Visual Inertial 3D Modeling

Improving the performance of dense 3D point cloud reconstruction algorithms

Q. Dekker

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Visual Inertial 3D Modeling

Improving the performance of dense 3D point cloud reconstruction algorithms

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Q. Dekker

September 24, 2020

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

ERICSSON

The work in this thesis was supported by Ericsson. Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Visual Inertial 3D Modeling

by

Q. Dekker

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: September 24, 2020

Supervisor(s):

dr. ir. T. J. J. van den Boom

dr. ir. M. Kok

Reader(s):

dr. V. Grancharov

dr. C. S. Smith

Abstract

Dense 3D modeling based on monocular visual data is a powerful process of gaining spatial 3D understanding from 2D observations. The use of visual data to reconstruct such 3D models is still a challenging topic. To obtain the accurate dimensions, additional metadata is required such as a Global Positioning System (GPS) which is not always available. Besides this, dealing with challenging visual situations such as bad-lightening conditions or motion blur remains a difficult subject. Furthermore, since visual data is highly dimensional, most algorithms lack scalability, meaning that they fail to reconstruct 3D models in acceptable time limits and are incapable of handling large data sets.

In this thesis, the objective is to mitigate these typical issues whilst preserving the quality of the dense 3D models. To this end, visual-inertial Simultaneous Localization and Mapping (SLAM) and Multi View Stereo (MVS) techniques are combined to form a dense 3D modeling architecture that is capable of mitigating the typical challenges of classical dense 3D modeling approaches. Besides this, the architecture is extended with additional improvements in the MVS system. These improvements further increase the scalability by abstracting the input data in a highly compact representation by leveraging image segmentation techniques. The result is a novel, visual-inertial dense 3D modeling system.

The novel system is tested on benchmark data sets and within a lab setting, where a remote-inspection case-study is performed. The presented system is compared against the industrial and academic state-of-the-art systems. A thorough comparison is made by evaluating the pose accuracy, computation time, and reconstruction quality.

It is shown that the presented system improves the state-of-the art systems by a significant margin in terms of computation time. Furthermore, the presented system is capable of computing 3D models with accurate geometric scale without relying on external metadata, showcasing the effectiveness of the presented system. This work contributes to the lack of research in dense 3D modeling based on visual-inertial SLAM and paves the way for a new direction of efficient MVS algorithms.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank the research team of Ericsson AB - dr. V. Grancharov, dr. S. Sverrisson, dr. J. Araújo and dr. H. Pobloth for their support and guidance during the writing of this thesis. It was a great learning experience to be part of their research team for this brief period of time. I am in particular grateful for their continuing support when I was forced to work remotely in light of the 2020 pandemic.

Furthermore, I am very grateful for the fruit full discussions and interactions that I had with my supervisors from the Delft University of Technology - dr. ir. T. J. J. van den Boom and dr. ir. M. Kok.

Delft, Q. Dekker
September 24, 2020

"It is impossible for a man to learn what he thinks he already knows."
— *Epictetus, Discourses, Book II, ch. 17*

Chapter 1

Introduction

1-1 Research motivations

As a global telecom infrastructure provider, Ericsson plans to expand a global 5G network worldwide starting from 2020. Such a network would service more than 1.9 billion subscribers and contains approximately 1-2 million cellular sites [32, 43]. Such sites are thoroughly inspected after installation to ensure that it is functioning properly when it is handed over to the customer - this is the process of acceptance of the installation from Ericsson's client. In this process, all findings have to be documented which is very time-consuming. After acceptance, the sites are visited again to upgrade the equipment or to perform troubleshooting when technical problems arise. The field of 3D modeling has opened up an opportunity to do this in an alternative, more time-efficient and save manner. Instead of requiring a technician to visually inspect the cellular site, a drone is flown to the tower which records visual data. Using this data, a 3D point cloud model is created of the cellular site with a 3D modeling architecture. Then, the technician can use this digital 3D point cloud model, a digital twin, from the convenience of his or her office. The difference between manual and remote inspection is shown in Figure 1-1.

Figure 1-1: Manual inspection (left) can be replaced with a digital twin, captured using a remotely operated drone (right).

Such a digital model offers several use-cases including but not limited to:

- Minimizing safety hazards associated with climbing the cellular towers.
- Automating the acceptance tests for Ericsson's customer.
- Planning expansion or upgrading projects ahead by using digital models of the cellular tower that have correct dimensions.

The current process works by flying the drone around the cellular tower and capturing visual data (i.e. images) from different heights, orientations and angles. With this data, the 3D modeling architecture computes a 3D point cloud model. This model is then used by the technician to automate the acceptance tests, check for faults or wear and tear and plan maintenance ahead of visiting the site. An overview of a typical acquisition pattern and an inspection example is shown in Figure 1-2.

Figure 1-2: The 3D model is used to inspect an antenna of a cellular tower (left). Specifically, in this example, the technician is inspecting the bottom part of the equipment where the connectors are located. This 3D model is acquired by flying a pattern with a UAV around the cellular tower (right) and capturing a set of pictures.

Although the quality of the current 3D modeling architecture is satisfactory, there exist three major shortcomings: robustness, scalability and the ability to observe geometric scale without metadata. Robustness problems are observed in challenging visual settings such as in indiscriminate image regions or places with bad lightening. A lack of robustness is then defined as the incapability of handling such situations. The model quality may suffer or the reconstruction might even fail completely. Scalability is defined as the required computational resources as a function of the input data. In the current architecture, the required resources scale quadratically with the amount of input images [30], making it difficult to obtain results when the technician is still present at the site. This is important since the data-acquisition process may have failed and if this becomes clear after the technician is back, it means that he or she has to return to the site at a later point. Finally, in the current architecture, it is impossible to directly observe geometric scale without relying on captured metadata such as a Global Positioning System (GPS) information. This is problematic since it is important to be able to perform accurate measurements on the 3D point cloud model. If the geometric scale is unknown, this is not possible. Combined, these shortcomings limit the current use-case for remote inspection. This directly leads to the motivation of this work. In this thesis, an alternative architecture is proposed which has the purpose of mitigating the shortcomings of the current architecture. It tries to answer the question:

> "What is an architecture that reconstructs 3D point cloud models with comparable quality as the current state-of-the-art architectures whilst mitigating the typical shortcomings?"

Since it is required to obtain 3D reconstructions that are comparable to the current architectures, the proposed system would aim to minimize the required computational resources whilst preserving the 3D model quality, improve the performance in challenging visual environments and be able to extract the geometric scale of 3D point cloud models without relying on external metadata.

1-2 Research contributions

There are four main contributions in this work. First, the performance of the current state-of-the art visual 3D modeling architectures are studied on a real-life data set of a cellular site provided by Ericsson AB in the context of remote inspection. Secondly, visual-inertial Simultaneous Localization and Mapping (SLAM) and Multi View Stereo (MVS) techniques are combined to produce a dense 3D modeling architecture that offers a reduction in terms of computational complexity and is able to directly observe geometric scale. Thirdly, a further reduction in the computational complexity of the MVS module is achieved by leveraging image over-segmentation techniques. Both approaches are combined which yield a novel architecture for dense 3D modeling. Fourth and finally, the proposed method is thoroughly tested on public benchmarks as well as lab-experiments, recorded on representative data sets. The proposed system is compared against the current academic and commercial state-of-the-art visual 3D modeling architectures. In all experiments, the computation time, accuracy of the reconstructed poses and the dense 3D model quality are evaluated.

1-3 Outline of the thesis

The research question is answered in a structured approach in four steps. First, the current visual-based architecture will be introduced and according to a set of requirements, two baseline methods will be chosen. Secondly, the shortcomings of the current architectures will be studied by means of several representative experiments on data sets that are provided by Ericsson AB. Thirdly, an alternative visual-inertial based architecture will be proposed which improves the current architecture in two steps. In the first step, inertial information will be integrated in the pipeline. In the second step, the dense 3D modeling module will be accelerated by using a form of image-segmentation. Fourth and finally, the proposed architecture will be tested in end-to-end benchmark and lab-experiments. These results will be compared against the baseline methods and used to establish whether the proposed architecture indeed mitigates the current shortcomings.

Following this description, the thesis is divided in the following chapters:

- **Chapter 2** will introduce the currently used architecture. It will show how the current architecture frames the problem of dense 3D modeling and how this is solved. Furthermore, two suitable architectures will be chosen which can serve as a baseline method.

- **Chapter 3** showcases the shortcomings of the current architectures by means of several representative examples. Based on these shortcomings, the main problem of this thesis is formulated where the goal is to mitigate these limitations.
- **Chapter 4** shall introduce the proposed architecture which includes inertial information in the pipeline for 3D model generation. It will treat how the inertial information is included and what a suitable architecture is for the use-case of remote inspection.
- **Chapter 5** introduces further improvements with the purpose of optimizing the scalability of the dense 3D modeling module. It will derive the proposed method to achieve this and show the performance in several benchmark experiments.
- **Chapter 6** will test the complete system and compare it against the baseline methods. Two types of experiments are run. First, benchmark experiments are run. Next, a lab set-up is created to run end-to-end experiments to test the proposed method on representative data sets for remote inspection.
- **Chapter 7** concludes the thesis and presents directions for future research.

Chapter 2

Visual Based 3D Modeling

Visual-based 3D modeling is the process of transforming visual data into 3D models. In the context of this work, a 3D model is defined as a dense 3D point cloud representation. It can be seen as an inverse problem where the goal is to recover a 3D structure of a captured scene when only its 2D projections are given. This is visualized in Figure 2-1.

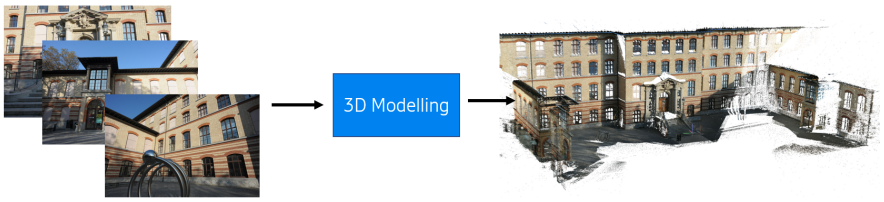

Figure 2-1: The inverse problem in 3D reconstruction - given are 2D projections (images) and the goal is to recover the 3D structure (right) from this information.

In this setting, there are two unknowns that need to be solved - the poses of the input images and the 3D structure. This problem is tackled in two distinct stages defined as sparse and dense 3D modeling which is shown in Figure 2-2.

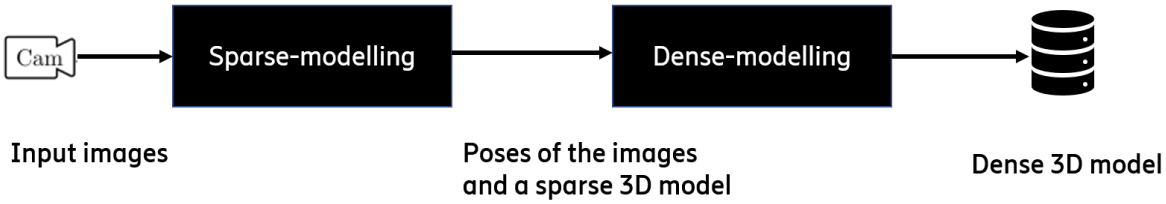

Figure 2-2: Overview of a typical 3D modeling architecture. A dense 3D point cloud model is obtained by sparse 3D modeling (left) and dense 3D modeling (right).

Sparse 3D modeling uses a set of input images and computes two sets: the set of poses or 6 Degrees Of Freedom (DOF) of each image and a sparse 3D point cloud model. Using this output, the dense modeling module will estimate a dense 3D point cloud model. The dense

3D point cloud model is the final output of the architecture. In the following two sections, the sparse and dense modeling modules will be treated more in depth.

2-1 Sparse 3D modeling

In sparse 3D modeling, the input is a set of N images defined as $\mathcal{I}$

$$\mathcal{I} := \{\mathbf{I}_j \mid 0 \leq j \leq N-1\}, \tag{2-1}$$

where an image $\mathbf{I}_j \in \mathbb{R}^{w \times h \times 3}$ is defined by the three dimensional tensor where w and h are the image width and height, respectively. The goal is to estimate a set of corresponding poses $\mathcal{T}$ for each image and a sparse 3D point cloud model $\mathcal{X}_s$ containing M_s points where the set of poses are defined as

$$\mathcal{T} := \{\mathbf{T}_j \mid 0 \leq j \leq N-1\}. \tag{2-2}$$

Each pose is represented by a homogeneous rigid body transformation

$$\mathbf{T}_j = \begin{pmatrix} \mathbf{R}_j & \mathbf{t}_j \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \in SE(3), \tag{2-3}$$

where $\mathbf{R}_j \in SO(3)$ is the rotation matrix and $\mathbf{t}_j \in \mathbb{R}^3$ the translation vector. In this notation it holds that $SO(3)$ and $SE(3)$ are the 3D Special Orthogonal (SO) and Special Euclidean (SE) group, respectively. The sparse 3D point cloud model is defined as

$$\mathcal{X}_s := \{\mathbf{X}_i \mid 0 \leq i \leq M_s - 1\}, \tag{2-4}$$

where each point $\mathbf{X}_i \in \mathbb{R}^3$ represents a point in 3D. From this definition it can be concluded that two sets have to be estimated simultaneously, the poses and 3D structure. This makes the problem inherently difficult to solve. When the poses are known, it is relatively simple to solve for the 3D point cloud via triangulation methods [21]. In the opposite situation, when the 3D point cloud is given, the poses can be obtained by solving the Perspective-n-Points problem where popular approaches are described in [27, 45].

In the computer-vision community, the dual problem of solving for poses and 3D structure simultaneously is typically defined as Structure-from-Motion [41]. The key-driver to solve this problem is by means of visual correspondence search [41]. Visual correspondence search is the process of recognizing locations in different images that correspond to the same scene. Such locations need to be robust against changing viewpoints, scale or lightening conditions [33]. Typically, these locations correspond to high-intensity gradient areas such as corners or lines. An example of extracted key-points is shown in Figure 2-3.

Figure 2-3: Image of the statue Ignatius obtained from the Tanks-and-Temples data set [25]. A total of 2500 ORB [36] key-points (blue) are extracted.

In Figure 2-3 it can be clearly seen that features are extracted near edges (windows of the building) or highly textured areas (flowers). Using these key-points, it becomes possible to extract a feature vector which can be seen as an unique identifier of the key-point. Repeating this feature extraction process for each image in the input set, a set of extracted features for each image is obtained.

Now that a set of extracted features is available for each image in the set, it becomes possible to match them. This is the core of visual-correspondence search where the purpose is to determine where key-points are co-observed in the input image set. This yields a set of correspondences in the form of a connected graph. An example of such graph is shown in Figure 2-4 for two images where matched features are indicated by connected blue lines.

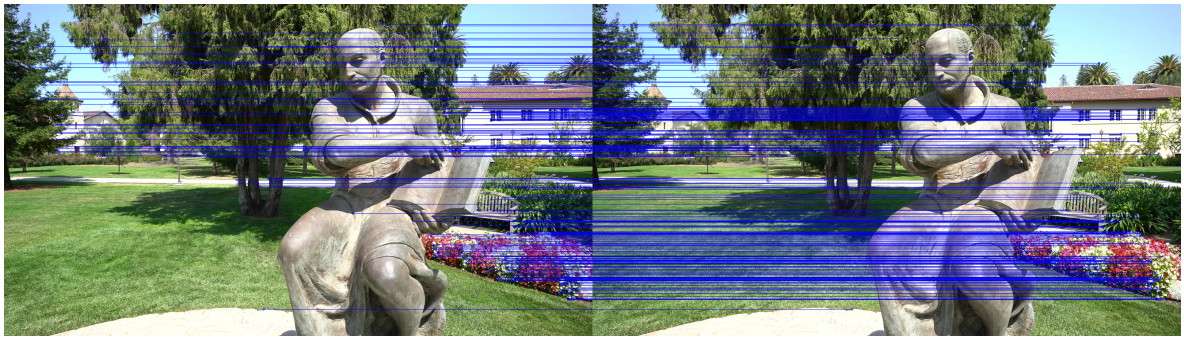

Figure 2-4: Matched features (blue lines) in two images of the Ignatius data set from the Tanks-and-Temples benchmark [25]. By searching for key-points that share similar feature-descriptors it becomes possible to obtain pairs of key-points that are part of the same object but observed in different images.

Such a graph spans the whole input image set and contains an edge between each set of key-points, observed in different images, that are matched. Many different types of feature extraction and matching approaches exist and, for a more complete review, the reader is referred to [26, 35]. Many of the extracted matches between two images are faulty, meaning that they do not map to the same scene points. To filter out the erroneous matches, the feature matching process is often combined with a geometric verification step. In this step, a mapping is estimated between two images based on the matched features using projective geometry. Then, if the obtained mapping projects a sufficient number of features between

two images, the matches are geometrically verified [38]. With the set of matched key-points, the dual problem of estimating the set of poses $\mathcal{T}$ and the sparse 3D point cloud $\mathcal{X}_s$ from Equation (2-2) and (2-4), respectively can be framed. For each unique and new match

$$\{\mathbf{x}_{in} \leftrightarrow \mathbf{x}_{im}\} \quad n \neq m, \tag{2-5}$$

a variable $\mathbf{X}_i \in \mathcal{X}_s$ is created. In this notation, a key-point $\mathbf{x}_{ij}$ is coupled to a 3D point i and observed in pose j. This process is shown in Figure 2-5.

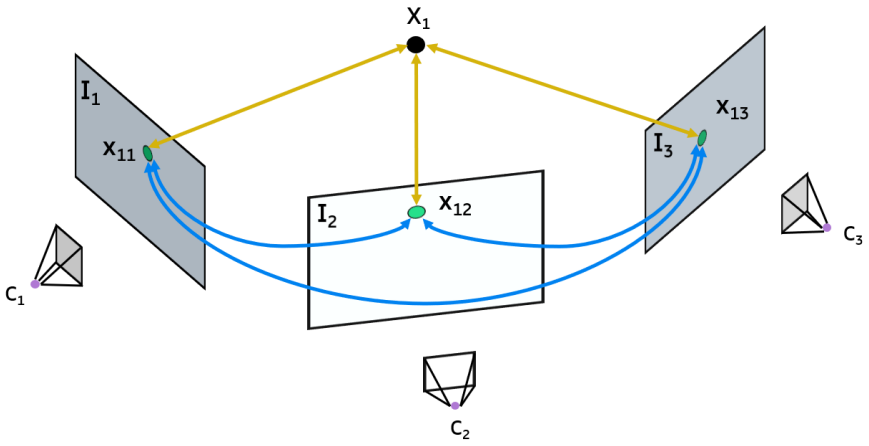

Figure 2-5: Matching key-points $\mathbf{x}_{ij}$ across different images $\mathbf{I}_j$ creates a new variable $\mathbf{X}_i$ for each connected graph. This example shows three images with three matched (blue edges) key-points (green dots) with one new scene point. Each key-point is associated with a scene point (yellow edges). The result is a correspondence graph.

The variable point $\mathbf{X}_i$ can be projected back into the images in which it was observed as a function of its estimated location and the poses of the cameras. This results in a projected point $\hat{\mathbf{x}}_{ij}$ and is visualized in Figure 2-6.

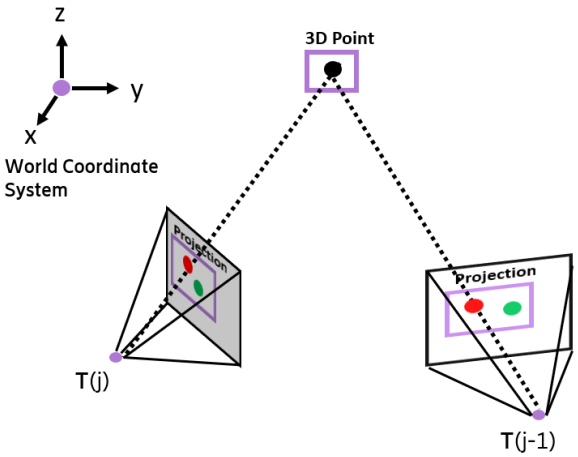

Figure 2-6: Two images with pose j and $j-1$ showing the extracted key-points (green) and re-projected 3D points (red).

