Technische Universität Braunschweig

Institute of Flight Guidance

Platzhalter

Real-time Photogrammetry using monocular SLAM for Unmanned Aerial Vehicles

Master Thesis
June 11 2018

Alexander Kern - 4227786

Institute of Flight Guidance
Prof. Dr. Peter Hecker

supervised by:
M.Sc. Markus Bobbe
Dipl.Ing. Simon Batzdorfer

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Date and Signature

Contents

1 Introduction

The present thesis devotes to the implementation and evaluation of a real-time photogrammetry pipeline for unmanned aerial vehicles. As a result of this work the open source project REALM (REal-time Aerial Localization and Mapping) emerged and was published on GitHub. It is ready-to-use and scalable for a variety of scenarios and input data, allowing to create

- 2D maps from generic images (RGB, IR, ...) based on GNSS position and heading
- 2D maps from RGB images based on GNSS position and visual pose estimation
- 2.5D elevation maps from RGB images based on GNSS positioning and visual pose estimation using CPU and sparse cloud only
- 2.5D elevation maps from RGB images based on GNSS positioning and visual pose estimation using GPU accelerated stereo reconstruction

The proposed framework relies on several existing open source projects that are mainly introduced in chap. 3 and 4 and were crucial to achieve the previously listed features in the limited amount of time for this thesis. To remain modular a variety of interface classes have been implemented for future contributions, especially regarding the two core functionalities: pose estimation using monocular SLAM and densification of a sparse point cloud to reconstruct surface elevation.

Accompanied to the following chapters source code and videos can be found on

- GitHub: https://github.com/laxnpander/OpenREALM
- YouTube: ....

1.1 Motivation

The surveying of the world has always been an elementary part of human curiosity. With invention of the aircraft and the success of commercial aviation in the past century an additional viewpoint has been added to this curiosity. Observing the world from above never fails to impress. But it is not only an astonishing perspective, aerial imagery has made its way into diverse areas of life. Precision agriculture, urban planning and disaster risk management are only some of the most promising topics in the upcoming years, that will profit from advancements in this field [1]. Besides resolution of the acquired data, several other factors will thereby define the success. Availability, costs and time of the mapping process are of crucial importance to almost any of the use cases. The rapid progress in the field of Unmanned Aerial Vehicles (UAVs) bears a chance to significantly contribute to a solution for these challenges. Thanks to their potentially lightweight design and small dimensions UAVs are quickly manufactured in large number of pieces. With the development of open source autopilots [2][3] their autonomy has reached a level, where almost any untrained user can safely plan and execute flight missions. This trend already democratised aerial imagery and led to numerous software for processing and evaluation of acquired aerial data, such as Pix4D [4], Agisoft Photoscan [5], DroneDeploy [6] or Colmap [7]. The technique behind such software is typically referred to as "photogrammetry". While in the classical sense any kind of spatial measuring considering shape and/or position of objects in images is referred to as photogrammetry, today a very specific workflow is understood under this term. It typically involves algorithms such as bundle adjustment, that compute highly accurate camera pose and 3D surface information of the observed scene in a large, geometric optimization formulation. Despite bundle adjustment being exceptionally efficient and fast considering the usually large number of parameters to be refined [8], most of the mentioned frameworks exclusively work after flight with all images captured. This is not optimal in several ways. Intervention during the mission based on the acquired data is not possible, although all information is theoretically available. The overall time that passes from mission start to observation of the global map can take up to several hours, which in case of a search and rescue scenario is critical. And finally the acquisition time is not used for processing, even though the UAV is operating autonomously.

1.2 Problem Statement

In the scope of this thesis a real-time aerial photogrammetry pipeline should be developed. As input serves a continuous image stream acquired by a calibrated, 2-axis gimbal stabilized camera attached to a UAV, as seen in fig. 1.1. Generated output must include an estimate of the camera motion, depth maps for each view, an incrementally updated 2D map considering surface elevation (orthophoto), as well as a sequentially expanded dense 3D point cloud of observed scene.

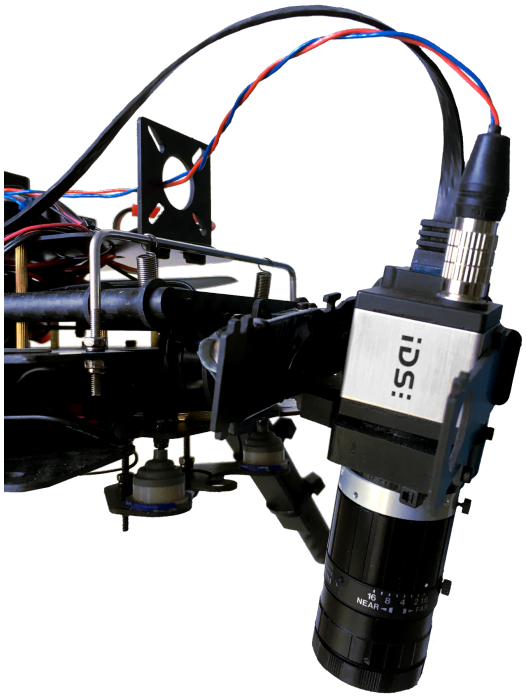


Figure 1.1: Two-axis gimbal stabilized camera attached to a UAV.

The fundamental problem of this task is explained with fig. 1.2. The consecutively acquired images are the main source of information, on which basis the described output must be extracted. Yet, the images contain no direct information about the motion, distance of the scene, nor what is displayed. They are an ordered grid of measured intensities for different wavelengths of the light that describe the observed scene. To somehow align the two images to a global map a straightforward approach is to identify unique points in one image, that might be easily recognizable in the other. This can work well, assuming the scene has such unique points, but in the next step the alignment by 2D image points allows no definite answer about the motion of the camera itself. The image might be captured by a UAV from several hundred meters altitude, but it might as well be zoomed and taken from an airplane from thousands of meters height.

Figure 1.2: Consecutive image pair captured by a UAV.

The current state of the art can answer these basic questions, yet the answers also depend on the underlying model assumptions. Existing solutions can be categorized according to fig. 1.3. On the one hand the classical photogrammetry exists, that is able to reconstruct the scene very accurately and in high resolution with many free model parameters (camera pose, calibration, 3D surface). However, it typically requires all input data at once and can not be updated incrementally. On the other hand there are the solutions for fast mapping using 2D-stitching techniques, which in turn are limited in their validity and quickly produce distorted maps as soon as the underlying assumptions are violated.

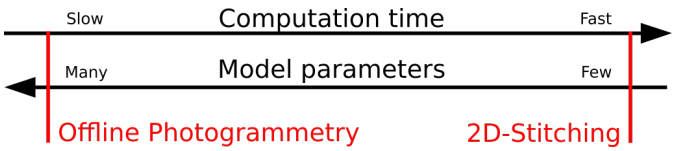


Figure 1.3: Classification of existing solution.

Addressing this issue consequently leads to two possibilities. The first is to make existing offline photogrammetry frameworks faster and allow them to compute images incrementally. The second is to improve the state of the art for real-time mapping, so it is able to produce the desired outputs. This thesis follows the latter approach and describes how this is achieved in the consecutive chapters.

1.3 Outline

In chap. 2 the required background for this thesis is presented. At first, single view geometry is described, which sets 2D image plane and 3D world into mathematical relation. Afterwards the fundamental principles of multiple view geometry are outlined. The focus lies on the theory of surface reconstruction and camera pose estimation. This is followed by an analysis of the state of the art for all relevant topics, including visual SLAM, real-time 3D dense reconstruction and mapping. Based on all this, the proposed implementation developed for this thesis is then described in chap. 4. It is designed as classical pipeline consisting of several coupled processing stages, which in turn are extensively characterized subsequently. The evaluation of the proposed framework is finally presented in chap. 5 using a captured test dataset. Conclusions and a short discussion about future research end this work in chap. 6 and 7.

2 Background

In the following chapters an overview of the principles involved with geometric computer vision is given. These principles are crucial to the understanding of the proposed real-time aerial photogrammetry framework. At first, the mathematical background of single view modelling is presented. Projective geometry, transformations in 3-space and the pinhole camera model are the main topics covered. Subsequently section 2.2 follows with the principles of 3D reconstruction from multiple views. Especially the concepts of depth and depth maps are important. As source of information served the work of Hartley and Zisserman "Multiple View Geometry in Computer Vision" as well as Richard Szelski's "Computer Vision: Algorithms and Applications", which cover all chapters and more in great detail.

2.1 Single View Geometry

Single view geometry describes the mapping between 3D world and 2D image, which is involved with every photo taken from a camera. Section 2.1.1 therefore starts with the transition between 3D and 2D utilizing projective geometry, homogeneous coordinates and the projective transformation. Afterwards in sec. 2.1.2 transformations in 3D will be described. The concrete application to the camera as measuring device is subsequently presented using the pinhole camera model.

2.1.1 Projective Geometry

A point in a plane is typically represented as pair of coordinates $(x, y)^T$ in $\mathbb{R}^2$. In consequence every point that fulfills the equation

$$ax + by + c = 0 \tag{2.1}$$

lies on the same line within the plane. Considered in vector space the point $x = (x, y)^T$ is also a vector and the line can be formulated as

$$(x, y, 1)(a, b, c)^T = 0. \tag{2.2}$$

This representation is also referred to as the „homogenous coordinates" of point x or the projective space $\mathbb{P}^2$ [9, S.27]. It is a mathematical formulation of the central projection,

where all possible lines in space meet at one distinct root point, which is referred to as "projection center". The additional coordinate dimension allows to treat points as lines and vice versa [9, S.30]. Figure 2.1 illustrates this relationship. By tracing the straight line between a point x in plane π and the projection center O the intersection with an arbitrary plane π' can be calculated. This intersection point x' is the projection of x to plane π'.

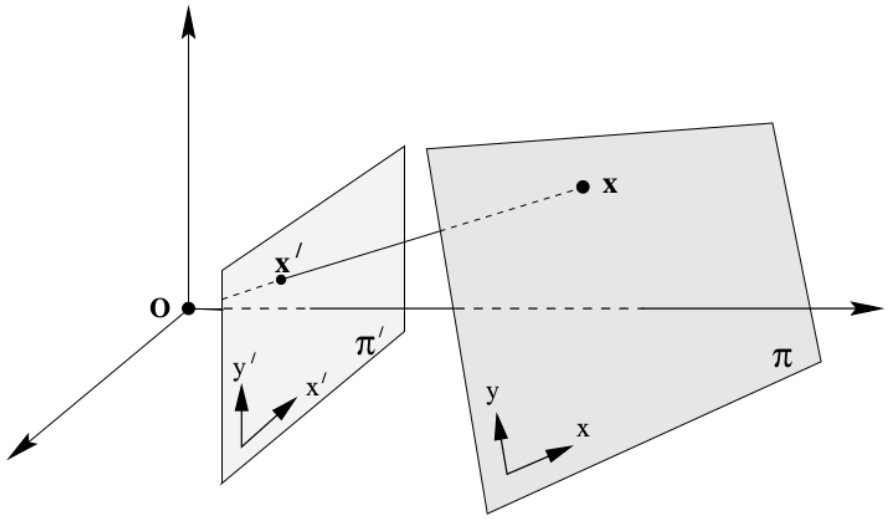


Figure 2.1: Central projection [**Hartley2004**].

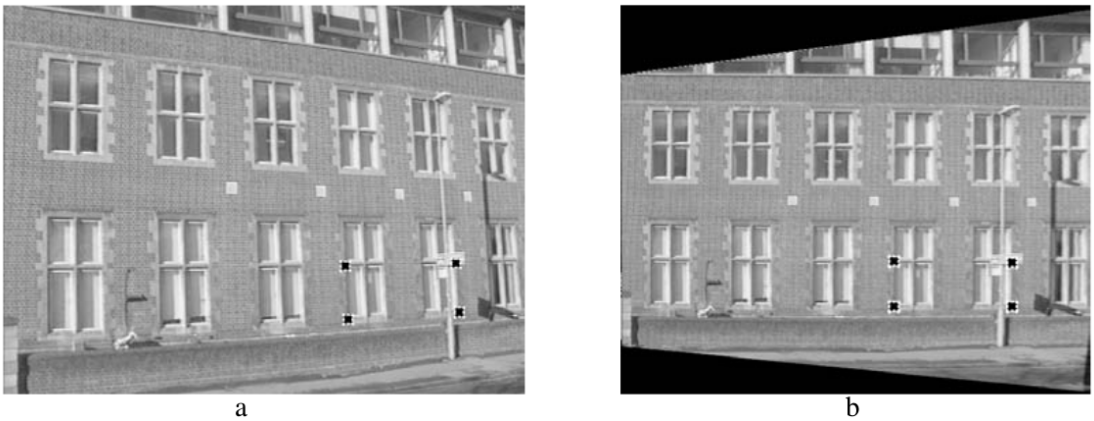


Figure 2.2: Removing perspective distortion [**Hartley2004**].

In most use cases multiple points from plane π should be projected to π'. To describe this transition efficiently a 3x3 transformation matrix called "Homography" can be determined from a set of corresponding points in both planes. Figure 2.2 illustrates this with an example photo taken of an exterior wall. The photographer observes the wall under a small angle (a). By selecting four corners of a window and choosing them to form an appropriately sized rectangle, the photo can synthetically be transformed to look like

the photographer stood right in front of the wall. The transformation removes the perspective distortion. Depending on the exact relationship of the planes to be mapped, the maximum degrees of freedom (dof) for this transformation ranges from a full projection with 8 dof as in the example, to a simple euclidean with 3 dof (2 translations, 1 rotation). Fig. 2.3 gives an overview of all possibilities. Note that this representation is only valid to map points from one plane to another. If point sets do not form an ideal plane, the computed homography can only be obtained with an error from a best-fit plane.

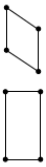
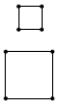
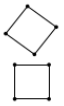
Group	Matrix	Distortion
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$	
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	

Figure 2.3: Perspective transformations [**Hartley2004**].

2.1.2 Transformations in 3D

Although formulations of section 2.1.1 are mostly demonstrated for 2D coordinates, they are also valid but less vivid for higher dimensions. In the context of this work the mapping of 3D coordinates $(x, y, z)^T$ to projective space $X = (x, y, z, 1)^T$ is of equal interest as the 2-space case. While the latter is used to model the acquisition process itself, 3-space formulations are often applied to the resulting measurements. These measurements are typically point clouds that represent the real world object at discrete samples. To be able to move, rotate, scale and deform the points efficiently, linear algebra provides a matrix formulation based on homogeneous coordinates. Fig. 2.4 shows an overview of the 3D coordinate transformations starting with the fewest degrees of freedom at the top and increasing to the bottom. A translation only has three dof, the x-, y- and z-displacement.

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{3\times4}$	3	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{3\times4}$	6	lengths	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{3\times4}$	7	angles	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{3\times4}$	12	parallelism	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{4\times4}$	15	straight lines	

Figure 2.4: Hierarchy of 3D coordinate transformations [10, S.39].

While the figure states, this preserves the orientation, it also sustains all the other properties of the transformations listed below that. In consequence a rigid transformation preserves all the properties except for the orientation. This kind of schema is therefore also referred to as "hierarchy" of transformations [10, S.39].

Of special interest for this work is the similarity transformation. It allows to rotate, translate and scale point clouds uniformly. Because of the information loss through projection from 3- to 2-space, the overall scale of monocular measurements is ambiguous and often chosen randomly. To restore this information, additional assumptions or techniques have to be used (e.g. known distances). Similarity transformations can be utilized to apply the identified real world scale to the randomly scaled data through

$$T_{sim} X = \tilde{X}, \tag{2.3}$$

where T_{sim} is the 3x4 similarity transformation matrix, X a data point of the randomly scaled point cloud in homogenous representation and $\tilde{X}$ the data point in real world scale. Often there is also the need to apply several transformations from a root to a destination coordinate system. This can be achieved by concatenating multiplications in the form of

$$T_{dst} T_2 T_1 T_{root} = T_{root2dst} \tag{2.4}$$

where T are all 4x4 transformation matrices in homogenous representation and the direction is defined from right to left. To transform points in the opposite direction (here: destination to root), the inverse of the matrix must be computed. This can be formulated as

$$T^{-1}_{dst2root} \tilde{X} = X. \tag{2.5}$$

Note that the inverse of a matrix is only defined for squared matrices, which is why all non-homogeneous transformation matrices (3x4 dimensions) must be provided with an

additional row of (0,0,0,1). As additional remark, for Euclidean transformations there is a more efficient way to determine the inverse by computing

$$T^{-1}_{euclidean} = (R^T | -R^T t) \tag{2.6}$$

The inverse through transposition is only valid for orthogonal matrices and can therefore not be applied to the rotation of a similarity, affine or projective transformation.

2.1.3 Pinhole Camera Model and Distortion

For computer vision formulations of the previous chapters can be applied to the simplified image acquisition process of a pinhole camera shown in figure 2.5. Rays of light from an external source hit an object. Depending on the surface of the object the resulting ray is refracted and proceeds through the image plane and the projection center of the camera. As measurand a point with $(x, y)^T$ is acquired. Note that the image plane is just another representation of the sensor plane, but scaled and often displayed on the opposite side of the projection center. With known internal parameters (intrinsics) as well as rotation and

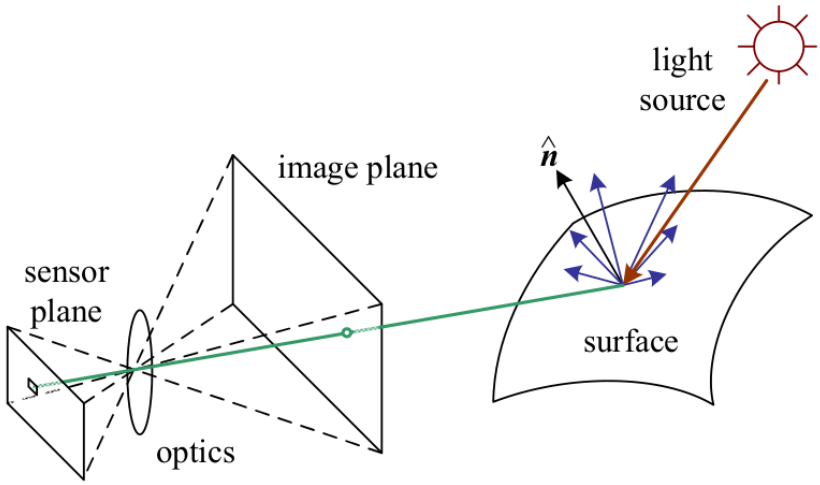


Figure 2.5: Simplified model of the camera's image acquisition process [10, S.61].

translation of the camera in the world frame (extrinsics), the ray of light can be traced back from the original world coordinate using the homogenous representation of the central projection. Formulated as simple matrix multiplication this is

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = PX, \tag{2.7}$$

where P is the camera projection matrix, X the homogeneous world point, $(u, v, 1)^T$ the 2D projection of X in the image plane and s the arbitrary scaling factor [10, S.51] [9, S.153].

In some literature s and $(u, v, 1)^T$ are integrated into one representation with

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = x \tag{2.8}$$

and therefore equation X also

$$x = PX. \tag{2.9}$$

The projection matrix P is only valid for linear modelling and can also be expressed by previously mentioned intrinsic and extrinsic camera parameters through the equation

$$P = K(R|t), \tag{2.10}$$

where K is the 3x3 intrinsic camera calibration and $(R|t)$ the 3x4 extrinsic camera pose. While the latter is described with a 3D transformation matrix consisting of a 3x3 rotation matrix R and a 3x1 translation vector t, the camera calibration matrix K is yet undefined.

In figure 2.6 the schematic composition of the pinhole camera model is displayed. It consists of a principal point $c = (cx, cy)^T$, an image with dimensions WxH and a focal length f. Because the pinhole camera is a linear projection model, it can be expressed as 3x3 matrix in the form

$$K = \begin{pmatrix} f & 0 & cx \\ 0 & f & cy \\ 0 & 0 & 1 \end{pmatrix} \tag{2.11}$$

which provides the proposed matrix K. Extended formulations add skew α for non-rectangular pixels and aspect ratio for non-square image dimensions on the focal length with

$$K = \begin{pmatrix} fx & \alpha & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.12}$$

These parameters will often be computed before complex computer vision tasks in a separate camera calibration step. They are crucial to any reconstruction approach, as small inaccuracies can lead to big errors due to the projective nature. This is also the reason why state of the art algorithms model and estimate the distortion of the lens in addition to the parameters of K. Due to the non-linear behaviour of lens distortion it can not be expressed as linear transformation matrix and a common approach is therefore to define a high order polynomial to describe the deflection. Afterwards a minimization problem is set up and the coefficients are estimated through levenberg-marquardt optimization. In consequence for every image point $x = (x, y)^T$ an undistorted position can be computed.

Figure 2.7 illustrates this context. A rectangular image that is captured by a camera with a distorted lens (e.g. barrel or pincushion distortion) must be modified so it fulfills the

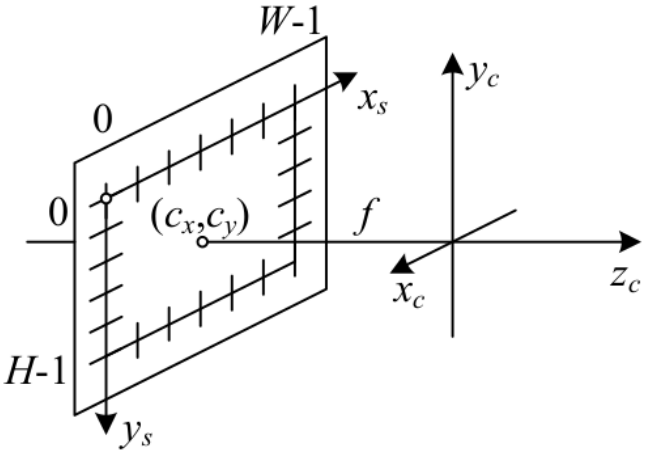


Figure 2.6: Simplified camera intrinsics [10, S.52].

underlying linear model again. Or in other words: to avoid the modelling of "curved" light rays, the image itself must be remapped as if the light rays run straight. In the open source library "OpenCV" for example the undistortion of x is documented at [11] The distortion coefficients returned are k_1, k_2, p_1, p_2 and k_3. This formulation follows the proposed approach in [12].

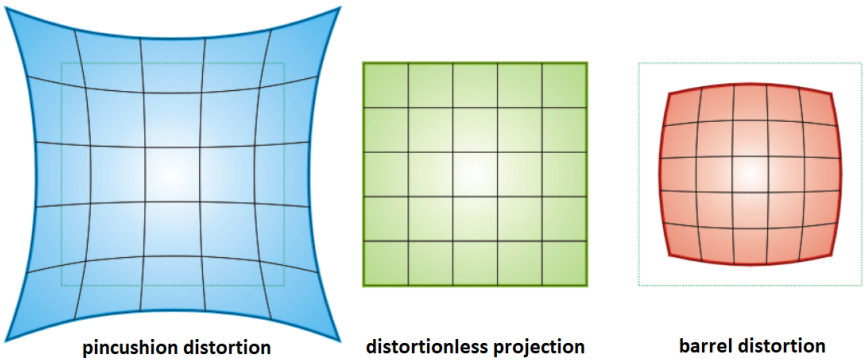


Figure 2.7: Different types of lens distortion.

2.2 Multiple View Geometry

Following eq. 2.7 a naive approach to reconstruct 3-space world from 2-space image points might be to invert the camera projection matrix P. In consequence, by knowing the intrinsic and extrinsic camera calibration a full reconstruction of the observed surface from a single photo would be possible. However, this is not consistent with our everyday experience. There is no way of knowing the distance or size of an unknown object just by looking at a photo of it. It can either be small and close to the camera or big and far away.

Mathematically this is depicted by the fact, that all 3D world points along one projection ray are mapped into the same image coordinate. In return, a measured image coordinate can be mapped into a projection ray, that represents all possible world coordinates along its way.

This fundamental problem can be solved by utilizing additional information. State of the art approaches use deep learning to train special properties within the picture (e.g. diminishing lines) to estimate the depth through experience [13]. Other techniques rely on structured light to triangulate between a projector and the camera system [14]. For aerial photogrammetry in an unknown environment the triangulation between multiple images, also known as "structure from motion", has proven to be a reliable standard. Therefore the geometric relationship between two views is explained in section 2.2.1. Afterwards, the concepts of depth and depth maps are described. At last, a sparse geometric optimization formulation called "bundle adjustment" is outlined. It can be seen as final step in many reconstruction approaches to simultaneously achieve the best solution for viewing parameters as well as 3D world points.

2.2.1 Epipolar Geometry and Triangulation

The reconstruction of 3D world points can be accomplished by triangulation from multiple viewpoints. The minimum number of views is defined by the intersection of rays and therefore two. To efficiently identify corresponding points between images and extract projection rays the basic relationship between multiple cameras must be analyzed. One important principle evolved from that is the epipolar geometry [9, S.241]. Its algebraic representation is described by the fundamental matrix F, which is very similiar to the Homography describing the mapping between two related planes.

