landmarks. In such a case, the covariance matrix $\Sigma_\Theta$ cannot be simplified as shown in equation (5.1.7) of section 5.1.1, since the assumption about identical distribution of the image measurements error is dropped. However, the landmarks are still assumed to be independent, and thus $\Sigma_I$ takes the form of a block-diagonal

111
matrix. Each block, denoted $W_i$, is a $2 \times 2$ covariance matrix associated with the $i$'th landmark’s measurements. The relaxed optimization program therefore becomes:

$$
\arg \min_\alpha \text{tr} \left[ \Sigma_R \cdot \tilde{\Sigma}_\Theta(\alpha) \left( \sum_{i=1}^{n} \alpha_i \cdot \tilde{J}_i^T W_i \tilde{J}_i \right) \tilde{\Sigma}_\Theta(\alpha) \right]
$$

s.t.:

$$
\sum_{i=1}^{n} \alpha_i = k \quad 0 \leq \alpha_i \leq 1.
$$

where $\tilde{\Sigma}_\Theta(\alpha)$ is defined in (5.3.23).

As before, this program can be solved using any nonlinear optimization toolbox to obtain the fractional solution, which later can be rounded into an integral solution, as described in the beginning of this section. However, it is not clear how to translate this generalized problem into a semi-definite programming formulation, and thus it will be more time consuming to solve.

### 5.4 $\Sigma_R$ Construction

This section elaborates a method for translating the requirements of the navigation task into the symmetric positive semi-definite $6 \times 6$ requirements matrix $\Sigma_R$. At first, a pose-severity function, denoted $S(\Theta)$, is defined by the system designer. This function evaluates how “bad” the pose is for the specific task. For example, if our task is to photograph an object $O \in \mathbb{R}^3$ in the scene, then a proper severity function could be the 2D distance between the object’s projection and the principle-point:

$$
S(\Theta) = \| \text{pp} - \text{proj}(R^T(O - p)) \|.
$$

where pp is the principal-point. Such a severity function reflects the desire to keep the object at the center of the image. Another classical example appears when the task is to follow some predefined trajectory. In this case, a reasonable severity function could measure the distance of the camera from the trajectory. Landing an airplane is an example of such a task: the trajectory leads the airplane along the landing track in a smooth and tangential manner. In the simpler case of a straight trajectory, defined by a source-point $s$ and a direction-vector $d$, the severity function takes the form of:

$$
S(\Theta) = \| (I - d \cdot d^T)(p - s) \|.
$$

In order to obtain $\Sigma_R$, I choose a close camera pose $\Theta_0$ which is optimal according to the severity function $S$. Thus, the value and gradient of $S$ at this pose vanish. In this simplified case, the second order
approximation of $S$ at any perturbed $\Theta$ is:

$$ S(\Theta) = \frac{1}{2} (\Theta - \Theta_0)^T H_S(\Theta_0) (\Theta - \Theta_0), $$

(5.4.35)

where $H_S$ is the Hessian matrix of $S$. The vector $(\Theta - \Theta_0)$ represents the perturbation direction in the pose’s configuration space, and therefore can be replaced by any of $\Sigma_0$’s eigenvectors, $\hat{m}_i$:

$$ S(\hat{m}_i) = \frac{1}{2} \hat{m}_i^T H_S(\Theta_0) \hat{m}_i. $$

(5.4.36)

Comparing the above result to (5.2.8), we observe that $S(\hat{m}_i)$ is proportional to the weights in the sum of the eigenvalues. Therefore, it is concluded that the Hessian may serve as the requirements matrix.

### 5.5 Results

In this section the performance and the advantages of the proposed algorithm are demonstrated through simulations and lab experimentation.

#### 5.5.1 Simulations

Obtaining the dual value as part of the SDP solution is advantageous: it is a lower bound on the primal grade optimum, which is in itself a lower bound on any feasible integral solution of the original problem. Hence, one can use the dual value to obtain an upper bound on the approximation factor of any evaluated landmark subset. In Fig. 5.2 the approximation factors obtained by the algorithm are evaluated. A set of 100 landmarks was synthetically generated, from which subsets of different sizes were selected. One can see that the obtained approximation is very good, almost 1 for any subset size larger than 3. For comparison, groups of 10 and 100 subsets were selected uniformly as well (the dotted and dashed lines in Fig. 5.2). As could be expected, the approximations obtained by this method were similar to those of the proposed algorithm when the subset size was near 100. However, a clear and drastic advantage can be observed in the more realistic scenarios where small subsets are selected.