From Figure 2-6 it can be concluded that such a projection almost never exactly aligns with the location of the key-point in an image. This may be caused due to a faulty match, noise or pose errors. The misalignment between the projected scene point $\hat{\mathbf{x}}_{ij}$ and extracted key-point $\mathbf{x}_{ij}$ is shown in Figure 2-7.

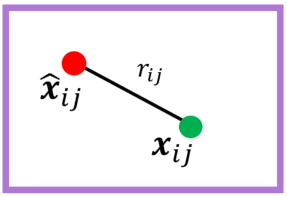

Figure 2-7: Projection of 3D point (red) with extracted feature (green). The difference between the two points yields a residual vector r_{ij} (black edge).

Depending on the exact projection, a residual can be defined as a function of the projection of the 3D scene point $\mathbf{X}_i$ and pose $\mathbf{T}_j$

$$\mathbf{r}_{\mathbf{x}_{ij}} = \hat{\mathbf{x}}_{ij} - \mathbf{x}_{ij}. \tag{2-6}$$

Such residual corresponds to the Euclidean distance between an extracted key-point $\mathbf{x}_{ij}$ and a projected 3D point $\hat{\mathbf{x}}_{ij}$. With a given camera projection model this results in

$$\mathbf{r}_{\mathbf{x}_{ij}} = f(\mathbf{K}, \mathbf{T}_j, \mathbf{X}_i) - \mathbf{x}_{ij}, \tag{2-7}$$

where $\mathbf{K}$ is the intrinsic camera matrix which projects a reconstructed scene point $\mathbf{X}_i$, defined in the camera coordinate system of a pose j, to an observation in the image plane as described in Appendix A. The resulting projection function $f(\cdot)$ is then defined by a mapping that is nonlinear in the pose and 3D scene parameters [44]

$$f(\mathbf{K}, \mathbf{T}_j, \mathbf{X}_i) := \mathbf{K} \begin{pmatrix} \mathbf{R}_j & \mathbf{t}_j \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{X}_i \\ 1 \end{pmatrix}. \tag{2-8}$$

Next, all residuals are collected by summing over all M_s points and N images. Solving the problem can then be defined as minimizing a robustified non-linear least-squares problem of the form

$$\min_{\mathcal{T},\mathcal{X}_s} \sum_{j=0}^{N-1} \sum_{i=0}^{M_s-1} \rho\left(\mathbf{r}^2_{\mathbf{x}_{ij}}\right), \tag{2-9}$$

where $\rho(\cdot)$ is a robust loss function which down weighs the effect of large residuals [44]. This is necessary since the problem can contain gross outliers due to for example faulty matches. Typically, this is chosen to be a Huber's loss function [22]. The output of the sparse 3D modeling for the Tanks and Temples example is shown in Figure 2-8.

Figure 2-8: Collection of input images with known poses $\mathcal{T}$ and sparse 3D point cloud model $\mathcal{X}_s$. The data set was obtained from the Ignatius data set of the Tanks-and-Temples benchmark [36].

To solve Equation (2-9), a good initial estimate is required due to the non-convexity of the problem [29]. Solving directly will generally not result in good reconstructions. To circumvent this, most state-of-the-art architectures use an incremental approach. Starting from an initial pose $\mathbf{T}_0$, new poses are registered sequentially and the sparse 3D point cloud model is expanded with each new pose. Each new pose is typically added in two steps where the first step infers the pose based on the current point cloud model and the second step uses the newly registered pose to triangulate new 3D points. After some threshold has been reached, the current estimate can then be refined by solving Equation (2-9) where this refinement step is typically denoted as bundle-adjustment. There are many variations in the literature available that have different protocols for initialization, registering new images, bundle-adjustment and determining which view to add next. For a comprehensive overview of these approaches the reader is referred to [4, 29, 38].

After solving Equation (2-9) for the set $\mathcal{T}$ and $\mathcal{X}_s$, the sparse 3D model can be densified in the final step which will be treated next.

2-2 Dense 3D modeling

The second and final step in 3D modeling is the densification step. It uses the computed poses and sparse 3D model from the previous module and densifies the sparse 3D model. In the previous section, it was explained that the sparse 3D model was generated by working on a sparse set of pixels in each image: key-points. This is exactly the main difference between sparse and dense 3D modeling. Instead of operating on sparse pixel locations in each image, dense 3D modeling directly operates on the pixel level. The main goal is to obtain a dense 3D point cloud representation $\mathcal{X}_d$ with M_d points. The point cloud model is defined as

$$\mathcal{X}_d := \{\mathbf{X}_i \mid 0 \leq i \leq M_d - 1\}, \tag{2-10}$$

where each point $\mathbf{X}_i$ represents a point in 3D. The dense 3D model is created through a process defined as Multi View Stereo (MVS) and fusion. In this process there are three steps

to infer the dense 3D point cloud model. First, it infers a set of depth maps $\mathbf{D}_j \in \mathcal{D}$ for each input image where $\mathbf{D}_j \in \mathbb{R}^{w \times h}$. In this notation it holds that w and h are defined as the image width and height, respectively and each element in the depth map contains a depth value d. Secondly, it uses inverse projection to obtain a point cloud model from each depth map. Thirdly, it fuses the independent point cloud estimates into one consistent dense 3D point cloud model $\mathcal{X}_d$.

As explained, the first step is to infer a depth map $\mathbf{D}_j \in \mathcal{D}$ for each image $\mathbf{I}_j$ in the input set $\mathcal{I}$. Besides estimating a depth value d for each pixel, in most state-of-the-art applications, this is combined with the estimation of a normal map to yield more accurate results. In this scenario, each pixel location also contains a unity normal vector

$$\mathbf{n} = (n_x, n_y, n_z)^T, \quad ||\mathbf{n}||_2 = 1. \tag{2-11}$$

Each pixel is then represented by an oriented plane which has a depth-normal value. The problem of estimating a single depth-normal map is then defined as inferring the depth map $\mathbf{D}_j \in \mathcal{D}$ and normal map $\mathbf{N}_j \in \mathcal{N}$ for a reference image $\mathbf{I}_{ref}$ given the source images $\mathbf{I}_{src} = \{\mathcal{I} \setminus \mathbf{I}_{ref}\}$. In this notation a normal map is represented by a three dimensional tensor $\mathbf{N}_j \in \mathbb{R}^{w \times h \times 3}$ since each entry contains a normal vector. Such combination of depth and normal is defined as the hypothesis for a pixel. To perform inference, a measure of likelihood is needed which describes how likely a certain hypothesis is given the set of source images that are available.

Such likelihood is typically defined as photo-consistency $\rho(\cdot)$ and it yields a matching cost for a given depth and normal hypothesis in one other view. This is shown in Figure 2-9.

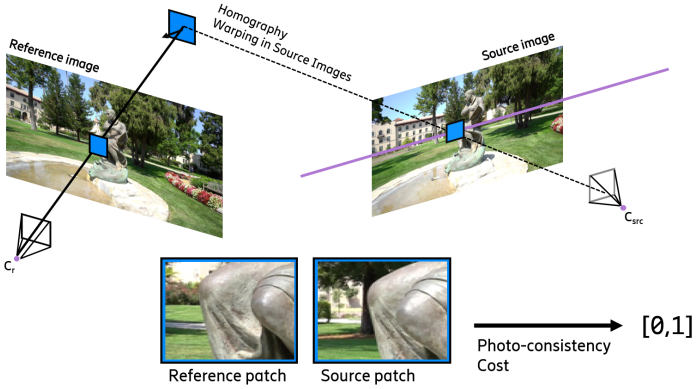

Figure 2-9: The matching cost for the depth-normal hypothesis of a pixel is computed by comparing two patches in the reference and source image. A reference image (left) extracts and projects a patch in a source image (right). This yields a reference and source patch which are compared to compute a photo-consistency cost (bottom).

In Figure 2-9, homography warping is defined as the projection of an oriented (slanted) plane from the reference image to a source image. Photo-consistency returns a value between 0 and 1 where a higher value indicates a higher similarity score between two patches. The score is computed by using a rectangular support region Ω around the current pixel. This yields a rectangular patch of $|\Omega|$ pixels for the reference and source view, respectively. Multiple photo-consistency measures exist but the most state-of-the-art algorithms such as Gipuma

[18], COLMAP [38] or PMVS2 [16] use the zero-mean Normalized Cross Correlation (NCC). For a more complete review of different type of photo-consistency costs the reader is referred to the work of C. Hernández et. al [17]. Prior to computing the NCC, most implementations convert color images into greyscale intensity images. The NCC can then be computed by taking the dot product between the vectorized reference and source patches, with subtracted mean and normalizing them with the standard deviation

$$\rho_{NCC}(f,g) = \frac{(f-\bar{f})\cdot(g-\bar{g})}{\sigma_f \sigma_g} \in [-1,1], \tag{2-12}$$

where $f \in \mathbb{R}^{|\Omega|}$ and $g \in \mathbb{R}^{|\Omega|}$ are the vectorized patches of the reference and source view, respectively. These vectors contain the pixel values of the support region Ω. In this notation $\bar{f}$ and $\bar{g}$ denote the average pixel intensity value in both patches. Furthermore, σ_f and σ_g are the standard deviations of both patches. Although the NCC yields a score between -1 and 1, where a higher score indicates a better match, it can be scaled such that it returns a value between 0 and 1. A more clear example of how the vectors f and g are extracted is depicted in Figure 2-10.

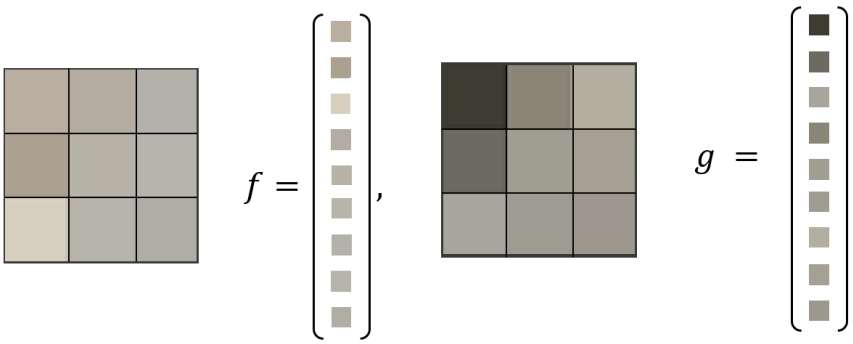

Figure 2-10: A reference patch with a support region Ω of 3×3 pixels and vectorized patch f (left) and a source patch and vectorized patch g (right). Both vectors contain the greyscale intensities of the pixel locations.

When the hypothesis is evaluated for different depth-normal values, a cost-volume can be build. This is shown in Figure 2-11.

The matching cost volume is constructed by aggregating the costs from the set of source images. If one would use all the source images to obtain the matching cost, the result could become ambiguous. This is a result of the occlusion problem. The occlusion problem is caused by including source images that are not observing the specific part of the scene of the reference patch, that is being matched with the source views. This is depicted in Figure 2-12.

When these images are included, they obscure the cost-volume since they map to incorrect image regions that are not visible in the reference image. In the example of Figure 2-12, the two images are taken at two opposing locations of the statue. Then, when the middle part of the statue (red patch) is given the correct depth-normal hypothesis, it can be projected to the source view. In this scenario, this will map to the back of the statue which will yield a completely different image patch. Since this patch will be different from the reference patch, even at the correct depth-normal hypothesis, it will return a low photo-consistency

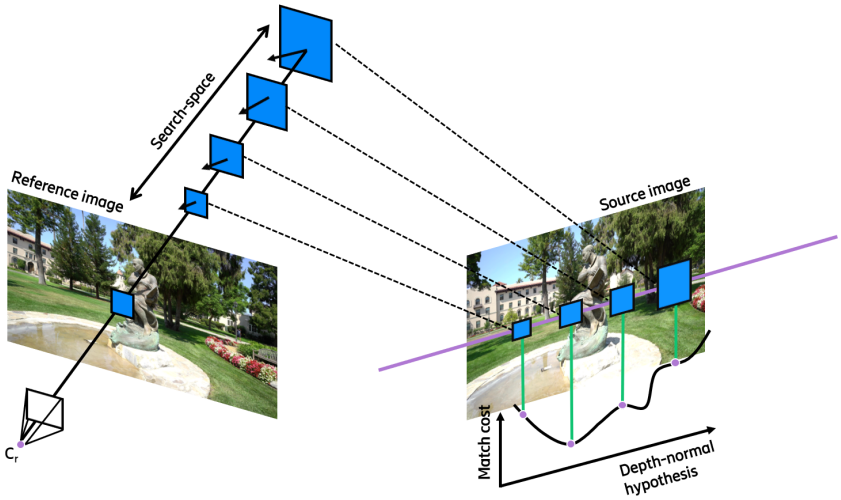

Figure 2-11: Matching cost evaluated for different depth-normal hypothesis for a single pixel. Exploring the search-space for a single pixel yields a cost-volume.

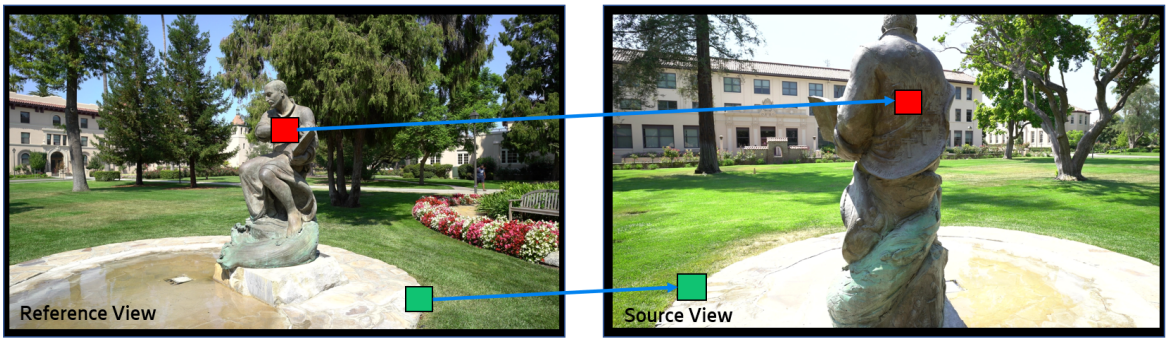

Figure 2-12: The occlusion problem obscures the photo-consistency cost in MVS. In this example the reference patches (left) are given the true depth-normal hypothesis and are projected in the source image (right).

score. Projecting a non-occluded part of the scene (green patch), when it is given the correct depth-normal hypothesis, will not result in low photo-consistency costs.

Typically, to circumvent the occlusion problem, most state-of-the-art MVS algorithms also propose methods to select a set of source images which are likely observing the same part of the scene as the reference patch. For example, COLMAP [38] adopts a pixel-wise view selection scheme that is directly integrated in the depth-normal inference. Having devised an approach to handle the (partial) occlusions in source images, it becomes possible to solve for the correct depth-normal hypothesis. This boils down to choosing the hypothesis with the lowest aggregated matching cost in non-occluded source views.

However, this problem quickly becomes intractable to solve since computing a photo-consistency score in each reference image is costly and the number of pixels is very large. A typical smartphone camera is capable of shooting 12 Mega Pixels (MP) pictures. This means that there are 12 million depth-normal values that need to be estimated for only one image. Since each depth-normal has 4 dimensions (a depth value and three normal-vector components) the

problem space is $48 \cdot 10^6$ for one image. Solving this problem quickly becomes intractable and as such, very efficient algorithms are required to handle the large problem-space.

The current state-of-the-art algorithm that overcomes this problem is defined as PatchMatch and was originated by M. Bleyer et al. [5]. The algorithm is able to tackle the large problem space by leveraging structured region information within a scene. Structured region information means that fairly large groups of pixels share similar depth-normal values since they belong to the same or similar objects. The algorithm exploits this fact by operating an iterative scheme that works in four steps. First, it random initializes the depth-normal map. It assumes that most random guesses are completely off but a few will be close to the correct depth-normal value. Then, it allows those good estimates to spread to neighboring pixels in a propagation step. Next, it refines the current depth-normal map and repeats these two steps until the depth-normal map converges to a stable state. Finally, the depth-normal map is filtered to remove spurious estimates or outliers. An overview of this procedure is given in Figure 2-13.

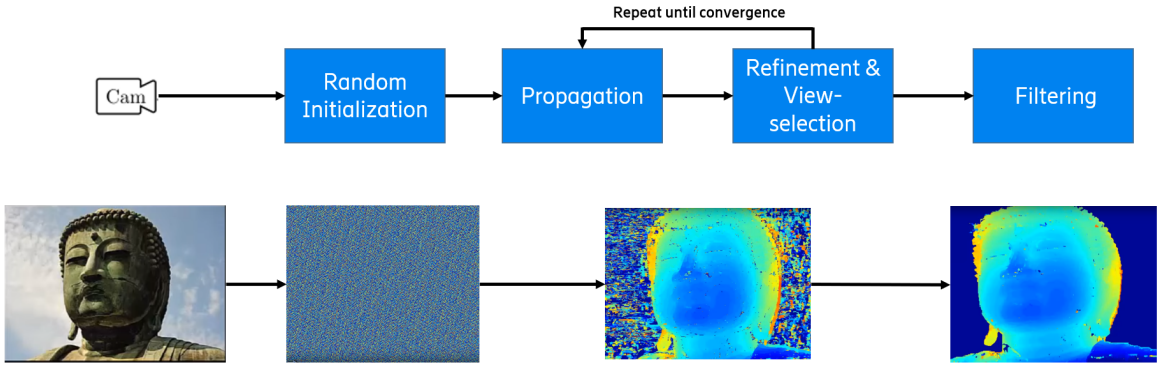

Figure 2-13: Overview of the PatchMatch optimization scheme (top) and example flow (bottom). The optimization scheme operates in four steps which are shown from left to right. First, a depth-normal image is random initialized. Next, the algorithm iterates between propagation and refinement steps until convergence. Finally, it removes spurious estimates in a filtering step to obtain the final result.

In MVS, this procedure is repeated for each image $\mathbf{I}_j \in \mathcal{I}$. The final step is to use inverse projection to obtain a point cloud model of each depth map. Those independent point cloud models are then fused into a consistent dense 3D point cloud model. Several variations exist for the fusion step. However, most approaches search for consistent depth-normal values by performing a cross-consistency check across different images and requiring the values to be consistent with a minimum number of views. The final depth-normal point in this case is then the combination of all the independent estimates. For example, COLMAP [38] uses the median value of the independent estimates to obtain the final point. The end result of the fusion process is the final output in the 3D modeling architecture and is shown in Figure 2-14 for the Tanks-and-Temples example.

Several different variations of dense 3D modeling exist and in this work a choice needs to be made to determine what a suitable architecture is. This will be treated in Section 2-3.

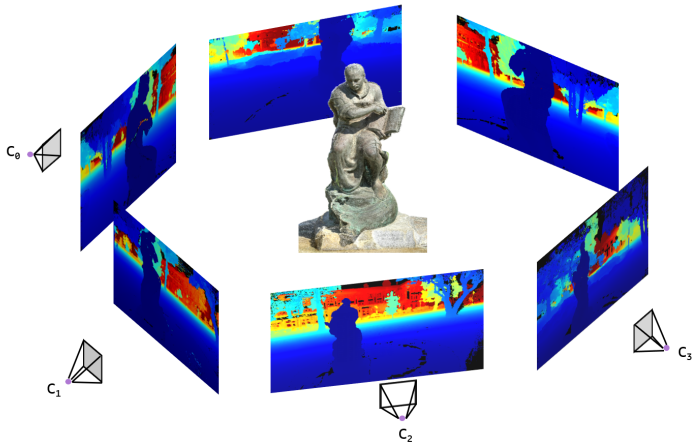

Figure 2-14: Depth-normal maps of all input images are fused into a dense 3D model. The depth-normal maps of this example were computed using the Ignatius data set from the Tanks-and-Temples benchmark [25].