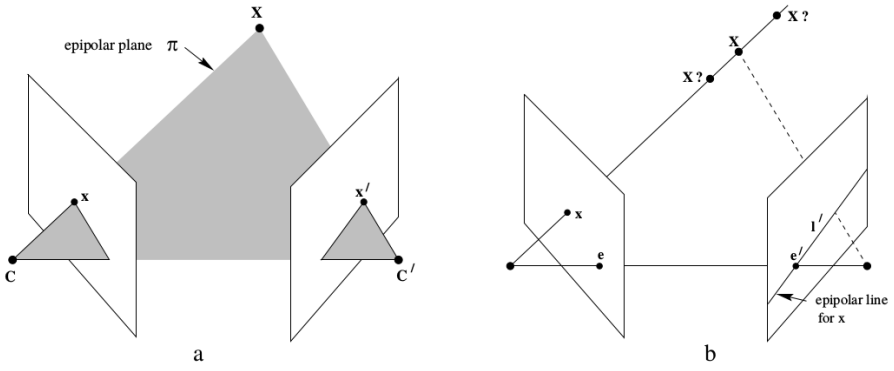


Figure 2.8: Epipolar geometry between two views [9, S.240].

Figure 2.8 illustrates the background of the epipolar geometry. For an image coordinate x in the left view together with the projection centers C and C' of both cameras a plane π

is spanned (a). This plane is called the "epipolar plane" and contains as well the observed 3D world point as the corresponding image coordinate x' in the right view. Consequently for every point in the left view the search for the same observed world point in the right view reduces to a 1D problem along the so called epipolar line l' in the image (b). In addition, the connecting line between C and C' is called the baseline. Its point of intersection with each image plane is the epipole e, respectively e'. Every epipolar line intersects with all other possible epipolar lines within an image in the epipole due to this fundamental relationship between the views. Note that for e.g. pure rotational motion $C = C'$ and therefore no unique solution for F can be obtained.

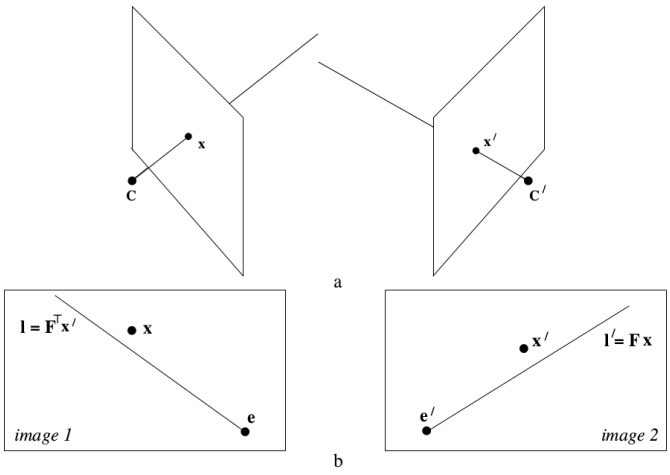


Figure 2.9: Misaligned image pair [9, S.231].

However, in reality the pose of two cameras can never be perfectly estimated. Uncertainties in the model and acquisition process will always yield to misalignments, in which case the search along the epipolar line will fail to give the correct corresponding point. Figure 2.9 illustrates this problem describing two error prone rays in (a). The resulting epipolar lines are displayed in their relative image in (b). It is obvious, that a straightforward triangulation under this prerequisites inherits the errors made up to this step, because there is no point X which exactly satisfies the projective equations

$$x = PX, x' = P'X \tag{2.13}$$

for the views. A good reconstruction approach has to consider this and different techniques therefore produce different results depending on their additional assumptions or constraints.

In the linear triangulation method eq. 2.13 is combined into a form $AX = 0$ with

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \tag{2.14}$$

where p^{iT} are the rows of P and (x, y),(x', y') the coordinates of x, respectively x'. The system of linear equations can be transferred to an optimization formulation, consequently the error between the rays is minimized.

2.2.2 Depth and Depth Maps

Based on the epipolar geometry and its algebraic representation a 3D world point can be triangulated from different views using eq. 2.14. For a dense reconstruction such triangulation is ideally performed once for every pixel of the image. This circumstance allows a very efficient representation of the observed surface. Instead of directly computing all 3D-coordinates, the distance of each pixel to the scene can be used. These distance values are typically referred to as depth and single channel images with depth values encoded at each pixel as "depth maps" (see fig. 2.10). While the contained information stays the same, depth maps are more memory efficient and illustrative.

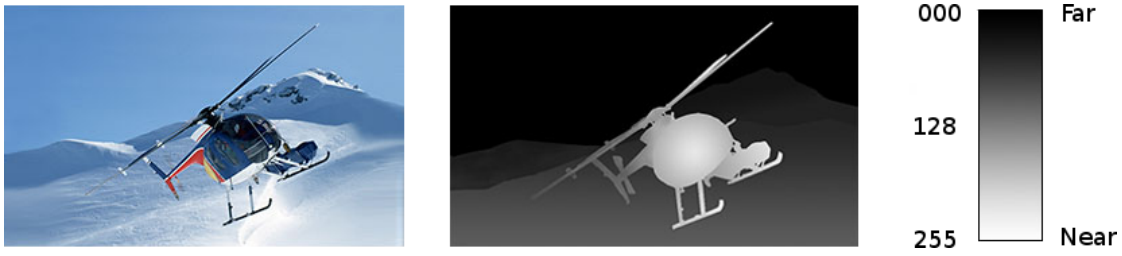


Figure 2.10: Depth map of an observed object [9, S.231].

For depth map computation according to eq. 2.9 a world point X can be projected to the ray $x = w(u, v, 1)^T$ using the camera projection matrix with $x = PX$. The last row of this matrix multiplication is

$$(p_{31}, p_{32}, p_{33}, p_{34}) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = w. \tag{2.15}$$

Following [9, S.162] w can be interpreted as depth, if

$$det(n) > 0, ||n|| = 1 \tag{2.16}$$

applies for the principal axis direction $n = (p_{31}, p_{32}, p_{33})^T$. The pinhole camera model $P = K(R|t)$ can provide such properties as rotation matrix R is orthonormal. In consequence depth d is computed with

$$(r_{31}, r_{32}, r_{33}, t_z) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = d \tag{2.17}$$

To reproject a depth map to a 3D point cloud under the same prerequisites eq. 2.9 can be utilized again in the form

$$x = KRX + t \tag{2.18}$$

and with $R^{-1} = R^T$, $x = d(u, v, 1)^T$ follows

$$R^{-T}K^{-1}d \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} - R^T t = X. \tag{2.19}$$

It is often convenient to define the camera pose as transformation from the camera to the world frame. With eq. 2.6 as the inverse of the euclidean transformation the linear projection matrix turns to

$$P = K(R^T | - R^T t) \tag{2.20}$$

and therefore eq. 2.19 to

$$RK^{-1}d \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + t = X. \tag{2.21}$$

2.2.3 Bundle Adjustment

In the previous sections the fundamental relationship between camera pose, calibration, image and world points has been evaluated. It was shown, that all these quantities are tightly coupled in the central projection (eq. 2.9). To achieve highest accuracies, robust non-linear optimization has to be carried out to jointly estimate 3D structure and viewing parameters (pose and/or calibration) [10, S.363]. Bundle adjustment is a generic term for such optimization formulations in visual reconstruction and photogrammetry. The name refers to the 'bundles' of light rays, derived from the simplified image acquisition model described in section 2.1.3, that connect 3D world points and projection centers (see fig. 2.11). While each camera can be considered a separate system, they are linked to one another through the same observed scene. For world point X_i this is formulated as

$$x_{1i} = P_1 X_i, x_{2i} = P_2 X_i, \ldots, x_{ni} = P_n X_i. \tag{2.22}$$

The constructed rays are adjusted, so that a defined cost function including optional constraints is minimized. One common measure for deviations is the geometric error, or reprojection error [9, S.314]. Its definition is visualized in fig. 2.12. A world point X is observed by two cameras with projection centers C and C' at the corresponding point positions $x \leftrightarrow x'$. However, due to noise the true position $\overline{x} \leftrightarrow \overline{x}'$ which exactly fulfills the epipolar constraint $\overline{x}'^T F \overline{x} = 0$ is unknown. By minimizing the euclidean distance $d(*, *)$ in the cost function f with

$$f(x, x') = d(x, \hat{x})^2 + d(x', \hat{x}')^2 \tag{2.23}$$

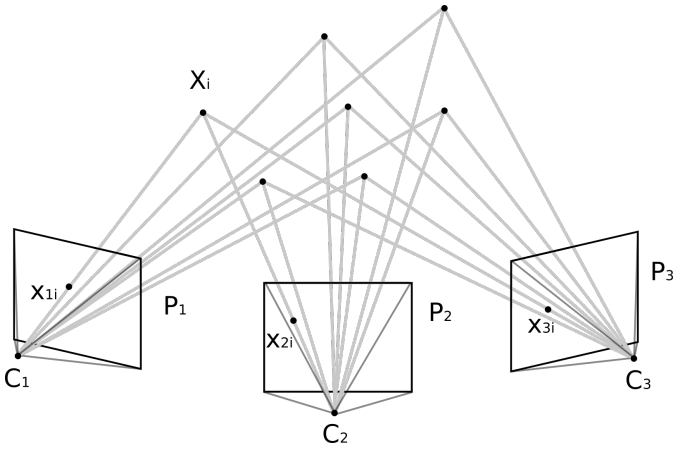


Figure 2.11: Bundles of light rays for multiple cameras observing the same scene.

the points $\hat{x} \leftrightarrow \hat{x}'$ can be found, that are the Maximum Likelihood Estimates (MLE) of the true position under the assumption of gaussian noise.

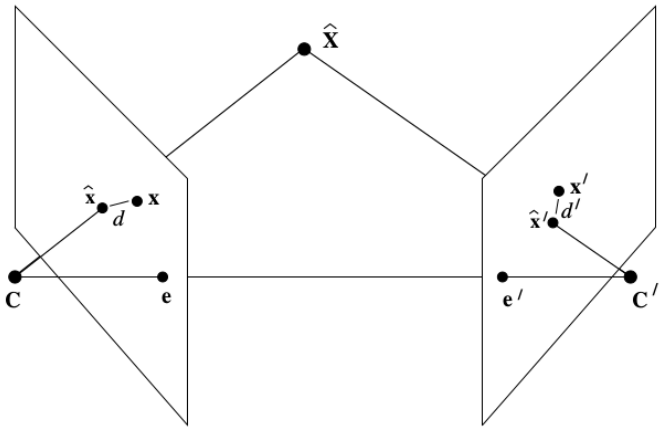


Figure 2.12: Reprojection error visualized [9, S.314].

These constraints can be set up for all cameras and observed points, resulting in a huge overdetermined system of non-linear equations. Several millions of measurements (feature matches) and hundred thousands of unknown parameters (3D point positions, camera poses, intrinsics) can result from a bundle adjustment formulation [10, S.364]. This might seem like an overwhelming amount of data whose solution has to be complex. However due to the fact every feature point x_{ij} only depends on one world point position X_i and one camera pose $(R|t)$, the problem is sparse and allows very efficient solutions. In their publication from 2000 Triggs et. al. summed up the state of the art of bundle adjustment in a generic framework and highlighted the benefits and potential for the future. They expect it to predominate the computer vision branch in a similar way as it has done in photogrammetry. For a deeper insight into this topic therefore refer to [8].

2.3 Camera Pose Estimation

In the context of this work the 3D transformation of a camera relative to another coordinate frame is also referred to as “camera pose” or “pose”. If not explicitly defined, it is always a rigid body motion described by a 3x4 transformation matrix

$$M = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix}. \tag{2.24}$$

The estimation of it is crucial to the proposed implementation in chapter 4. The prerequisites given by the scene and application define the chosen approach for pose estimation. First, the Perspective-n-Point-Algorithm (PnP) is outlined, working with 3D world points as prior information and allowing to extract M from a single view. Approaches based on homography or fundamental matrix however need multiple views, but can be applied even to unknown scenes. Consequently this section utilizes both single (2.1) and multiple view theory (2.2). Note also, that the focus is on the use cases for the respective approach. For detailed descriptions refer to the specified sections and references.

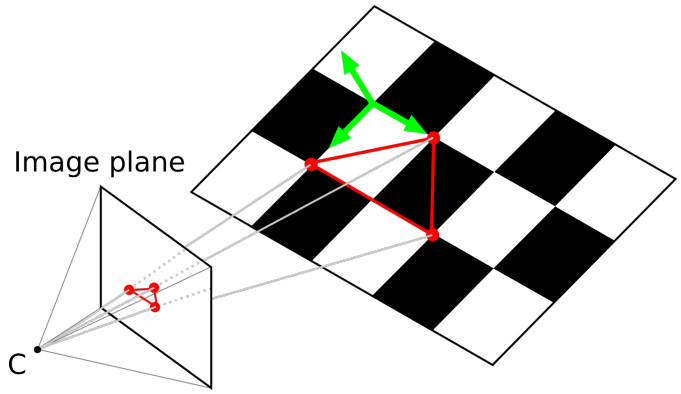


Figure 2.13: Chessboard calibration prerequisites for PnP-Algorithm.

2.3.1 PnP-Algorithm

The Perspective-n-Point-Algorithm allows a single view reconstruction of the camera pose. It is applied, if the 3D world points are already known. This is for instance the case during geometric camera calibration with a chessboard pattern (see fig. 2.13). The coordinate system of the world frame can be chosen as one of the edges of the pattern. The rest of the black and white junctions form a common plane, in which case one coordinate axis (usually the z-axis) can be set to zero. In the next step, it is assumed that the visual angle of any pair of 2D points must be the same as the angle between their corresponding 3D points [10, S.323]. Depending on the number of points ‘n’ selected (n >= 3) for estimation,

different formulations exist to solve this problem (e.g. P3P-, P4P-, ..., PnP-Algorithm). It is important to note, that the PnP-Algorithm can handle non-planar points, but in case of the calibration they are advantageous to initialize a fixed coordinate system without additional views required.

2.3.2 Homography

The homography H was already introduced in section 2.1.1 and describes a 3x3 transformation matrix in projective space. Its estimation is a multiple view operation that uses corresponding features in the images to identify the camera movement. It follows the equation

$$x = Hx', \tag{2.25}$$

where x and x' are both homogenous 2D projections of world point X, but from different viewpoints. The underlying assumption implies, that all observed points lie in the same plane (see also the example of section 2.1.1). While the PnP-Algorithm can handle non-planar points, the homography can not. Instead, it is able to reconstruct camera movement in an unknown environment up to a set of four ambiguous solutions. Afterwards, these can be decomposed with the intrinsic camera calibration K into pose matrix M following e.g. the approach of Faugeras et. al [15]. If more than four corresponding points are provided, a best-fit optimization has to be performed. This can be a challenging task, if the scene is highly non-planar. In this case the estimated pose comes with an error, that might not be negligible.

2.3.3 Fundamental matrix

The fundamental matrix F is a 3x3 matrix similar to the homography. While it is also computed from point correspondences without knowledge of the camera intrinsics or relative pose [9, S.239], it is in contrast to H independent of the scene structure and described by

$$x'^T F x = 0, \tag{2.26}$$

where again x and x' are both homogenous 2-space projections of world point X, but from different viewpoints. It is an algebraic representation of the described epipolar geometry of section 2.2.1 and sets two viewpoints into relation. Even though its independence of the scene, in some special cases, e.g. planar scene or pure rotational motion, the problem might not be well constrained [9, S.250]. Consequently utilizing both, fundamental matrix and homography can yield better solutions than restricting to either one of them.

3 State of the Art

The proposed implementation of this thesis addresses the problem of real-time aerial photogrammetry in 3D using monocular SLAM. The consecutive state of the art description therefore includes an overview of current visual SLAM frameworks (sec. 3.1), 3D reconstruction techniques (sec. 3.2) and recent, related work in the field of real-time aerial mapping (sec. 3.3).

3.1 Visual SLAM

Simultaneous localization and mapping (SLAM) is a general term for techniques that obtain both the structure of an unknown environment and the sensor motion in that environment [16]. It was originally developed for autonomous control of robots [16], but has found a variety of new applications in e.g. augmented reality or self-driving cars since then. Practicable solutions exist for a wide range of sensors today, such as rotary encoders, laser scanner, GNSS receivers and cameras. Especially the latter has undergone active research in the last years due to the simplicity and flexibility of the hardware setup. A camera comes with every modern smartphone or tablet and can be mounted to nearly any sized robot. The analysis of the sensor data in contrast is challenging. Even micro cameras of a few centimeters are able to acquire several megapixels in a decent framerate. To still achieve real-time performance visual SLAM algorithms have to invest significant effort to extract only the essential information for pose estimation and map creation.

The main modules most of modern vSLAM algorithms share are as follows [16]:

- Initialization
- Tracking
- Mapping

During initialization a reference coordinate system is defined and the first 3D map is triangulated using e.g. the principles presented in sec. 2.2. Afterwards in the tracking, the initialized map is traced by recognizing known points in the current image and estimating the position according to sec. 2.3. As the camera moves, unknown regions might be discovered. Mapping extends the original map with 3D points of the new environment and

simultaneously keeps the tracking running. For stable and accurate visual SLAM often additional modules are needed. Two very common are [16]:

- Relocalization
- Global Map Optimization

Relocalization allows to recover the camera pose if it was lost due to rapid motion or other disturbances. The global map optimization on the other hand is responsible for maintenance of the reconstructed map. Due to incremental nature of SLAM approaches errors in the estimation process tend to accumulate over time. By recognizing places that were visited before, so called “loops” can be closed. Loop closing typically involves rearrangement of previously computed camera poses and map points for the sake of global error minimization. One of the most famous techniques to achieve such global consistency was already introduced with bundle adjustment in sec. 2.2.3.

While this general structure of visual SLAM is followed by most of the modern open source frameworks, their respective implementation for every module is diverse. Recent publications can be distinguished into “feature-based"/"indirect" and “direct” approaches. Feature-based approaches like ORB SLAM 2 [17] detect unique points in the input image and abstract the observed scene into a geometric constellation of these points. Direct approaches like DSO [18] in contrast work with the whole image and minimize a photometric error function to estimate model parameters. Both methods are explained subsequently. As a hybrid solution between these two competitors SVO is presented afterwards in sec. [19].

3.1.1 Indirect vSLAM: ORB SLAM 2

ORB SLAM 2 is a feature-based visual SLAM approach. It was developed by Raulmur et al. and published in 2016 as open source framework on Github [17]. Its name goes back to a feature descriptor called ORB (Oriented FAST and Rotated BRIEF) [20], which characterizes unique image points in binary format for efficient computation and evaluation. Just as any other feature-based approach ORB SLAM 2 works with points as abstraction from the actual image. The geometric relationship between these points is of prior interest and analyzed in several ways. Fig. 3.1 shows the schematic structure and most essential datatypes of the implementation.

As first step after feature extraction the map initialization is performed. Its goal is to compute the relative pose between two frames to triangulate an initial set of map points. The authors propose two different strategies to achieve this. The first is based on the fundamental matrix described in sec. 2.3.3 and aims for use on non planar scenes. The second in contrast will be applied, if the scene is assumed to be planar. In that case the

fundamental matrix is not well constrained [9, S.250] and the homography produces better results. To decide which approach is selected, Raulmur et al. identified a robust heuristic with

$$R_H = S_H/(S_H + S_F), \tag{3.1}$$

where S_H and S_F are scores for homography and fundamental matrix depending on the symmetric transfer error computed for the respective assumption. If $R_H > 0.45$, then the homography is selected. Otherwise the motion of the camera can be better described with the fundamental matrix. After model selection, the recovery of the pose is performed using e.g. the approach by faugeras et. al [15]. If the overall error is too high or there are too few inliers for either one of the methods, no model is selected and the algorithm waits for a new image pair.

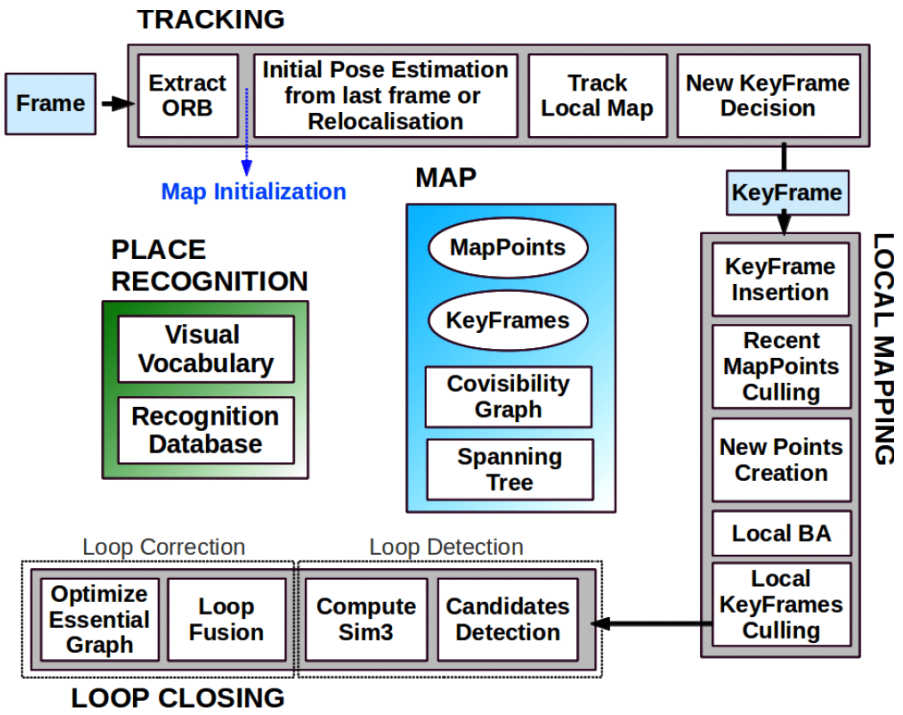


Figure 3.1: Structure of ORB SLAM 2.

After initialization the tracking starts. For every new frame first the ORB features are extracted. In the next step a constant velocity motion model is used to predict where matching features of prior frames should be located. If not enough correspondences were found, a wider search region is applied. In case the pose is still uncertain, tracking is marked as lost and relocalization is started. Input frames are then transferred to a "bag-of-words" representation [21] and queried in a global database. As soon as known regions are recognized, PnP algorithm is utilized to compute the camera pose and recover tracking.

As the previous steps suggest, at least 3D points and 2D image features must be saved permanently. This can be problematically if the camera acquires images at high framerates.

In that case even small amounts of data per frame will quickly stress the processing hardware. ORB SLAM 2 follows a wide spread strategy, where only a subset of all frames is kept to represent most of the informations. These frames are also referred to as "keyframes". It is also this subset, that is passed to the next processing step.

During local mapping bundle adjustment between neighbouring keyframes is performed to gain higher accuracy of the camera pose and observed points. Additionally, new map point candidates are created and existing candidates confirmed or discarded. To further reduce redundant data, existing keyframes are checked for uniqueness. This so called "survival-of-the-fittest" approach also allows to insert a lot of keyframes during rapid movement and filter for the most essential ones afterwards.

In the last step processed keyframes and their correspondences are passed to the loop closing module for global consistency. The so called essential graph describes keyframes as nodes and overlap between them as edges. Using the prior mentioned relocalization module connections with all prior keyframes can be detected and the consequences for the essential graph be estimated. If a loop candidate is validated by a 3D similarity transformation, corrections are performed and the accumulated error is minimized.

3.1.2 Direct vSLAM: DSO

Direct Sparse Odometry was developed by Engel et al. and published open source in 2016 [18]. It is not strictly speaking a visual SLAM, as it lacks loop closing and global map optimization. However, these so called "visual odometry" implementations are seen as part of the same problem statement.

DSO is categorized as direct approach, which means it works on pixel intensity values, instead of relying on feature detection and description. Benefit of this is, that even in seemingly featureless regions small gradients can be detected and tracked. Additionally, the theoretical formulations are not limited to certain points and their close environment. Lines, edges and other structural elements of an image can be integrated in the joint optimization to extract the camera parameters (intrinsics, extrinsics and inverse-depth for created points). Basis for that is a technique called photometry. It utilizes the fact, that the reflection of an object contains information about the object's surface normal. By observing the same point from different views, the measured intensity this world point generates in the cameras as pixel is used to identify this normal and subsequently triangulate the point position. Because a lot of quantities influence this process, e.g. surface structure, camera response function, vignette, so called photometric error functions are defined. Only points who produce a low photometric error according to the function are reasonable for reconstruction and/or tracking.

As for DSO's general workflow, it is structured into frame and point management similar to the classical design of visual SLAM algorithms. Frame management represents the tracking and contains three essential steps. At first, utilizing a constant velocity model new frames are positioned to the last created keyframe only, so that the difference in pixel intensities is minimal. This technique is also referred to as "direct image alignment". In the next step, the requirements for keyframe creation are checked. Similar to ORB SLAM 2 more keyframes are generated than needed. This increases robustness during rapid motion. Correction to keep only the ones with unique information is performed in the third step.