In some scenarios there exists a clear intuition about which landmarks should be selected for the subset. Fig. 5.3 demonstrates two such scenarios. In Fig. 5.3(a), most of the landmarks are located as a dense cluster while only two landmarks are placed at distant locations. If one needs to select a subset of size 4 in order to estimate the camera position, it is obvious that the two distant landmarks should be included, while the rest of the subset will be selected from the cluster. As can be seen, the obtained fractional $\alpha$ values of these two features are 1, guaranteeing their inclusion as desired. In Fig. 5.3(b) a 10 landmark subset was selected.
Figure 5.2: Approximation factors that were obtained for different subset sizes. The solid line was obtained by the algorithm, the dotted line by taking the best of 10 uniformly selected subsets, and the dashed line by taking the best of 100 uniformly selected subsets.

from 100 synthetic landmarks that were placed on a plane parallel to the image plane. This time, computing the roll angle of the camera was defined as relevant for the task. In this case, landmarks should be selected from the image periphery in order to maximize their arc-length displacement for roll movement. Indeed, the relaxed $\alpha$ values are equally distributed between the most extrinsic landmarks, necessitating that the subset be chosen from them.

Figure 5.3: Synthetic landmarks were placed on a plane parallel to the image plane $z=30$ (black dots). The obtained fractional $\alpha$ values (vertical lines) and the selected subset (circles) are presented. (a) A subset of 4 landmarks was selected when the navigation task required the camera position. (b) A subset of 10 landmarks was selected when only the roll angle of the camera was required.
5.5.2 Lab Experimentation

In order to demonstrate the advantage of the proposed algorithm in real scenarios, two lab experiments were conducted: one with still images and the other with a video that was captured while a robotic arm was performing some tasks.

**Still Images Experiment**

For the first experiment two environments were constructed: the first one contained 100 coplanar landmarks that were defined by the squares’ corners on a $10 \times 10$ chessboard (see Fig. 5.4(a) or 5.4(b)), and the other contained 300 landmarks from 3 orthogonal chessboards (see Fig. 5.4(c)). A calibrated camera was placed in various positions and orientations in the two environments and images of $640 \times 480$ were captured. Using the Harris corner detector, the image location of the visible landmarks was found up to sub-pixel resolution and the ground-truth camera pose was calculated from all available landmarks. Next, different tasks were defined and subsets were selected accordingly using the task-oriented algorithm. In order to study the advantages of the algorithm for different sizes of subsets, I examined subsets ranging from 4 to 50 landmarks.

Fig. 5.4 shows the selected subsets of size 5 and 40 for three examined tasks: the first requires only the X-component of the camera position, the second requires the Y-component, and the third task is one in which an object is photographed as described in section 5.4. One can see that different subsets were automatically selected as a result of the different task definitions. For example in Fig. 5.4(d), only 25 out of the 40 selected landmarks were repeatedly selected regardless to the task, while the other 15 landmarks (37%) were selected differently from one task to another.

Next, for each examined task and subset size, 500 additional subsets were uniformly selected for comparison. Fig. 5.5 compares the weighted error of the pose obtained by the algorithm’s selected subset to the mean weighted error of the poses when using the uniformly selected subsets. All these subsets were selected from the environments presented in Fig. 5.4(c). The weighted error was defined as the Mahalanobis norm of the pose parameter error. A clear advantage of the proposed algorithm can be observed for all subset sizes, although this advantage diminishes for large subsets, as in Fig. 5.2. Note that the graphs are not always monotonic. This is due to the specific measurement errors of the selected points, which affect the quality of the result.
Figure 5.4: Subset selection of 5 and 40 landmarks for different tasks. (a) and (d) show coplanar landmarks parallel to the image plane, (b) and (e) show coplanar landmarks in general position, (c) and (f) show landmarks placed on 3 orthogonal planes. In all six images the markers represent the selected landmarks according to different navigational tasks: an ’x’ marker – a task of computing the X component of camera position, a diamond – a task of computing the Y component of camera position, a circle – a photographing task.