2-3 Choice in Baseline Systems

In this work, visual-based 3D modeling architectures are considered the baseline method. Such an architecture is capable of transforming a set of visual input data (i.e. images) into a dense 3D point cloud model. Being the baseline approach, this architecture will be compared against an alternative proposed architecture. That is why it is important that the visual-based 3D modeling architecture is capable of delivering state-of-the-art results. For this reason, two systems are chosen: PIX4D and COLMAP [1, 38]. PIX4D and COLMAP represent the industrial and academic standard, respectively.

PIX4D, is a widely used commercial 3D modeling architecture. It has been shown to deliver state-of-the-art results on several benchmarks including the Tanks-and-Temples benchmark [25]. It is capable of transforming a set of visual input data with attached metadata such as Global Positioning System (GPS) information into dense 3D point cloud models with the correct geometric scale, thereby offering an end-to-end reconstruction system.

The COLMAP [38] system is chosen as the second baseline method. COLMAP is a general purpose, academic 3D modeling architecture which provides an end-to-end solution. It is widely studied by the research community on multiple benchmarks such as the Tanks-and-Temples [25] and ETH3D [40]. As such, its state-of-the-art performance has been proven across a wide variety of settings.

Although many high-performing learning-based methods exist that are academically interesting, they need a great quantity of domain specific training to perform well [49]. As such, these approaches are not feasible for real-world applications which is the focus of this work. Popular examples of learning-based approaches include [31, 48, 24].

Chapter 3

Problem Formulation

In this chapter, the shortcomings of the baseline methods introduced in Section 2-3 will be studied. First, several experiments on real data sets will be performed both with PIX4D (the commonly used industrial solution) and COLMAP (one of the most widely studied open-source methods). Using these experiments, the set of issues will be specified. With the identified issues, the research problem for this thesis work is formulated with the general purpose of mitigating the typical issues.

The experiments are run on a real recording of a cellular site provided by Ericsson AB. The visual data is collected with a DJI Phantom drone which is capable of capturing 5472×3648 RGB images. With each image, the DJI Phantom drone receives Global Positioning System (GPS) information - which includes latitude, longitude and altitude information and the yaw, pitch and roll angles. These angles define the orientation of the drone during the image acquisition process. Multiple experiments are available from this site. In this work, two experiments are chosen since they are particularly interesting. The first experiment is a standard flying pattern with a downward facing camera. This pattern is used to capture a complete 3D model of the whole site and is defined as the 3D Model data set. The second experiment, which has the goal of capturing a piece of the equipment with an upward facing camera, is defined as the Level 1 up-look set.

The first experiment is representative of a standard setting which can be used to study the general performance whilst the second experiment contains a challenging visual data set. This is caused by the large portion of sampled sky-pixels due to the upward facing camera.

In the experiments, two sets of reconstructions are computed using PIX4D and COLMAP, respectively with standard settings unless stated otherwise. Furthermore, to perform the computations, the computing platform described in Appendix B is used.

3-1 The 3D model data set

In this experiment, two 3D point cloud models are computed using PIX4D and COLMAP. In this setting, the goal is to capture a complete 3D point cloud model of the scene and its

surroundings. To achieve this, the technician operates a drone and flies a circular pattern at different heights. Then, he or she flies several columns at different locations to connect the circular runs at different heights. The resulting flight-plan is visualized in Figure 3-1.

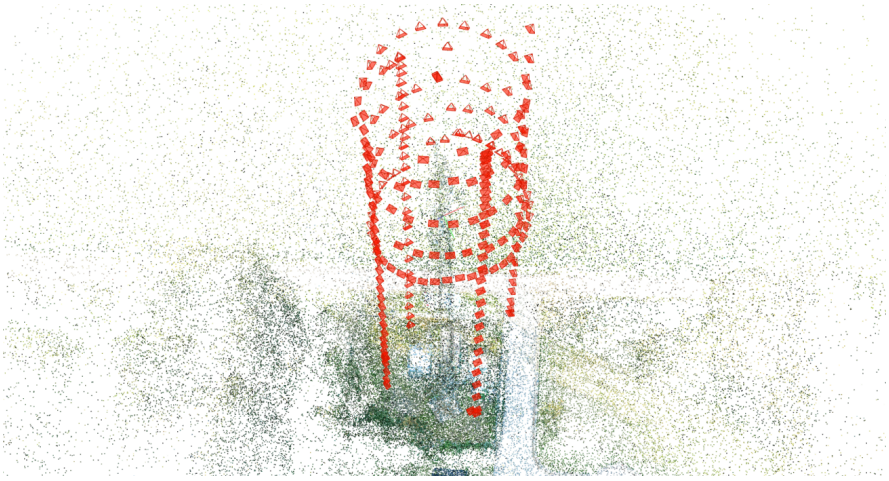

Figure 3-1: Overview showing the flight-plan of the 3D model experiment. Camera poses are shown as a red projection objects. The figure was obtained from COLMAP's sparse 3D modeling output.

Using the standard settings in both methods, the two dense 3D point cloud models are computed. The resulting models are shown in Figure 3-2.

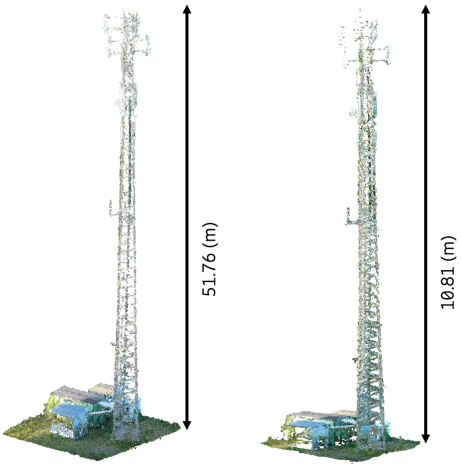

Figure 3-2: PIX4D (left) and COLMAP (right) reconstructed 1,346,612 and 921,634 3D points respectively. Both 3D point clouds are computed using the 3D model experiment data set. For each reconstruction, the cellular tower's height is computed in meters.

From Figure 3-2 it can be concluded that both 3D point cloud models look quite comparable. The large difference in number of points could be caused by different filtering settings in both methods. However, there is one important difference. PIX4D is able to read the attached metadata of the DJI drone. Leveraging this information, the algorithm is able to deduce

the correct geometric scale. This results in a 3D point cloud model which has accurate dimensions up to its accuracy of 2 (cm). COLMAP is not able to do so and the implications are significant. Although visually very similar, the dimensions in COLMAP's model are inaccurate. As means of an example, since there is no ground-truth reconstruction available, the computed height of both towers is also shown in Figure 3-2. In Figure 3-2, it can be seen that PIX4D's reconstruction has computed a cellular tower that has a height of 51.76 (m) which is accurate up to 2 (cm). However, COLMAP's reconstruction yielded a tower that has a height of 10.81 (m) which is clearly inaccurate. This is caused by the inability to leverage the available metadata information attached to the images.

It is important to note that if there is no (accurate) metadata available, both methods would be unable to recover the accurate geometric scale. This is problematic in the context of remote inspection since the technicians need to be able to measure pieces of the equipment and parts of the structure to perform their tasks. If this is not possible, the use-case of the 3D point cloud models is significantly hindered. The absence of metadata in the form of a GPS signal is not an unusual situation since there are many cellular sites located at remote instances where the coverage is not reliable. Besides lack of GPS information due to a remote location, alternatively, it can be required to create indoor digital models of the sites. For example, indoor-equipment such as battery cases or racks with base-bands are sometimes scanned as well. These pieces of equipment are located indoor and therefore, the acquired GPS information may not be reliable. By relying on the metadata only, it is not possible to extract 3D point cloud models with accurate geometric scale in these settings.

3-2 The level 1 up-look data set

The second set of reconstructions is computed using the challenging up-look experiment. In this scenario, the goal is to capture the antenna equipment from the bottom with a camera facing upwards. This is important since the technician is often interested in inspecting the underside of the antenna equipment, where the connectors are located, which are a crucial piece of equipment. This was also depicted in Figure 1-2. In this setting, the drone is also flying a circular pattern around the tower similar to the previous case. However, the drone is now flying significantly closer to the tower to capture the equipment more in detail. This is visualized in Figure 3-3.

Feeding the exact same input data to PIX4D and COLMAP and using the standard settings, both methods failed to reconstruct the equipment in this scenario and crashed in the sparse 3D modeling phase. This is attributed to the fact that the large portion of indiscriminate sky-pixels make the system unstable. Although the settings of PIX4D can not be changed, COLMAP has the possibility of tweaking the parameters. Specifically, the number of extracted features is increased in the first phase. This allows the sparse 3D modeling module of COLMAP to find more accurate correspondences across the images to guide the reconstruction. Using the new settings a reconstruction is obtained and is shown in Figure 3-4.

Notice that since the drone flew closer to the tower, the number of details that are captured is higher than the previous scenario. Especially, the thin structural elements of the tower's framework are reconstructed more accurately. However, since COLMAP is unable to use the

Figure 3-3: The Level 1 up-look run. The drone has an upward oriented camera and is flying a circular pattern around the site. Camera poses are shown as red projection objects. The figure was obtained from COLMAP's sparse 3D modeling output.

attached metadata, the dimensions of the reconstruction are inaccurate. Furthermore, it is noticeable that there is noise present around the edges of the reconstruction. The blue colored noise originates from the sky-pixels and is caused by errors in the depth-normal hypothesis generated by propagating the depth-normal values from the structural framework elements to the sky-pixels. This can be seen more clearly when one of the depth-normal map generated by COLMAP is analysed. This is shown in Figure 3-5.

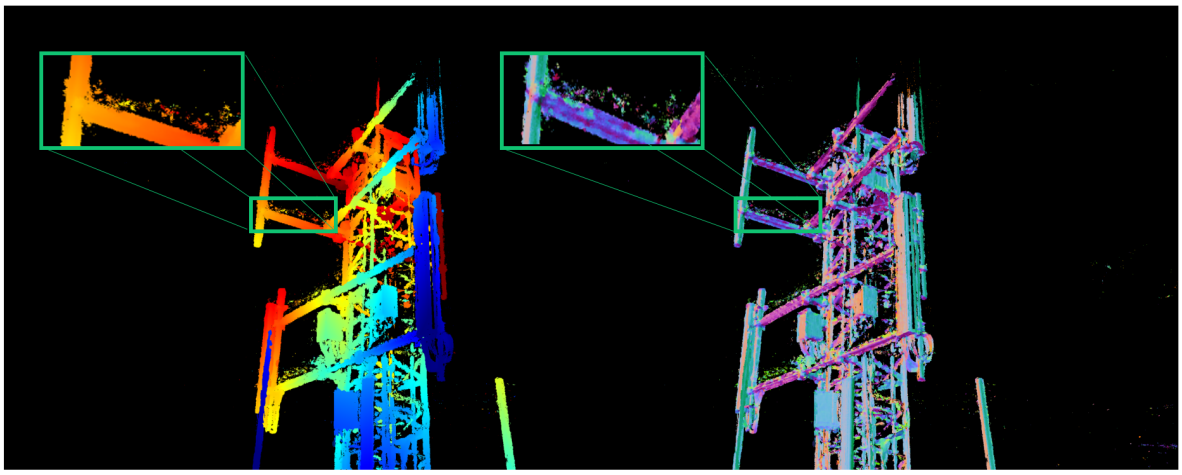

Figure 3-5: The depth-normal map of COLMAP for a single image where the depth map is shown on the left and the normal map on the right. Showcasing the noise around thin edges in the structure.

From Figure 3-5 the location of the noise can be clearly seen to focus around the edges of the thin members. Although visually displeasing, it is not problematic for the use-case of remote inspection since it can be filtered out easily in a post-processing step.

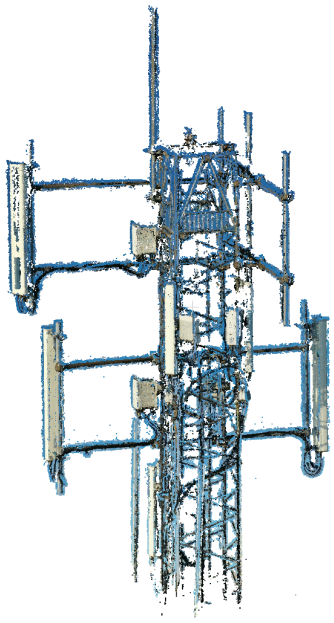

Figure 3-4: The reconstructed dense 3D point cloud model of COLMAP after increasing the number of features. Note that PIX4D failed to compute a reconstruction.

3-3 Defining the problem

Based on the previous experiments several shortcomings were observed. Summarizing, there are three key-shortcomings of the current architecture:

- Inability to directly extract geometric scale.
- Lack of robustness.
- Scalability.

First, there is a geometric scale ambiguity. This ambiguity means that the obtained dense 3D reconstruction is correct up to an unknown scale-factor S if there is no external metadata available. The geometric scale ambiguity is visualized in Figure 3-6.

In the context of remote inspection this is problematic. This stems from the fact that the main use-case of dense 3D point cloud models is the ability to perform measurements on the structure. The current solution as implemented in for example PIX4D solves this problem by relying on external information sources in the form of EXchangeable Image File-format (EXIF) metadata attached to the images. Such metadata contains for example the GPS coordinates of where the image was acquired. Having GPS information attached to the images resolves the scale ambiguity of the 3D point cloud model. The ambiguity is solved since the GPS information provides additional prior information terms on the camera poses. This constraints the reconstruction to only one scale factor. However, some mobile cellular towers are located in remote locations where the GPS signal is absent or inaccurate. Or the visual data is captured indoors which yields the same problems. In the aforementioned

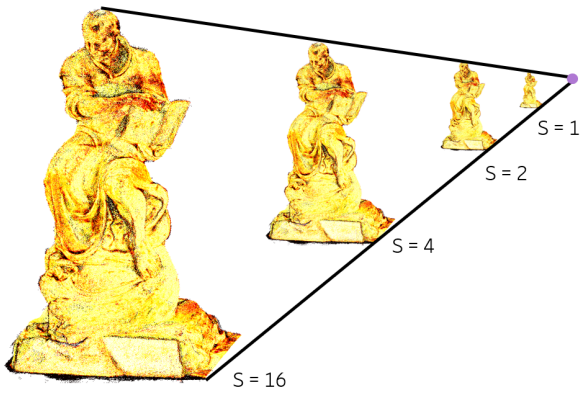

Figure 3-6: Scale ambiguity in classical 3D-modeling where S denotes the 3D model scale.

scenarios, it becomes difficult or even impossible to obtain the accurate geometric scale. Not having accurate geometric scale of the 3D point cloud model renders it almost useless for remote inspection.

Secondly, there is a lack of robustness in challenging visual situations. As mentioned in the introduction, the current architecture relies solely on visual information (excluding the GPS signal for scale-extraction). This means that the quality of the final 3D point cloud model is entirely dependent on the quality of the visual input data. Whenever a challenging visual environment is encountered, such as bad-lightening conditions, homogeneous surfaces or motion blur, the quality degrades. More so, the system may even fail to reconstruct anything at all as was showed in the comparison of PIX4D and COLMAP on the Level 1 up-look data set. This is problematic as well since bad-lightening or homogeneous image regions are encountered frequently in site monitoring. Two examples are shown in Figure 3-7.

Figure 3-7: Challenging situations in remote inspection scenarios. Bad lightening (left) due to a sun-facing camera and homogeneous sky image-regions (right). Images were obtained from the recording provided by Ericsson AB.

Thirdly, the whole process of transforming the visual data into usable 3D point cloud models is computationally very expensive. Although not considered directly in these experiments, the 3D modeling algorithms scale very poorly with an increasing size of input data. In the context of remote inspection it is important to obtain the dense 3D point cloud models in a

time that allows the visiting technician to observe the results whilst still being present at the site. This is important since the data acquisition process is non-trivial and as such may have failed and require an additional run. If the technician is able to assess this immediately at the site, he or she is not required to visit the site again. With the current architecture, obtaining the dense 3D point cloud models within a time window that allow this, is not possible.

In this research, an alternative architecture is proposed which is able to mitigate the current shortcomings. It is required that the alternative architecture is capable of computing dense 3D models with comparable quality as the current solutions. Since PIX4D is capable of computing reconstructions with 2 (cm) accuracy [1], this will be used as the minimum requirement. In the following two chapters the alternative architecture will be introduced. After this, it will be tested and compared against the baseline methods: PIX4D and COLMAP.

Chapter 4

Visual-inertial Based 3D Modeling

Over the recent years, the topic of sparse 3D map and pose reconstruction has gained increasing attention from outside the computer vision community in the field of Simultaneous Localization and Mapping (SLAM). The topic of SLAM is mostly studied by the robotics community where the goal is to reconstruct a sparse 3D map and the pose of the system in real-time to perform for example collision avoidance and path-planning. Essentially, SLAM and Structure from Motion (SfM) try to solve the same problem of computing a sparse 3D map and set of poses. However, SLAM is mostly interested in obtaining efficient and online results where the sparse output is the end-goal of the pipeline. On the contrary, SfM typically operates offline and serves as the first step in a dense 3D reconstruction pipeline where the end goal is a dense 3D point cloud model. Since both methods have the same intermediate output and SLAM approaches have shown to yield more favorable results in terms of computational efficiency and accuracy, it would make sense to combine state-of-the-art SLAM algorithms with Multi View Stereo (MVS) to reconstruct dense 3D models. However interesting, this topic has not been studied extensively in the research community.

In this chapter, an architecture that combines SLAM with MVS algorithms will be designed. Besides introducing SLAM algorithms to replace SfM, in this work, additional information sources will be integrated in the system. Specifically, it is proposed to include inertial information. The inclusion of inertial information has the potential to mitigate some of the typical issues that are observed in the current architecture such as the scale ambiguity. Then, the resulting architecture will leverage the additional information source by relying on visual-inertial SLAM algorithms.

In this chapter, first, it will be explained how inertial information is absorbed in the problem structure by deriving the new measurement model. Next, it will be explained how the new measurement model is used to frame the general optimization problem. After this, the new architecture will be introduced where a motivated choice in specific method will be made. Using the presented architecture, a set of reconstructions will be computed using the current academic baseline solution (COLMAP) and the presented system on a benchmark data set. Based on this experiment, further improvements are suggested to further increase the performance of the presented system.

4-1 Deriving the inertial measurement model

The inertial information is included by adding an additional sensor in the form of an Inertial Measurement Unit (IMU). An IMU measures the system's accelerations and angular velocities. This is shown in Figure 4-1.

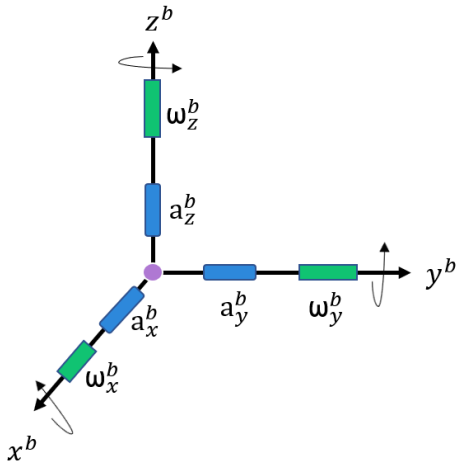

Figure 4-1: IMU measurements over the body-axis (superscript) over the 3 axes (subscript). The origin of the coordinate system coincides with the body centroid of the system. The IMU measures accelerations (blue columns) and angular velocities (green columns).