Point management in contrast is responsible for the 3D map and contains the optimization backend. It works with a fixed number of points and keyframes, which are referred to as "active". New keyframes are always active and are accompanied by $N_f - 1$ other keyframes (typically $N_f = 7$). Similar to the frame management three different steps are sequentially performed. At first, for the new input keyframe point candidates are created, that might be "activated" in the future. For that, the current image is separated into 32x32 blocks. For each block the pixel with the largest intensity gradient is selected, if it surpasses a certain region-adaptive threshold. If no pixel of a block is above the threshold, the whole block is dropped. In the next step for every point candidate an epipolar search is performed on newly incoming frames. Matches are identified by minimizing the photometric error. After a set of old points is out of sight, new point candidates are activated to replace them and fill up the fixed number. DSO then performs optimization on the currently activated keyframes and points to achieve local consistency and minimize as well pose as point depth errors. This last step is comparable to the known sparse bundle adjustment formulations, that other visual SLAM frameworks like ORB SLAM 2 or SVO use. However, it does not rely of specific, geometric features.

3.1.3 Hybrid vSLAM: SVO

Fast Semi-direct monocular Visual Odometry (SVO) is an open source visual SLAM framework developed by Forster et. al in 2014 [19]. It is a combination of both, direct and indirect methods, and therefore referred to as hybrid in the literature [16]. Its schematic structure is displayed in fig. 3.2.

SVO follows the classical design of two separate threads, one for tracking and a second for mapping. Within the first two frames the scene is assumed to be planar and the homography H is computed. Afterwards H is decomposed and the rough camera pose is extracted. With it, an initial 3D point cloud is triangulated and motion estimation starts. Now, as soon as a new frame is acquired so called sparse image alignment is performed. This strategy is similar to other direct approaches, e.g. DSO. By minimizing the photometric error around a set of image patches the current frame is aligned relative to the previous one

and the camera pose is extracted. Subsequently, comparable to indirect approaches the 2D locations of the reprojected patches are refined individually and a pose and structure optimization concludes by minimizing the reprojection error.

If the euclidean distance of the current frame is larger than 12% of the average scene depth, a new keyframe is created. These are then passed to the mapping thread. Meanwhile, in the mapping thread a probabilistic depth filter is initialized for every 2D feature patch of incoming keyframes and updated as soon as new observations are available. Only points whose uncertainty converged below a certain threshold are reconstructed and used for further tracking. In the most recent version of SVO besides the plane visual odometry functionalities also global optimization is performed. For that it relies similar to ORB SLAM 2 on the graph optimization framework g2o [22], which is used to simultaneously minimize pose and point errors through bundle adjustment.

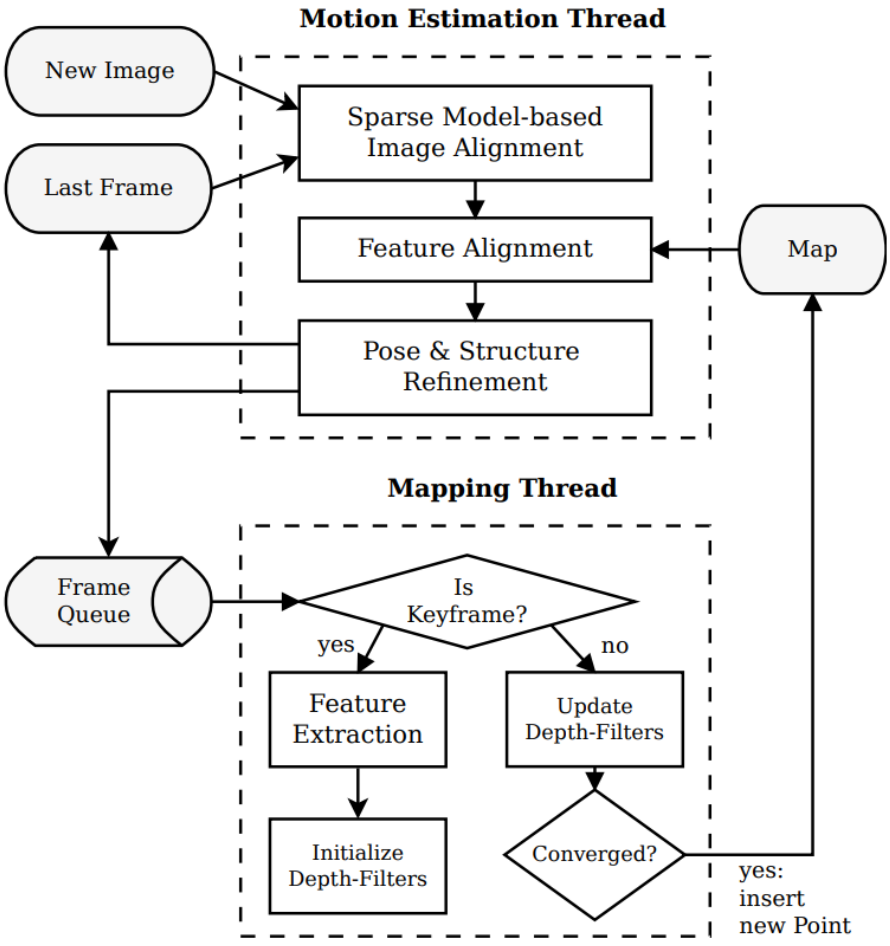


Figure 3.2: Workflow of the tracking and mapping thread of SVO.

3.2 Monocular Dense Reconstruction

Considerable efforts have been made in the past decades to realize monocular dense reconstruction, which is also often known as structure-from-motion in computer vision [9], photogrammetry in geodesy [23] or partially also as monocular SLAM in robotics [24]. The simplistic sensor setup of a single, moving camera motivates different fields for research, as it can be attached to almost any technical system and delivers extensive amounts of data of the environment. The literature for dense stereo reconstruction is comprehensive and for a detailed comparison refer to [25]. However, only few addressed the problem of actual real-time performance.

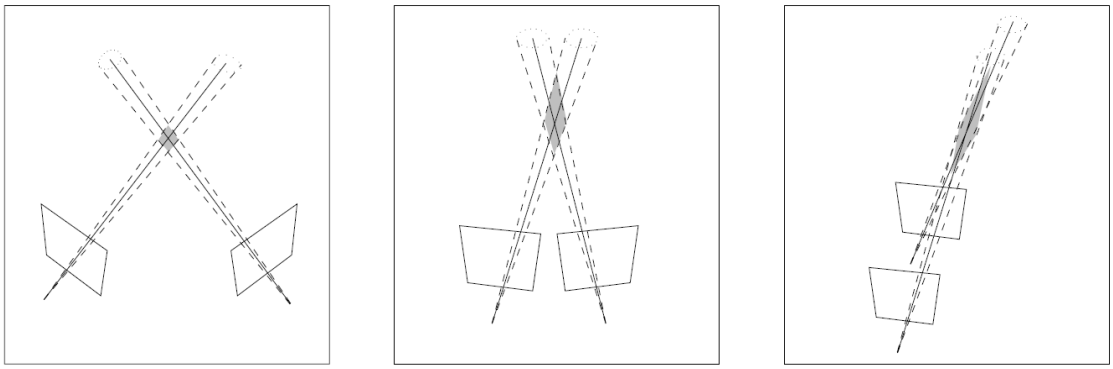


Figure 3.3: Uncertainty of triangulation [9, S.321].

The reason for this becomes clearer, if the fundamental problem of such dense reconstruction is visualized in fig. 3.3. For every pixel of an input image a 3D point should be triangulated. Though, the measurement for corresponding points is error-prone, which results in rather a cone for triangulation than an actual ray. Depending on the arrangement of the cameras very different scenarios can occur. On the left the triangulation angle between the two views is large, the region of uncertainty consequently small and compact. The closer the cameras are placed to each other, the larger this region of uncertainty grows and depth estimation becomes vague. Stereo cameras cope with this by a fixed baseline and calibrated dimensions. However, for monocular reconstruction the baseline is not fixed und must be estimated as well. Additionally, for real-time approaches it is typically small due to the continuous motion. Proposed visual SLAM algorithms overcome this by their very selective nature for point creation and tracking. Only the ones with high probability and consistency over several frames are used for reconstruction. Dense methods in contrast have to deal a lot more with this problem, as they are otherwise referred to as sparse. Noise handling is therefore an essential property of real-time monocular dense reconstruction algorithms. In sec. 3.2.1 an approach using Bayesian probability formulations is presented, followed by a "plane sweep" approach in sec. 3.2.2.

3.2.1 Using Bayesian Formulation: REMODE

In 2016 Pizzoli et al. published the open source framework “Open REMODE”, which handles depth uncertainty with a Bayesian formulation and distance weighting to account for the baseline length [26]. Additional smoothness of the surface is enforced by minimizing a regularized energy functional. The integrated visual odometry module is used to achieve subpixel accuracy by refining an initial, already accurate pose with the dense estimation. In consequence, not only the depth map of the scene is reconstructed, also the pose is improved and an estimate of the uncertainty for observed regions is given. Further parallelized formulations allow to compute REMODE on a GPU and achieve real-time performance. On the downside, the implementation only provides depth for points whose uncertainty converged below a certain threshold as seen in fig. 3.4. In (a) the raw input image is displayed and (b) shows the resulting depth map, whereas bright areas are far away and dark areas close to the camera. In (c) and (d) in contrast regions that converged (blue) or diverged (red) are illustrated. Due to this selective reconstruction REMODE is referred to as "semi-dense". In the limited time of this thesis the parameters of the framework could not be tuned in such a manner, that a consistent reconstruction with the provided aerial image data was acquired. Additionally, REMODE maintains its own, growing dense cloud whose memory consumption adds up to one used for the implementation of this thesis. This is not ideal for especially large global mosaics, which is why other frameworks were finally preferred.

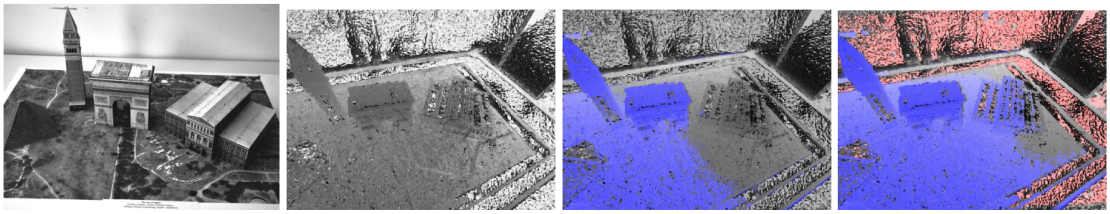


Figure 3.4: Monocular dense reconstruction based on Bayesian formulation.

3.2.2 Using Plane Sweep Algorithm: PSL

Such a preferred framework is the “Plane Sweep Library” (PSL) developed by Häne et al., which was published in 2016 [27]. The plane sweeping algorithm itself was introduced as technique to perform matching between multiple images without the need for rectification [28]. The basic idea is as follows. Every world point is represented as pixel position in the respective image plane of an observing camera. Two or more image planes in turn are related by the perspective transformation (see also chap. 2.2). While classical feature detection and matching assumes corresponding points all lie in the same plane and subsequently compute the homography, plane-sweeping inverts this process. Utilizing several plane hypothesis the one fitting best for each individual pixel is identified. Consequently

instead of assuming the scene consists of one plane only, a variety of differently oriented and positioned planes is suggested. That way the technique is able to reconstruct the 3D scene with prior known camera parameters (intrinsics and extrinsics) and a scene depth estimate. The latter is necessary to initialize plane hypotheses and limit their creation to a volume of interest. Performance of PSL is furthermore achieved by parallelizing it on a GPU. Due to its algorithmic, especially reconstructions in urban environment profit from the plane based approach. Even the surface of featureless regions can be estimated, as long as a plane hypothesis describes it approximately as to be seen in fig. 3.5. On the left the reference image is shown, on the right in contrast the different sweeping directions are displayed on which basis the 3D is generated. The homogenous asphalt is identified as one continuous plane and will subsequently also be treated as such. This characteristic of the plane sweep will turn out to be very beneficial for the aerial mapping scenario in chap. 4, also.

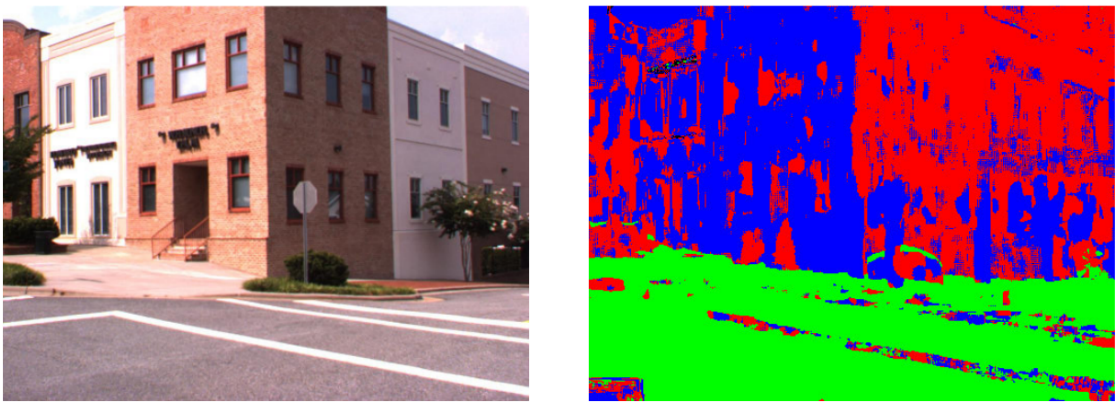


Figure 3.5: Different directions of a plane sweep for an urban scene [28].

3.3 Real-time Aerial Mapping

As suggested in the problem statement of chap. 1 classical approaches to real-time aerial mapping perform mainly 2D panorama stitching by detection and matching of feature points between consecutive images, as for example in [29]. However, this strategy mainly relies on computation of the perspective transformation matrix (homography), which in turn only describes the motion between two image planes. This representation lacks flexibility, as it does not allow to use well known techniques from the field of photogrammetry and structure from motion. The limitation to a planar surface excludes consideration of 3D-data, which in turn would improve the results especially in lower altitudes and with considerable ground elevation. Additionally, almost any use case for real-time aerial mapping would benefit from a digital surface model, as for example for situational awareness in search and rescue scenarios. Therefore in the following sections two approaches are presented, which address this issue. In sec. 3.3.1 a 2D real-time mapping approach using visual SLAM for pose estimation is presented. Afterwards a 3D implementation relying on sensor fusion and dense reconstruction is described.

3.3.1 Using visual SLAM: Map2DFusion

To overcome the fundamental problem of classical panorama stitching approaches for mapping, the 3D camera pose is required. Bu et al. made a step in this direction by publishing their open source framework "Map2DFusion" in 2016 [30]. It replaces the image alignment module of a typical stitching pipeline with a state of the art visual SLAM. In consequence, loop closing, global optimization and robust tracking in visual challenging environments was externally provided by a matured, well investigated framework. Thus, the 3D camera pose had to be utilized to generate 2D maps. The authors suggested a workflow that is presented in fig. 3.6.

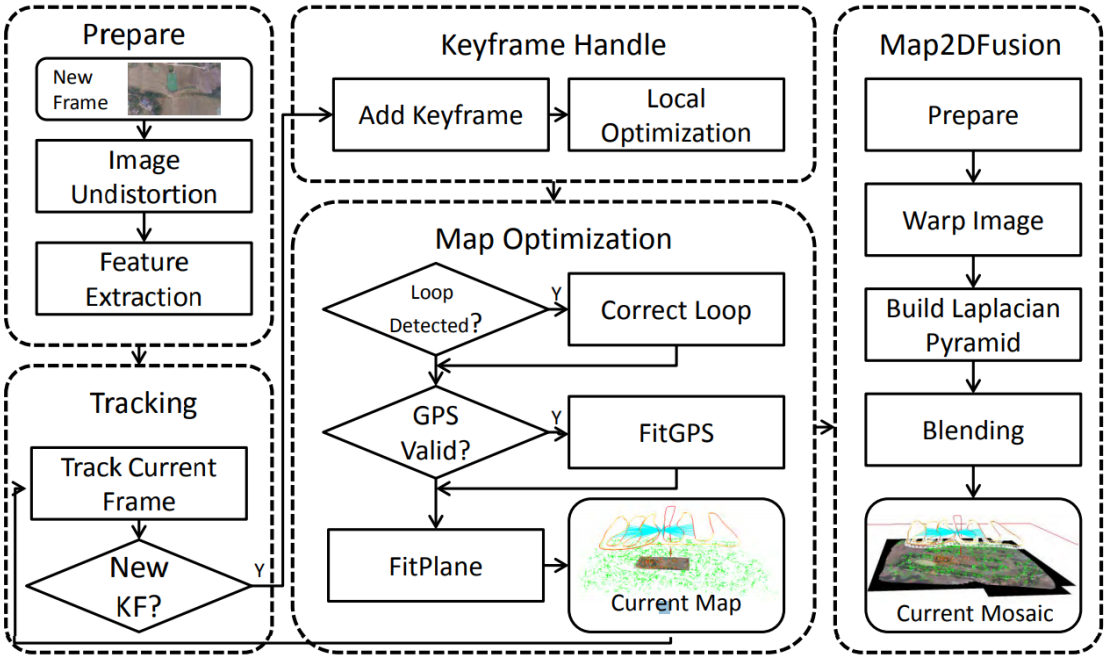


Figure 3.6: Structure of Map2DFusion [30].

At first, input images are undistorted and features extracted. With the tracking capabilities of the visual SLAM the identification of unique keyframes is made. After a local optimization and loop detection, the provided, synchronized GNSS measurements for each image are used to transfer the pose to a geographic reference. Exploiting the 3D triangulated sparse cloud of the scene, a 2D best fitting plane is computed afterwards. With the pinhole camera model the images are then projected into the reference plane and are consequently aligned to a global mosaic. With the additional real-time blending module results of the framework come close to the orthophotos provided by commercial solutions from e.g. Agisoft Photoscan or Pix4D. While the approach by Bu et al. provides a solid solution for 2D mapping, it suffers from several architectural problems. The implementation itself was designed to be used after the flight with all images taken for quick map creation. However, a large benefit of real-time mapping comes from the use while the

UAV is still flying. Additionally, the planar assumption of the surface was avoided for pose estimation, but by projecting images into a reference plane this error is made again.

3.3.2 Using Sensor Fusion and Dense Reconstruction: Aerial Mapper

To achieve what is recognized as aerial photogrammetry today, not only the camera pose must be reconstructed, but also the surface structure. Just recently Hinzmann et al. published their framework for such real-time aerial 3D mapping [31]. For the main duration of this thesis, several modules were still closed source, which is why an extensive practical investigation could not be performed. However, their theoretical work inspired essential algorithms in the implementation of chap. 4. An overview of the system is given in fig. 3.7. While Bu et al. computed the camera pose with a visual SLAM, Hinzmann et al. only partially rely on visual techniques. A Kanade–Lucas–Tomasi feature tracker is fused with an IMU and a GNSS module for continuous state estimation. The resulting 3D camera pose is then utilized to reconstruct a dense point cloud of the observed scene. This point cloud is afterwards transferred to a multi-layered grid map, which represents the surface as 2.5D digital elevation model. Based on this model the final orthomosaic is generated.

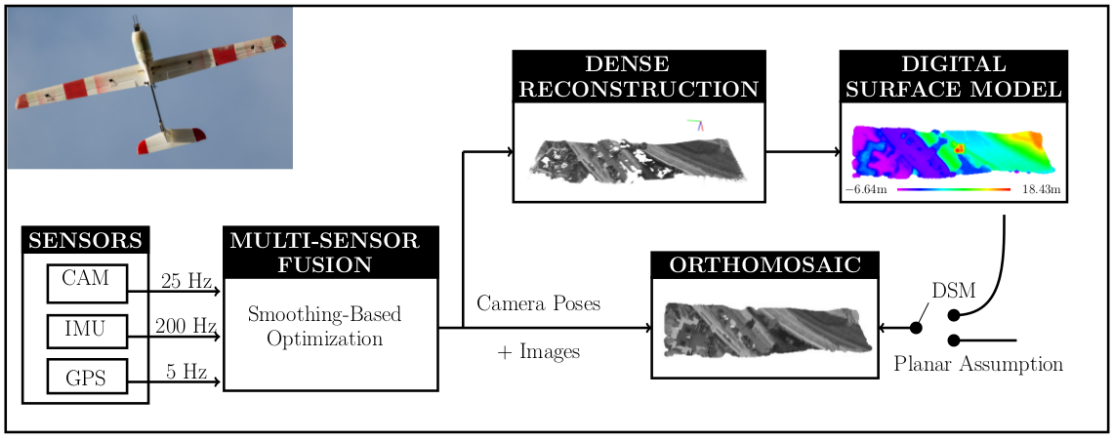


Figure 3.7: System overview of Aerial Mapper 3.7.

The proposed steps are very similar to the classical, offline aerial photogrammetry. As well the camera pose as the surface is recognized as 3-space quantity and the resulting map is therefore rectified in regards of the geometric distortion of the surface. In direct contrast to Map2DFusion the design of Aerial Mapper aims for computation mainly onboard the UAV. While this information might be used for navigation, the benefit for a user is low. The global map is highly valuable on the ground station (GCS) and should therefore be transported as fast as possible. Additionally, the acknowledgement of a ground station as part of the overall mapping process allows to split certain tasks on either the UAV or the GCS. Especially computationally intensive tasks, which might require GPU acceleration are suitable for that. In the long run this strategy allows higher resolution maps and more benefit for practical use cases.

4 Methodology

In the following chapters a complete framework for real-time aerial photogrammetry is outlined. Its main contributions are the following:

- Integration of camera pose estimation based on monocular SLAM for popular frameworks like ORB SLAM 2, DSO and SVO
- Fallback pose estimation based on GNSS and heading information
- Flexible georeferencing of visual SLAM data
- Real-time 2.5D surface reconstruction using GPU accelerated Plane Sweep Library (PSL)
- Efficient mosaicing of rectified images in a multi-layered grid map approach
- A modular structure to allow future contributions on visual SLAM, surface reconstruction or data transportation from the UAV to a ground station

Sec. 4.1 starts with an overview of the proposed system and displays the essential software dependencies. Afterwards a detailed description of the workflow is presented, which is distinguished into the five processing stages pose estimation (sec. 2.3), densification (sec. 4.4), surface generation (sec. 4.5), ortho-rectification (sec. 4.6) and mosaicing (sec. 4.7). Note, that the proposed framework is currently limited to a downward facing camera, that is stabilized by a gimbal in pitch and roll axis

4.1 Software Architecture

The implementation proposed in this thesis aims to provide a functional solution for real-time aerial 3D mapping. However, strict division of tasks leads to a more generic framework with a variety of use cases and applicable on a large scale of processing hardware. As a first introduction it is therefore useful to analyze the overall structure.

Fig. 4.1 shows the schematic composition of the developed system. In the library layer all essential functionalities and mathematical formulations are realized. It contains for example the camera model, as well visual SLAM as 3D reconstruction frameworks and

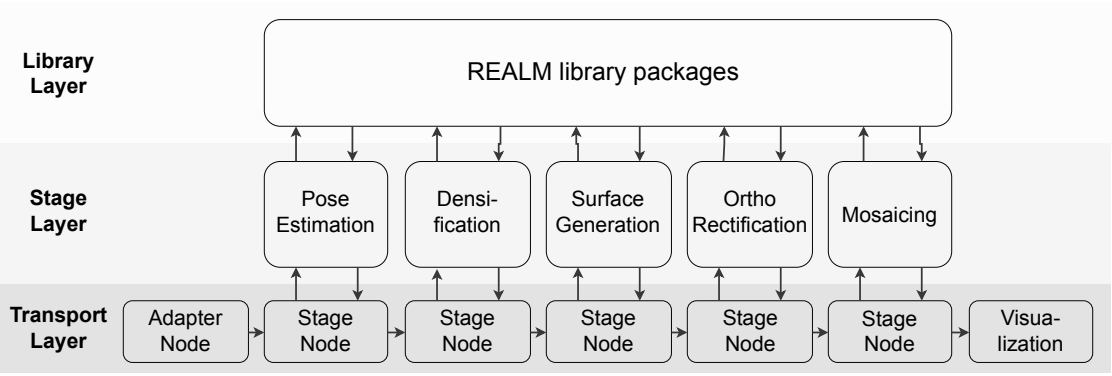


Figure 4.1: Schematic overview of the proposed implementation.

interfaces. It is the backend of the implementation and makes heavy use of OpenCV [11] for visual, GDAL [32] for geodetic and Eigen3 [33] for linear algebra tasks.

In the stage layer in contrast library functions are combined and interact to fulfill a dedicated purpose. Each stage is implemented as a separate worker thread, allowing to process incoming data in parallel. Pose estimation is the first stage using image and GNSS data to generate georeferenced camera poses. Successfully tracked frames are passed to the densification, which utilizes the visual pose and integrated surface reconstruction algorithms to extract depth maps from the observed scene. These depth maps are reprojected into point clouds, that are the basis to produce 2.5D elevation maps in the surface generation stage. With such a digital surface model (DSM) the input images can be corrected during ortho rectification, so that the perspective distortion induced by the terrain is removed. As last step the rectified images are integrated into the global mosaic.