Figure 5.5: The weighted error obtained by different subset sizes for the scene presented in Fig. 5.4(c). The dashed line shows the mean error of the pose when using the uniformly selected subsets, the solid line shows the pose error when using the selected subset of the algorithm. In (a), the navigation task requires the X component of the camera position, while (b) shows the results for the photographing task.
Robot Experiment

In this experiment a video camera with a resolution of $720 \times 428$ pixels was attached to a robotic arm (see Fig. 5.6). This arm can be manipulated in 6 DOF and supplies the trajectory in which it was maneuvered up to sub-millimetric accuracy. The positional information that was gathered from the robot was not used during the navigation task but rather was collected and saved as a ground truth for the algorithm evaluation.

Six different scenarios were examined. These scenarios were a result of three scenes with different landmark constellations and two tasks to be performed. The first scene contained 90 coplanar landmarks homogeneously dispersed on a rectangular grid (see Fig. 5.7(a)). Only 34 landmarks from the first scene were kept in the second: the upper 30 (forming a dense cluster) and 4 additional landmarks, widely dispersed in the lower part of the scene (see Fig. 5.7(b)). The third scene was constructed from 34 landmarks lying on three orthogonal planes. Once again, 30 of them were located in a relatively dense cluster while the other 4 were dispersed in different locations (see Fig. 5.7(c)).

Using subsets of these landmarks, the robotic arm was supposed to perform two types of tasks: one was to accurately move along a predefined trajectory (path-following task), and the other was to keep a
Figure 5.7: Subsets of 10 landmarks that were selected during the six experiments. In (a), (b) and (c) a path-following task was performed. In (d), (e) and (f) a photographing task was performed. (a) and (d) show the first scene, (b) and (e) the second scene, and (c) and (f) show the third scene. The stars mark the selected landmarks. The circles in the sub-figures of the photographing task show the location of the sight.
predefined target in a fixed location on the image (referred to as the *sight*) for as long as the camera was in motion (photographing or targeting task).

While the camera was in motion, landmark subsets were selected whenever required: when one of the landmarks left the field of view, or after the camera pose shifted beyond a certain threshold. As part of the control loop, the camera pose was constantly estimated on the basis of the selected landmarks and, as a consequence, the robotic arm trajectory was periodically adjusted. Fig. 5.7 shows examples for the selection results when using the proposed algorithm in the different scenarios. Note that in the non-homogeneous landmark constellations the distant landmarks were always selected by the algorithm. This conforms to the intuition discussed in 5.5.1.

For each examined scenario, subsets of different sizes were tested (6-15 landmarks). For each subset size, ten experiments were performed: in five of them the subsets were selected using the proposed algorithm while in the others the subsets were selected arbitrarily for purposes of comparison. The mean and maximum task-related errors are presented in Figures 5.8 and 5.9. In all these figures the solid line represents the errors obtained when using the selection algorithm, and the dashed line represents the errors obtained when selecting the subsets arbitrarily. For the path-following task, the 3D distance between the ground-truth camera location and the predefined trajectory was computed. For the photographing task, the target was located in the captured images at sub-pixel accuracy and its 2D distance from the predefined sight was computed. It can be observed in all the experiments that there is a clear advantage to utilizing the proposed algorithm. As could be expected, the merit of the selection algorithm becomes more significant when the subset size decreases and when the distribution of the available landmarks is non-homogeneous. One illustration of this is the vanishing gap between the mean photographing errors in Fig. 5.9(a) as the subset size increases. This is due to the homogeneous ordering of the landmarks in that experiment.
Figure 5.8: Mean and maximal millimetric errors obtained for different subset sizes when a path following task was performed. In (a) and (b) the first landmark constellation was used, in (c) and (d) the second constellation, and in (e) and (f) the third constellation. Solid line - the error obtained when using the selection algorithm, dashed line - the error obtained when selecting the subsets arbitrarily.
Figure 5.9: Mean and maximal projection errors (in pixels) obtained for different subset sizes when a photographing task was performed. In (a) and (b) the first landmark constellation was used, in (c) and (d) the second constellation, and in (e) and (f) the third constellation. Solid line - the error obtained when using the selection algorithm, dashed line - the error obtained when selecting the subsets arbitrarily.
Chapter 6

Conclusions and Future Research Directions

In this dissertation I’ve presented new approaches for vision based navigation. I have introduced algorithms for computing the pose (position and orientation) and motion (translation and rotation) of a calibrated camera with respect to an external reference system.

All pose estimation algorithms utilize the available data – both the captured images and the information about the surrounding environment – in order to define a system of constraints on the navigation parameters. Next, some sort of optimization procedure is activated to estimate the sought parameters. The accuracy and robustness of the different algorithms are a direct result of the constraint system quality. As we have more information at our disposal the conditionality of the constraint systems may improve which in turn yield stronger algorithms.