The IMU sensor is typically described using the measurement-model which is adopted from [13] and defined as

$$\begin{aligned} \tilde{\boldsymbol{\omega}}^b(k) &= \boldsymbol{\omega}^b(k) + \mathbf{b}_g(k) + \boldsymbol{\eta}_g(k), & \boldsymbol{\omega}^b(k) \in \mathbb{R}^3 \\ \tilde{\mathbf{a}}^b(k) &= \mathbf{R}(k)\left(\mathbf{a}^w(k) - \mathbf{g}^w\right) + \mathbf{b}_a(k) + \boldsymbol{\eta}_a(k), & \boldsymbol{a}^b(k) \in \mathbb{R}^3 \end{aligned} \tag{4-1}$$

where $\tilde{\boldsymbol{\omega}}^b(k) \in \mathbb{R}^3$ and $\tilde{\mathbf{a}}^b(k) \in \mathbb{R}^3$ are defined as the measured angular velocity and the acceleration and described in the body frame at time-step k. Besides this, $\boldsymbol{\omega}^b(k) \in \mathbb{R}^3$ and $\mathbf{a}^b(k) \in \mathbb{R}^3$ are the actual angular velocity and acceleration vectors, $\mathbf{g}^w \in \mathbb{R}^3$ is defined as the gravity vector described in the world coordinate system and $\mathbf{R}(k) \in \mathrm{SO}(3)$ is the rotation matrix that transforms vectors defined the world coordinate system to the body frame. In Equation (4-1) it further holds that $\mathbf{b}_g(k) \in \mathbb{R}^3$ and $\mathbf{b}_a(k) \in \mathbb{R}^3$ are the time-varying gyroscopic and acceleration biases, respectively. Furthermore, $\boldsymbol{\eta}_g(k) \in \mathbb{R}^3$ and $\boldsymbol{\eta}_a(k) \in \mathbb{R}^3$ are zero-mean white-noise which evolve the the gyroscopic and acceleration biases via a random-walk noise-model.

The measurements of Equation (4-1) can be related to the pose of the system by using a kinematic motion model of the form

$$\begin{aligned} \dot{\mathbf{R}} &= \mathbf{R}\hat{\boldsymbol{\omega}}, \\ \dot{\mathbf{v}}^w &= \mathbf{a}^w, \\ \dot{\mathbf{p}}^w &= \mathbf{v}^w, \end{aligned} \tag{4-2}$$

where $\mathbf{R} \in \mathrm{SO}(3)$ and $\dot{\mathbf{R}} \in \mathrm{SO}(3)$ are the rotation and the rotation time-derivative, respectively and the rotation matrix transforms coordinates from the world to the body coordinate system. Furthermore, $\hat{\boldsymbol{\omega}} \in \mathfrak{so}(3)$ is the Lie-algebra of the angular velocity vector. The Lie-algebra relates an angular velocity vector to a difference in rotation matrix. Using Lie-algebra to represent differential rotation matrices stems from the fact that rotation matrices have undesirable degrees of freedom which make it an inefficient representation. Essentially, Lie-algebra solves this problem by leveraging the fact that the rotational increments are typically very small. For a more thorough treatment of describing rotations using Lie-algebra, the reader is referred to [46]. Next, $\dot{\mathbf{v}}^w \in \mathbb{R}^3$ and $\mathbf{a}^w \in \mathbb{R}^3$ are defined as the time derivative of the velocity and the acceleration vector, respectively. Finally, $\dot{\mathbf{p}}^w \in \mathbb{R}^3$ and the $\mathbf{v}^w \in \mathbb{R}^3$ are the time-derivative of the position and velocity vector, respectively.

Integrating the kinematic model over a period Δj and relating this to the IMU measurement model, yields a pose $\tilde{\mathbf{T}} = \left(\tilde{\mathbf{R}}, \tilde{\mathbf{p}}\right) \in \mathrm{SE}(3)$ and velocity measurement $\tilde{\mathbf{v}} \in \mathbb{R}^3$ at time j, which is the point at which a new camera image $\mathbf{I}_j$ becomes available. To relate the measurement-model of (4-1) to the kinematic motion model defined in (4-2), the procedure outlined in [13] can be used which yields

$$
\begin{aligned}
\tilde{\mathbf{R}}(j) &= \tilde{\mathbf{R}}(i) \prod_{k=i}^{j-1} \mathrm{Exp}\left(\left(\tilde{\boldsymbol{\omega}}(k) - \mathbf{b}_g(k) - \boldsymbol{\eta}_g(k)\right)\Delta t\right), \\
\tilde{\mathbf{p}}(j) &= \tilde{\mathbf{p}}(i) + \sum_{k=i}^{j-1}\left[\mathbf{v}(k)\Delta t + \frac{1}{2}\mathbf{g}^w \Delta t^2 + \frac{1}{2}\mathbf{R}(k)\left(\tilde{\mathbf{a}}^b(k) - \mathbf{b}_a(k) - \boldsymbol{\eta}_a(k)\right)\Delta t^2\right], \\
\tilde{\mathbf{v}}(j) &= \tilde{\mathbf{v}}(i) + \mathbf{g}^w \Delta t_{ij} + \sum_{k=i}^{j-1} \mathbf{R}(k)\left(\tilde{\mathbf{a}}^b(k) - \mathbf{b}_a(k) - \boldsymbol{\eta}_a(k)\right)\Delta t,
\end{aligned}
\tag{4-3}
$$

where Δt_{ij} is defined as the time between two consecutive images $\mathbf{I}_i$ and $\mathbf{I}_j$ and Δt is defined as the time between consecutive IMU measurements. Furthermore, $\mathrm{Exp}\left(\cdot\right)$ is defined as the exponential map which transforms an element of the Lie-algebra back to its Lie-group

$$
\mathrm{Exp} : \mathfrak{so}(3) \rightarrow \mathrm{SO}(3), \tag{4-4}
$$

where in this context the transformation means that the exponential map will transform an angular velocity vector, assumed constant over a small time increment Δt, in a rotation matrix. It can be concluded that the integration of the inertial terms yields three additional type of variables for each measurement at time k. Specifically, the velocity $\mathbf{v}(k)$, acceleration bias $\mathbf{b}_a(k)$ and gyroscopic bias $\mathbf{b}_g(k)$ need to be estimated as well and are included in the optimization scheme as the sets $\mathcal{V}$ and $\mathcal{B}$, respectively. Typically, these are assumed constant between consecutive images i and j. Starting from Equation (4-3), the measurements are collected in a single, relative positional, velocity and rotation term which yields a compact set of residual terms. For a complete derivation of this term the reader is referred to [13].

4-2 Framing the new sparse 3D modeling problem

The next step is to include this residual term into the optimization problem of (2-9). It is similar to visual-based sparse 3D modeling but in this case there are additional information terms originating from the IMU. This is shown in Figure 4-2.

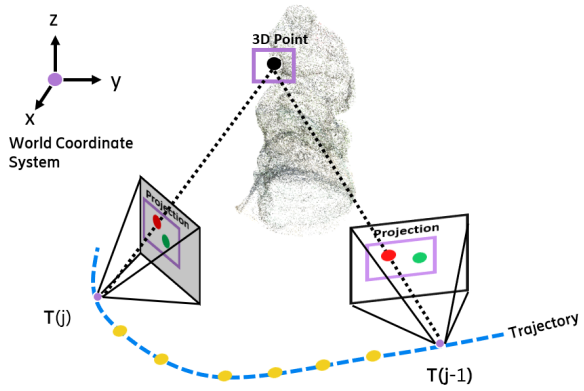

Figure 4-2: Inertial measurements at step k (yellow) between pose at time $j-1$ and j yield extra information terms.

This term originates from the pose and velocity measurements and results in two additional residual terms which are visualized in Figure 4-3.

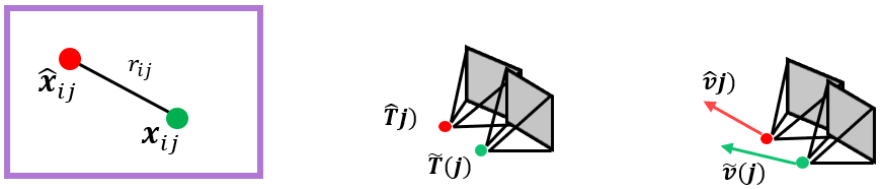

Figure 4-3: Projection of 3D point (red) with extracted feature (green), difference between estimated pose (red) with measured pose (green) and estimated velocity (red) with measured velocity (green) shown on the left, middle and right respectively.

Specifically, there is a set of residuals originating from the 3D landmarks $\mathcal{X}_s$, inertial pose $\mathcal{T}$, velocity information $\mathcal{V}$ and bias estimates $\mathcal{B}$ defined as

$$\begin{aligned} \mathbf{r}_{\mathbf{x}_{ij}} &= \hat{\mathbf{x}}_{ij} - \mathbf{x}_{ij}, \\ \mathbf{r}_{\mathbf{T}_j} &= \hat{\mathbf{T}}_j - \tilde{\mathbf{T}}_j, \\ \mathbf{r}_{\mathbf{v}_j} &= \hat{\mathbf{v}}_j - \tilde{\mathbf{v}}_j. \end{aligned} \tag{4-5}$$

The new residuals are then included in the optimization scheme. Similar to the visual-based case, not the squared loss but a robust loss is minimized to account for outliers. This is typically chosen as Hubers's loss [22] due to its convex properties [44]. The resulting problem can then be defined as

$$\min_{\mathcal{T},\mathcal{X}_s,\mathcal{V},\mathcal{B}} \sum_{j=0}^{N} \sum_{i=0}^{M_s} \rho\left(\mathbf{r}^2_{\mathbf{x}_{ij}}\right) + \sum_{j=0}^{N} \rho\left(\mathbf{r}^2_{\mathbf{T}_j}\right) + \sum_{j=0}^{N} \rho\left(\mathbf{r}^2_{\mathbf{v}_j}\right), \tag{4-6}$$

where the minimizer yields the set of poses $\mathcal{T}$ and sparse 3D model $\mathcal{X}_s$. Note that the inclusion of the inertial information required to add the set of velocities $\mathcal{V}$ and IMU biases $\mathcal{B}$ as additional variables. Depending on the used architecture, probabilistic information may be included in Equation (4-6). In this setting, a weighted least-squares objective of the residuals is minimized where the weights are obtained from the information matrix of the measurements.

4-3 The new architecture

To obtain the new architecture, the structure-from-motion based sparse 3D modeling module is replaced with a visual-inertial based 3D modeling architecture. Both methods yield the same output: a sparse 3D model. In both the current as the proposed architecture, the 3D models are densified using COLMAP's multi-view stereo algorithm. However, this leaves out the question of which visual-inertial sparse 3D modeling architecture is a suitable choice in this work. There are two key-requirements in the desired architecture.

First, the chosen architecture should be able to deliver state-of-the art sparse 3D modeling results that can compete with the baseline methods (i.e. COLMAP and PIX4D). Secondly, the ideal architecture should also have a demonstrated performance on drone systems. This is relevant in the context of remote inspection since Ericsson AB wants to be able to produce dense 3D point cloud models with data captured on board of drone systems. Having an architecture that is proven to work on such systems is advantageous.

For this reason, in this work, the Maplab [37] framework is chosen. Maplab is an open-source visual-inertial mapping framework. It offers a state-of-the-art visual-inertial SLAM front-end: ROVIO [6]. ROVIO is a direct Extended Kalman Filter (EKF) based approach that has been shown to deliver state-of-the-art performance in benchmark experiments on Unmanned Aerial Vehicle (UAV) systems [12]. Besides an integrated front-end, Maplab also offers extensive post-processing tools such as loop-closure and global map optimization. These tools are necessary in order to obtain globally consistent maps. The proposed architecture is shown in Figure 4-4.

To conclude, it is proposed to integrate inertial information in the sparse 3D modeling module by adding an IMU sensor and using the Maplab framework. Next, the resulting sparse 3D model $\mathcal{X}_s$ and set of poses $\mathcal{T}$ are exported to COLMAP. COLMAP takes the output of Maplab and runs its dense 3D modeling module. This yields the dense 3D point cloud model $\mathcal{X}_d$ obtained by means of a visual-inertial based architecture.

4-4 Performance of the visual-inertial method

To showcase the performance, the visual-inertial architecture is compared against one of the baseline methods - COLMAP. The goal in this experiment is to assess the differences in

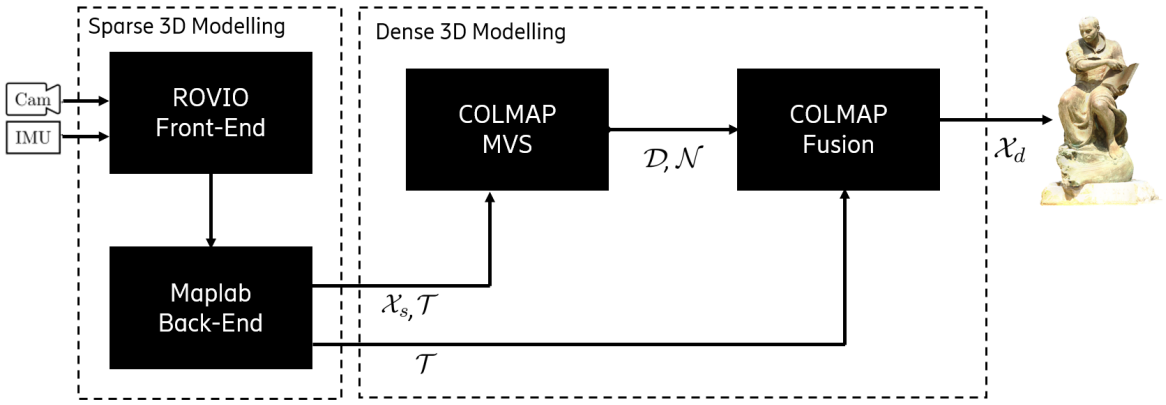

Figure 4-4: The proposed SLAM based architecture computes a dense 3D point cloud model $\mathcal{X}_d$ using Maplab's framework and COLMAP's MVS module.

accuracy and computation time in both methods. To achieve this, the two architectures are tested on the EuRoC Micro Aerial Vehicle (MAV) benchmark [9]. This benchmark has an available ground-truth trajectory and offers synchronized visual-inertial data captured with a small MAV system. Specifically, for this experiment, the machine hall data set MH01 Easy is chosen. The benchmark also offers more challenging data sets but for now this is not required since the aim is to compare the differences in accuracy and computation time and not robustness. Robustness properties in both approaches will be evaluated further in Chapter 6.

In the experiment, two dense 3D reconstructions will be computed based on the current academic (COLMAP) and the proposed architecture (Maplab). All calculations are performed on the computing platform that is described in Appendix B. In the comparison, PIX4D is not considered since it does not provide means to evaluate the pose-accuracy. In Chapter 6, PIX4D will also be evaluated and compared against the presented method and COLMAP.

First, the sparse 3D modeling will be compared. Both outputs have to be aligned with the ground-truth trajectory since they use different global coordinate systems. Furthermore, since COLMAP computes a reconstruction which is accurate up to an unknown scale-factor, the scale needs to be estimated as well in order to compare the results. To achieve this, Umeyama's [45] Sim(3) and SE(3) alignment methods are used for COLMAP and the proposed architecture, respectively based on the implementation of [20]. Here, for the current architecture, a Sim(3) alignment is computed which estimates a rigid-body transformation and a scale-factor to align the trajectory with the ground-truth. For the proposed method, a SE(3) alignment is estimated which yields a rigid-body transformation and no scale-factor. This is chosen since Maplab is able to observe the correct scale directly.

To compute a reconstruction in COLMAP, the standard settings are used and all the recorded visual data is provided as input to the pipeline. For Maplab, to compute the sparse 3D reconstruction, two set of steps are taken. First, the visual-inertial data is run on Maplab's SLAM front-end, ROVIO. Secondly, the resulting sparse 3D reconstruction is post-processed by running the loop-closure module and visual-inertial bundle adjustment of the Maplab framework. The output is a sparse 3D reconstruction. Running the first step of both methods yields a sparse 3D model with an estimated trajectory. These results are aligned using the

aforementioned method and is visualized in Figure 4-5.

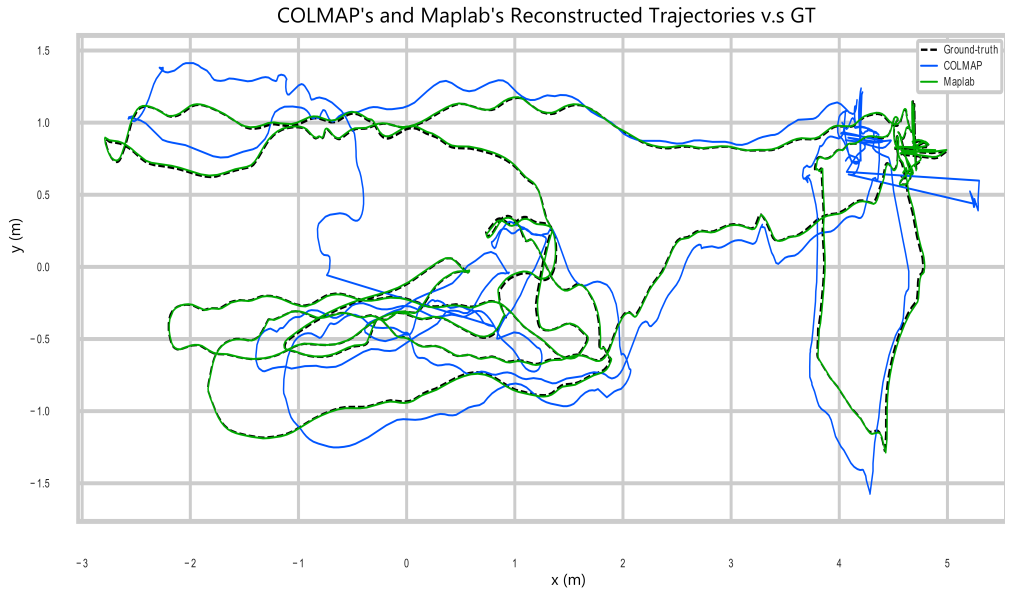

Figure 4-5: COLMAP and Maplab trajectories $\mathrm{SO}(3)$+Scale aligned with the ground-truth poses.

From the result shown in Figure 4-5 it can be concluded that the Maplab-based approach yields significantly more accurate results in terms of pose-estimation. Next, it becomes possible to compute a dense 3D reconstruction based on the output of both methods and compare the results. The resulting reconstructions are shown in Figure 4-6.

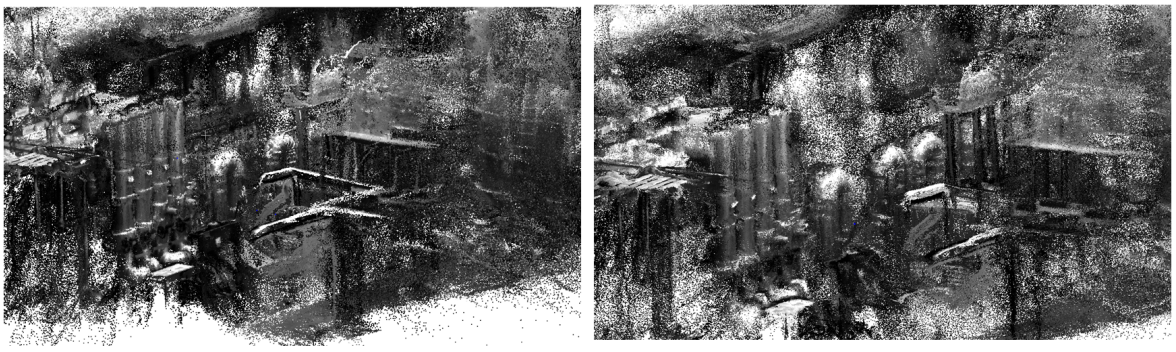

Figure 4-6: The current (left) and proposed (right) architecture resulting 3D model. Using the same fusion settings 2.6M and 2.35M points are extracted respectively.

In Figure 4-6 it can be concluded that both methods produce dense 3D reconstructions that appear visually very similar. Although the current architecture reconstructed a dense 3D point cloud model of 2.6M points, which contains 10.6 (%) more points than the proposed architecture, it is important to note that this result also contained more outliers which are not clearly visible in Figure 4-6. For this data set it is not possible to compare both methods with a ground-truth 3D point cloud model since that is not available. However, in Chapter 6, both approaches will be evaluated on several data sets which contain ground-truth 3D point cloud models.

To make a more rigorous comparison between the accuracy in pose-estimation in the two approaches, the Absolute Pose Error (APE) defined in meters can be computed. This metric computes the Euclidean distance between the ground-truth and reconstructed pose. Then, several statistics can be computed from this metric such as the Root Mean Squared Error (RMSE) or median, which is less sensitive to outliers. The resulting metrics and computation times are depicted in Table 4-1.