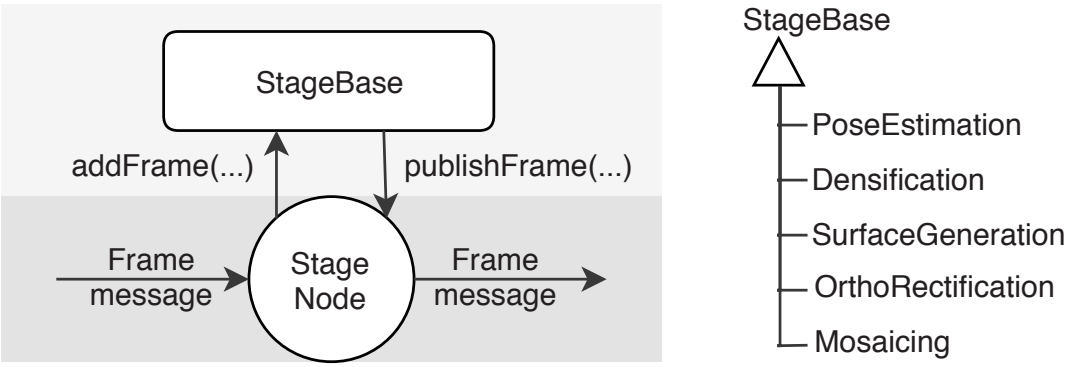


Figure 4.2: Composition of a transportation and processing unit.

The third layer is responsible for transportation and is currently realized as publisher and subscriber network using robot operating system [34] (ROS). Due to the fact that all stages share a universal message, that is modified with every processing step, allows to centralize the transportation into one single implementation. Therefore instead of maintaining e.g.

five different ROS nodes, a single one is sufficient. Fig. 4.2 shows this basic idea. On the left a generic transportation and processing unit is displayed. It can be concatenated up to any number of stages. On the right the currently derived ones from the base class are listed. This architecture allows to replace the transportation with moderate effort and keeps the modularity of the framework high. On the downside of this design it is more difficult to integrate transportation features for one specific stage only. If for example pose estimation should be provided with a reset functionality that is triggered through the stage node, then all other stages should implement this functionality too. In case of a reset this is not a problem, but there might be certain cases for which this leads to laborious solutions. In general, to keep the framework generic and maintainable specializations should be avoided.

At last the adapter node is of interest (see fig. 4.1). Because there is no universal way of how to acquire sensor data and send them to the mapping module, it can be adjusted for every application. For the analysis of chap. 5 images with Exif tags were read from a folder, transferred to a ROS message and then passed to the first stage. However, other solutions might as well be viable and implemented in the future.

4.2 Conventions

The proposed framework makes use of coordinate transformations. To unify the representation fig. 4.3 visualizes the general convention. The world frame is a ENU coordinate system that can be transformed to the camera frame by matrix T_{w2c}. The camera frame in contrast points in the opposite z- and y-direction to align with the image plane. Consequently matrix T_{c2w} is the inverse of T_{w2c} and transforms points from the camera to the world frame. Furthermore sensors of the UAV, such as magnetometer for heading, are measured in a NED coordinate system. This approximately corresponds to the camera coordinate system except that the y-axis is inverted.

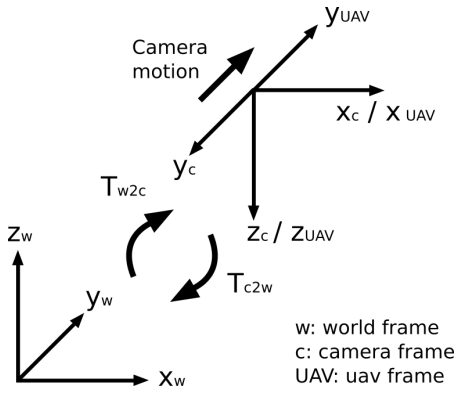


Figure 4.3: Coordinate frame conventions used in the implementation.

4.3 Pose Estimation

In sec. 4.3.1 the general workflow of the pose estimation stage is outlined. Further descriptions and extensions to the visual SLAM interface (sec. 4.3.2) and the georeferencing module (sec. 4.3.3) are presented afterwards.

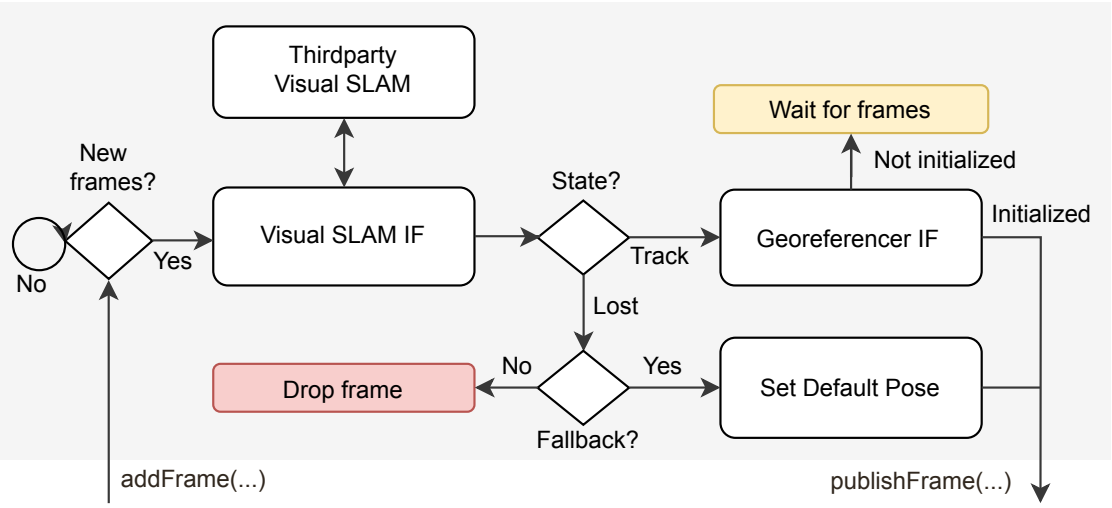


Figure 4.4: Basic structure of the pose estimation stage.

4.3.1 Workflow

The schematic workflow of the pose estimation stage is displayed in fig. 4.4. On start, it creates a visual SLAM interface (IF) for one of the supported frameworks. This can currently either be ORB SLAM 2, DSO or SVO, depending on the initial arguments passed. Incoming frames are redirected by the interface to the actual SLAM implementation, where the camera pose matrix M is estimated. Note that M is defined as transformation from the camera to the world frame. If tracking was successful, frames are processed in the so called "georeferencer". This module tries to identify the transformation from the local, visual to the global, geographic coordinate system utilizing visual and GNSS position. Because currently an initial set of measurements is necessary for a robust computation, incoming frames might be queued until the estimated error is below a certain threshold. After solving for the arbitrary scale and aligning visual and GNSS trajectory, as well all frames on hold as new ones are published. After this step the computed pose is described with

$$M = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_E^c \\ r_{21} & r_{22} & r_{23} & t_N^c \\ r_{31} & r_{32} & r_{33} & t_{Alt}^c \end{pmatrix} \tag{4.1}$$

where r_{ij} are the components of the (3x3) rotation matrix R, (t_E^c, t_N^c) UTM coordinates of the global camera position and t_{Alt} the relative altitude of the camera. In case the visual SLAM framework is not able to track the current frame due to e.g. featureless surfaces like water or plane fields, the state switches to "Lost" and no visual pose is set. To avoid a failure of the mapping process in this scenario, a fallback solution can be computed with

$$M_{Default} = \begin{pmatrix} cos(-\phi) & -sin(-\phi) & 0 & t_E^{UAV} \\ sin(-\phi) & cos(-\phi) & 0 & t_N^{UAV} \\ 0 & 0 & 1 & t_{Alt}^{UAV} \end{pmatrix}, \tag{4.2}$$

where ϕ is the magnetic heading of the UAV, (t_E^{UAV}, t_N^{UAV}) UTM coordinates of the GNSS position and t_{Alt}^{UAV} the relative altitude of the UAV. This substitute pose can only be assumed, if the conventions from sec. 4.2 are valid. Therefore the camera is facing the ground, is stabilized around x- and y-axis and its visual and GNSS position are approximately the same.

4.3.2 Visual SLAM Interface

The visual SLAM interface class ensures, that state of the art frameworks can be easily integrated into the pose estimation process. However, it also implies that all frameworks share a certain workflow. Fig. 4.5 shows a UML diagram of the basic functionalities provided by the interface. At first, $track(*)$ is called whenever a new frame acquired and has to estimate an initial guess of the motion matrix M. As a return the current state of the visual SLAM is computed, which is either "tracking" or "lost". Additionally, "tracking" is further splitted into "frame insertion" and "keyframe insertion" to identify whether the current frame contains essential new informations or not. It is also useful to add at least a sparse cloud of observed surface points to the current input frame during $track(*)$ to have an estimate of the median scene distance later on.

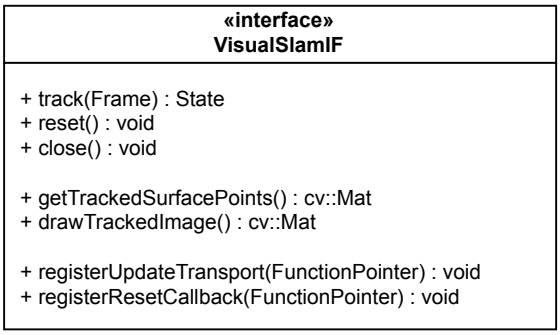


Figure 4.5: UML diagram of the visual SLAM interface class.

In general visual SLAM frameworks can only compute a good guess of the camera pose during tracking and refinement is performed afterwards (e.g. bundle adjustment). It is

therefore reasonable to implement a callback functionality into the interface, which transports pose and/or point updates to the pose estimation stage as soon as they are available. This necessary linkage is done with *registerUpdateTransport*(∗). In a similar way the pose estimation stage can be informed about a reset inside the visual SLAM framework with *registerResetCallback*(∗). These resets invalidate the visual coordinate system, which in turn is critical if no georeference was computed yet. At last *drawTrackedImage*(∗) allows to write extended information (e.g. features, matches, …) into the currently tracked image for user feedback and debugging.

4.3.3 Georeferencer

Usually, Visual SLAM frameworks compute camera poses in a local coordinate system only. In addition, monocular algorithms lack the ability to recover real world scale. It is often chosen randomly, by e.g. assuming the median distance to the first observed point cloud is equal to "1". By measuring this distance and applying it as similarity transformation to both, points and pose, the data can be transferred from the local to a global coordinate system. For aerial photogrammetry it is convenient to display the resulting map in a geographic information system (GIS). This however implies not only to estimate the correct scale, but also to align it with a geographic frame, which is referred to as "georeferencing".

A classical approach to achieve this is to compute the best fitting solution between two sets of 3D points, one consisting of all estimated visual positions and the other of all measured global positions in e.g. UTM coordinates. The transformation between these sets is then identified by following the algorithm of Umeyama [35]. However, in real-time aerial photogrammetry the constraints to apply this approach are more difficult. The overall problem can be badly restricted in the typical mapping scenario. In that, the UAV flies serpentines above the selected area, but should initialize the georeferencer as fast as possible. In consequence, the transformation from visual to geographic coordinate system is computed with trajectories along a straight line instead of homogeneously spread along the axes. While this will cause no trouble to the residual errors of the positions up to this point, the estimated attitude will likely be wrong.

This problem is visualized in fig. 4.6. In the upper depiction the trajectories lie approximately in the same plane, which is the case for flights in fixed altitude. The transformation computed with Umeyama from visual to geographic coordinate system delivers a best fit for the point positions and simultaneously good results for the camera attitude. In the illustration underneath the typical real-time case is presented. All possible solutions (I, II, III) deliver similar residual errors for the point positions but fundamentally different camera attitudes. These attitudes will turn out to be wrong, as more measurements are collected, but the created map will suffer from high distortion by then and must be up-

dated afterwards. To avoid this, the proposed approach first estimates the scale and then performs a bounded Umeyama.

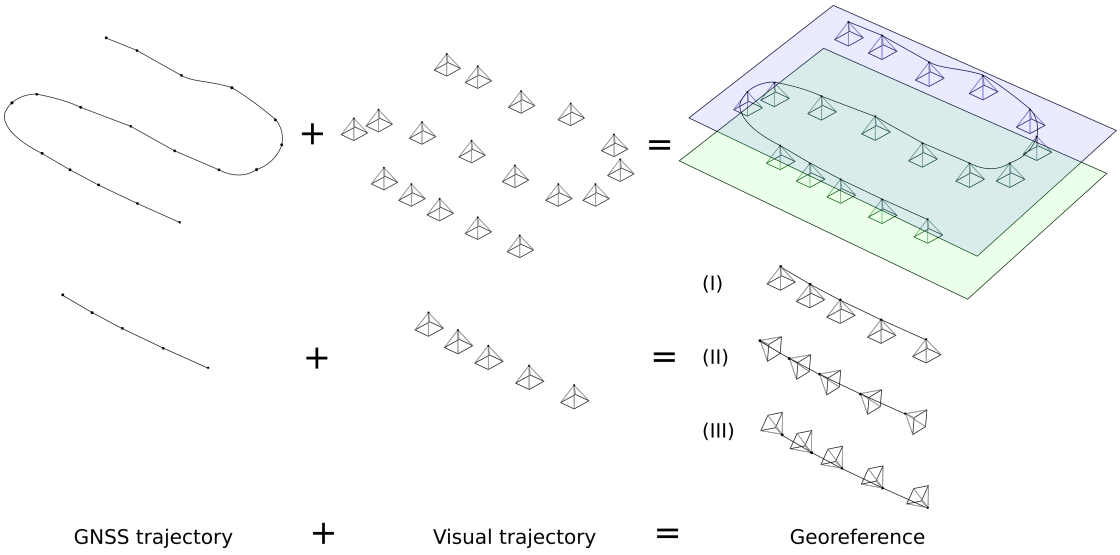


Figure 4.6: Visualization of the georeference ambiguity.

Algorithm 1 and 2 show the implemented strategy for initialization and further updating of such a georeferencer. During initialization a set of at least 3 frames is required. These frames are first checked for validity, so all of them contain visually estimated pose and surface points, a GNSS position measurement and heading information of the camera (or UAV, see convention sec. 4.2). In the next step the remaining frames are tested for uniqueness. For that, the distance of each visual and GNSS measurement to the previous one is computed. Following distance thresholds are defined:

- Geographic distance of current to previous frame: > 10m
- Visual distance of current to previous frame: > 10% median scene depth

These conditions ensure a high signal-to-noise ratio and prevent division through zero for subsequent calculations. For all unique frames the distance to all other unique frames is computed and a real world scale estimate is identified with

$$Scale = d_{gnss} / d_{visual}. \tag{4.3}$$

By averaging all scale estimates an initial guess is extracted. On the first attempt of initialization this average scale is saved and the function returns to the pose estimation stage to wait for more frames. From the second attempt onward the relative error to the previous attempt is computed and the next processing step is not performed until this error is below a certain threshold (e.g. < 0.1m). The smaller the threshold, the more frames must be acquired until the georeferencer is initialized, but the better is the initial guess.

Fig. 4.7 shows an example plot of this process. The blue graph displays the relative error, that ideally converges to zero, if the scale converges to a fixed value. However, the dashed graph represents the final scale that was computed after all frames ($n = 350$) of a mapping procedure were processed. Even though the relative error falls below 0.1m, the absolute scale is still a bit off the final measure. A possible explanation for this is on the one hand the very small subset of all frames on which the initialization is computed (here: 16/350 frames). On the other hand the overall uncertainties of the GNSS receiver can be up to several meters, which influences the computation of d_{gnss}. As soon as the relative error is below the defined threshold, all visual camera poses are transformed to a new coordinate system using previously estimated scale and eq. 2.4.

Algorithm 1 : Georeference Initialization

Input: Vector of frames with valid visual pose and unique GNSS information

```
1:  vector<Frame> valid frames = identifyValidFrames(input_frames)
2:  vector<Frame> unique frames = identifyUniqueFrames(valid_frames)
3:  vector<double> scale estimates
4:  for all frames f_i in unique frames do
5:    for all frames f_i in unique frames do
6:      if f_i == f_j then
7:        continue
8:      else
9:        d_visual = computeVisualDistance(f_i, f_j)
10:       d_gnss = computeGnssDistance(f_i, f_j)
11:       addToScaleEstimates(d_gnss/d_visual)
12:     end if
13:   end for
14: end for
15: double scale average = computeAverage(scale estimates)
16: if abs(scale average - scale old) < threshold scale then
17:   Matrix4x4 georeference = boundedUmeyama(unique frames, scale avr)
18:   is initialized = true
19: end if
20: scale old = scale avr
```

In the next step a bounded Umeyama is performed. To avoid the presented attitude misalignment, additional constraints are incorporated. Fig. 4.8 displays how to include the convention from sec. 4.2. With the magnetic heading of the UAV and its rough alignment with the negative y-axis of the camera virtual 3D points (grey) can be created. These virtual points extend as well every visual as every GNSS position by three points for the best fit computation. Two points are along the x- and y-axis at $(1, 0, 0)$ and $(0, 1, 0)$. The third is in direction of the z-axis with $(0, 0, w)$ to prevent unrestricted rolling of the camera. Depending on how strong this constraint should be, w can chosen to be high (strong con-

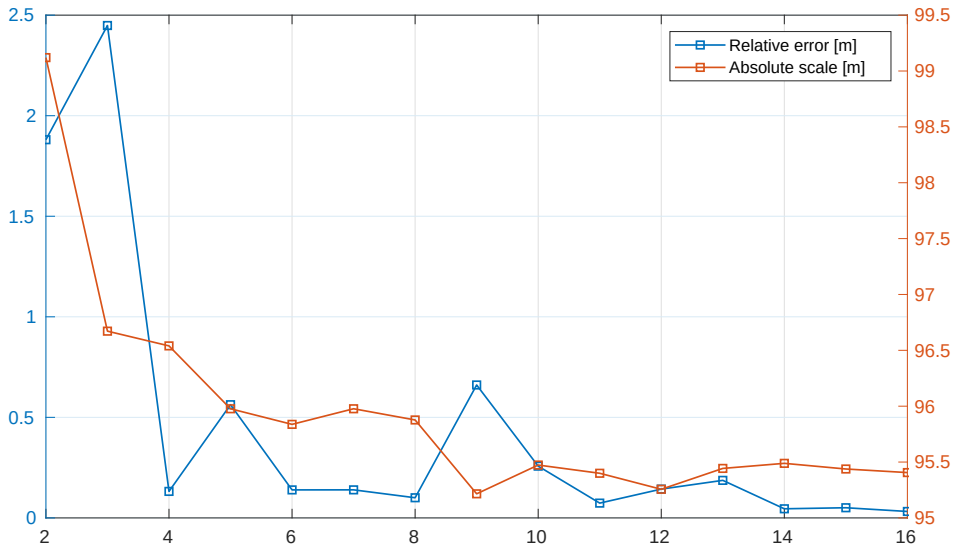


Figure 4.7: Relative error vs. absolute scale on georeference initialization.

straint) or low (weak constraint). In addition, for best fit computation the virtual points must be transformed to their respective world frame with

$$X_{w_i}^{virt} = T_{j_{c2w}} s X_{c_i}^{virt}, \tag{4.4}$$

where $T_{j_{c2w}}$ the homogenous representation of motion matrix M is, j indicating M as either visually estimated or based on GNSS position and heading, $X_{c_i}^{virt}$ a virtual point in the camera coordinate frame (e.g. $(0, 1, 0)$, ...) and $X_{w_i}^{virt}$ the virtual point in the world frame. After applying Umeyama to fit visual and GNSS trajectory, the transformation describing this operation is passed to the pose estimation stage and the georeferencer is set as "initialized".

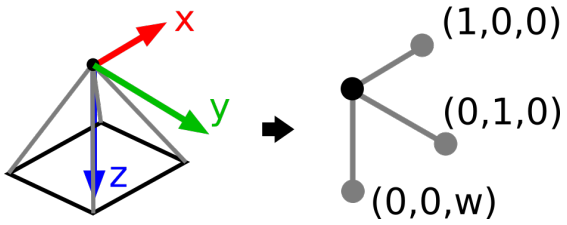


Figure 4.8: Additional constraints per camera for Umeyama trajectory fit.

To update the georeference later on, algorithm 2 shows the basic workflow. Again, the input frame is checked for uniqueness, but scale estimation is not required. Consequently only the bounded umeyama including the new frame is performed and the refined transformation passed back to the pose estimation stage.

Algorithm 2 : Georeference Updating

Input: Current frame, unique frames and georeference of last iteration

1: **Frame** previous frame = lastFrameOf(unique frames)
2: **double** d_{gnss} = computeGnssDistance(frame, previous frame)
3: **if** $d_{gnss} > 0$ **then**
4: addToUniqueFrames(frame)
5: **Matrix4x4** new_georeference = boundedUmeyama(unique frames, georeference)
6: **end if**

4.4 Densification

In the previous stage the camera pose for the current input frame was computed in a geographic coordinate system. This pose can either be visually estimated or based on GNSS and heading information only. The former achieves high accuracies, but is lacking robustness in featureless regions. The latter on the other hand is usually always computable but uncertainties are high and attitude is fixed. In the densification stage only frames with a visually estimated pose are used to reconstruct dense 3D point clouds of the observed surface following the workflow in fig. 4.9.

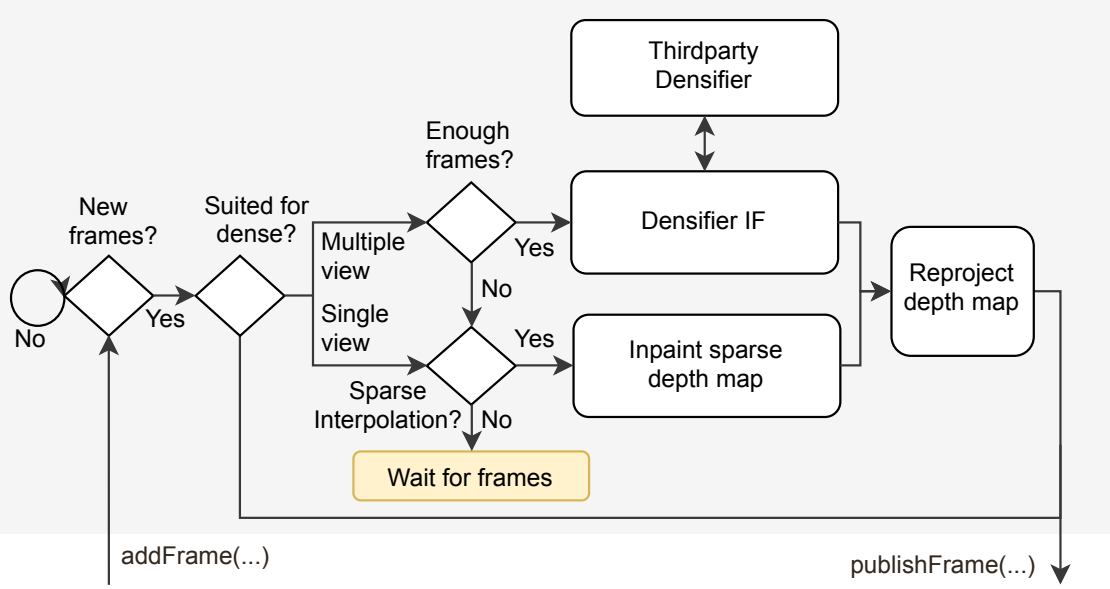


Figure 4.9: Densification workflow.

At first, the input frame is checked for suitability. If the pose is identified as visually estimated one of two methods can be performed to generate a depth map of the scene. The first option is to use multiple view geometry and other frames for a surface reconstruction. To do so, a set of frames for the chosen implementation is passed to the densifier interface. This in turn provides, analogous to the visual SLAM interface in the pose estimation stage,

the possibility to integrate state of the art reconstruction frameworks. Currently only Plane Sweep Library is integrated, which is described in chap. 3.2.2 in more detail.

As alternative to the classical reconstruction a sparse cloud interpolation of the current input frame can be performed. This is viable for a single frame and has no further need for other measurements. However, as it is only interpolating existing data it is not able to generate new information. Also accuracy and resolution depend on the source of the sparse cloud, which is usually the visual SLAM framework beforehand. Either way, both approaches generate a depth map that are afterwards reprojected to a dense 3D point cloud overwriting any existing sparse points.

4.4.1 Densifier Interface

Surface reconstruction frameworks are integrated into the densification stage by implementing the densifier interface class shown in the UML diagram in fig 4.10. In contrast to the visual SLAM interface it is currently limited to very basic operations. A vector of input frames and a reference index is passed in by $densify(*)$. The reference index identifies for which frame the depth map should be generated, which is returned after reconstruction as OpenCV floating point matrix type to the densification stage. Because different frameworks need a varying number of frames, $getNrofInputFrames()$ is an overloaded function to return the required amount. The currently set resize factor to which the images are downsampled is accessible through $getResizeFactor()$.