In light of the above observation, the main concept which stands behind the new proposed approach is that the environmental information should be integrated in all stages of the algorithm. It is claimed that such a scheme utilizes the available data to the fullest and thus the accuracy of obtained results is improved. Digital Terrain Maps (DTM) were chosen as the source of environmental data since they encode all the structure of the scene and thus do not require that a set of specific features will be detected and match (unlike landmark based algorithms).

Two algorithms were presented and studied. The first algorithm uses correspondence between feature points in two images and the information provided by a DTM to build a simultaneous constraint on the pose and motion variables. The constraint requires a priori information about the pose of the camera in the first frame and assumes that the DTM can be linearized in the sense discussed in chapter 3. The final constraint is non-linear and hence, in general, needs to be solved by using numerical techniques. The constraint characterizing the two views plus DTM geometry presents several interesting features. First, at
least six correspondence points are required to solve the different variables. Second, the formulation includes epipolar geometry, showing that the DTM effectively encodes additional valuable information about the 3D scene. Third, the constraint does not suffer from the ambiguity that haunts the SFM problem. A study of the degenerate scenarios was also presented. In addition to the theoretical results, this dissertation contains implementation details for the algorithm and a numerical study on synthetic and model data. Performance comparison between the novel algorithm and an alternative three-step algorithm constructed from “state of the art” building-blocks is also included. A clear advantage has been shown for the novel algorithm in most reasonable scenarios.

The aforementioned algorithm, like some other counterpart algorithms, starts by computing feature correspondences between the images which is known to be rather challenging task in some scenes and suffers from the well known aperture problem. This difficulty was circumvented in the second presented alternative by following a direct-method scheme, meaning that it constructs a system of constraints based directly on the brightness of the images pixels. This yields a very robust and efficient algorithm and it was shown that it can perform very well in extremely difficult scenes and even in totally featureless environments. Once again, the constructed constraints are non-linear, and thus can only be solved using an iterative optimization algorithm which requires an initial guess of the navigation parameters. In order to overcome large errors of the a priori estimates a hierarchial scheme was followed: first low resolution images were used to correct most of the poses a priori errors and then higher resolutions were used to further refine the estimates. The applicability and accuracy of the algorithm were tested both under controlled simulated environment and on real data that was acquired during a flight of an ultra-light airplane. As expected, it was shown that by using dead reckoning navigation and activating the presented algorithm once in a while, one can maintain navigation errors bounded below a reasonable magnitude.

According to my examination, there are two important scenarios in which the proposed algorithms fail to compute an accurate pose estimates. The first scenario takes place when the airplane make a turn. Our constraints are based on the overlapping area of the two selected images, whether they use feature correspondences or compare the pixels brightness directly. Thus, in order to obtain an accurate estimate we are compelled to select an image couple with large overlap. However, during a turn the overlap requirement lead us to select image couple with short time gap and thus also with rather small baseline. The accuracy of the algorithm is known to deteriorate in such a case. This problem can be easily solved by adapting the algorithm to use omni-directional images. Indeed, it was shown that such panoramic acquisition system can significantly improve the stability of the feature based algorithm. Such a panoramic view of the scene prevents another problematic scenario when flying in the proximity of the terrain, e.g., close to a mountainside.
A regular camera will see only a small patch of the terrain which might have low structural complexity, that in turn, yields a degenerate constraints system. An omni camera, on the other hand, will still capture significant part if the surrounding scene and thus the performance of the algorithm will be maintained.

A possible continuation of the presented research is to extend the direct method version of the algorithm for handling omni-vision data as well. Secondly, the fact that position and translation appear linearly in the constraints strongly suggests the possibility of using the current schemes in a filtering scheme. It seems that fusing the approach with inertial navigation may provide an effective and accurate inertial/optical navigation algorithms. In such a case, using optical flow may be advantageous. Unlike the feature correspondences, optical flow measures the features’ velocities. Using such measurements the camera’s ego-motion can be derived in terms of its velocities instead of Euclidian transformation. These velocities are more naturally combined with the inertial measurements.