Architecture	**Sparse modeling Time**	**RMSE APE (m)**	**Median APE (m)**	**Dense modeling Time**
Current	325	0.9971	0.6579	332.4
Proposed	19	0.0148	0.0124	333.9

Table 4-1: Computation time defined in minutes and trajectory accuracy evaluated for the current and proposed architecture. In this notation APE is defined as the Absolute Pose Error. The metrics are computed using [20].

In Table 4-1, the observation in Figure 4-5 is confirmed where the proposed architecture yields poses which are an order of magnitude more accurate than the current architecture. Besides the increase in accuracy, it is also observed that the sparse 3D modeling computation time has been decreased from 332.4 to 19 (min), a decrease of 94 (%). And, since the same dense 3D modeling architecture is used in both methods, the dense 3D modeling time is approximately equal.

Concluding, the addition of inertial information solved the scale ambiguity problem of the current architecture. Furthermore, the computation time for sparse 3D modeling was reduced with a large margin whilst producing more accurate results in terms of pose-estimation. As of right now, the dense 3D modeling module is the bottleneck in the pipeline where both architectures rely on the same dense 3D modeling module of COLMAP. As it is of interest to obtain dense 3D point cloud models of cellular sites when the technician is still at the site, the next chapter will focus on optimizing the computational complexity of the dense 3D modeling module.

Chapter 5

Accelerating Dense 3D Modeling

As mentioned in Chapter 2, dense 3D modeling works through a process called Multi View Stereo (MVS) and fusion. In this process, a depth-normal value was estimated for each pixel in each image in the input set.

As such, MVS algorithms scale very poorly with increasing resolutions where the computation time is directly proportional to the number of processed pixels [30]. In many practical applications, it is necessary to produce dense 3D point cloud models in near real-time for a direct spatial understanding such as autonomous driving or EXtended Reality (XR) [11, 39]. As the quality of consumer-grade camera's is rapidly growing, a need is created to improve the scalability of dense 3D modeling algorithms to be able to handle larger image resolutions. In this chapter, a novel approach is proposed to achieve a reduction in computational complexity of dense 3D modeling systems to better handle high-resolution images.

5-1 Limiting the number of processed pixels

To improve the scalability of dense 3D modeling algorithms, the most widely used solution aims at decreasing the number processed pixels. To achieve this, the input images are downscaled by a certain factor. This effectively limits the number of processed pixels by decreasing the image size. The decrease in computation time is directly proportional to the downscaling factor that is applied to the input images. This is shown for an image in Figure 5-1.

In this scenario, not only computation time and dense 3D model resolution are traded-off but also coarsity. In this definition, coarsity describes the ability to capture fine details and edges in 3D structures. To explain why this is the case, it is important to remember how multi-view stereo processes an image to compute the depth-normal map. It extracts a rectangular patch around the pixel which it uses to compute a matching cost. Intuitively, downscaling can be seen as resizing the size of pixels where the size of one pixel increases. With equal patch-sizes, this means that the relative size of a patch increases with a decreasing scale-factor. This is visualized in Figure 5-2

Figure 5-1: Downscaling an image with a different scale-factor S. The decrease in total pixels with respect to the original image is directly proportional to the downscale factor.

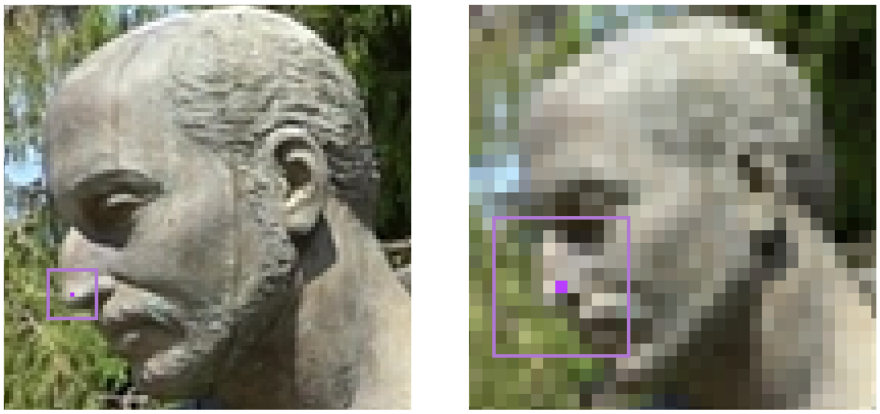

Figure 5-2: Full image (left) and factor 4 downscaled image (right). In both images a 6×6 patch (purple rectangular) is visualized for a pixel (purple square) which is used to compute depth-normal values.

The increase in relative patch-size reduces the capability of discriminating fine details in an image where examples include edges, boundaries or complex shapes such as curves. One may wonder how this affects the final 3D reconstruction quality. To study the trade-off in computation time and dense 3D model quality, a series of experiments are run on the Ignatius data set from the Tanks-and-Temples benchmark [25]. All calculations are performed on the computing platform described in Appendix B. The Tanks-and-Temples benchmark provides an accurate ground-truth 3D dense point cloud reconstruction for evaluation and a pre-computed sparse 3D model. For this experiment, the downscale factors $S = \{1, 0.5, 0.25, 0.125\}$ are chosen. In each reconstruction, the input images are first downscaled with a factor S using a bi-linear filter which is a widely used image rescaling algorithm that is implemented in COLMAP. More information on the precise workings of a bi-linear filter and other image scaling techniques can be found in [19]. Then, the dense 3D point cloud model is calculated with COLMAP's dense 3D modeling module using standard settings. The resulting dense 3D

point cloud models are shown in Figure 5-3.

Figure 5-3: Dense 3D point cloud reconstructions at different downscale factors S compared against the ground-truth (GT).

The reconstructions are compared using two metrics: precision and recall which are combined into a F_1 score. The F_1 score is obtained by computing the harmonic mean of both metrics. Intuitively, precision can been seen as the accuracy and recall as the completeness of the reconstructed 3D point cloud model. In this definition precision is defined as the fraction of points of the reconstruction that have a closest neighbor to the ground-truth for a given threshold distance τ. Recall is defined exactly the other way around. It is computed by evaluating the closest neighbor for each point in the ground-truth reconstruction. The recall score is then computed by evaluating the fraction of ground-truth point which have a closest neighbor within the distance τ. For this experiment, τ is set to 3 (mm).

The differences in processed pixels per image, F_1 score, precision, completeness and computation time are summarized in Table 5-1.

S	**Processed Pixels**	**F_1**	**Precision**	**Recollection**	**Time (min)**
1	2,073,600	0.7700	0.7252	0.8207	129.2
0.5	518,400	0.4425	0.5836	0.3564	57.4
0.25	129,600	0.1196	0.3585	0.0718	28.6
0.125	32,400	0.0217	0.1780	0.0116	14.2

Table 5-1: Experimental results showcasing the effect in dense 3D point cloud quality when downscaling the image. The downscale factor S limits the number of processed pixels per image which accelerates the dense 3D modeling.

From Table 5-1 it can be clearly seen that the computation time is directly proportional with the downscaling factor S. Furthermore, it can be concluded that not only the recall score decreases for smaller downscaling factors S. Besides recall, also precision drops which is undesirable since it destroys fine details and edges in the reconstruction.

In this work, an alternative is proposed to improve the scalability of the dense 3D modeling algorithms. Instead of downscaling the image, the full resolution is used as input to the multi-view stereo algorithm. However, instead of processing each pixel, a subset of pixels is

processed. Specifically, in this work, it is proposed to process every k^{th} pixel which results in a uniform grid. This is shown for an example image-grid of 20×20 pixels in Figure 5-4.

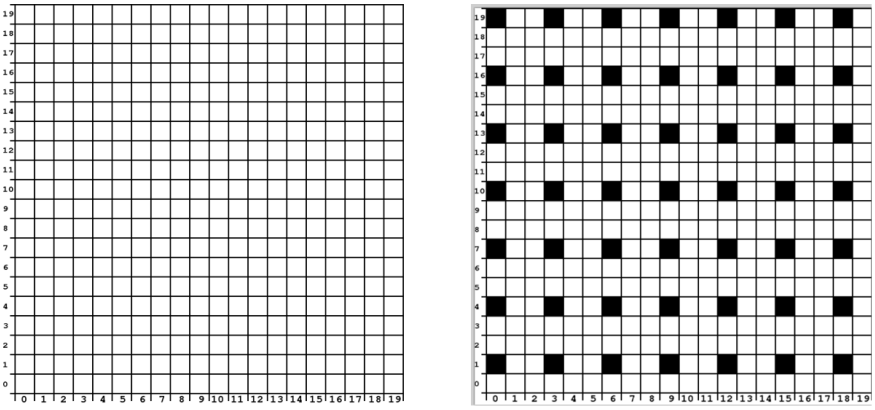

Figure 5-4: Full image (left) sparse pixel computations (black) where every $k^{th} = 3$ pixel is processed. The total reduction in processed pixels is equal to k^2.

Using this scheme, the number of processed pixels is reduced with a factor k^2. This method circumvents the problem of larger relative patches since the original image is kept during processing. To implement this, the code of COLMAP is changed such that it skips every k^{th} pixel and outputs a sparse set of depth-normal maps defined as $\mathcal{D}_s$ and $\mathcal{N}_s$, respectively.

By implementing this change and choosing a sampling-rate k, the computation time per image is reduced by limiting the number of processed pixels. Since COLMAP's parallel implementation processes one image row at a time, the computation time is reduced by a factor k. This is visualized in Figure 5-5.

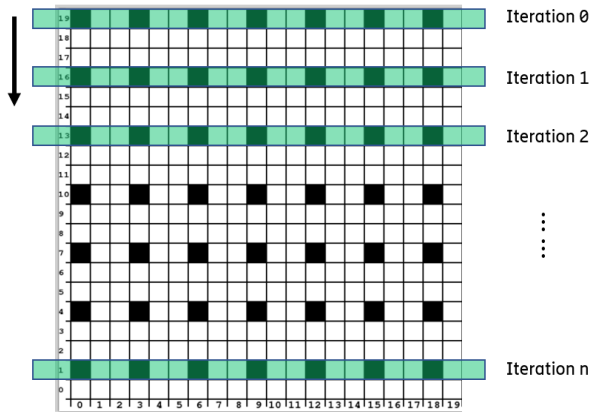

Figure 5-5: COLMAP's parallel implementation processes one image row at a time. In this example, the MVS computations are accelerated with a factor 3 which is directly proportional to the sampling-rate $k = 3$.

However, the output is a sparse depth-normal map where many pixels contain missing values. As described in Chapter 2, a full depth-normal map is necessary to extract a dense 3D model

$\mathcal{X}_d$. The next section will propose a method to achieve this and answer the question of how the dense depth and normal information may be recovered.

5-2 Dense depth recovery

In this section, a method will be derived to retrieve the missing pixel information. The goal is to transform the sparse set of depth maps $\mathcal{D}_s$ and normal maps $\mathcal{N}_s$ into dense maps $\mathcal{D}$ and $\mathcal{N}$, respectively. Using the proposed method, a dense depth-normal map can be estimated from the sparse multi-view stereo output.

5-2-1 Problem definition

Given the sparse depth-normal map from running COLMAP's adjusted multi-view stereo algorithm, the goal is to compute a depth and normal estimate for all the missing pixel values. For an image j there are three information sources available: the color image $\mathbf{I}_j \in \mathcal{I}$, the sparse depth map $\mathbf{D}_j \in \mathcal{D}_s$ and the sparse normal map $\mathbf{N}_j \in \mathcal{N}_s$.

To solve for the missing pixel values, in this work, it is proposed to leverage the available full color image. It is argued that there exists a local correlation between depth and color space that can be used to estimate the missing depth-normal values whilst preserving details. In this definition, the term local indicates that this assumption holds for small neighborhoods of pixels $(u, v) \in \Omega_k$ in an image. This relation is depicted in Figure 5-6.

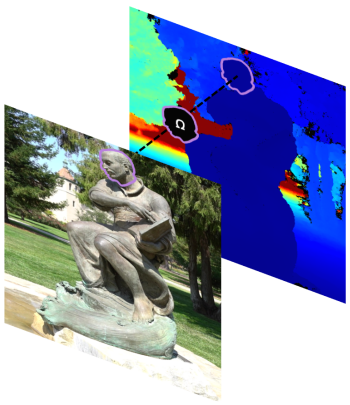

Figure 5-6: The assumption in this work is that similar colored pixels in a local neighborhood $(u, v) \in \Omega_k$ are linearly correlated with their associated depths $d(u, v)$.

To use this correlation, it is argued that, given a small enough neighborhood Ω_k of pixels in an image, the surface that is reconstructed within this local area can be well approximated with a 3D plane $\pi_k \in \mathbb{R}^2$

$$\pi_k := \mathbf{n}^T \cdot \mathbf{p} + d_c = 0, \tag{5-1}$$

where $\mathbf{n} = (a, b, c)^T$ represents the normal vector of the plane, $\mathbf{p} = (x, y, z)^T$ a point lying on the plane and d_c a constant defined as

$$d_c = -\mathbf{n}^T \cdot \mathbf{c}, \tag{5-2}$$

where $\mathbf{c} = (c_x, c_y, c_z)^T$ represents the centroid of the plane. The problem of retrieving the missing pixel values is then solved by estimating a series of 3D planes within each local window Ω_k. This is shown in Figure 5-7.

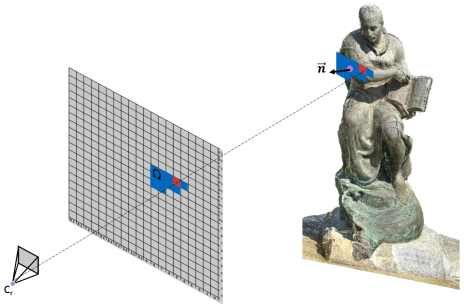

Figure 5-7: For a local neighborhood of pixels $(u, v) \in \Omega_k$ (blue) in an image it is argued that the 3D scene that is reconstructed can be well approximated by a 3D plane π_k. Missing pixels are retrieved by back-projection (red square).

For a given plane π_k valid in a local neighborhood of pixels $(u, v) \in \Omega_k$, the missing pixel estimates for depth $d(u, v)$ can be retrieved by projecting the 3D plane back into the 2D depth map which may be computed as

$$d(u, v) = -\frac{d_c}{(ae + bf + c)} \quad (u, v) \in \Omega_k. \tag{5-3}$$

In this equation it holds that u and v are the pixel coordinates and $\{a, b, c, d_c\}$ the plane-parameters. Furthermore, e and f are defined as

$$\begin{aligned} e &= \frac{u - f_x}{\bar{c}_x}, \\ f &= \frac{v - f_y}{\bar{c}_y}, \end{aligned} \tag{5-4}$$

where the parameters $\{f_x, f_y, \bar{c}_x, \bar{c}_y\}$ are obtained from the intrinsic camera calibration matrix $\mathbf{K}$ as described in Appendix A. Since a 3D plane has a constant normal-vector, to obtain the missing normal values, all pixels within the local neighborhood are set equal to the orientation of the 3D plane $\mathbf{n}$.

Using this method, there are two key assumptions made to solve the problem of retrieving missing pixel values. First, it is assumed that there exists a local correlation between the color and depth space. And secondly, within a small neighborhood Ω_k where this correlation holds, the local surface can be well approximated with a planer 3D model.

However, that still leaves the question unanswered of how the local neighborhoods Ω_k can be obtained. In this work, it is proposed to compute the local neighborhoods by computing

an over-segmentation of the color image: super-pixels. Such super-pixels have the desirable properties in that they locally cluster the image based on spatial (locality) and color (correlated with depth) distance. What exactly super-pixels are and how they are computed will be treated next.

5-2-2 Super-pixel segmentation

Super-pixels are a set of clusters computed from a color image. Each cluster is defined by its centroid location and boundary pixels. Super-pixels are an abstraction of the color image and can been seen as a low-level representation of the image. In such a representation each pixel in the image is assigned to a specific cluster. Super-pixels are widely used in the computer-vision community to accelerate or improve the accuracy of vision applications. Examples include 2D-tracking and optical-flow estimation problems [50, 47]. The key insight of these applications is that operating on a super-pixel abstraction of an image provides a more efficient representation of an image without much loss of information. This efficient representation may be leveraged to improve the results. However, super-pixels have never been applied in the context of multi-view stereo to reduce the computational complexity. That is the main idea behind the proposed approach - leverage super-pixel level abstraction of images to reduce the computational complexity in multi-view stereo.

There are multiple ways of computing super-pixels but the most widely used one is defined as Simple Linear Iterative Clustering (SLIC) [2]. It computes the over-segmentation in two steps: initialization and iteration until convergence. First, the initialization distributes a set of K cluster-centers in a uniform grid. Then, secondly, each pixel in the image is assigned to a cluster-center k based on color and spatial distance. Next, the cluster-centers are updated with a new location based on the assigned pixels. Then these steps are repeated until convergence. This yields a set of cluster-centers with assigned pixels. This process is shown in Figure 5-8.

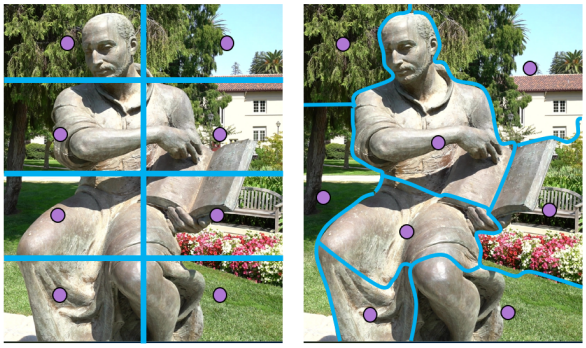

Figure 5-8: Computation of $K = 8$ super-pixels in two steps. Initialization of cluster-centers (left) and convergence into a set of super-pixels (right).

The parameter K which defines the number of super-pixels can be controlled. To showcase the effect of this parameter, several over-segmentation's are computed using the SLIC algorithm and shown in Figure 5-9.

Figure 5-9: Super-pixel segmentation of the image computed using the SLIC algorithm [2]. In this example, the number of clusters (cyanide borders) is varied by increasing the value K.

Due to their properties, super-pixel clusters can be used to leverage the assumed correlation between depth and color to obtain a set of local neighborhoods Ω_k.

Summarizing, the dense depth-normal estimation process can be divided in three steps. First, it will compute a set of super-pixels from a color image which are used as local neighborhoods Ω_k. Next, it will collect the sparse set of depth-normal samples and compute a 3D plane for each super-pixel. Thirdly, it will project the 3D planes back into the depth map to obtain the missing depth values. The missing normal values are extracted by setting them equal to the constant normal vector of each plane. This process is depicted in Figure 5-10.

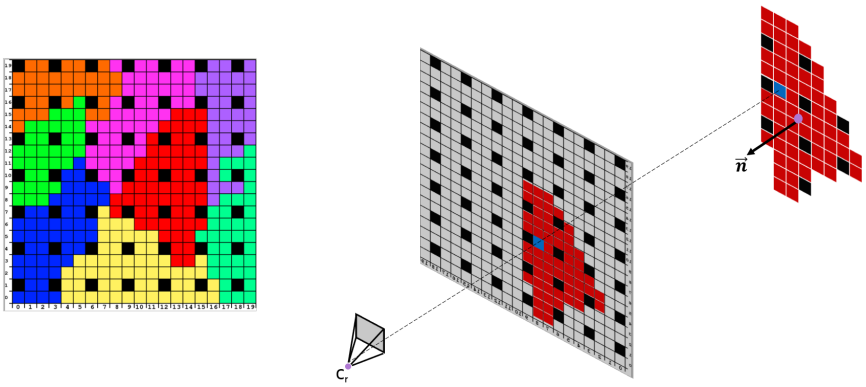

Figure 5-10: Super-pixels overlaid on the sparse depth-normal map (left). Interpolation is executed by estimating a plane for each super-pixel (right) using the sparse information (black rectangular) and projecting it back into the depth-normal map (middle).