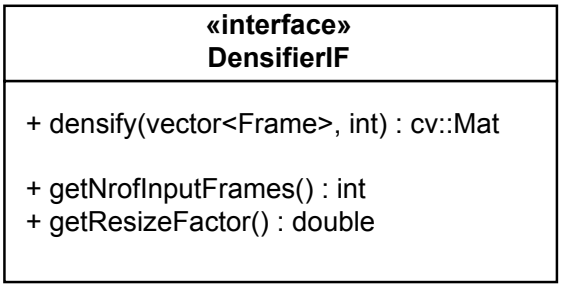


Figure 4.10: UML diagram of the densifier interface class.

4.4.2 Inpainting of sparse depth maps

Utilizing multiple view geometry to reconstruct depth maps from a set of resized input frames allows to extract accurate, high resolution spatial surface information. While this should be the preferred choice, whenever the overall constraints allow it, there are use cases for which a less complex approach can be sufficient. Especially with limited processing power a fast, single view densification based on sparse point cloud interpolation might be useful.

For the proposed implementation this is achieved by the following workflow. First, all 3D points of the visual SLAM sparse cloud are reprojected into the image plane and depth values are written at the pixel location according to eq. 2.15. In consequence, a very sparse depth map is recovered. Afterwards inpainting is performed to fill the gaps in between the sampling points [36]. Fig. 4.11 shows the resulting depth map for an input image (left) reconstructed with multiple other images and the Plane Sweep Library (middle), and interpolated from a sparse depth map using image inpainting (right). As seen the multiple view approach generates more detailed structures, but with higher frequency noise. The inpainting on the other hand is not able to recover data at the trees due to missing information and has blurry contours, but is more homogenous with less noise. Both approaches and their effect on the final 2.5D elevation map are further analyzed in chap. 5.

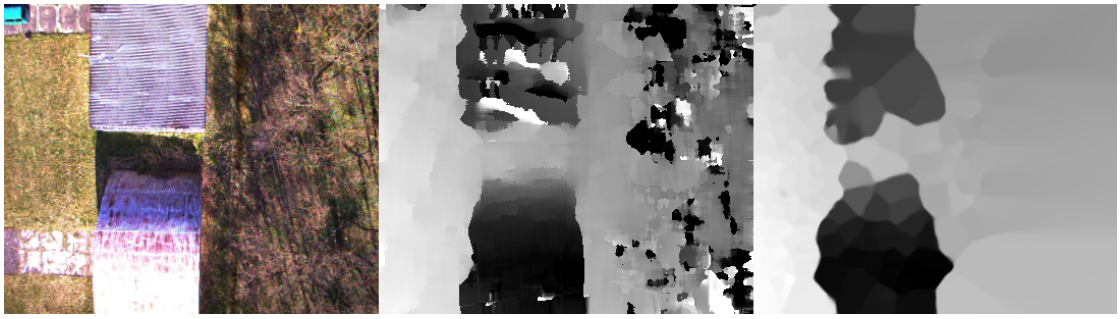


Figure 4.11: Comparison of depth extraction.

4.5 Surface Generation

In the previous stages for all processed frames a georeferenced pose was estimated. This could either be based on the image data using visual SLAM, or based on the GNSS position and convention of sec. 4.2. Afterwards, only for the former depth maps were generated and reprojected into dense point clouds. In contrast, the latter are directly published for the next stage. A subsequent stage therefore has to handle three different types of frames, that might occur:

1. Frames with GNSS based position, fixed attitude and no point cloud (if visual pose estimation failed),
2. Frames with visually estimated, accurate pose and no / sparse point cloud (if only densification failed),
3 Frames with visually estimated, accurate pose and dense point cloud (if all previous stages were successful).

Surface generation stage evaluates all data of the current input frame and proposes a digital surface model, which describes the observed scene with a 2.5D elevation map. Its workflow is presented in fig. 4.12. At first, the surface assumption is identified. Frames with geographic pose information but no or sparse surface information are categorized as "Planar" (1,2), those with a dense point cloud as "Elevated" (3). After creation of the digital model and adding it to the frame data, it is published to the next stage. Sec. 4.5.1 describes the basic class used for the surface model, while sec. 4.5.2 shows how it is filled with data based on the underlying assumption.

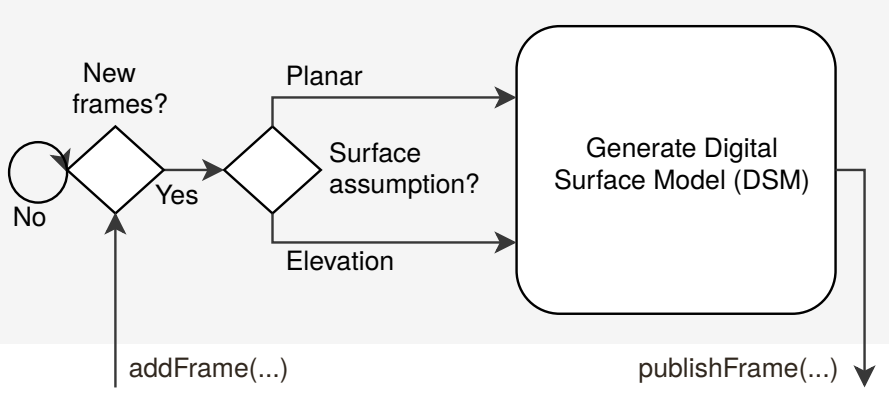


Figure 4.12: Workflow of the surface generation stage.

4.5.1 CvGridMap

The implementation for the digital model is an adaption of Péter Fankhauser's "Grid Map Core" library [37]. The basic idea is to have a structure that is defined by a region of interest with $(x, y, width, height)$ and a resolution for sampling. This region of interest has multiple layers of data (see fig. 4.13). For every sample point within this layers information can be set. For example, in case of an elevated surface one layer contains the height of an observed point from a reference plane, while another one might save the surface normal for this exact point. Different layers can then be easily accessed with the following example code:

Algorithm 3 : CvGridMap Code Example

```
1: CvGridMap map("elevation", "normal")
2: Layer elevation = map["elevation"]
3: Layer normal = map["normal"]
4: Point p = map.atPosition(x,y,"elevation")
5: Normal n = map.atPosition(x,y,"normal")
```

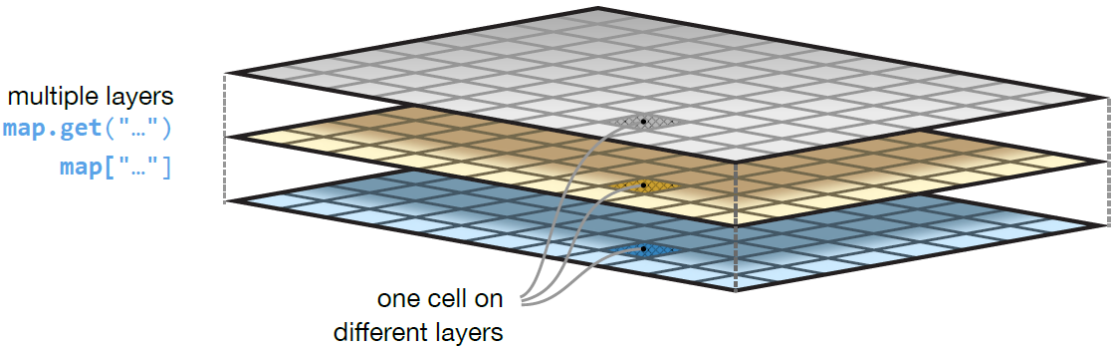


Figure 4.13: Multi layered grid map approach visualized [37].

Modifications to the original library were carried out due to several reasons. The most important is, that Fankhauser designed his library as fixed to a moving robot. Consequently dynamic map movement is efficient, but incremental expansion is not. While this might suite a moving UAV as well, it is not applicable to the global map created ultimately. Additionally, the original library realizes all layers as Eigen matrices [33] and cell values as float. While this simplifies layer spanning mathematical operations, it complicates the proposed computer vision tasks relying mainly on OpenCV [11]. For the framework of this thesis therefore "CvGridMap" was developed, which follows Fankhauser's general architecture, but with an OpenCV matrix type backend, extension to dynamic map expansion and overlap computation.

4.5.2 Digital Surface Model

As described in the previous sections the observed surface is represented by a 2.5D grid map containing elevation data. Yet, it is undefined how this elevation data is generated from the input frame. For the planar surface assumption this is trivial and shown in algorithm 4. The region of interest is set by the projection of the input frame to the reference plane. Due to the fact all cells contain the same information, the resolution, respective ground sampling distance (GSD), has no further influence as long as at least one cell is created. The elevation layer is afterwards set to zero and all elements are marked as valid.

Algorithm 4 : DSM Creation for Planar Surfaces

Input: Region of interest for current frame (ROI)

```
1: CvGridMap map("elevation", "valid")
2: map.setGeometry(ROI, GSD = 1.0)
3: map.add("elevation", zeros)
4: map.add("valid", all)
```

In contrast, the creation of the elevated surface is more complex and visualized schematically in fig 4.14. Additionally, algorithm 5 shows the implementation as pseudocode. It follows mainly the workflow presented by Timo Hinzmann et al. [31]. At first the dense cloud's x- and y-coordinates are structured by a 2-dimensional, binary k-d tree [38]. In the next step this k-d tree is used to compute the nearest neighbour distance for 1% of all points. The result is assumed as GSD for the subsequent grid map creation, while the region of interest (ROI) is again provided by the projection of the frame into the reference plane. After adding the layers "elevation" and "valid", a nearest neighbour search for every cell (x_{cell}, y_{cell}) of the grid is carried out. Note, that x_{cell} and y_{cell} are both UTM coordinates. For all detected neighbours the z-component is extracted from the dense cloud and the resulting height of the cell is finally interpolated.

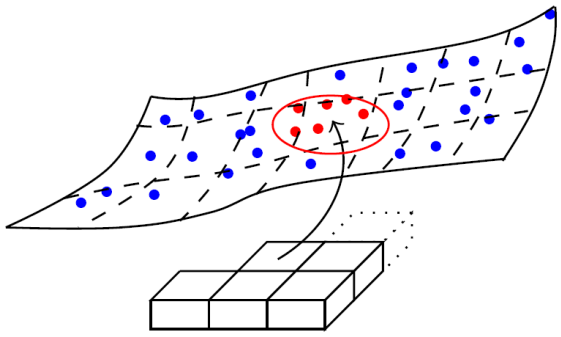


Figure 4.14: Elevation of a cell interpolated from a dense cloud [31].

Algorithm 5 : DSM Creation for Elevated Surfaces

Input: Region of interest for current frame, dense cloud

```
1: KdTree kdtree = initKdTree(dense cloud)
2: double resolution = estimateAverageNearestPointDistance(kd tree, dense cloud)
3: map.setGeometry(ROI, GSD = resolution)
4: map.addLayer("elevation", zeros)
5: map.add("valid", none)
6: for every cell in map do
7:   Point query point = (x_cell, y_cell)
8:   vector<Point> neighbours = kdtree->findNearestNeighbours(query point)
9:   if neighbours found then
10:    map.at(x_cell, y_cell, "elevation") = interpolateHeight(neighbours);
11:    map.at(x_cell, y_cell, "valid") = true;
12:  end if
13: end for
```

4.6 Ortho Rectification

Goal of the ortho rectification stage is to use the previously estimated surface model and camera pose to rectify the visual distortion of the image induced by the viewing angle and surface structure. At best, the resulting orthophoto is in high resolution so points of interest (e.g. humans, cars, ...) can be easily detected. Hinzmann et al. [31] presented two different approaches to achieve such correction:

1. Point Cloud-Based Orthomosaic (Forward Projection)
2. Grid-Based Orthomosaic (Backward Projection)

While method 1) had the lowest computation time, the authors remarked the strong dependency on the reconstructed dense cloud. Especially the smaller area coverage due to holes in the point cloud were noted. But there are more reasons to consider method 2). By saving the elevation of the observed scene as 2.5D grid map, it can be treated as a regular single channel image with floating point data instead of intensity values. Therefore it can also be efficiently resized to whatever resolution is necessary, just like a regular image. In conclusion the spatial and texture resolution can be treated as two separate parameters. The spatial resolution mainly depends on the densification and surface generation stages, the texture is independent and can be set to a value of choice only limited by the raw image resolution. This is especially useful, if the input images of the mapping pipeline are significantly larger than the multiple view reconstruction can process. For this implementation therefore the "Grid-based Orthomosaic" technique was chosen.

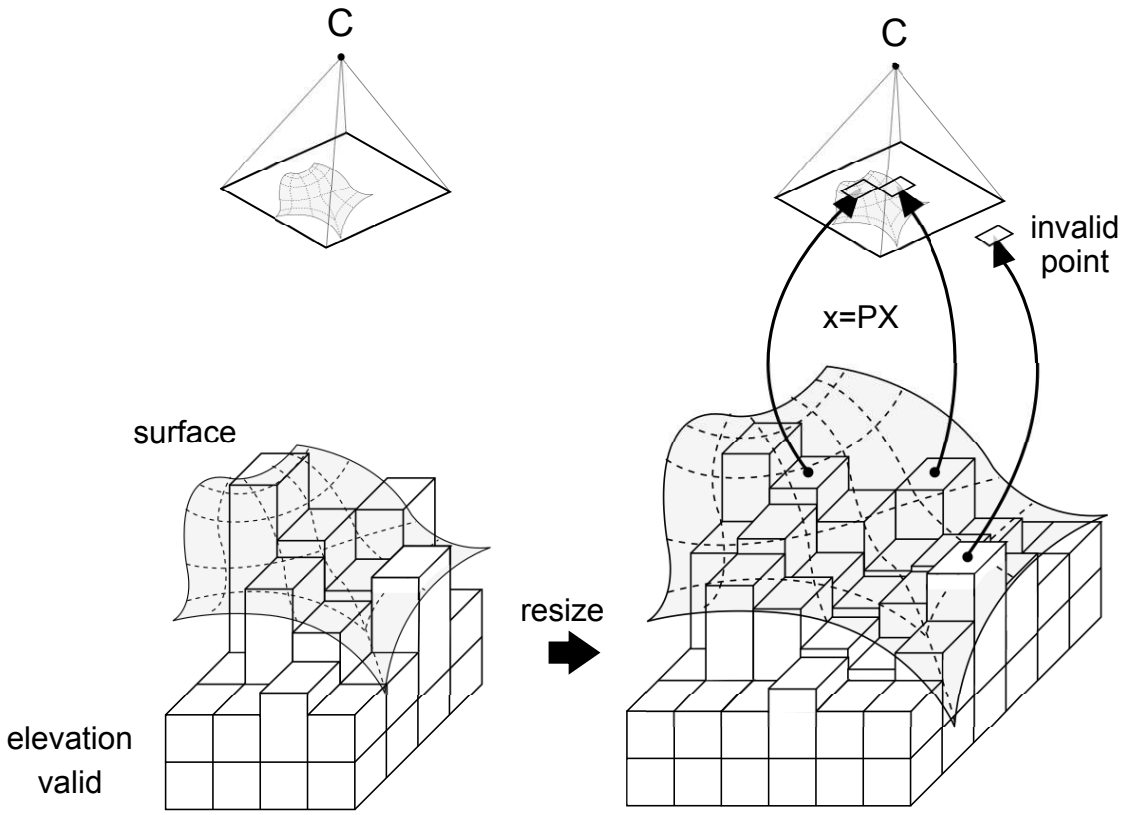


Figure 4.15: Grid-based Orthomosaic visualized.

Fig. 4.15 visualizes the basic workflow of the ortho rectification. At first, the input grid map containing elevation and validity layer will be resized to the desired orthophoto ground sampling distance. In the next step for each cell of the grid a 3D point X will be created with

$$X = (x_{cell}, y_{cell}, h_{cell})^T, \tag{4.5}$$

where (x_{cell}, y_{cell}) the coordinates and h_{cell} the elevation in the grid at a specific position represent. This step is followed by a back projection into the camera image with eq. 2.19. Note, that in case of the planar surface assumption the math does not change, h_{cell} will just be zero for all grid cells. The observed pixel position x for the specific cell is now determined and can be set inside a new color layer containing RGB information. Due to noise in the elevation map or a weak pose estimation back projected points may be identified as outside the image boundaries. These points will be consequently marked as invalid.

Besides the rectification of the image, the angle of observation is computed for every cell during this stage. It is an additional parameter to achieve high orthogonality in the final mosaic. Fig. 4.16 shows all layers of the current input frame after ortho rectification.

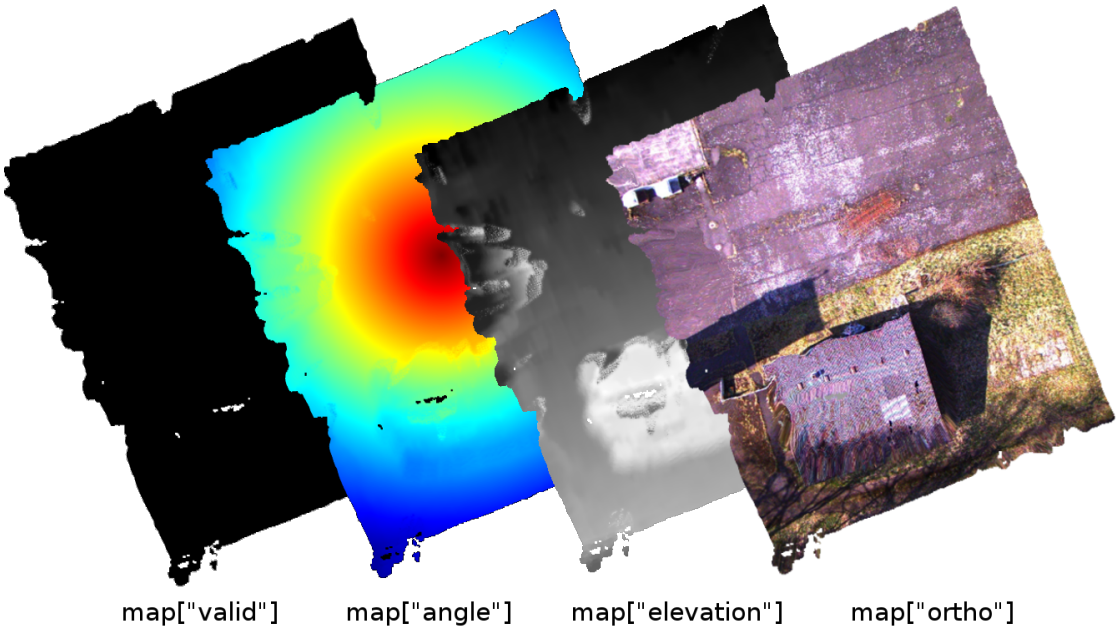


Figure 4.16: All layers after ortho rectification.

4.7 Mosaicing

Mosaicing is the final processing stage and fuses all previously collected data into a single scene representation. While all prior stages are able to keep the computing resources roughly constant over time, mosaicing does not. All sequentially densified, reconstructed and rectified frames are composed into a high resolution mosaic. The main challenge is therefore to keep the required resources minimal.

In fig. 4.17 the workflow is visualized. By receiving the first frame from the ortho rectification stage the global map is initialized. Afterwards new frames are referred to as "map update" and can be separated into regions with no prior information and overlapping regions. The former are directly written into the global map. The latter however are extracted so that two submaps are present, both describing only the overlap for the respective data (either global map overlap region or map update overlap region). In the next step for each grid element of the submaps one blended value is computed, which best describes the surface in each layer. This blended region is finally written back in the global mosaic.

For the blending of the grid cells a variety of strategies can be applied. In this thesis a probabilistic approach was chosen and is described in the following. The underlying

problem statement can be summarized as: "If two different hypothesis for the elevation of a grid cell exist, which one is chosen?". To decide this basic question, three additional layers are added to the global map, the "elevation variance", "elevation hypothesis" and the "number of observations". As soon as a map update arrives, for every cell of the overlapping region a temporary, floating average of the new elevation is computed with

$$\hat{x}_{ij} = \frac{n_{ij}}{(n_{ij}+1)} \hat{x}_{ij}^{global} + \frac{1}{(n_{ij}+1)} x_{ij}^{update}, \tag{4.6}$$

where $\hat{x}_{ij}^{global}$ is the current averaged elevation in the global map, n_{ij} the number of observations of $\hat{x}_{ij}^{global}$ and x_{ij}^{update} the elevation of the map update. In a similar way the floating sample variance s_{ij}^2 is estimated with

$$s_{ij}^2 = \frac{n_{ij}-1}{n_{ij}} s_{ij}^{global\,2} + \frac{(x_{ij}^{update} - \hat{x}_{ij}^{global})^2}{n_{ij}+1}. \tag{4.7}$$

If s_{ij}^2 is below a certain threshold, the new values for variance and average are written to the grid map layers. Additionally, the number of observations for the current cell is incremented. In case s_{ij}^2 exceeds the variance threshold, two different hypothesis exist for the specific cell. At the first appearance of such a new hypothesis, it can not be decided and is therefore written into the "elevation hypothesis" layer, while existing data is kept as is. As soon as a new update arrives in the mosaicing stage for the specific cell, the floating average and sample variance are compared to both, the elevation set and the possible second hypothesis. Now the one with the lower sample variance is selected as the most likely one, while the other is written to the hypothesis layer.

The above strategy aims to reduce noise in the elevation by minimizing its variance. It implies, that with ongoing mapping procedure the estimates of a point get more accurate. This is especially true, if "sparse cloud interpolation" was chosen as densification strategy, because the sparse cloud will typically get optimized by bundle adjustment from time to time. In case of a full 3d surface reconstruction every measurement can be treated as new, independent guess, not better or worse than the one before. In such a scenario the proposed strategy might not be optimal. If two hypothesis exist, always the one with the lower variance is selected, despite one having a lot higher number of observations.Though, the approach saves memory as no additional layer for the number of hypothesis observations is needed. It would also be difficult to decide, when to drop a hypothesis for a new one without, again, saving the number of observations.

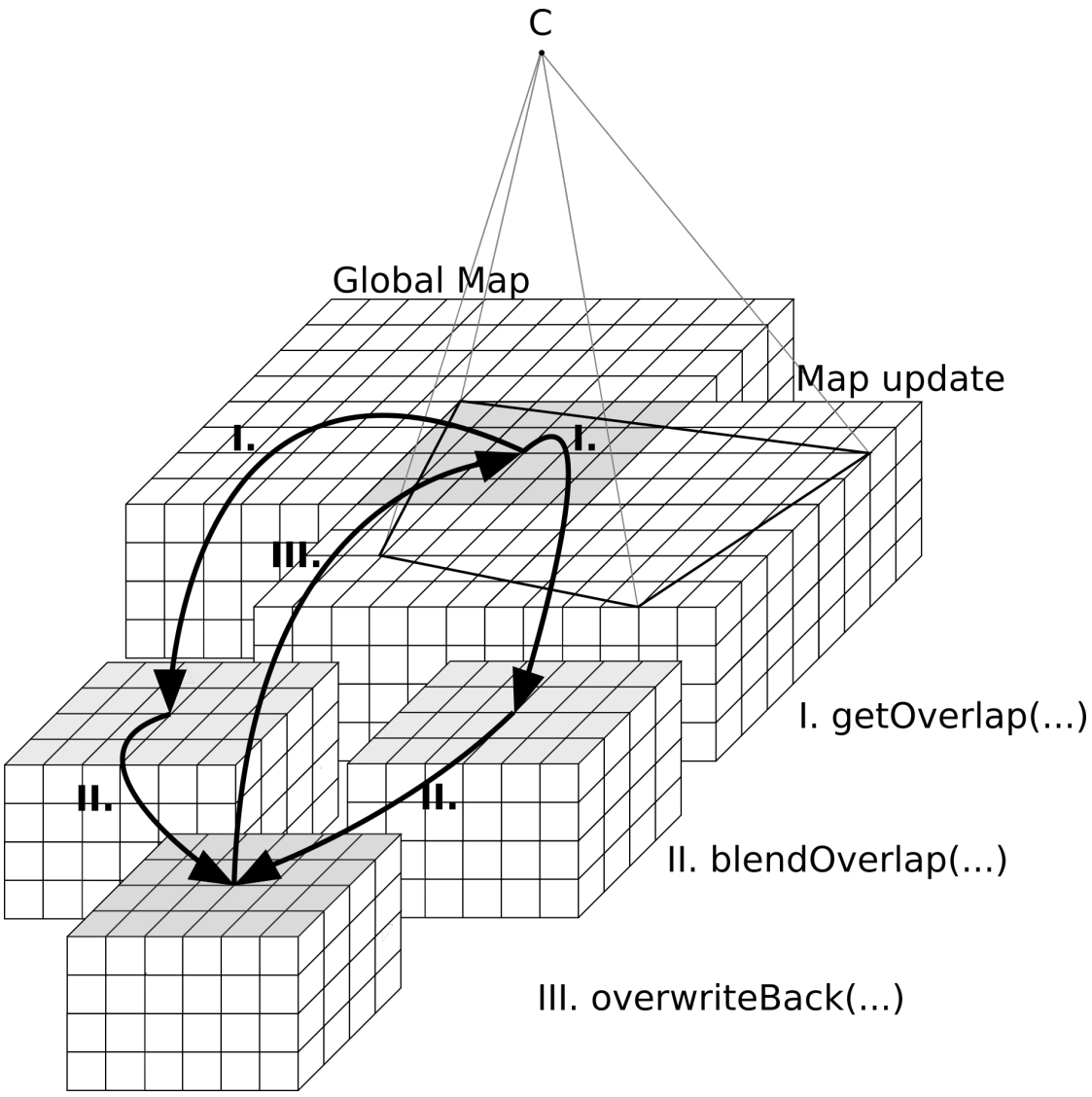


Figure 4.17: Integration of an input frame into the global mosaic.