Another problematic scenario appears when flying above planar (or almost planar) terrain. As shown in section 3.1.5, a degenerate (or ill-conditioned) constraints system is constructed in such a case which prevent the computation of an accurate estimate. In such a scenario the available data simply do not contain the information about the camera pose. Indeed, shifting the camera trajectory in any constant direction which is parallel to the terrain’s plane will yield the very same image movements. The only way to obtain accurate pose estimates in such uninformative scenario is by adding additional information that enable us to distinguish between different locations on the terrain. A preferable method to supply additional information to the algorithm is by adding ortho-photo of the terrain to the navigation algorithm inputs. While such input can be used to solve the aforementioned ambiguity, it brings with it a whole set of new problems that need to be addressed. One of the most serious difficulties is that while the terrain topography does not change (or changes extremely slow) its appearance does change between different seasons of the year, different weathers and even different hours of the day according to the illumination conditions. In order to properly utilize ortho-photos this difficulty should be overcome.

In addition to the above mentioned research, another interesting question was addressed: how can one compare the results of two pose estimation algorithm? Since the pose has six degrees of freedom, the accuracy of its estimates is represented by a $6 \times 6$ covariance matrix. A new approach that incorporates task-related considerations was presented. It was shown that by defining the specific task requirements in the form of a requirements matrix a proper comparison framework can be applied. Under this framework, a new algorithm for task-oriented landmark selection was proposed. Due to performance limitations, a real-time navigation system can usually use only a small number of landmarks to compute the camera pose. It was shown that by properly defining the task requirements, different subsets from the available landmarks are
automatically selected. The obtained subset yields minimal uncertainty for the pose parameters according to the Mahalanobis metric, which is defined using the requirements matrix. Simulations and experimentation verify the advantages of integrating the proposed algorithm in real-time navigation systems.

The landmark selection problem should not be perceived as the only possible application of the task-oriented comparison framework. A variety of selection algorithms can be developed in the spirit of the proposed scheme. Such implementations may include for example the selection of optimal features for classification mechanism, the selection of flow vectors for ego-motion estimation and SFM algorithms, and even for the selection of corresponding features or image segments for the navigation algorithms that were proposed at the beginning of this work.
Bibliography


127


129


הערכה מאולצת של תגרשהOMPI

רונן לברכר
הערכה מאולצת של תחיצה במיקום

היבחר על מחקר

לשם מילוי מחיק של הדירישה לקבלת התואר
ורשות לفيلוסופיה

רונית לזר

הוגש לשכת התשכ"ו – משרד.setCellValue ליוינשלט
שבעת ושש"ב ת"ת
יחוד
מרץ 2008

Technion - Computer Science Department - Ph.D. Thesis PHD-2008-07 - 2008
המתקר נעצה בברחיבת

פרופ' אהוד ריבלין בפכוקלט' לימודים המחשב

הכרת תודת

ﺏﺭ.LoggerFactory לחרות ארוחי ריבלי' על החינוך, התוכנות, היצורה הכל
שלב החרת

ﺏﺭ LoggerFactory לחרות גמ לאסטר, למשתתפי הלוחרים על התוכנה.

אמר מודא לשלום על התוכנה המפעילה הנדיבות בה Sahara lentil.
תקציר

פחתת מרוכזת גיוון עצמאית,chaining, היא פעילות מפורצת שנותנת כנף רחב יותר לroleIdיה, תמיכה בשיתוף פעולה أشهرות מבית לблокי מספרי בלוק, ו quàת התוכנית הגלובלי, ותא הנחיתת החברה של השתייה עם התוכנות שלה. בגישה זו, הגלובליים מבקריות פעילותם ופריקטיזציה המישה, כמו גם התמונה וירטואלית מאית, ו yan. תנוהל הגלובליים וקומנה המעט מתכנתים, וירגון המשוחרר בו צורת הקשת המנטיטים בין מערכות ציר.

ה全世界 תכנית בשתייה בין התוכנות וקופים.

מרבית כמות מידע בלתי משכילה מדינה של מרוכזת ארגונאותו. המיתר של פרקיטיזציה, שהתקיפה, מונעת תכנית במחזור וגרדיאן של GPS, למעט פתרון זה, יכון גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובעテンר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובעテンר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובעテンר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, למתכנתים שבע, פתרון מהמחזור. גורם למסגרת הגלובליים בדואר. לסופי המחזור, שבע בווידאו בעלים שולחן רך או גורם GPS, ובע텐ר והנה, L."GPS."
structure from motion
Technion - Computer Science Department - Ph.D. Thesis PHD-2008-07 - 2008