5-2-3 Choosing the parameters

The number of required super-pixel clusters are governed by the parameter K. When the goal is to accelerate the multi-view stereo scheme by abstracting the image in super-pixels, it is desirable to have an as compact as possible abstraction. Specifically, as small as possible

K. However, from Figure 5-9 it became clear that at smaller values of K the super-pixels fail to grasp fine details in the scene. In this example, they fail to capture the boundaries of the Ignatius statue accurately. Hence, there is a lower-bound on the required super-pixels. In this work, it is proposed to estimate such bound by defining the minimum width or height in pixels p_{min} of the smallest details in the image that needs to be retained in the abstraction.

If an image has a width w and height h the total number of pixels is equal to $w \cdot h$. Let K be the number of super-pixels then the average length or width of a super-pixel L is equal to

$$L = \sqrt{\frac{w \cdot h}{K}}. \tag{5-5}$$

When the smallest details in the image are of size p_{min} then it is required that at least the average super-pixel length and width is equal to this. This yields

$$p_{min} = L = \sqrt{\frac{w \cdot h}{K}}. \tag{5-6}$$

Solving Equation (5-6) for K then results in

$$K = \frac{w \cdot h}{p_{min}^2}. \tag{5-7}$$

This yields a minimum number of super-pixels that are required to capture the details of interest. This only leaves the parameter k (sampling-rate) to be considered. Ideally, the sampling-rate should be chosen as high as possible since the number of processed pixels is reduced with a factor k^2. To make the problem tractable it is required that the minimum number of samples k_{min} is equal to 1. The number of samples per super-pixel cluster is given as

$$k_{min} = \frac{p_{min}^2}{k^2}. \tag{5-8}$$

Solving for k will result in

$$k = \sqrt{\frac{p_{min}^2}{k_{min}}}, \tag{5-9}$$

where k is rounded down to its closest integer value. However, $1 \leq k$ so Equation (5-22) is adjusted to always be greater than or equal to 1

$$k = \max\left(1, \sqrt{\frac{p_{min}^2}{k_{min}}}\right). \tag{5-10}$$

Concluding, a method was obtained for setting the required parameters K and k to run the proposed scheme. However, the process of transforming the sparse depth-normal map into a set of 3D planes π_k has not been introduced. This will be treated next.

5-2-4 Robust M-estimation for dense depth-normal map recovery

The dense depth-normal map estimation problem is defined as identifying a set of 3D planes using the sparse depth-normal map. Since each depth-normal map contains a portion of outliers due to noise in the MVS algorithm, the problem is framed as a robust model fitting problem. To this end, it is chosen to use a maximum likelihood type or M-estimation framework where a robust loss function is minimized to solve for the planar parameters. In this section, the framework which estimates the set of planes using the M-estimation scheme will be derived.

The available information is the sparse set of depth-normal pixels and the super-pixel mask which contains the local neighborhoods $(u, v) \in \Omega_k$. For a super-pixel k, the goal becomes to infer a 3D plane π_k which is described by its centroid $\mathbf{c}$ and normal vector $\mathbf{n}$. The normal vectors are given in Cartesian coordinates. This is problematic since this means that there is a norm-constraint on the vector. To circumvent this, the Cartesian coordinates are transformed to spherical coordinates via the mapping $(x, y, z) \rightarrow (r, \theta, \varphi)$

$$\begin{aligned} \theta &= \arctan2\left(\frac{\sqrt{a^2+b^2}}{r}\right), \\ \varphi &= \arctan2\left(\frac{b}{c}\right), \end{aligned} \tag{5-11}$$

which satisfies the norm constraint by fixing the radius r to 1. The plane π_k is then defined by the state

$$\mathbf{x}_k = \begin{pmatrix} c_x, c_y, c_z, \theta, \varphi \end{pmatrix}^T, \tag{5-12}$$

which needs to be estimated. The state $\mathbf{x}_k$ is directly related to the plane which was described in Equation (5-1) via the mapping

$$\begin{aligned} a &:= \sin(\varphi) \cdot \cos(\theta), \\ b &:= \sin(\varphi) \cdot \sin(\theta), \\ c &:= \cos(\varphi), \\ d_c &:= -\left(a \cdot c_x + b \cdot c_y + c \cdot c_z\right). \end{aligned} \tag{5-13}$$

For a super-pixel, one processed pixel contains a 3D point and a normal vector defined in spherical coordinates $\mathbf{z}_i = (z_x, z_y, z_z, z_\theta, z_\varphi)^T$. This yields the term

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{A}_i} \underbrace{\begin{pmatrix} c_x \\ c_y \\ c_z \\ \theta \\ \varphi \end{pmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{pmatrix} z_x \\ z_y \\ z_z \\ z_\theta \\ z_\varphi \end{pmatrix}}_{\mathbf{b}_i}, \tag{5-14}$$

where $\mathbf{A}_i \in \mathbb{R}^{5\times5}$, $\mathbf{x}_k \in \mathbb{R}^5$ and $\mathbf{b}_i \in \mathbb{R}^5$. Collecting all N measurements within one super-pixel k yields

$$\mathbf{A}_k = \begin{pmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \cdot \\ \cdot \\ \mathbf{A}_N \end{pmatrix}, \quad \mathbf{b}_k = \begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \cdot \\ \cdot \\ \mathbf{b}_N \end{pmatrix}, \tag{5-15}$$

where $\mathbf{A}_k \in \mathbb{R}^{5N\times5}$ and $\mathbf{b}_k \in \mathbb{R}^{5N}$. Using this result, the error for one super-pixel is defined as

$$\boldsymbol{\epsilon}_k = \mathbf{A}_k\mathbf{x}_k - \mathbf{b}_k, \tag{5-16}$$

where $\boldsymbol{\epsilon}_k \in \mathbb{R}^{5N}$. Let the total number of super-pixels be equal to K, then the total system can be defined by stacking the results for each super-pixel

$$\underbrace{\begin{pmatrix} \boldsymbol{\epsilon}_0 \\ \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_K \end{pmatrix}}_{\boldsymbol{\epsilon}} = \underbrace{\begin{pmatrix} \mathbf{A}_0 & & & \\ & \mathbf{A}_1 & & \\ & & \ddots & \\ & & & \mathbf{A}_K \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{pmatrix}}_{\mathbf{x}} - \underbrace{\begin{pmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \cdot \\ \cdot \\ \mathbf{b}_K \end{pmatrix}}_{\mathbf{b}}. \tag{5-17}$$

From which the plane estimation problem can be formulated in a M-estimator framework by formulating it as a least-squares problem where the Huber's loss function is minimized

$$\min_{\mathbf{x}} J(\boldsymbol{\epsilon}) = \sum_{i=0}^{K-1} \rho\left(\boldsymbol{\epsilon}_k^2\right), \tag{5-18}$$

where $\rho(\cdot)$ denotes the Huber's loss function [22]. This loss function is designed such that large residuals get down-weighted. It can be solved by the Iteratively Reweighted Least Squares (IRLSQ) algorithm which iterates between solving the weighted least-squares problem and using that result to update the weights until convergence. The convergence criteria is met when the cost-change between two iterations is smaller than a threshold Δ_{min}

$$|1 - \frac{J_{t-1}}{J_t}| \leq \Delta_{min}, \tag{5-19}$$

where the index t is defined as the iteration number. In this setting, the weighted least-squares problem is defined as

$$\min_{\mathbf{x}_k} \sum_{k=0}^{K-1} \mathbf{w}_k \cdot \boldsymbol{\epsilon}_k^2, \tag{5-20}$$

where $\mathbf{w}_k \in \mathbb{R}^{5N \times 5N}$ is a diagonal weighting matrix. The weights are derived from differentiating Huber's loss function and are defined for a single residual as

$$w_i(\epsilon_i) = \begin{cases} 1 & \text{if } |\epsilon_i| \leq c \cdot \hat{\sigma} \\ |\frac{1}{\epsilon_i}| & \text{otherwise.} \end{cases} \tag{5-21}$$

In equation (5-21), c is chosen as 1.345. The parameter c is chosen such that it has 95 (%) efficiency when there are no outliers compared to the ordinary least-squares solution [22]. Furthermore, $\hat{\sigma}$ is defined as the estimated co-variance of the state belonging to ϵ_i. In practice, the co-variance σ of the states is unknown and needs to be estimated as well. This is achieved by following the procedure as described in the works of P. J. Rousseeuw et al. [34]. It estimates the co-variance by computing the Median of Absolute Deviations (MAD)

$$\hat{\sigma}_{x_i} = 1.4826 \cdot \text{median}\left(|\boldsymbol{\epsilon}_{\mathbf{x_i}}|\right), \tag{5-22}$$

where $\boldsymbol{\epsilon}_{x_i}$ is the set of residuals of state i. The MAD is scaled with a constant 1.4286 to obtain a break-down point of 50 (%) when the distribution is assumed to be Gaussian [34]. The break-down point is a measure of robustness and indicates at what percentage of outliers in the sampled set the estimate becomes biased. For example, to compute $\hat{\sigma}_\theta$, the residuals of the state θ in all K super-pixels are collected and the median value of this vector scaled with the constant 1.4286 is used as the robust co-variance estimate.

Summarizing, the method can be formulated for a single image in the following pseudo-code:

Algorithm 1 Accelerated Dense 3D modeling

procedure PRE-PROCESSING
  $p_{min} \leftarrow$ smallest pixel-size of details
  $K \leftarrow$ (5-6);
  $k \leftarrow$ (5-10);
  compute SLIC super-pixels;
  if each super-pixel has one sample **then**
    continue;
  else
    $K \leftarrow K^+, \quad K^+ \leq K$;
procedure PROCESSING
  run multi-view stereo on the sparse uniform grid of pixels;
  store sparse depth-normal image;
procedure POST-PROCESSING
  build system matrix using super-pixels and sparse multi-view stereo output (5-17);
  $\mathbf{w}_k \leftarrow \mathbf{I}_k$;
  while not converged (5-19) **do**
    $\mathbf{x}^* \leftarrow \underset{\mathbf{x}}{\text{argmin}} \sum_{k=0}^{K-1} \mathbf{w}_k \cdot \boldsymbol{\epsilon}_k^2$ according to (5-20);
    $\hat{\boldsymbol{\sigma}} \leftarrow 1.4286 \cdot \text{median}\left(\boldsymbol{\epsilon}\right)$ according to (5-22);
    $\mathbf{w}_k \leftarrow$ Huber's weights obtained from (5-21);
  interpolate depth-normal images using (5-3);
  store dense depth-normal image;

5-3 Verifying the color-depth correlation

The dense depth estimation scheme was created on the basis of two key-assumptions. First, it was assumed that there exists a local correlation between color and depth for a small enough neighborhood Ω_k. This implied that color information can be used to define a small neighborhood of pixels such that the depth of those pixels are linearly correlated. The second assumption stated that if such neighborhood exists, the surface within that neighborhood can be well-approximated with a 3D planar model π_k.

In this section, the first assumption will be verified by means of an experiment. For this experiment a high-accuracy ground-truth depth map is required with corresponding color image. To this end, the Middlebury stereo data set is suitable [36]. This data set contains several color images with highly accurate ground-truth depth maps.

To perform the experiment, a data set is chosen which contains the color image and the corresponding ground-truth depth map. Next, for a different number of super-pixels, the color information is defined as the independent variable and used to predict the depth within each super-pixel. This yields a linear predictor of the form

$$\hat{\mathbf{y}}_k = \mathbf{A}_k \mathbf{x}_k, \tag{5-23}$$

where the predictor uses the available color information. This results in a predictor of depth of the form

$$\begin{aligned} \hat{d}(u,v) &= \alpha_k \cdot L + \beta_k \cdot A + \gamma_k \cdot B \quad (u,v) \in \Omega_k, \\ \hat{d}(u,v) &= \begin{pmatrix} L & A & B \end{pmatrix} \begin{pmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{pmatrix} \quad (u,v) \in \Omega_k, \end{aligned} \tag{5-24}$$

where L, A, and B are the colors in LAB-space at a pixel location (u, v) and α_k ,β_k,γ_k are the estimated coefficients for the super-pixel k. To estimate the coefficients, the ground-truth depths and respective color pixels can be collected in a standard linear regression form

$$\underbrace{\begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_N \end{pmatrix}}_{\mathbf{y}_k} = \underbrace{\begin{pmatrix} L_0 & A_0 & B_0 \\ L_1 & A_1 & B_1 \\ & \vdots & \\ L_N & A_N & B_N \end{pmatrix}}_{\mathbf{A}_k} \underbrace{\begin{pmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{pmatrix}}_{\mathbf{x}_k} + \underbrace{\begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_N \end{pmatrix}}_{\boldsymbol{\epsilon}_k}, \tag{5-25}$$

where $\mathbf{y}_k \in \mathbb{R}^N$, $\mathbf{A}_k \in \mathbb{R}^{N \times 3}$, $\mathbf{x}_k \in \mathbb{R}^3$ and $\boldsymbol{\epsilon}_k \in \mathbb{R}^N$. This is the result for one super-pixel k with N pixels. Expanding this result to all K super-pixels by stacking the results yields

$$\underbrace{\begin{pmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_K \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} \mathbf{A}_0 & & & \\ & \mathbf{A}_1 & & \\ & & \ddots & \\ & & & \mathbf{A}_K \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{pmatrix}}_{\mathbf{x}} + \underbrace{\begin{pmatrix} \boldsymbol{\epsilon}_0 \\ \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_K \end{pmatrix}}_{\boldsymbol{\epsilon}}. \tag{5-26}$$

The least-squares solution to Equation (5-26) is

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{5-27}$$

From this result, the prediction error is equal to

$$\boldsymbol{\epsilon} = \mathbf{A}\mathbf{x}^* - \mathbf{y}. \tag{5-28}$$

The correlation coefficient for multiple regression (regression involving more than one independent variables) is defined as the portion of the sample variance (ground-truth depths) that is explained by the predictor $\hat{\mathbf{y}}$

$$R^2 = 1 - \frac{Cov(\boldsymbol{\epsilon})}{Cov(\mathbf{y})}, \tag{5-29}$$

where $Cov(\cdot)$ is the sample co-variance. This procedure is executed on the Midlebury stereo benchmark [36] for 5 data sets - Vintage, Pipes, Playtable, Sticks and Jadeplant. The ground-truth depth-maps and color images are shown in Figure 5-11.

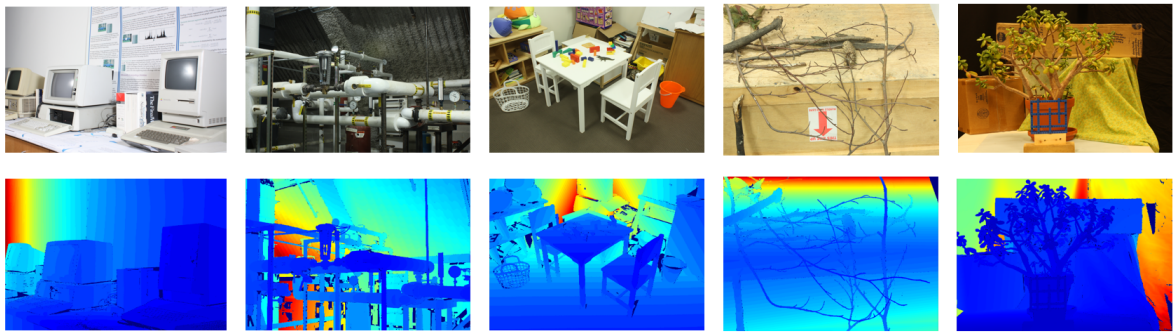

Figure 5-11: Ground-truth depth-map (bottom) and color-image (top). For this experiment, the Vintage, Pipes, Playtable, Sticks and Jadeplant (left to right) data sets were chosen from the Midlebury stereo benchmark [36] to represent a diverse set of scenes.

The correlation coefficient (R^2) is plotted for increasing number of super-pixels on each data set. Specifically, the correlation coefficient is computed for

$$K = \{10, 40, 90, 160, 250, 360, 490, 640, 810, 1000\}. \tag{5-30}$$

The results for the different data sets are shown in Figure 5-12.

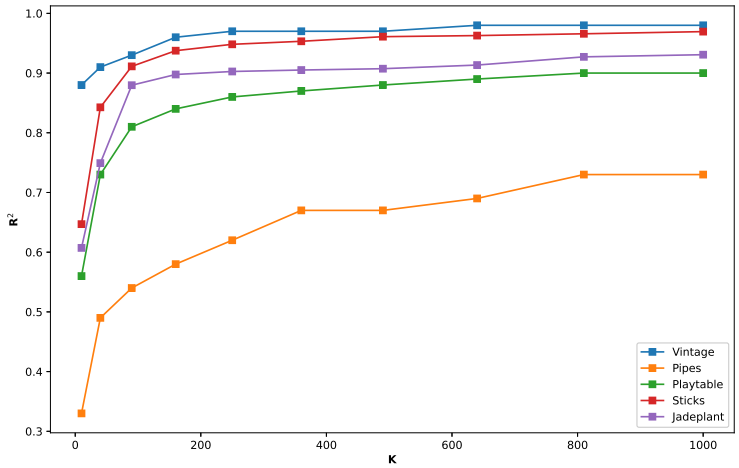

Figure 5-12: The correlation coefficient (R^2) is plotted for different number of super-pixels (K). Each colored line represents the results obtained on a different data set from the Midlebury stereo benchmark [36].

From Figure 5-12 it can be concluded that for increasing number of super-pixels there is a clear correlation between color and depth. Starting with 10 and increasing to 1000 super-pixels, the correlation coefficient increases from an average value of 0.61 to 0.90. Although the pipes data set seems to be an outlier in the sense that its correlation coefficient only increases to $\sim$0.7, it is still a significant result and supports the assumption that there exist a strong local linear correlation between the color and depth space. Concluding, the experimental results indicate that for sufficiently small-sized super-pixels (higher number of super-pixels), a linear correlation exists between the local neighborhood of color pixels $(u, v) \in \Omega_k$ and depth values $d(u, v)$ which can be exploited with the proposed scheme.

5-4 Experimental validation

In the previous sections, a method for accelerating dense 3D modeling was proposed. Contrary to downscaling, it was proposed to process a sparse set of pixels on a uniform grid for each image $\mathbf{I}_j$. Specifically, it was proposed to process only every k^{th} pixel. This procedure effectively limits the number of processed pixels with a factor k^2. However, the resulting sparse output needed to be densified in order to extract dense 3D point cloud models. To this end, it was proposed to leverage the local correlation between color and depth. Using this relation, the problem was defined as estimating a set of K 3D planes, using the sparse depth map $\mathbf{D}_j \in \mathcal{D}_s$ and normal map $\mathbf{N}_j \in \mathcal{N}_s$. The local neighborhoods Ω_k where such planes would be computed are obtained by means of a super-pixel over-segmentation on the color image. The local neighborhoods of pixels were grouped together based on color and spatial distance. It was expected that this procedure has the advantage of preserving edges and boundaries in a scene.

The goal of this section is to validate whether the proposed approach yields accurate and complete dense 3D point cloud models at decreased computational complexity. To this end, the proposed approach will be tested on the Tanks-and-Temples benchmark [25] on the Ignatius data set. The Ignatius data set is chosen since it contains complex curved shapes within the scene. This makes it possible to assess how well the approach holds for non planar objects. It is important to note that the experiment is performed on only one data set, the Ignatius data set. Although the experiment is only run on one data set, it should provide a good indication whether the planar surface approximation yields good 3D reconstructions.

5-4-1 Method

The input data set that is provided in the Tanks-and-Temples benchmark consists of a sparse 3D model with a set of images with a resolution of 1920×1080 and corresponding poses. Furthermore, it provides a highly accurate ground-truth 3D reconstruction for evaluation purposes. For a given sampling-rate and number of super-pixels, a sparse set of depth and normal maps are computed using COLMAP's MVS algorithm. Next, the proposed method is used to estimate a dense set of depth and normal maps. Then, the result is fused into a dense 3D point cloud model. Finally, this dense 3D point cloud model is evaluated by using the ground-truth 3D model.

In this experiment, a series of reconstructions will be computed where the sampling-rate will be increased to study the effect on computation time and 3D model quality. It will showcase the performance in reconstruction quality at reduced computational complexities and compare this with the downscaling of images.