5 Evaluation

In the following chapter the proposed implementation of this thesis is evaluated. At first, in sec. 5.1 the general testing constraints are presented. This includes the used dataset and hardware, as well as a quick overview of the applied ground truth. In sec. 2.3 pose estimation is analyzed. Especially accuracy of the visual SLAM frameworks in context of aerial mapping is of importance. Additionally, the influence of the proposed georeferencing is outlined. Afterwards the surface for both discussed approaches, sparse interpolation and 3D reconstruction, is compared to the results of a classical offline photogrammetry. Sec. 5.4 concludes with an analysis on the acquired orthophoto. At last, the performance is checked to validate its real-time capability.

5.1 Testing Constraints

For the sake of comparability all tests in this chapter were performed on the same dataset, if not explicitly mentioned otherwise. The hardware used to acquire this dataset is described in sec. 5.1.1, details about the containing information in turn are outlined in sec. 5.1.2. A note on the generated ground truth is shown in sec. 5.1.3.

Figure 5.1: Hardware setup for the used dataset.

5.1.1 Hardware

The dataset was acquired by a custom 560mm wheelbase quadrotor with 2 kg take-off weight and an estimated total flight time of 15 minutes (fig. 5.1). The UAV navigation stack consists of a Pixhawk 2.1 autopilot running APM with a ublox NEO-7 GNSS module and HMC5883L digital compass. It is equipped with an Odroid XU4 companion computer that is wired to the Pixhawk via UART on telem2 port. The Odroid runs Mavros and a custom camera node to grab images, create Exiv2 tags (e.g. GNSS and heading information) and write all data to the harddrive. Both, image stream and global position, are updated with 10 Hz but are not synchronized. The vision setup consists of a UI-5280CP Rev. 2 manufactured by IDS Imaging Development Systems GmbH with global shutter, 5 MPix resolution and an image size of 2456x2054 pixels. However, for the sake of bandwidth reduction and increased performance of the companion computer the images were subsampled to 1228x1027 pixels. As processing hardware for evaluation all tests were performed on a XMG P406 laptop with Intel i7-6700HQ cpu, NVIDIA GeForce GTX 970M graphics card and 16 GB RAM.

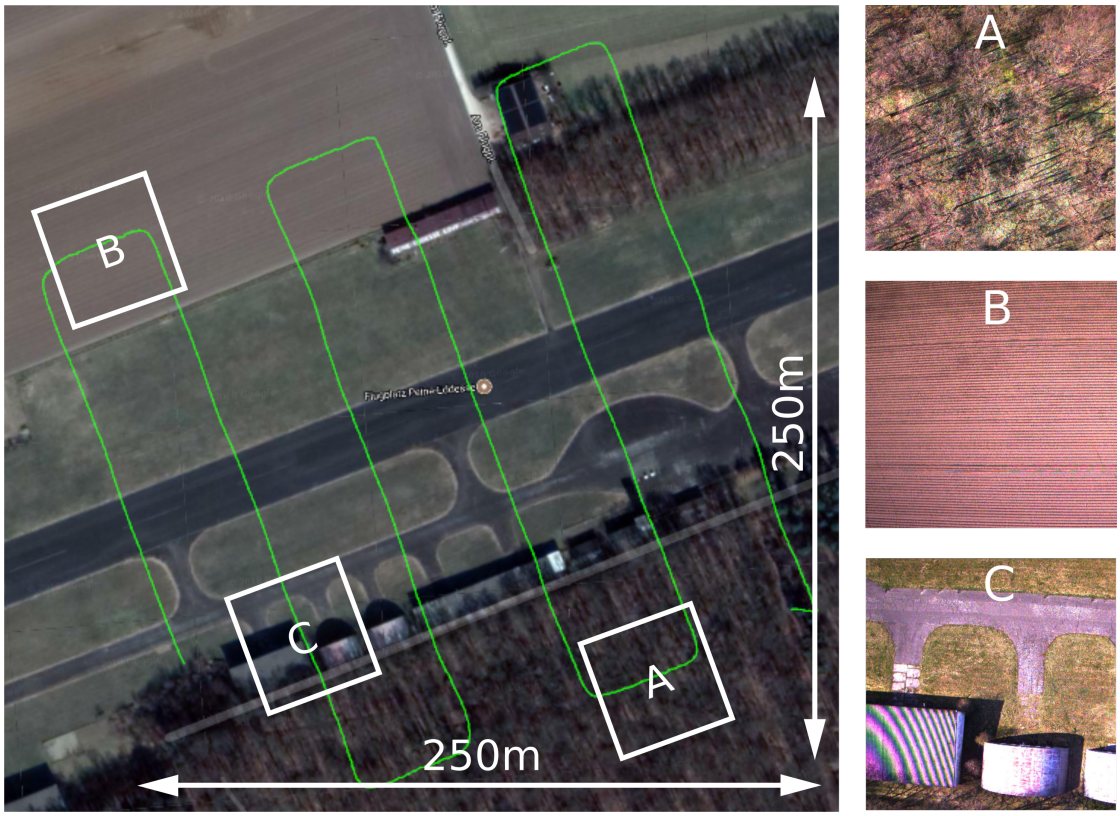


Figure 5.2: Aerial view of the mapped area [39].

5.1.2 Dataset

With the hardware described in sec. 5.1.1. a set of 3276 images was acquired. Appendix X presents an overview of the Exiv2 tags written as meta information into the data. The UAV trajectory described serpentines with 99% front and 50% side overlap above the observed scene (fig. 5.2, green), which has the size of roughly 250x250m and shows an abandoned airport in Edemissen, Germany. The dataset has some challenges for visual algorithms, which are also displayed. In A the forest in the south can be seen, that has very repetitive texture which hardly allows to extract unique points for feature-based visual SLAM. Same goes for B, but induced by the homogenous surface. Illustration C in turn reveals a regional aliasing effect on the corrugated iron roofs caused by the low resolution of the camera.

5.1.3 Ground Truth

Based on the dataset presented in the previous section ground truth was generated using classical photogrammetry. For the camera trajectory all 3276 images were processed with Agisoft Photoscan [5] on the high accuracy profile and afterwards exported to XML format. For the dense reconstruction the front overlap of the images was reduced to 80% to save computation time.

5.2 Performance of Pose Estimation

As feasible visual SLAM framework three candidates have been selected, ORB SLAM 2, DSO and SVO, which were already schematically outlined in chap. 3. According to the literature [40][18] DSO is the latest and most promising approach with high accuracy and an indirect, full image analysis. However, it comes with the disadvantage of being a visual odometry module only, which means relocalization and global map optimization are not part of the implementation. ORB SLAM 2 in contrast contains all necessary modules and is the most complete and matured framework of all three candidates, but works only feature-based, which tends to lack robustness in homogenous, featureless regions. SVO is a hybrid approach, similar to DSO, but also with relocalization and global map optimization. However at least the open source version is slightly outdated.

State of the art evaluation of such visual SLAM implementations is always based on public available datasets, provided by several universities [41][42] with high accurate ground truth to achieve comparable standards. Yet, all these datasets cover very specific use cases, of which most are related to indoor scenarios (AR, VR, GNSS-denied navigation) or autonomous driving. None of these fit the general settings of a UAV flying in low altitudes

above mixed rural and urban landscape. Therefore sec. 5.2.1 starts with a summary of the metrics applied for pose trajectory comparison. Afterwards in sec. 5.2.2 the proposed visual SLAM frameworks are analyzed with the presented dataset of sec. 5.1.2. For all evaluation purposes in this section the open source python package "Evo" is used [43].

5.2.1 Error Metrics

To compare the results of visual SLAM frameworks the extracted maps can be measured against a ground truth. However, different map representations exist (e.g. 3D point cloud, occupancy grid, ...), which is why Kümmerle et al. [44] developed a metric to evaluate the pose trajectories only. If the trajectory is represented as graph, then the nodes can also be treated as point masses and the edges as springs. The deformation energy needed to transform the poses to their ground truth is then described by

$$\epsilon(\delta) = \frac{1}{N} \sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2, \tag{5.1}$$

where N is the number of relative relations, $trans(*)$ and $rot(*)$ used to separate and weight translational and rotational components and $\delta_{i,j} \ominus \delta^*_{i,j}$ formulates the transformation from measured ($\delta_{i,j}$) to reference pose ($\delta^*_{i,j}$). This metric scores the relative pose error (RPE), which is consequently robust to global misalignment, but sensitive to drift. In contrast, the second metric of interest in the following section is called the absolute trajectory error (ATE), often also referred to as absolute pose error (APE), which measures the absolute pose to pose difference and detects global misalignments.

5.2.2 Pose Accuracy of visual SLAM frameworks

All three frameworks introduced in chap. 3. were tested with the dataset of sec. 5.1.2 and are subsequently compared to the ground truth generated by Agisoft Photoscan. To minimize the error induced by global misalignment, alg.2 was used to update the georeference with every new keyframe in all cases. Poses are saved as they are published, therefore georeference updates are applied to all subsequent estimations, but not retroactive.

In fig. 5.3 the results for ORB SLAM 2 are shown. On the left side the translational error for every axis is displayed. Especially the xy-alignment has almost no visible displacement. The z-axis on the other hand is equatable to the estimated depth of the scene, which is reconstructed using multiple view geometry. This axis should therefore be the most uncertain one, and indeed a slight deviation to the ground truth can be measured. Between timestamp 0.5 and 1.0 (*1e11) the first turning point occurs, which is also the visual obstacle B in fig. 5.2. Therefore several factors come into play at this position. First is the low number of features in this region, which reduces the overall accuracy. Second is the

georeferencing which gets its first update, that is not along a straight line and whose estimation is consequently stabilized. While this is positive in the sense of a better global alignment of the trajectory, it reveals the attitude misalignment up to this point. This can also be observed in the rotational component of the motion (fig. 5.3 right). Pitch has stacked up to the highest error of the whole mapping procedure, but is corrected at the turning point and converges back in direction of the ground truth. The relative pose error (RPE) in fig. 5.4 (left) supports this thesis, as it has its highest peak before the turn, and quickly reduces afterwards. Yaw in contrast, is well estimated due to the fact, that it is perpendicular to the likewise accurate xy-plane. At last, roll shows alternating behaviour for both, ground truth and SLAM, which might be caused by numerical issues of the Evo package, e.g. range mapping of the angle to -180 to 180 degrees. It is therefore difficult to analyze, however evaluation of the final mapping results in the following sections will show, that it must at least roughly fit the true rolling of the camera. The absolute pose error (APE) in fig. 5.4 (right) supports the overall good performance of the ORB SLAM 2, as its standard deviation is 0.53m with only a small average offset of 1m to the optimized offline photogrammetry.

DSO and SVO were analyzed in the same way and the results are presented in fig. 5.5 and 5.7. However, neither of them was able to perform over the whole sequence completely. While SVO failed at the first turning point due to weak tracking and is therefore excluded from a detailed review, DSO worked for three legs well aligned to the ground truth. However, at the third turning point it suffered from a scale drift. Depth estimation (z-axis) diverged and carried over to all other axes. Correction by a global optimization module might improve this, but the quick escalation from timestamp 1.75 onward points more in the direction of weak tracking in this area. This is supported by the fact, that the relative pose error (RPE) is consistent over the whole procedure and does not increase steadily (fig. 5.6). It should be noted, that DSO was used without photometric calibration, vignette or exposure times for every image. Literature points out, that its performance significantly increases, if these are provided as well [40]. However, due to the limited time of this thesis and the focus on selection of a proper “out of the box” solution, no deeper tuning was performed.

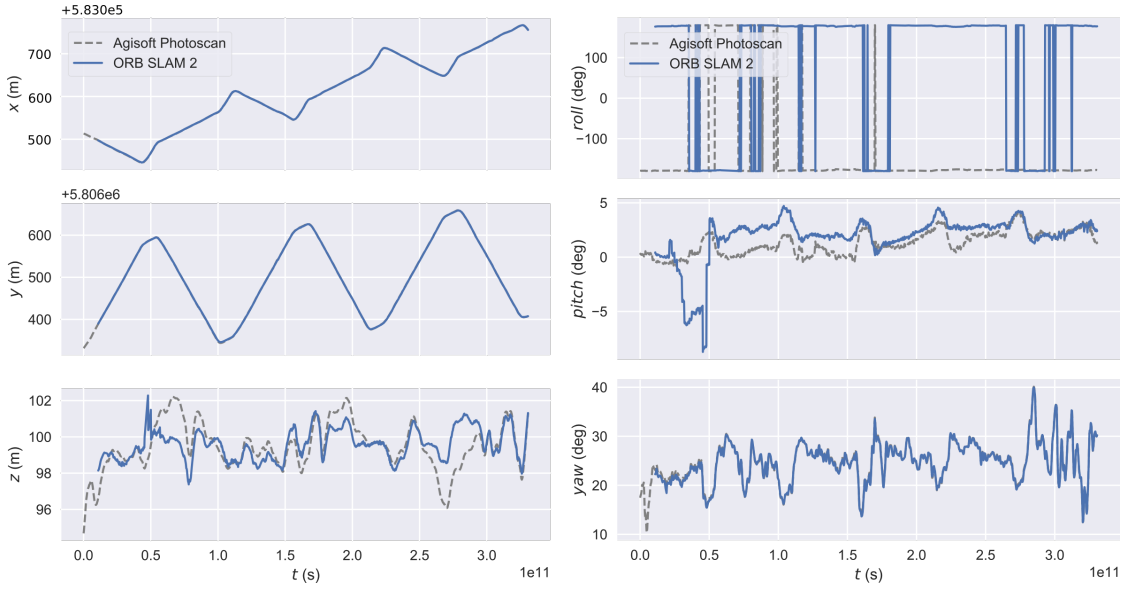


Figure 5.3: Trajectory plotted for ORB SLAM 2.

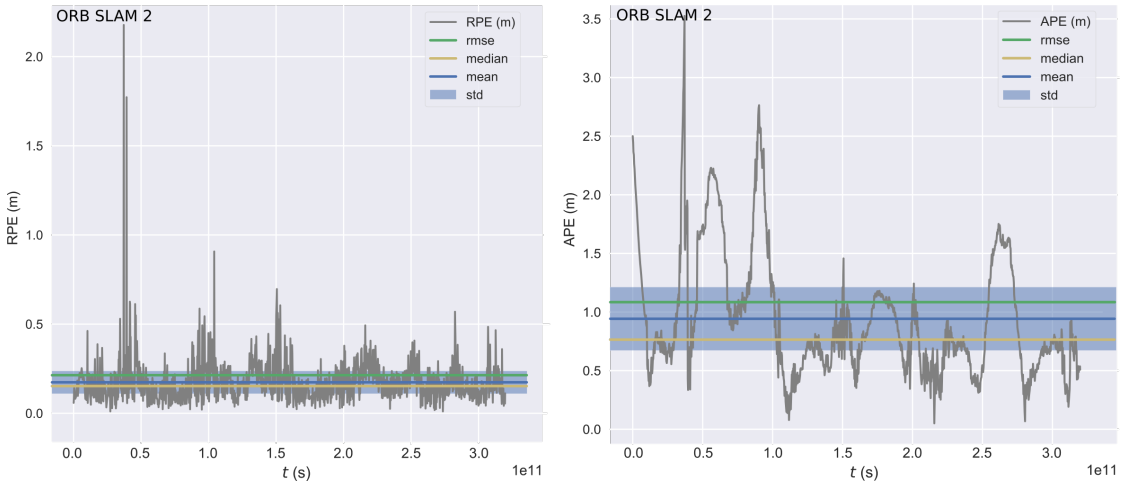


Figure 5.4: Pose error for ORB SLAM 2.

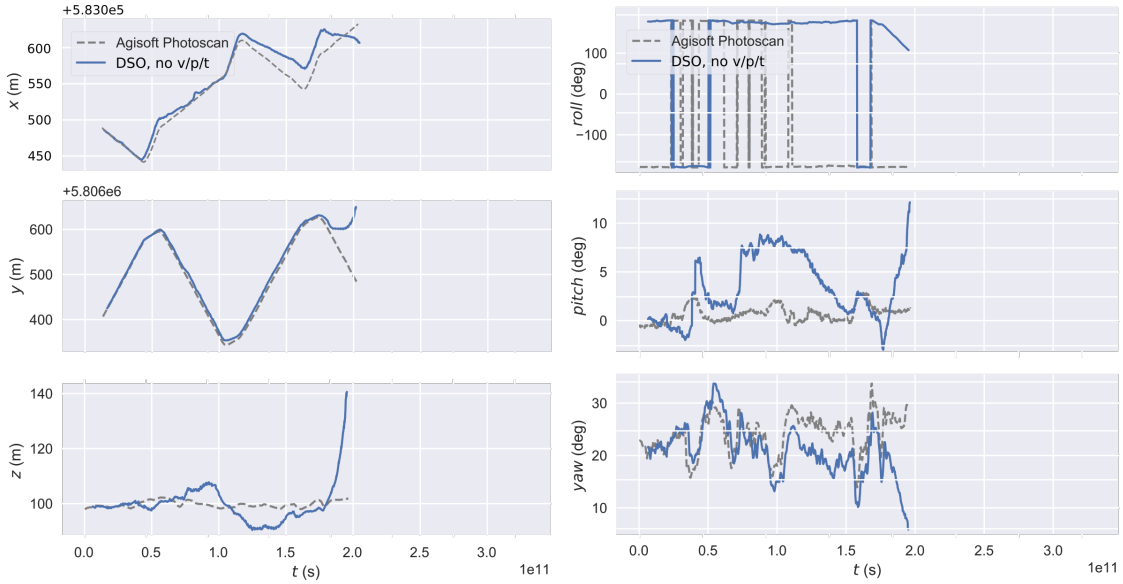


Figure 5.5: Trajectory plotted for DSO.

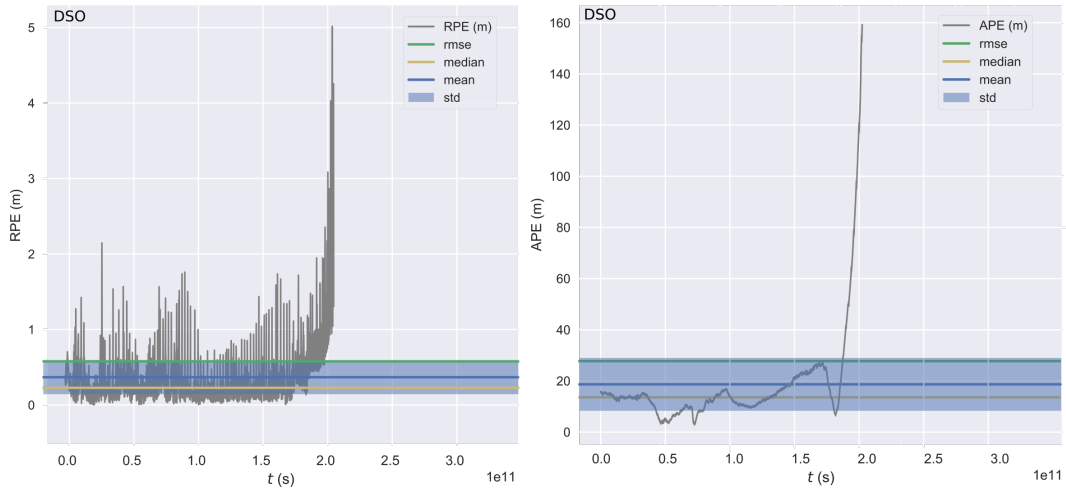


Figure 5.6: Pose error for DSO.

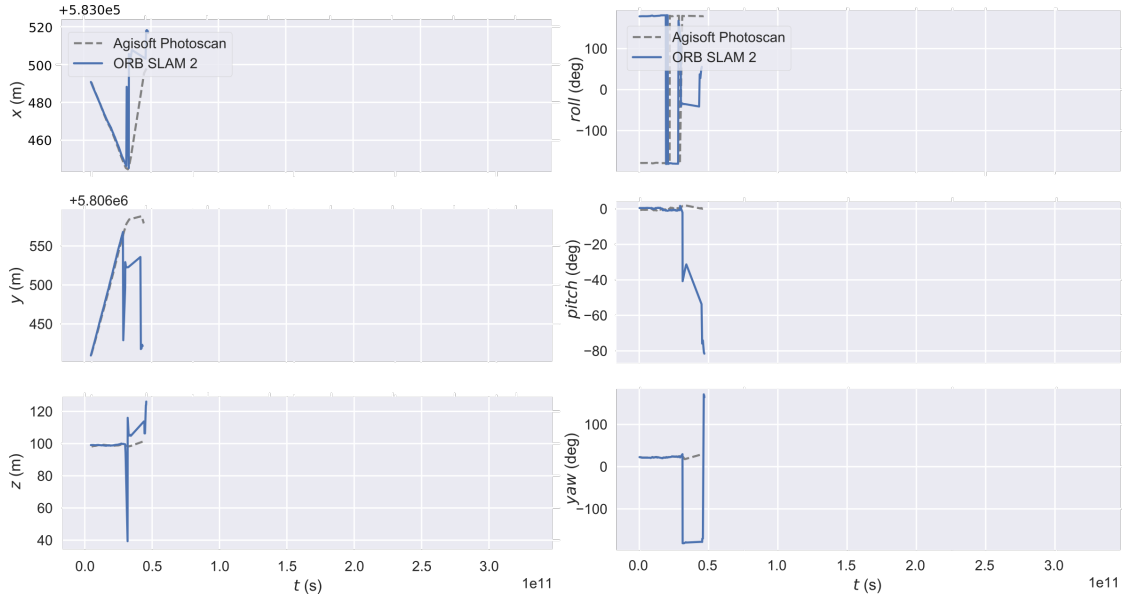


Figure 5.7: Trajectory plotted for SVO.

5.2.3 Influence of the Georeference

For the rest of this chapter ORB SLAM 2 is selected as the visual pose estimator of choice. In sec. 5.2.2 the update functionality of the georeferencing was activated, to reduce the effect of a bad initialization and to achieve equal testing constraints. However, it stays open if the updating of the georeference is necessary or if the initialization is good enough. For that purpose a full pose estimation was performed on the proposed dataset, but without further updating. The resulting relative pose error is displayed in fig. 5.8 (left) and is very similar to the results of sec. 5.2. It leaves out the peak at the first turn, but the rest of the poses are in equal range of 0.0 to 1.0m. The absolute error (right) in contrast is very high, which results from a global misalignment. Such a pose trajectory produces distorted final maps. Updating of the georeference is therefore highly recommended. Alternatively a longer initialization time for at least two flight legs will also stabilize the process.

Fig. 5.3 showed, that especially pitch suffered from high deviations before the first turn, which resulted in a high relative pose error as seen in fig. 5.4. Sec. 4.3.3 described, how the georeference is stabilized with additional constraints to reduce exactly this effect. It is therefore to be evaluated, if the taken measures improve the result, or if they are ineffective. Fig. 5.8 again shows the relative and absolute pose error, this time without additional constraints during initialization. Georeference is computed during the first leg and not updated afterwards. Attitude is badly estimated resulting in an impractical initialization. The absolute error is roughly 5 times higher compared to the constrained approach (fig. 5.9).

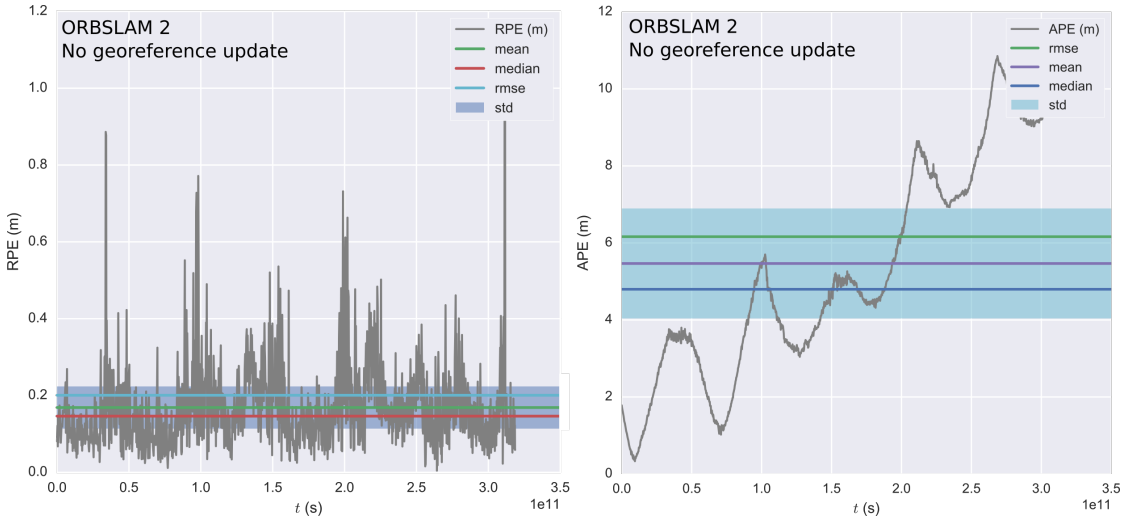


Figure 5.8: Pose error of ORB SLAM 2 without georeference updating.

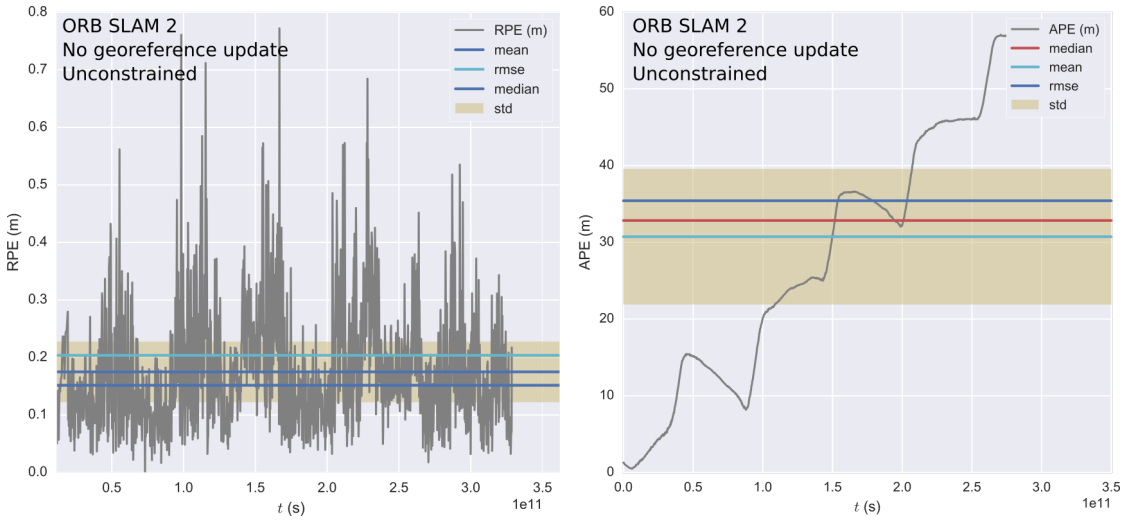


Figure 5.9: Pose error for unconstrained georeferencing.

In conclusion, alignment to the georeference and relative accuracy of the pose estimation are influencing each other negatively. As long as the updates of the georeference are not applied to previous measurements, a trade-off solution is the only practical way. This however would require more computational power, as the global map must be permanently modified, which is especially costly for dense 3D surfaces and orthophotos. For now, constrained georeferencing and updates for subsequent poses is the best compromise between performance and accuracy.

5.3 Quality of the Surface Reconstruction

In the previous section three state of the art visual SLAM frameworks were tested for their performance on aerial image data. ORB SLAM 2 has proven to be the best among the candidates and is subsequently selected for further evaluation. With it, estimation of the camera pose comes very close to the well optimized, offline photogrammetry results generated by Agisoft Photoscan. In the next step the quality of the reconstructed surface is analyzed. Literature suggests, that a highly accurate pose is the backbone of consistent 3D reconstruction [31]. However, it does not guarantee good results. Stereo cameras for example have a fixed baseline and calibrated dimensions. Yet, the reconstruction depends on the algorithms, quality of the hardware, observed scene and other quantities.

To validate the performance of the proposed implementation in the following, as well the interpolation of the sparse cloud as the 3D reconstruction using Plane Sweep Library are tested. For each of them a full run of the mapping pipeline was performed on the dataset of sec. 5.1.2. Settings for the stages were chosen, so that processing is still live on the hardware, but the spatial resolution is as high as possible. After each run, the resulting dense cloud of the final map was exported to the open source software "CloudCompare" [45]. There, the datasets were aligned with the ground truth using iterative closest point algorithm (ICP) to reduce the influence of the georeference. Afterwards every point of the real-time created dense cloud is projected onto the ground truth reference mesh. The resulting distance is encoded by a color value and displayed at its respective position on the 3D data. In sec. 5.3.1 the outcome for the interpolation method is evaluated, in sec. 5.3.2 the one for the 3D reconstruction using Plane Sweep Library.

5.3.1 Mesh Comparison for Sparse Cloud Interpolation

Fig. 5.10 presents the final surface, generated with the sparse cloud interpolation approach, compared to the ground truth. The heat map shows deviations in the range of -1.5m to 1.5m, whereas outliers are red or blue. GSD of the grid map is 0.15 m/cell. However, this is only the interpolated resolution. The true spatial GSD is equal to the resolution of the sparse cloud, which is non homogeneously spread across the map and roughly 0.5-5.0 m/cell.

Especially at the borders and in the woods the deviations are high, as there are only few repeatable observations of the surface. Additionally, at the transitions of the buildings to the ground more outliers were detected. This might be caused by the general principle of the reconstruction. Sharp edges can not be dissolved and interpolation is particularly vague in these regions. Another remarkable area is close to the turning point of the first leg (see also fig. 5.2). The deviations are mainly located in the overlap and change from negative

to positive. As previously described in sec. 5.2 the pose estimation has its highest error in this exact region due to the attitude in the georeferencing. While the global alignment error was minimized through the ICP, the local integrity of the poses get corrupted by the continuous updating of the georeference with every new keyframe. This error might propagate to the other stages and results in a deteriorated surface reconstruction. This effect should therefore also be observable in the reconstruction with the Plane Sweep Library (sec. 5.3.2). The rest of the airport is surprisingly well estimated. Even though spatial details can not be observed, buildings and the overall structure of the scene can be identified. The runway aligns well to the ground truth and has a predominant error range of 0.0 - 0.5m.

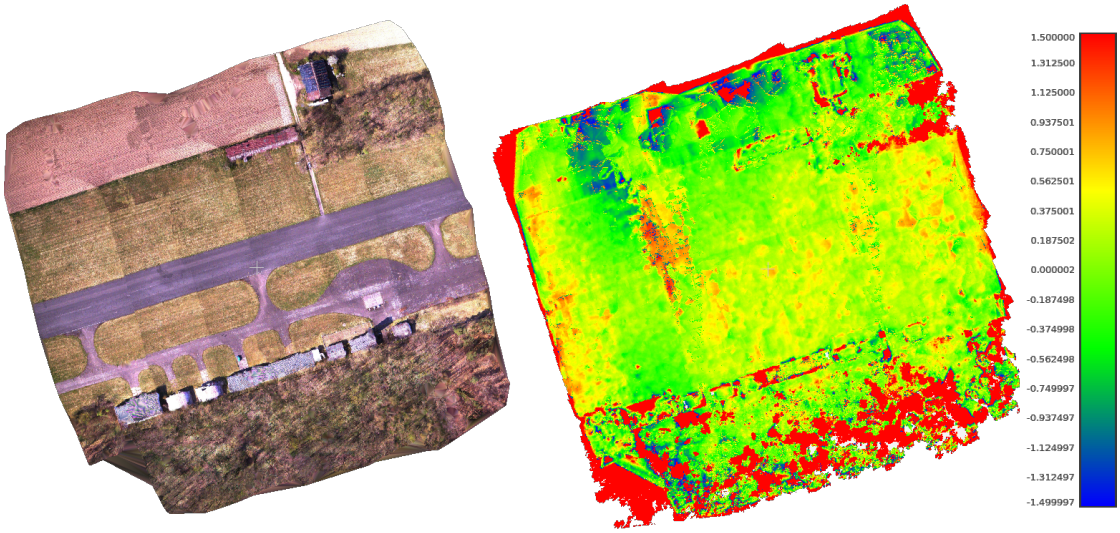


Figure 5.10: Surface comparison of the sparse cloud interpolation approach.

5.3.2 Mesh Comparison for Plane Sweep Library

In fig. 5.11 the mesh of a full 3D reconstruction is compared to the ground truth provided by Photoscan. Densification of the proposed implementation was processed using Plane Sweep Library. The achieved GSD is 0.15 m/cell, which is in contrast to sec. 5.3.1 the true spatial resolution of the surface. While the overall result looks very similar to the sparse interpolation, especially the corrugated iron roofs were badly reconstructed. The aliasing effect in the dataset described in sec. 5.1.2 seemed to have a more significant influence on the reconstruction process. This is not surprising, as the Plane Sweep Library tries to compute a depth value for every single pixel, even though the triangulation is very uncertain. In contrast, the interpolation approach is based on the sparse cloud of ORB SLAM 2, which uses highly selective features for reconstruction. The influence of the aliasing is furthermore illustrated in fig. 5.12. The rounded hangars are only vaguely recognizable in the interpolated depth map in the middle, but can be well identified in the 3D reconstruc-

tion on the right. Yet, the aliased region is filtered by ORB SLAM 2 resulting in a better estimation of the corrugated iron roof.

To improve the surface of the Plane Sweep approach modifications in the mosaicing have to be carried out. As previously stated in sec. 4.7, blending assumes that the estimates of the surface are getting better with every observation. This is mainly true for the interpolation approach, but not keepable for the 3D reconstruction. An algorithm that forces smoothness in overlapping regions for different observations and minimizes a defined energy function should improve the results significantly. Such techniques are standard for offline photogrammetry, as for example proposed by Kazhdan et al. [46]. Yet, their applicability on real-time approaches with limited processing power is not sufficiently examined.

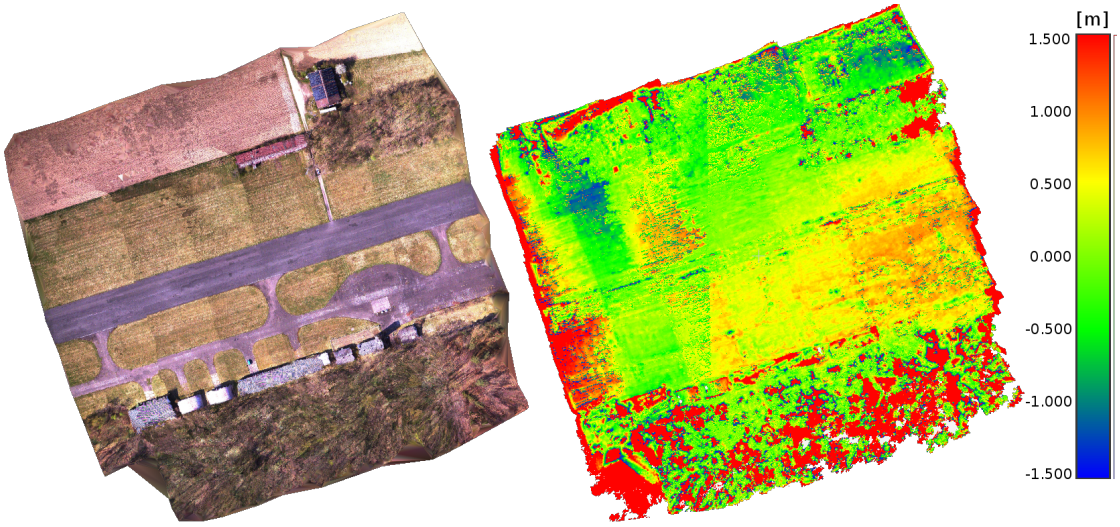


Figure 5.11: Surface comparison of the 3D reconstruction approach.

Figure 5.12: Depth of an image with aliasing.

Despite the negative influence of the aliasing, another observation in fig. 5.11 is of interest. As proposed in the previous section, deviations at the first turning point in the upper left of the map are extraordinary high. The suggested assumption in the previous sec. 5.2

pointed out, that the pose error in this region is high as well due to the estimated attitude during the georeferencing and its correction subsequently. Because it is observable in both approaches a systematic effect is likely and fits the theory. Consequently pose and surface accuracy are tightly coupled.

The rest of the runway aligns well with the ground truth. The plane field at the top of the map is slightly less deviated than in the interpolation approach (-0.5 to 0.5m against -1.5 to 0.5m), in contrast the error on the right part is slightly higher (0.5 to 1.0m against 0.0 to 0.5m). All in all the 3D reconstruction does not perform significantly better, even though the depth maps are visually superior. As stated, the workflow of the mosaicing might be improved to achieve higher quality meshes in the future.

5.4 Quality of the Orthophoto

Simultaneously with the surface reconstruction in the previous section a 2D orthophoto was generated. Orthophotos provide a fast overview of the mapped area and became essential to the aerial photogrammetry. They are aligned to a global coordinate frame of choice and allow to measure distances in real world scale. Evaluation of such is difficult, as visual distortions can easily be spotted by humans, but are more challenging to be detected automatically. For that reason in sec. 5.4.1 a manual (visual) inspection of the orthophoto results is presented using open source software QGIS 3.0 [47]. Afterwards in sec. 5.4.2 the relative integrity of the map is evaluated by measuring certain dimensions and distances in both, the generated ground truth and the maps produced by the proposed implementation. At last in sec. 5.4.3 the absolute positional errors are analyzed. For the sake of comparison all possible approaches are tested. This includes those with

- planar surface assumption and pose based on GNSS and heading only
- planar surface assumption and pose based on visual SLAM pose estimation
- elevated surface assumption, pose based on visual SLAM estimation and densification using sparse cloud interpolation
- elevated surface assumption, pose based on visual SLAM estimation and densification using PSL

5.4.1 Visual Inspection

A good orthophoto has high resolution, homogenous appearance, low geometric distortion and no visual artifacts. It must represent the observed scene with as much details as possible and should be accurately aligned globally. Ground truth for the visual inspection is shown in fig. 5.14 (top, middle). As expected, Agisoft Photoscan was able to reconstruct

the scene without major flaws. The map contains no ghosting or overlap edges and has a resolution of up to 0.07m/px. Exposure time changes in the camera slightly propagated to the field in the top of the map, which resulted in blooming of the area. But it does not influence the integrity of the map significantly.

The orthophotos generated by the proposed framework are shown below the ground truth. In the upper left the one generated by GNSS and heading pose is shown. It is mainly weak aligned with visual artifacts all over the map. Effects like these can be induced by an offset from UAV to camera heading, which is assumed to be equal (see also sec. 4.2). Due to the fact that in the current implementation no exposure correction or extensive color blending is performed, borders of the individual images can be seen in all approaches. The overall result is not sufficient for most use cases, as points of interest might be hidden by the misaligned image regions. However, it works for generic camera sensors of all kind (RGB, IR, ...) and is independent of the observed scene.

Figure 5.13: Detailed view of houses for elevated approaches.

The visual SLAM approach on the upper right in contrast shows improvement to the one prior. It represents the state of the art for real-time mapping, as it does not need a full reconstruction of the surface, but allows to create visually appealing results. With improved blending the orthophoto comes close to the ground truth. Though, geometric distortion is a fundamental problem which can not be overcome without surface information.

In the lower left and lower right the maps of the approaches with surface elevation are displayed. These are computationally more expensive due to the densification process, but show rectified, homogenous results similar to the Photoscan orthophoto. A major difference between these two can not be examined on first sight. Fig. 5.13 presents a detailed view on two buildings in the global map. The Agisoft results on the left look well reconstructed. Sparse interpolation in contrast had difficulties at sharp elevation transitions. This was already observed in the previous sec. 5.3 while analyzing the 3D surface data. The low resolution of the sparse cloud propagates to uncertain elevation, which in turn propagates to artifacts that distort the visual appearance. In the orthophoto of the

dense reconstruction approach these edges are much sharper, even though it suffers from noise and high frequent artifacts also. Suggested improvement of the mosaicing in sec. 4.7 will also result in a visually more consistent map.

Altogether the surface data improved the global map visually. Larger misalignments and geometric distortions are removed for the cost of high frequency noise. Though, the greatest benefit is the 3D impression of the observed scene. In use cases like search and rescue such information are crucial for coordination and situational awareness.

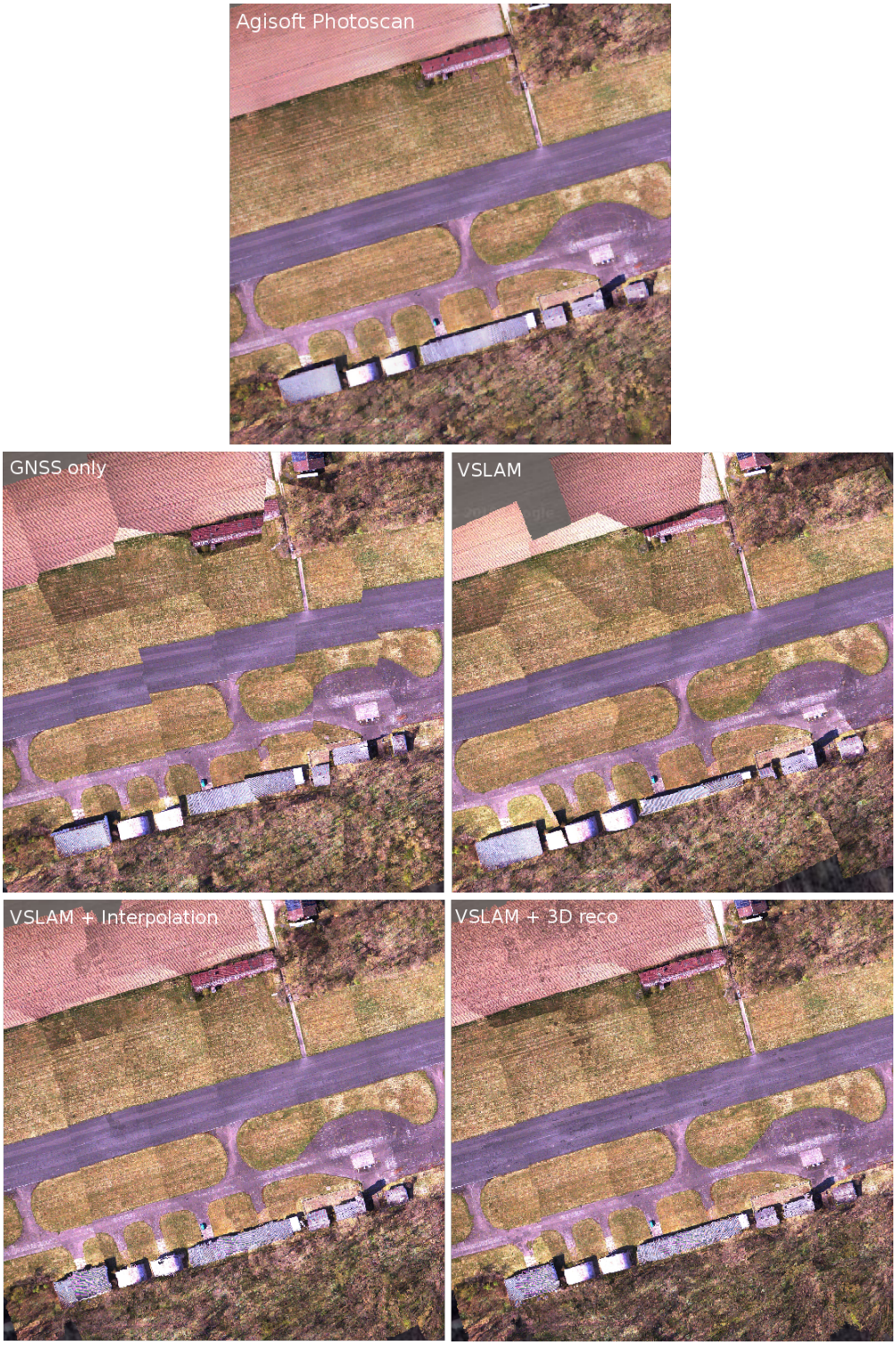


Figure 5.14: Visual inspection of the resulting ortho photo.

5.4.2 Relative Integrity

While the inspection of sec. 5.4.1 aims to detect visual distortions and misalignments, in this section an analysis of the relative integrity is performed. For that, dimensions and distances are measured in the global, georeferenced map for each of the approaches. Again, the orthophoto generated by Agisoft Photoscan is used as ground truth. In fig. 5.15 an overview of the objects is given, whose dimensions are checked. For buildings B1 to B8 both diagonal corner to corner distances are calculated and compared to the ground truth. Deviations for each building are listed in appendix X. Additionally, P1 to P4 represent points of interest, whose distances are measured according to fig. 5.15. The absolute deviations of B1 to B8 and P1 to P4 are then averaged and shown in table 5.1.

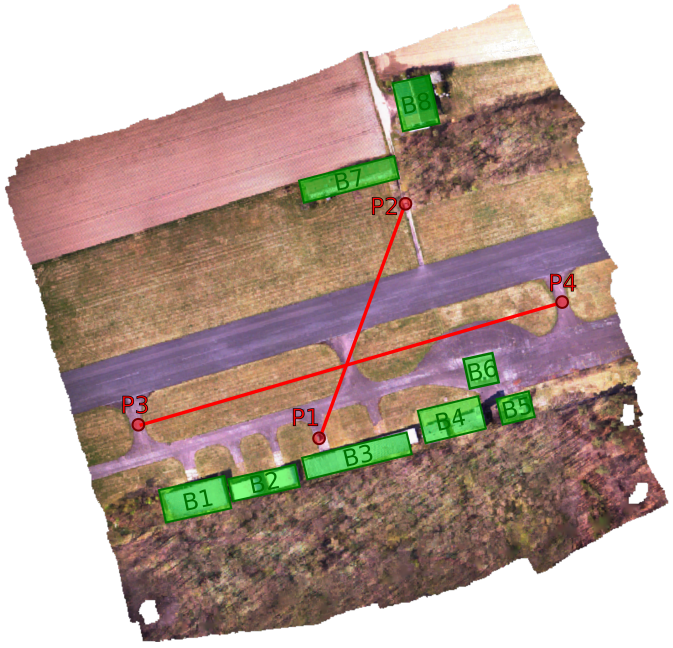


Figure 5.15: Objects to be evaluated.

As expected, the consideration of surface elevation results in a low dimensional error. While the sparse cloud interpolation achieves an average deviation of 0.86m to the ground truth, the dense reconstruction with PSL obtains 0.89m. Standard deviation for both is also low with 0.73m and 0.85m, which indicates an overall consistent solution. In contrast the GNSS based approach has an average error of 2.6m and the visual SLAM even more with 5.1m. While the visual inspection suggested, that the visual SLAM only approach produces a better map, the relative integrity seems to be worse. One possible cause may be the missing rotational component in the evaluation, which would detect and penalise the heading misalignment in the GNSS approach.

Table 5.1: Relative orthophoto integrity errors for all approaches.

Approach	Assumption	Relative Error [m]	
		Average	Std.dev.
GNSS + Heading	Planar	2.60	1.60
Visual SLAM	Planar	5.10	6.30
Sparse Interpolation	Elevated	0.86	0.74
3D-Reconstruction	Elevated	0.89	0.85

Another reason may be the distortion induced by the georeference updates. The latter effect was already presented in the previous sections and was expected to propagate to the orthophoto. In support of this, the errors at buildings B1 and B2, which lie in the flight path of the first leg, are among the highest of all deviations for the three SLAM based approaches. The GNSS method is unaffected by this, as it does not need georeferencing. Additionally, while the surface reconstructed approaches require at least two observations per grid cell and a low variance on the elevation before being declared as valid, the planar SLAM method has no such restriction. Therefore individual outliers may have a higher impact on the final map.

5.4.3 Absolute Alignment

After inspection of the visual appearance and relative integrity of the orthophoto, in this section its absolute alignment to the georeference is analysed. For that, points in the global map are compared to their position in the ground truth. At best, this ground truth is exactly calibrated using highly accurate techniques, such as differential GNSS with additional correction data. However, for the proposed dataset such information is not available. A comparison to the Agisoft Photoscan data should therefore be sufficient. To achieve this, the points of interest P1 to P4 in fig. 5.15 were measured in UTM coordinates for all four created maps and their translational deviation to the ground truth was computed. Results are presented in table 5.2.

Table 5.2: Absolute orthophoto alignment errors for all approaches.

Approach	Assumption	Absolute Alignment Error [m]					
		P1	P2	P3	P4	Average	Std.Dev.
GNSS + Heading	Planar	1.28	4.69	3.50	6.14	3.90	2.10
Visual SLAM	Planar	2.10	1.39	19.42	2.06	6.23	8.80
Sparse Interpolation	Elevated	0.32	0.41	1.00	0.14	0.47	0.37
3D-Reconstruction (PSL)	Elevated	0.45	0.7	1.60	0.32	0.77	0.58

Similar to the relative integrity, orthophotos from the elevated surface produce the most accurate results. The sparse interpolation approach achieves an average deviation of 37cm, whereas the 3D reconstruction obtains 57cm. The planar visual SLAM method has the highest error again, but it becomes clear that P3 is an outlier with 20m distance from the ground truth. It is likely, that this misalignment had its roots in the attitude error during georeferencing, which was already extensively described in the previous sections. The points following up in the timeline of the flight (P1, P2, P4) have low displacements of 1.4-2.1m.