5-4-2 Running the experiment

First, the number of required super-pixels need to be computed using Equation (5-7). Then, it is necessary to define the smallest pixel details that need to be captured. Since the Ignatius statue needs to be reconstructed a good starting-point is examining the fine details in the statue. By doing so, it is determined that most of the fine details such as the fingers can be captured by super-pixels of 15 (px) therefore setting $p_{min} = 15$. This is illustrated in Figure 5-13.

Figure 5-13: Zoomed in image showing the index finger of the Ignatius statue. Most details in the statue can be captured by approximately 15 (px) wide structures (purple).

Filling in (5-7) yields

$$K_{min} = \frac{w \cdot h}{p_{min}^2} = \frac{1920 \cdot 1080}{196} = 9259, \tag{5-31}$$

which is rounded upwards to 10,000 for the inclusion of a small margin. Using this amount of super-pixels and requiring at least a single sample in each cluster, the maximum sampling-rate can be calculated using Equation (5-22) as

$$k_{max} = \max\left(1, \sqrt{\frac{p_{min}^2}{k_{min}}}\right) = \max\left(1, \sqrt{\frac{15^2}{1}}\right) = 15, \tag{5-32}$$

which is sufficient for comparison with downscaling the image which was executed up until a factor of 8. With these results, a set of reconstructions is computed for $S = \{1, 0.5, 0.25, 0.125\}$. In this setting, the factor S denotes how many pixels are processed with each MVS iteration as a fraction of the full resolution input images. In this definition, the factor S is equivalent to the inverse of the sampling-rate k. As such, it is analogous to downscaling an image with a factor S since both approaches limit the number of processed pixels by a factor S^2. All reconstructions were computed using the hardware set-up described in Appendix B and using COLMAP's standard settings.

Although running $S = \{1, 0.5, 0.25\}$ yielded no problems, the number of super-pixels was set to $K = 5000$ for $S = 0.125$. This was chosen since otherwise there were a few super-pixels with no samples in the data set. The resulting reconstructions are compared with downscaling the input images and visualized in Figure 5-14.

Figure 5-14: Dense 3D point cloud reconstructions at different factors S compared against the ground-truth (GT). Showing the effect of downscaling (top row) and using the proposed scheme (bottom) on the dense 3D reconstruction.

From Figure 5-14 it can be concluded that the difference with downscaling is significant. Even at the coarsest scale, $k = 8$, which processed only 1.56 (%) of the total number of pixels, the reconstruction is visually not very different from the full information result. To make a more thorough comparison the reconstruction is compared with the available ground-truth 3D point cloud model which is obtained by a high-accuracy Light Detection And Ranging (LiDAR). For each reconstruction, an accuracy, completeness and F_1 score is computed. Where F_1 is the harmonic mean of the accuracy and completeness scores which yields a single performance index. All metrics are between 0 and 1 where a higher score indicates better performance. To determine the computation time, the interpolation time was not considered. This means that the processing time only consists of running COLMAP's MVS module. In Chapter 6, the post processing step will be considered to provide a more complete overview of the net benefit. The summary of these results are shown in Table 5-2.

S	**Processed Pixels**	**F_1**	**Precision**	**Recollection**	**Time (min)**
1	2,073,600	0.770	0.725	0.821	129.2
0.5	518,400	0.756	0.718	0.799	50.41
0.25	129,600	0.757	0.720	0.797	26.3
0.125	32,400	0.703	0.652	0.763	14.02

Table 5-2: Experimental results showcasing the effect in dense 3D point cloud quality when operating the proposed schemes at higher sampling-rates. For each sample-rate, computation time excluding interpolation, number of processed pixels per image, F_1, Precision, and Recollection metrics are computed for a threshold τ of 3 (mm). Higher scores indicate a better reconstruction.

Using the proposed method, the computation time of multi-view stereo is decreased with a factor S. Furthermore, it is surprising that reducing the number of super-pixels by half for the case $S = 0.125$ did not impact the performance significantly. This could be an indication that the method that was outlined to determine a minimum number of super-pixels is too conservative. Given the fact that in multi-view depth map estimation a set of independent depth maps are fused into a dense 3D point cloud model, it could be the case that having a set of coarser depth maps which do not capture all the details in a single image is still acceptable for the final result since it is fixed in the fusion step. This is considered interesting for future work. The differences in 3D point cloud quality between downscaling and the proposed approach are shown in Figure 5-15.

From Figure 5-15 it can be concluded that the proposed method is able to reduce the computational complexity at a more favorable trade-off in 3D point cloud quality when compared to downscaling the input images. Notably, at the coarsest setting for $k = 8$, the proposed method processed only 1.56 (%) of the pixels when compared to the full resolution images and reconstructed a point cloud model with a F_1 score of 0.703. To obtain an equal reduction in processed pixels per image, the input images would need to be downscaled with a factor 8. As demonstrated in Table 5-1, in this setting, COLMAP's MVS module produces a 3D point cloud model with a F_1 score of 0.022 which is 96.9 (%) lower than the proposed system. Although more experiments are necessary to generalize these findings, it can be concluded that there is a strong indication that the proposed system is capable of reconstructing complex shapes of scenes whilst decreasing the computational complexity in the MVS module when compared to downscaling the input images.

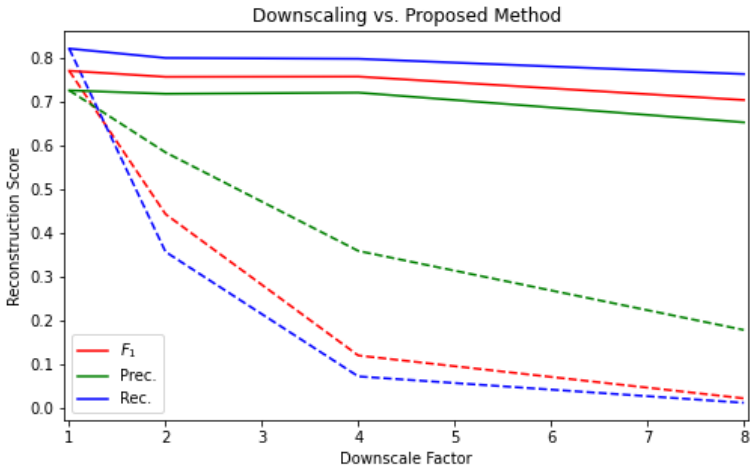

Figure 5-15: Comparison between downscaling (dashed lines) and the proposed method (unbroken line). For different downscale factors $(1/S)$, a dense reconstruction is computed and the F_1 (red), Precision (green) and Recollection (blue) scores are computed.

5-5 Overview of the proposed system

Now that the effectiveness of the acceleration scheme has been established in a first benchmark experiment, the complete proposed architecture can be defined. First, the input to the system is a set of synchronized visual-inertial recording and a calibrated camera system. Next, the proposed system will compute a dense 3D reconstruction in two steps. First, it will compute a set of poses $\mathcal{T}$ and sparse point cloud model $\mathcal{X}_s$ using the Maplab framework. Next, a sampling-rate and set of super-pixels is chosen and a dense reconstruction $\mathcal{X}_d$ is computed using the scheme outlined in this chapter. An overview of the proposed system is depicted in Figure 5-16

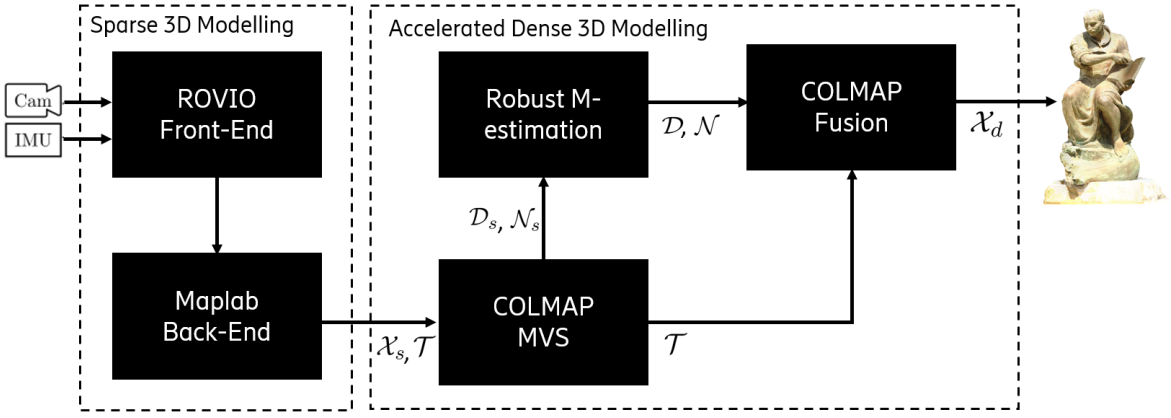

Figure 5-16: The proposed system computes a dense 3D point cloud model $\mathcal{X}_d$ in two steps. In the first step it computes a set of poses $\mathcal{T}$ and a sparse 3D map $\mathcal{X}_s$. Secondly, it computes the dense 3D point cloud model by running MVS on a sub-set of pixels, interpolating and fusing the sparse depth maps $\mathcal{D}_s$ normal maps $\mathcal{N}_s$.

5-6 Conclusion

In this chapter, a method was proposed to accelerate the dense 3D modeling module. It was shown that there exists a local linear correlation between color and depth for small enough neighborhoods Ω_k. This fact was exploited by moving away from per-pixel computations in multi-view stereo to higher-level clusters of pixels for which the local correlation holds. Such neighborhoods were obtained by computing an over-segmentation of the image: super-pixels. Then, it was proposed to compute only a sparse uniform grid of pixels defined by a sampling-rate k in an image which accelerated the dense 3D modeling by this factor. The sparse output in combination with the super-pixel grid was used to identify a set of 3D planes using robust regression. These planes were then used to obtain the dense depth-normal information again by back-projection in the image plane. Finally, the resulting dense depth-normal images were fused into a dense 3D point cloud model.

To study the performance, several experiments were run on the Tanks-and-Temples [3] benchmark. It was shown that the proposed method was successful in retaining details at sparser outputs. Even at very high sampling-rates (i.e. only processing 1.56 (%) of the pixels) the proposed approach was still capable of computing reconstructions at only a modest decrease in quality. For example, for the case $S = 0.125$, the F_1 score decreased from 0.770 to 0.703 compared to $S = 1$. The number of processed pixels was reduced from $2,073,600$ to $32,400$ per image.

However, the process of identifying the planar parameters was written in un-optimized and un-compiled code (Python). The interpolation time was not considered in these scenarios but was established to impair the net benefit of the approach. However, due to the independence of the computations, a parallel Graphical Processing Unit (GPU) implementation is possible which would decrease the interpolation time significant. This is considered interesting for future studies.

Now that the sparse and dense 3D modeling modules have been optimized, it is possible to put everything together and test the complete architecture in end-to-end experiments and compare against COLMAP and PIX4D.

Chapter 6

End-to-end Experiments

In Chapters 4 and 5, an alternative architecture was proposed to mitigate the typical issues with the current method: direct observation of scale, lack of robustness and scalability issues. As 3D modeling is executed in two steps - sparse and dense 3D modeling - the two steps were optimized sequentially. First, in Chapter 4, inertial information was included in the sparse 3D modeling system. This allowed the use of more efficient algorithms which used a form of visual-inertial Simultaneous Localization and Mapping (SLAM) instead of Structure from Motion (SfM). Following this approach, it was concluded that the scale is directly observable and the scalability was improved in the sparse 3D modeling step. Next, in Chapter 5, the dense 3D modeling was optimized for computational complexity. Specifically, the scalability was improved by operating not on the pixel level of the visual images but on a higher level abstraction - super-pixels. Using this method, it was concluded that a significant reduction in computational complexity was realized without compromising the 3D point cloud quality. Finally, the SLAM based approach was combined with the optimized dense 3D point cloud generation module, which resulted in an end-to-end visual-inertial 3D modeling architecture.

6-1 Test method

Now that the proposed architecture has been defined in Figure 5-16, it can be tested in end-to-end experiments. In this end-to-end setting the architectures will have a visual-inertial data set available as input. Using this data, the aim is to reconstruct a dense 3D point cloud model of the scene.

The goal of these experiments is to put everything together and showcase the performance in realistic scenarios of the current and proposed architectures. From these experiments it will be possible to make several conclusions whether the proposed architecture is capable of mitigating the typical issues of PIX4D and COLMAP. To achieve this, two types of data sets are used: multiple visual-inertial recordings from the EuRoC Micro Aerial Vehicle (MAV) benchmark [9] and recorded data in a lab-setting in the form of a case-study on a representative scene.

To be able to run a proper experiment, ideally, three types of information are required: a ground-truth trajectory, a ground-truth dense 3D point cloud model of the scene and a synchronized visual-inertial data set. First, the process of obtaining a ground-truth trajectory and a 3D model will be explained for the EuRoC MAV benchmark. Next, the process of evaluating the obtained trajectory and dense 3D point cloud model will be explained.

After having introduced the necessary concepts for evaluation purposes, several experiments will be run on the EuRoC MAV benchmark for which all information is available. After this, a set of experiments is executed on a lab-recording where the goal is to test the proposed system in a realistic remote inspection scenario and study its performance.

6-2 Obtaining a ground-truth trajectory and 3D model

Obtaining ground-truth information is the most critical aspect in running end-to-end experiments. Without this information the reconstructions can only be evaluated visually. To obtain a ground-truth set of poses, the typical approach is to use a high-accuracy motion tracking device. For example, in the well-known EuRoC MAV benchmark the 6D Vicon motion-tracker was used. Such a device directly measures the pose of the system. This information can be used as a ground-truth trajectory of an experiment.

The ground-truth 3D dense point cloud model is typically captured using a high-accuracy Light Detection And Ranging (LiDAR) sensor. LiDAR's are laser-based scanning devices capable of measuring a 3D scene with millimeter accuracy. Such dense 3D point cloud model can then be used to compare against the 3D reconstruction of the visual(-inertial) architecture.

6-3 Evaluating the poses

To evaluate the poses, the same procedure as outlined in section 4-4 is used. The procedure operates in two steps. First, it aligns the set of reconstructed poses with the ground-truth trajectory. This is achieved by Umeyama's [45] Sim(3) or SE(3) method, depending on the architecture. For PIX4D and COLMAP the SE(3) method is used. This is necessary since the recordings do not contain attached metadata and therefore, the reconstructed poses are correct up to an unknown scale factor. Furthermore, for the proposed architecture, only a Sim(3) alignment is required. The output of this first step is a set of reconstructed poses that are aligned with the ground-truth trajectory.

In the second step, two metrics are computed: the Absolute Pose Error (APE) and Relative Pose Error (RPE). The APE computes the Euclidean distance in meters between the ground-truth and reconstructed pose location. The RPE is defined as the relative difference in estimated pose. Intuitively, it can be thought of as a delta or difference between two consecutive estimated and ground-truth poses. Whereas the APE is a good measure of global consistency in the reconstructed poses, the RPE is commonly used to evaluate the drift over time in the estimation of the poses. Particularly, in SLAM applications this is interesting since drift builds up over the course of a trajectory and this metric provides insight as to how rapidly the accuracy of the reconstructed poses degrades. For both metrics, summarizing statistics are computed to evaluate the resulting trajectories. In this work, the Root Mean Squared

Error (RMSE) and standard deviation, σ are computed for the APE and RPE using the work of [20].

6-4 Evaluating the reconstruction

For evaluation purposes, the Tanks-and-Temples bench-marking procedure is used. Evaluating a dense 3D point cloud reconstruction boils down to comparing a set of two 3D point cloud models: the reference and the reconstruction. The quality of a point cloud model can be assessed by two characteristics: accuracy and completeness. Both characteristics can be computed quantitatively by computing the distance from one point cloud to the other and summarizing the results. Then, accuracy is defined as the percentile of points from the reconstruction that have a closest-neighbor with the ground-truth reconstruction within a threshold τ. Completeness is computed the other way around where the nearest distance neighbor is determined for each 3D point of the ground-truth reconstruction. Completeness is then defined as the percentile of ground-truth points that have a nearest neighbor point within a distance threshold τ. This process of calculating accuracy and completeness is visualized in Figure 6-1.

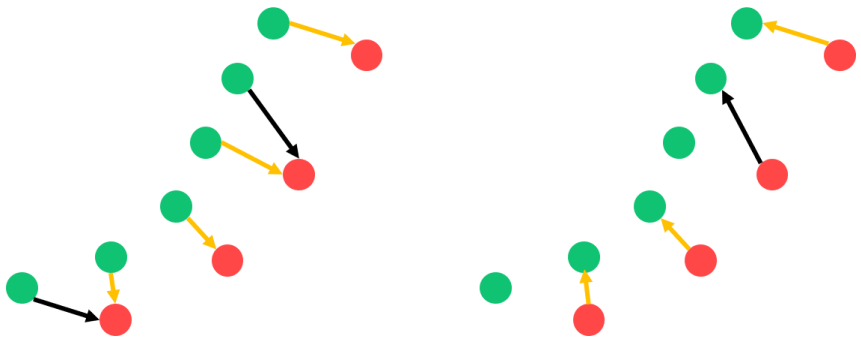

Figure 6-1: Ground-truth 3D (green) and reconstructed 3D model (red). Showing the completeness (left) and accuracy (right) score. Both metrics are computed by means of a nearest-neighbor search for each 3D point and thresholding on a distance τ. In this example the completeness and accuracy scores are 66.67 (%) and 75 (%), respectively.

When computing this score the threshold τ is typically chosen as the 2σ Mahalanobis distance of the specific LiDAR scanner that was used. This is chosen such that points that lie within the threshold τ have a $\sim$95 (%) probability of corresponding to the correct ground-truth point when the LiDAR's noise is assumed to be Gaussian. To provide one score which describes the quality of the point cloud, the authors of [3] propose to compute the harmonic mean of the recollection and precision metric defined as the F_1 score

$$F_1(\tau) = \frac{2P(\tau)R(\tau)}{P(\tau) + R(\tau)}. \tag{6-1}$$

This metric is preferred over for example the average of both metric since it has the property that when either $P(\tau) \to 0$ or $R(\tau) \to 0$, $F(\tau) \to 0$. Meaning, that lack of completeness and precision is both penalized. Taken to the extreme, when a reconstruction algorithm would

perfectly reconstruct a single 3D point, it would still have an average combined precision and accuracy score of 50 (%) when the mean is taken. On the contrary, using the harmonic mean would result in a combined score of 0 which better reflects the actual reconstruction quality.

Now that the evaluation method is outlined, the specific benchmark data set will be treated next.

6-5 The EuRoC micro aerial vehicle benchmark data Set

The EuRoC Micro Aerial Vehicle benchmark data set [9] was created for the aim of testing real-time SLAM algorithms. It provides a set of recordings that were acquired with the Asctec Firefly drone. On board of the drone, there is a visual-inertial architecture and a transmitter for the Vicon laser-tracker. The test set-up of the EuRoC MAV drone system is shown in Figure 6-2.

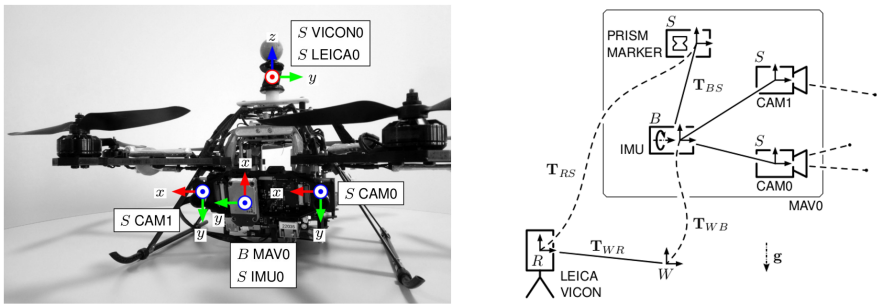

Figure 6-2: Asctec Firefly hex-rotor helicopter used during data set collection (left) and sensors and ground-truth instruments schematic overview (right). The figure and caption are obtained from [9].

The recordings are acquired in different locations but for this work, the Vicon hall was chosen as a suitable location since for this location there is a ground-truth dense 3D point cloud model available to compare against. This ground-truth point cloud model is captured with millimeter accuracy using the Leica MS50 laser-scanner and is depicted in Figure 6-3.