5.5 Processing Performance

As last step of this chap. the evaluation of the processing performance is carried out. In general, a stage should process data slightly faster than it receives new input. Due to the fact, that the proposed implementation is designed as multithreaded pipeline a simple time measurement for each stage however is not sufficient. If for example the densification stage has an average computation time of 0.2s, it might publish new frames for the next stage with 5 Hz, but it might as well publish at 1 Hz. A consecutive stage with an average computation time of 1.0s might then be okay, or overloaded by a factor of 5. By measuring the downtime also idle states are detected. Yet, a downtime of 0.0s might as well mean the stage is running just fine.

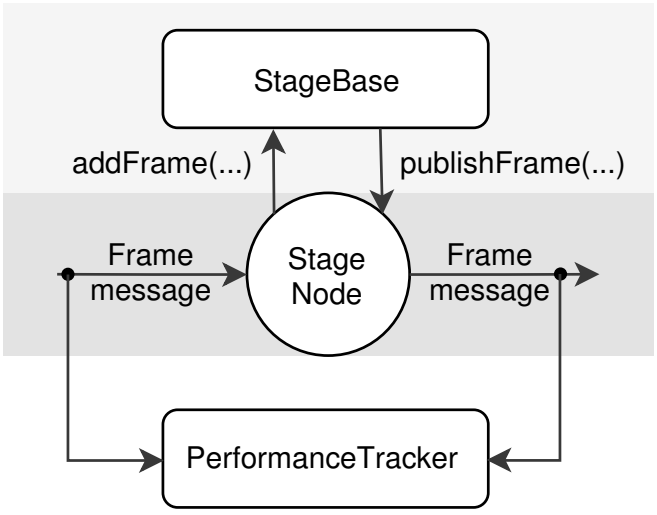


Figure 5.16: Performance tracker linked to each stage node.

Therefore another approach was chosen that is outlined in the following. The current transport layer is implemented in ROS. Therefore messages exchanged between the processing stages are transferred in the ROS infrastructure. To identify if each stage is performing within the limits the exchanged message rates are tracked. By measuring the input and output message frequency of each stage the overall workload can be estimated (see fig. 5.16). One assumption that is required is, that every stage processes every frame

and has no artificial throttle to reduce the output rate. Because of the keyframe selection this is only the case for the pose estimation stage, which is why it is neglected for now. All other stages process and publish every frame they receive. The performance measure delta is then defined as

$$\delta_{Perf} = \frac{f_{in}}{f_{out}}, \tag{5.2}$$

where f_{in} represents the input and f_{out} the output message frequency. Consequently a stage that is publishing messages as fast as it receives new data ($\delta_{Perf} = 1.0$) is declared as "real-time performant". Stages with a $\delta_{Perf} > 1.0$ are getting frames faster than they are able to publish them, which indicates high workload and is therefore labeled as overloaded. A $\delta_{Perf} < 1.0$ should not be possible as such a stage would generate more data than it receives. However, due to the fact that the performance tracker averages the frequency over time it might underlie temporary variations. Therefore only the steady-state is of interest.

In fig. 5.17 the performance log of the sparse interpolation approach is presented. It was recorded during the same run as sec. 5.2, 5.3 and 5.4. Therefore pose estimation used ORB SLAM2 as visual SLAM framework and processed incoming images in full resolution. GSD of the DSM was set to 0.15m/cell. On the x-axis the time from start of the mapping procedure to the end is plotted, while the y-axis shows the performance quotient δ_{Perf}. Additionally, the average pose estimation publish frequency was noted with 2.4Hz to get an impression of the final update rate of the global map. The log shows, that the sparse interpolation has no problems achieving the desired performance of $\delta_{Perf} = 1.0$. In the beginning it has higher fluctuation, which is caused by the mentioned averaging effect. All in all, due to the stationary convergence of all stages the processing can be labeled "real-time" under the defined prerequisites.

Fig. 5.18 in contrast shows the performance of the 3D reconstruction approach using PSL. Again, the settings were the same as in the previous sec. and also equal to the ones set in the sparse interpolation run. The resulting log exposes overall more variation in the performance, especially in the beginning. The densification stage has higher peaks once in a while. This might be explained by the additional involvement of the GPU or a more complex reconstruction of the scene. Yet, it also converges to $\delta_{Perf} = 1.0$ and is therefore performing within the requirements.

For comparison, in fig. 5.19 a performance log was plotted, that shows an overloaded mapping run. Densification was using 3D reconstruction and PSL, but with slightly higher resolution than in the previous attempts. This forces as well densification as surface generation to perform more operations, which results in a higher workload. The performance quotient of both stages is consequently converging to a higher value of $\delta_{Perf} \approx 1.6$. In conclusion, during the time they receive 3 frames only 2 are published. This underperformance will yield to gaps in the mapping result for long missions.

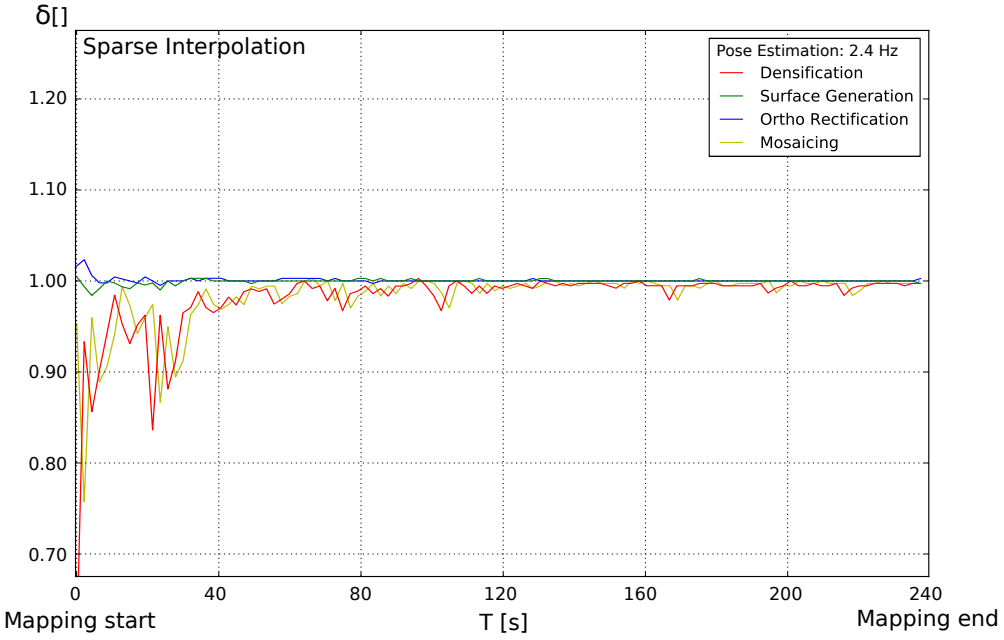


Figure 5.17: Performance measure δ_{Perf} for the sparse interpolation approach.

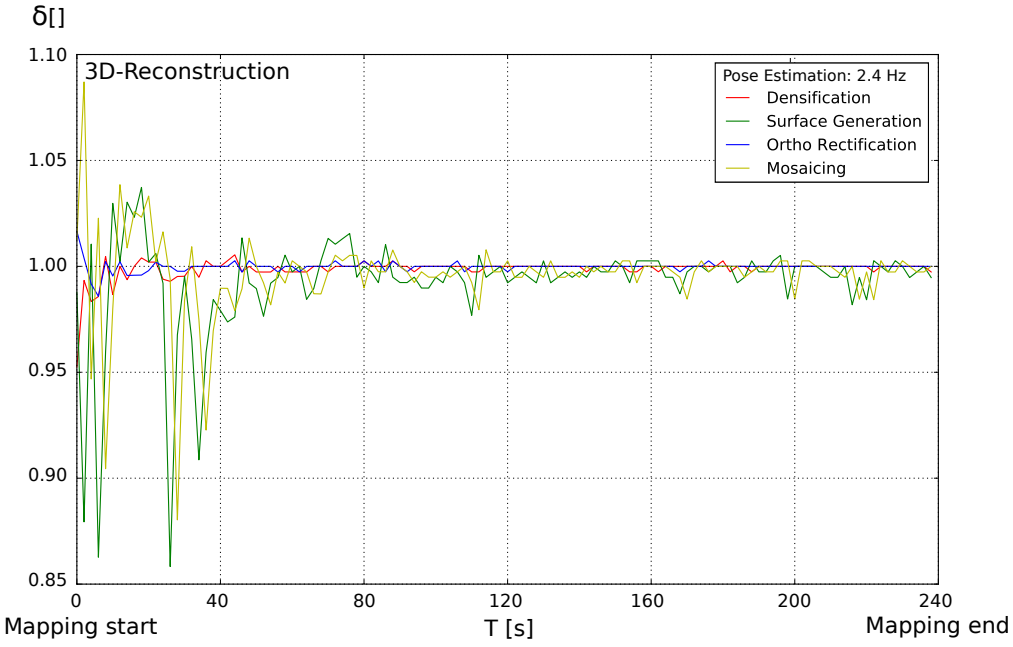


Figure 5.18: Performance measure δ_{Perf} for the 3D-reconstruction approach using PSL.

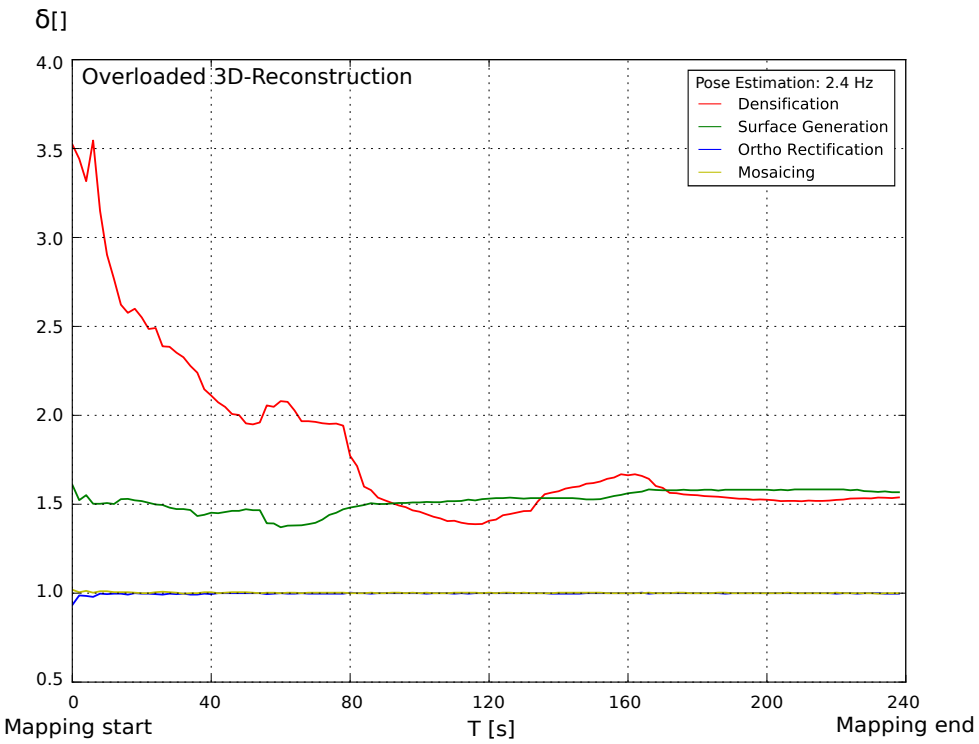


Figure 5.19: Performance measure for a overloaded run.

6 Conclusion

Goal of this thesis was to develop a real-time capable photogrammetry framework for unmanned aerial vehicles. Camera pose, 3D surface and the geometrically corrected orthophoto were the main objectives to obtain. After a roundup of the mathematical background and modelling an introduction to all relevant state of the art topics was given. On the one hand visual SLAM frameworks and monocular dense reconstruction were important, as the proposed implementation relies on such for pose and surface recovery. On the other hand the state of the art for real-time mapping was presented. While there are already open source frameworks available that allow mapping in (partially) 3D, none of them provided a sufficient architecture. Map2DFusion by Bu et al. [30] integrated a visual SLAM into their mapping pipeline to achieve 3D real-time performance on the pose estimation. However, afterwards images are projected into a common reference plane. A geometrical correction can not be performed that way. Aerial Mapper by Hinzmann et al. [31] comes closer to the desired goals as it uses sensor fusion to acquire a 3D pose and reconstructs the dense surface subsequently. Yet, for the main duration of this thesis essential modules were closed source, which made testing and modification difficult. Furthermore Aerial Mapper is designed to be computed onboard a UAV and only a basic 2D feature tracker is applied for the visual estimation.

Consequently a new implementation was proposed, which is structured in three layers. The first is the library layer that contains all essential algorithms. The second is the stage layer, which is designed as classical pipeline and was separated into five different stages. Each stage has a flow control that uses the library layer to incrementally process incoming frames and pass them to the next stage as soon as it finished. The data communication in turn was realized by the third layer, that implements a ROS[34] network. Consecutive stages are therefore each linked by an appropriate ROS node. To now achieve real-time mapping, acquired images with GNSS and heading information are first passed to the pose estimation stage. There, an interface class allows to integrate state of the art visual SLAM frameworks for motion recovery. Extracted camera poses in the local coordinate system are subsequently transferred to a geographic ENU frame using the global position information provided by the UAV and the best-fit Umeyama algorithm[35]. With an accurate, geographic pose and a sparse cloud of the observed scene the frames are then published for the next stage. Densification is responsible for the dense estimation of the surface and is realized analogically to the prior presented pose estimation stage. With an interface class existing, external 3D reconstruction frameworks can be integrated into

the mapping process to provide high quality, dense surface information. As fallback solution an interpolation based on the sparse cloud and image inpainting is provided. After extracting the depth map for each frame and reprojecting it to a dense cloud, the digital surface model is generated. This is an essential step, as previous mapping frameworks (except for Aerial Mapper) did not provide such surface model in real-time. A multi-layered grid map was chosen as representation and allows to save a variety of information for each world sample in an efficient manner. Afterwards the rectification of the image data is carried out. By using the linear pinhole model the prior extracted elevation grid map is reprojected into the camera so the visiual distortion induced by the terrain is corrected. In the last stage all incremental images are finally integrated into a global mosaic. Spatial information is blended by using a probabilistic approach and minimizing its variance consequently.

After presenting the implementation in details, a further evaluation was performed. Four essential goals were aimed for. The first was to identify the accuracy of three different state of the art visual SLAM frameworks. These have been tested extensively in the literature [40], but public available datasets do not cover the use case of a classical mapping scenario. Two of the implementations did not finish the complete sequence. Especially featureless regions that are common in aerial views (water, acre, ...) were major challenges. The second goal was to identify the overall reconstruction quality of the surface compared to the offline photogrammetry, which is already commercially applied. Both proposed approaches, the sparse cloud interpolation and the dense reconstruction using Plane Sweep Library (PSL) [27], showed equally good results. The surface was reconstructed in a consistent way using the pose generated by ORB SLAM 2 [17]. However, high frequency noise from the depth maps propagated to the elevation for the dense reconstruction case. The mosaicing stage, which should handle such noise by minimizing the variance of the spatial information, was not ideally designed for this scenario. It assumes, that elevation estimates get better with every observation. However, this is only the case for the interpolation method, as it relies on the steadily optimized sparse cloud of ORB SLAM 2. In a full reconstruction scenario like PSL new measurements are not better or worse than the ones before. After validating that the reconstructed surface represents the scene, the orthophoto quality was analyzed. For that, an extensive comparison to classical maps based on planar surface assumption was carried out. As well a visual inspection, as a measure on relative integrity and absolute alignment to the ground truth was performed. The orthophotos from elevated surfaces produced significantly better results showing a relative and absolute deviation of less than 1.0m to the results extracted by Agisoft Photoscan. Yet, the noise of the surface propagated as visual artifacts to the orthophoto. At last the real-time performance of the framework was validated by defining a performance quotient, which sets input and output message frequency for every stage into relation. It was shown, that in the steady-state all stages process incoming frames at least as fast as they receive new ones despite a high ground sampling distance.

7 Future Work

In the introduction to this thesis it was already stated, that the developed framework is ready-to-use and published open source. However, there are still interesting topics that should be covered in the future. Especially a proper integration of the dense reconstructed data in the mosaicing is an important step to achieve higher quality meshes and orthophotos. A possible solution should consider the heavy noise visual reconstruction algorithms can produce and force smoothness in an efficient formulation. Additionally, the global consistency can be further improved if updates of the georeference are applied retroactive. Therefore as soon as a better georeferencing was computed, poses of the past should be refined. This implies also to detect when and how to update past measurements and 3D information in the global mosaic.

For a better orthophoto quality especially exposure correction might be performed in the future. While this can be challenging to do in real-time, the general constraints of the mapping scenario might be exploited. After two flight legs with 50% side overlap the areas at the beginning will get no further updates as the UAV typically moves in serpentines. Therefore processing that is performed in such areas is likely to produce final results. That way even computationally intensive tasks can be incorporated in the real-time framework.

At last the modularity of the framework can be used in the future to develop a performance bench for different algorithms. If for example new visual SLAM or dense reconstruction implementations are published, they might be tested according to a standardized test procedure for application on the real-time aerial mapping scenario. That way the general standard for mapping can be improved in a similar way, as it was achieved by public datasets for visual SLAM. A researcher can then concentrate on a specific topic he is interested in, without the need for the implementation of a whole end-to-end pipeline.

References

[1] Technavio. *Technavio - Applications of aerial imaging from 2017-2021*. 2018. URL: https://www.technavio.com/blog/top-3-applications-of-aerial-imaging-in-2017-and-beyond (visited on 06/04/2018).

[2] PX4 Dev Team. *PX4 - Open Source Autopilot for Drones*. 2018. URL: http://px4.io/ (visited on 06/04/2018).

[3] Ardupilot. *Ardupilot - An open source autopilot system*. 2018. URL: http://ardupilot.org/ (visited on 06/04/2018).

[4] Pix4D S.A. *Pix4D - Commercial software for drone imagery*. 2018. URL: https://pix4d.com/ (visited on 06/04/2018).

[5] Agisoft LLC. *Agisoft - Stand-alone software product that performs photogrammetric processing of digital images and generates 3D spatial data*. 2018. URL: http://www.agisoft.com/ (visited on 06/04/2018).

[6] DroneDeploy. *DroneDeploy - Powerful Drone and UAV Mapping Software*. 2018. URL: https://www.dronedeploy.com/ (visited on 06/04/2018).

[7] Johannes Lutz Schoenberger et al. *COLMAP is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline with a graphical and command-line interface*. 2018. URL: https://colmap.github.io/ (visited on 06/04/2018).

[8] Bill Triggs et al. "Bundle Adjustment – A Modern Synthesis". In: *VISION ALGORITHMS: THEORY AND PRACTICE, LNCS*. 2000, pp. 298–375.

[9] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[10] Richard Szeliski. *Computer vision algorithms and applications*. London; New York: Springer, 2011. ISBN: 9781848829343 1848829345 9781848829350 1848829353. URL: http://link.springer.com/book/10.1007%2F978-1-84882-935-0.

[11] Intel and Willow Garage. *Radial and tangential undistortion model of OpenCV*. 2018. URL: https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html (visited on 06/04/2018).

[12] Zhengyou Zhang. "A Flexible New Technique for Camera Calibration". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.11 (Nov. 2000), pp. 1330–1334. ISSN: 0162-8828. DOI: 10.1109/34.888718. URL: http://dx.doi.org/10.1109/34.888718.

[13] Christopher B Choy et al. "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016.

[14] Thomas Luhmann et al. *Close Range Photogrammetry and 3D Imaging*. Nov. 2013. ISBN: 978-3110302691.

[15] Olivier Faugeras and F. Lustman. *Motion and structure from motion in a piecewise planar environment*. Tech. rep. RR-0856. INRIA, June 1988. URL: https://hal.inria.fr/inria-00075698.

[16] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. "Visual SLAM algorithms: a survey from 2010 to 2016". In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), p. 16. URL: https://doi.org/10.1186/s41074-017-0027-2.

[17] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[18] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry". In: *CoRR* abs/1607.02565 (2016). arXiv: 1607.02565. URL: http://arxiv.org/abs/1607.02565.

[19] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014.

[20] Ethan Rublee et al. "ORB: an efficient alternative to SIFT or SURF". In: *In ICCV*.

[21] Dorian Galvez-Lopez and Juan D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *Trans. Rob.* 28.5 (2012), pp. 1188–1197. ISSN: 1552-3098. DOI: 10.1109/TRO.2012.2197158. URL: http://dx.doi.org/10.1109/TRO.2012.2197158.

[22] R. Kuemmerle et al. "g2o: A General Framework for Graph Optimization". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949.

[23] Guenther Schelling et al. "Angewandte Geodaesie und Photogrammetrie in Graz". In: *Oesterreichische Zeitschrift fuer Vermessungswesen und Photogrammetrie* 73.1 (1985), pp. 56–61.

[24] Andrew J. Davison. "Real-Time Simultaneous Localisation and Mapping with a Single Camera". In: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV '03. 2003, pp. 1403–. ISBN: 0-7695-1950-4.

[25] Heiko Hirschmueller. "Evaluation of Cost Functions for Stereo Matching". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2007.

[26] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2014.

[27] Christian Häne et al. "Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo." In: *3DV*. IEEE Computer Society, 2014, pp. 57–64. ISBN: 978-1-4799-7000-1.

[28] David Gallup et al. "Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions." In: *CVPR*. IEEE Computer Society, 2007. ISBN: 1-4244-1179-3.

[29] Kekec Taygun, Yildirim Alper, and Unel Mustafa. "A new approach to real-time mosaicing of aerial images"". In: *Robotics and Autonomous Systems* 62.12 (2014), pp. 1755 –1767. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2014.07.010.

[30] Shuhui Bu et al. "Map2DFusion: Real-time incremental UAV image mosaicing based on monocular SLAM". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*. 2016, pp. 4564–4571. DOI: 10.1109/IROS.2016.7759672. URL: https://doi.org/10.1109/IROS.2016.7759672.

[31] T. Hinzmann et al. "Mapping on the Fly: Real-time 3D Dense Reconstruction, Digital Surface Map and Incremental Orthomosaic Generation for Unmanned Aerial Vehicles". In: *Field and Service Robotics - Results of the 11th International Conference*. 2017.

[32] OSGeo. *GDAL is a translator library for raster and vector geospatial data formats*. 2018. URL: http://www.gdal.org/ (visited on 06/04/2018).

[33] Benoît Jacob and Gael Guennebaud. *Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms*. 2018. URL: http://eigen.tuxfamily.org/index.php?title=Main_Page (visited on 06/04/2018).

[34] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[35] Shinji Umeyama. "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 13.4 (Apr. 1991), pp. 376–380. ISSN: 0162-8828. DOI: 10.1109/34.88573.

[36] In:

[37] Peter Fankhauser and Marco Hutter. "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation". In: *Robot Operating System (ROS) – The Complete Reference (Volume 1)*. Ed. by Anis Koubaa. Springer, 2016. Chap. 5. ISBN: 978-3-319-26052-5. DOI: 10.1007/978-3-319-26054-9{_}5. URL: http://www.springer.com/de/book/9783319260525.

[38] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching". In: *Commun. ACM* 18 (9 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007.

[39] Google LLC. *Google Maps ist ein Online-Kartendienst des US-amerikanischen Unternehmens Google LLC*. 2018. URL: https://www.google.de/maps (visited on 06/04/2018).

[40] Nan Yang, Rui Wang, and Daniel Cremers. "Feature-based or Direct: An Evaluation of Monocular Visual Odometry". In: *CoRR* abs/1705.04300 (2017). arXiv: 1705.04300. URL: http://arxiv.org/abs/1705.04300.

[41] Michael Burri et al. "The EuRoC micro aerial vehicle datasets". In: *The International Journal of Robotics Research* (2016). DOI: 10.1177/0278364915620033. eprint: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html. URL: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract.

[42] J. Sturm et al. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. 2012.

[43] Michael Grupp. *Python package for the evaluation of odometry and SLAM.* 2018. URL: https://github.com/MichaelGrupp/evo (visited on 06/04/2018).

[44] Rainer Kümmerle et al. "On Measuring the Accuracy of SLAM Algorithms". In: *Auton. Robots* 27.4 (Nov. 2009), pp. 387–407. ISSN: 0929-5593. DOI: 10.1007/s10514-009-9155-6. URL: http://dx.doi.org/10.1007/s10514-009-9155-6.

[45] EDF. *3D point cloud and mesh processing software Open Source Project.* 2018. URL: http://www.danielgm.net/cc/ (visited on 06/04/2018).

[46] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP '06. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70. ISBN: 3-905673-36-3. URL: http://dl.acm.org/citation.cfm?id=1281957.1281965.

[47] QGIS Development Team. *QGIS - Ein freies Open-Source-Geographisches-Informationssystem.* 2018. URL: https://www.qgis.org/de/site/ (visited on 06/04/2018).

Institute of Flight Guidance
Hermann-Blenk-Straße 27
38108 Braunschweig