For this location, there are three set of recordings. Each recording differs in the difficulty of the flown trajectory. Starting from a slow-flown path, in each new recording, the speed is increased thereby making it more important to deal with aspects like motion blur. In each recording, the drone flies a path through the room at various heights, orientations and speed. Note that the trajectories of the drone do not resemble a realistic pattern for remote inspection purposes. In these cases, it is important to obtain a controlled, slow pattern of the object of interest. Although the benchmark was not captured with this goal in mind, it is interesting to study nevertheless. Since the recordings are more challenging, the algorithms are tested more exhaustively by decreasing the input data quality. This makes it possible to study the robustness properties more in depth. Besides this, it will be interesting to observe how the reconstruction quality is influenced by feeding the different architectures with different data.

For each recording, three reconstructions are computed using the two current approaches (PIX4D and COLMAP) and the proposed architecture without the accelerated dense modeling scheme (i.e. $k = 1$). All reconstructions are computed on the hardware set-up that

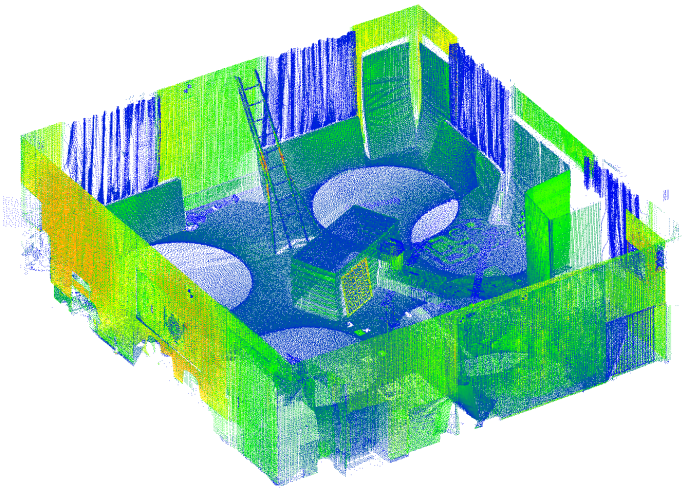

Figure 6-3: Ground-truth point cloud model of the EuRoC Vicon Hall captured with a Leica MS50 laser-scanner [9].

is described in Appendix B. Next, for each reconstruction, three aspects are compared: the computation time, the pose-accuracy in sparse modeling and the dense 3D model quality. After this, the complete proposed approach is tested. In this scenario, a dense reconstruction is computed for different sampling-rates on the easy data set. From this result, the dense 3D model quality will be assessed for these reconstructions and compared against the results of PIX4D and COLMAP.

6-5-1 Comparison of the current and proposed architectures

For the easy, intermediate and difficult recordings the sparse and dense reconstructions are compared. The same procedure as outlined in Section 4-4 is used where the result of COLMAP and PIX4D is scale-adjusted and aligned with the ground-truth trajectory and structure. This is necessary since the reconstructions of both methods are correct only up to an unknown scale factor. To compare all approaches, the results need to be aligned. Without alignment, the results can not be compared. It is important to keep in mind that since all recordings are obtained indoors, no external Global Positioning System (GPS) information is available. Both PIX4D and COLMAP would, without alignment to the ground-truth, be incapable of producing usable results where usable means reconstructions with accurate geometric scale.

The results for the sparse reconstruction are shown in Table 6-1.

When computation time is evaluated for the sparse modeling phase, there is a clear distinction between COLMAP, PIX4D and the proposed approach. On all recordings, the proposed approach yields at least an order of magnitude faster results. For example, on the first data set, the proposed approach computed a reconstruction in 18.2 (min), PIX4D in 892.8 (min) and COLMAP in 663.8 (min). This means that the proposed method is a factor 49 and 37 faster than PIX4D and COLMAP, respectively. When the quality of the reconstructed poses are evaluated, from which only the results of COLMAP and the proposed approach are available, an interesting phenomenon emerges. On the 01 data set, which was the easy data set, the proposed approach and COLMAP yielded a RMSE in APE of 0.0127 (m) and

	Sparse modeling Time (min)	APE (m) σ	APE (m) RMSE	RPE (m) σ	RPE (m) RMSE
01 PIX4D	892.8	**-**	**-**	-	-
01 COLMAP	663.8	0.0649	0.0230	0.0066	0.0044
01 Proposed	**18.2**	**0.0336**	**0.0127**	**0.0044**	**0.0022**
02 PIX4D	143.7	-	-	-	-
02 COLMAP	81.7	0.0273	**0.0095**	0.0045	0.0025
02 Proposed	**4.83**	**0.0245**	0.0096	**0.0036**	**0.0017**
03 PIX4D	168.5	-	**-**	-	-
03 COLMAP	66.6	**0.0329**	**0.0100**	0.0078	0.0051
03 Proposed	**4.2**	0.0464	0.0213	**0.0037**	**0.0019**

Table 6-1: Sparse 3D modeling results on the Vicon Room 1, data set 01, 02 and 03. Showing the sparse modeling time, the Absolute Pose Error and the Relative Pose Error in meters. Reconstructions are computed using COLMAP, PIX4D and the proposed method.

0.0230 (m), respectively. This means that the proposed architecture produced better results in terms of pose accuracy. However, when the data sets become more difficult the quality degrades. As can been seen in data set 03, COLMAP yielded better results. On this data set, the proposed architecture and COLMAP yielded a RMSE of the APE of 0.021 (m) and 0.010 (m), respectively. This could be caused by the fact that the proposed approach is constructed for real-time operation. When the data set is difficult, with fast movements and motion blur in the recordings, requiring real-time results degrades the performance. Since the post-processing step is executed after the recording, to find loop-closures and perform bundle adjustment, the errors are still larger than COLMAP. Although the RMSE and σ of the APE are 0.014 (m) and 0.011 (m) lower, respectively, it remains important to keep in mind that the reconstruction was obtained faster by a factor 16 than COLMAP, showcasing the favorable quality and performance trade-off. Furthermore, in the context of remote-inspection, the acquisition patterns are typically controlled, low-speed trajectories. Hence, data set 01 would be the more representative data set for such a use-case.

Now that the sparse reconstruction quality has been assessed, the dense reconstructions can be compared of the current approaches and the proposed approach. To achieve this, for each recording, a dense reconstruction is computed with COLMAP, PIX4D and the proposed approach. This yields 9 reconstructions. The reconstructions are aligned with the method outlined in Section 4-4 and evaluated using the procedure of Section 6-4. The results are depicted in Table 6-2

For these data sets the threshold τ was chosen to be 0.02 (m). This was chosen since the reported accuracy of the current approaches should be within this range. From Table 6-2 it can be concluded that PIX4D computes the best reconstructions on the easy and intermediate data sets 01 and 02 with a F_1 score of 0.7968 and 0.7063, respectively. Particularly, the precision of PIX4D is fairly high when compared to COLMAP and the proposed approach. Specifically, on the data set 01, PIX4D delivers a precision score of 0.8816 while COLMAP and the proposed approach only yield 0.5550 and 0.5843, respectively. Since PIX4D has a more extensive post-processing system, the higher score could indicate that PIX4D effectively removes more noise from the final reconstruction. Although the proposed approach delivers

	Dense modeling Time (min)	F_1	Precision	Recollection
01 PIX4D	346.6	**0.7220**	**0.8061**	0.6536
01 COLMAP	**343.4**	0.5643	0.4823	0.6798
01 Proposed	356.5	0.6421	0.5488	**0.7736**
02 PIX4D	**83.4**	**0.7063**	**0.7687**	0.6533
02 COLMAP	216.9	0.6845	0.5996	0.7973
02 Proposed	194.4	0.6279	0.5197	**0.7973**
03 PIX4D	**106.1**	0.5071	**0.6618**	0.4110
03 COLMAP	251.7	**0.6239**	0.5883	**0.6641**
03 Proposed	264.5	0.4395	0.3475	0.5977

Table 6-2: Dense 3D modeling results on the Vicon Room 1, data set 01, 02 and 03 (top to bottom). Showing the dense modeling time, F_1, precision and recollection scores using a threshold τ of 0.02 (m).

a better reconstruction on the easy data set when compared to COLMAP, the performance degrades on the more difficult data sets 02 and 03. From the previous comparison, it was noted that the pose accuracy degraded in the proposed approach and COLMAP delivered better results in terms of APE. Interestingly, the RPE of the proposed approach remained better than COLMAP. Since the reconstruction quality of COLMAP was higher on the data sets 02 and 03, the effect of the APE is apparently a more important predictor of reconstruction quality.

Now that the proposed method has been tested without leveraging the acceleration scheme introduced in Chapter 5, it is time to introduce this in the proposed architecture and assess the results on the EuRoC MAV data set.

6-5-2 Accelerated dense modeling

In this section, the acceleration scheme will be added to the proposed architecture. Using the sparse reconstructions of the proposed architecture, a set of dense reconstructions are computed on the EuRoC data set. All reconstructions are computed on the hardware set-up that is described in Appendix B. For this experiment, only the 01 data set is used since it is captured in a controlled fashion. This is more representative of a real-life scenario where a drone flies a controlled flight pattern around a cellular tower. The reconstructions of the proposed scheme will be compared against PIX4D and COLMAP in terms of the model quality and computation time. It will be studied whether the proposed architecture is capable of delivering comparable model quality at decreased computation times.

To use the acceleration scheme, the first step is to determine how many super-pixels should be used to properly capture the details in the EuRoC MAV benchmark. In Section 5-2-3, it was established that the number of super-pixels are dependent on the pixel-size of the smallest details that need to be retained. For the EuRoC MAV Vicon Hall 1 data set, p_{min} was chosen to be 19 px. This value captures most of the details in the images. This is shown in Figure 6-4.

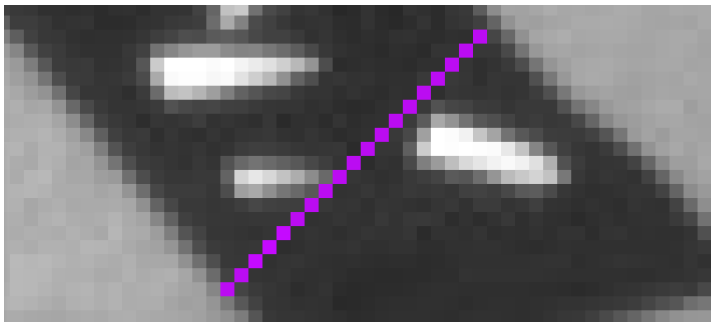

Figure 6-4: Choosing p_{min} on the EuRoC MAV Vicon Hall 1 data set. Using 19 pixels seem to capture the majority of the fine details in an image.

Now that p_{min} has been set, the number of super-pixels can be computed. Filling in Equation (5-7) yields

$$K_{min} = \frac{w \cdot h}{p_{min}^2} = \frac{752 \cdot 480}{196} = 999.89. \tag{6-2}$$

This result is rounded up to 1000 super-pixels. An example super-pixel mask extracted from using 1000 super-pixels is depicted in Figure 6-5.

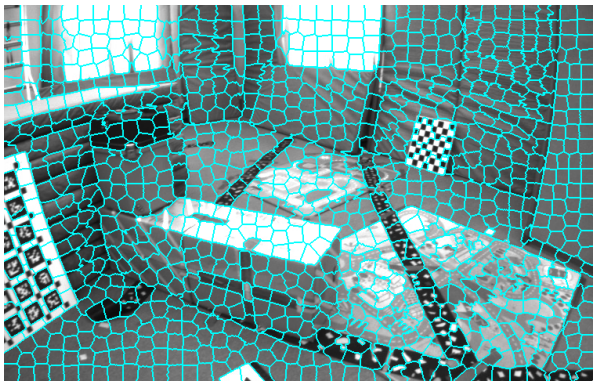

Figure 6-5: 1000 super-pixels extracted on an input-image in the EuRoC MAV Vicon room data set 01.

Using this amount of super-pixels and requiring at least a single sample in each cluster, the maximum sampling-rate can be calculated using Equation (5-22) as

$$k_{max} = \max\left(1, \sqrt{\frac{p_{min}^2}{k_{min}}}\right) = \max\left(1, \sqrt{\frac{19^2}{1}}\right) = 19, \tag{6-3}$$

which is more than the maximum sampling-rate of $k = 8$. Now that the parameters have been set, a set of reconstructions can be computed using the proposed scheme. Starting from processing 100 (%) of the pixels, the downscale factor S is decreased in each new reconstruction. This effectively limits the number of processed pixels. In this experiment four reconstructions are computed for $S = \{1, 0.5, 0.25, 0.125\}$ where the downscale factor was equal to the inverse

of the sampling-rate $S = 1/k$. This means that the four reconstructions are computed using 100 (%), 25 (%), 6.25 (%) and 1.56 (%) of the pixels. For each reconstruction, the same evaluation procedure is used as described in Section 6-5-1 using a threshold τ of 0.02 (m). The results are shown in Table 6-3.

S	Processed Pixels	F_1	Precision	Recollection	Time Excl. Interpolation	Time Incl. Interpolation
1	360,960	0.642	0.549	**0.774**	343.4	343.4
0.5	90,240	0.591	0.483	0.764	162.3	607.3
0.25	22,560	0.612	0.506	0.774	80.6	259.7
0.125	**5,640**	**0.652**	**0.593**	0.761	**47.1**	**117.3**

Table 6-3: Table showing the dense 3D model quality for different down-scale factors S. For each reconstruction, the number of processed pixels per image, precision, recollection and F_1 scores are computed for a threshold τ of 0.02 (m). Besides this, the computation time defined in minutes is shown including and excluding the interpolation step.

From Table 6-3 it can be concluded that the proposed scheme works as expected on the EuRoC Vicon Room data set. By processing only 5,640 instead of 360,960 pixels per image, the dense 3D modeling time excluding the interpolation time has been decreased from 343.4 to 47.1 (min) with a factor 7. Although not the focus of the work, since the implementation of the dense depth estimation scheme was written in non optimized code, a speed-up of a factor 3 was achieved when including the dense depth estimation. Furthermore, the F_1 score did not only decrease, it even slightly increased. In particular, the precision score improved slightly. This is an unexpected result since the reconstruction was obtained by relying on less information. From Table 6-3 it can be observed that the higher F_1 score is caused by an increase in precision whilst the recollection score remained approximately constant (0.7736 - 0.7612). Interestingly, the precision score decreased for $S = 0.5$ and then increased slowly. This effect may be caused by the suppression of noise due to an increase in super-pixel size. Besides this, the recollection score did not decrease either when decreasing the factor S. The results are further visualized in Figure 6-6.

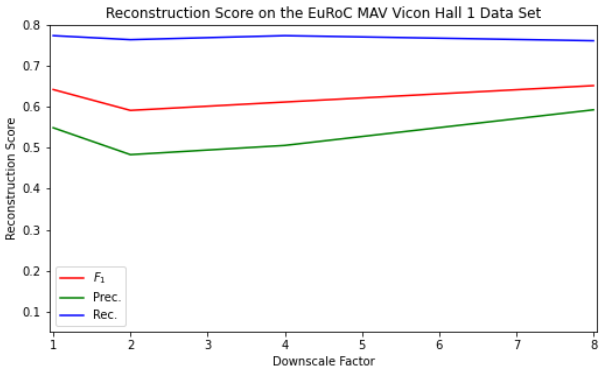

Figure 6-6: Reconstruction metrics for different downscale factors $(1/S)$. For each reconstruction the F_1 (red), Precision (green) and Recollection (blue) score is computed. The reconstructions were computed using the Vicon Hall 1 data set.

Although the metrics indicate that the sparsest reconstruction, processing 1.56 (%) of the pixels, shows comparable performance, it is important to also visually showcase the results. To this end, all four reconstructions are shown in Figure 6-7.

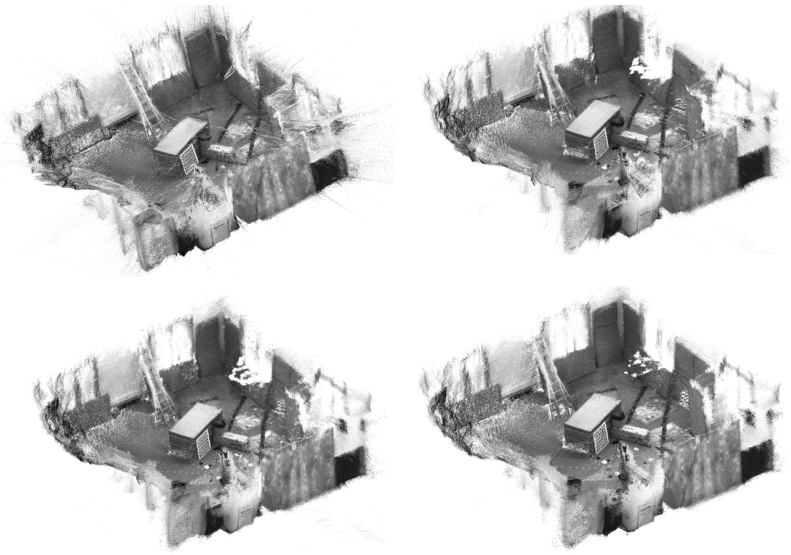

Figure 6-7: Reconstructions of the Vicon Hall 1 data set of the EuRoC benchmark [9]. Showing the reconstruction where 100 (%) of the pixels (top-left), 25 (%) of the pixels(top-right), 6.25 (%) of the pixels (bottom-left) and 1.56 (%) of the pixels (bottom-right) are processed.

From Figure 6-7 it can be concluded that the visual results confirm what was shown in the reconstruction scores. Also, the slight increase in precision which caused the higher reconstruction score for $k = 4$ can be explained. For $k = 1$ there is significant noise present at the edges of objects in the room. For example, near the edges of the desk, there is a cluster of noise on both sides. For the reconstruction computed using $k = 4$, a large percentage of noise present at the windows of the room and at the desk is removed. Since this noise is removed, which contain a cluster of outliers, the precision score is increased. Although interesting to observe, in practice, the reconstructions all look visually very comparable. It can be concluded that the first point cloud would need more manual cleaning to improve its usability but the overall result is highly similar. It is noteworthy that even at the coarsest level of computation small details remained preserved. This is depicted in Figure 6-8 .

The ability to retain fine details whilst improving the scalability of the dense 3D modeling architecture is the main objective of the proposed architecture. The results on the EuRoC MAV benchmark seem to confirm that this is indeed achieved. The proposed scheme is able to accurately estimate the dense depth-normal map using the sparse depth-normal information. An example of a full and interpolated depth map is shown in Figure 6-9.

Both results look very similar. The piece-wise planar assumption yields smooth depth-normal maps which preserves sharp edges and details in the scene by leveraging the super-pixel segmentation of the color image. Note that the depth-normal filtering step is not possible in the current implementation. This is causing the absence of the black pixels in the right depth image. The reconstructions are showed side-to-side with PIX4D and COLMAP in Table 6-4.

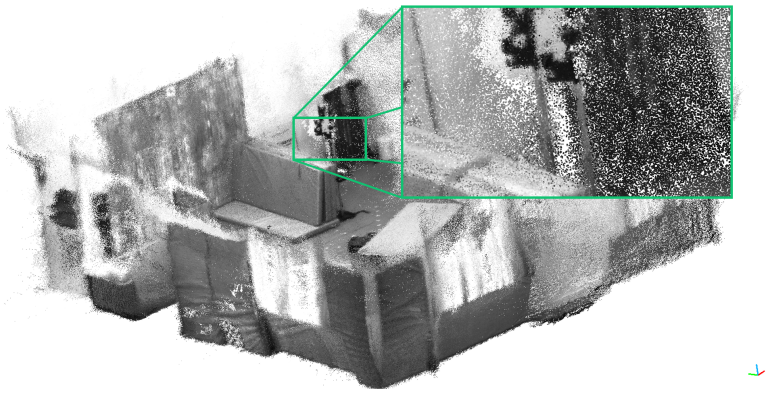

Figure 6-8: Close-up of the reconstruction computed using only 1.56 (%) of the pixels. Even at the sparsest experiment that was performed, the algorithm was capable of capturing fine details such as the pole in the Vicon room.

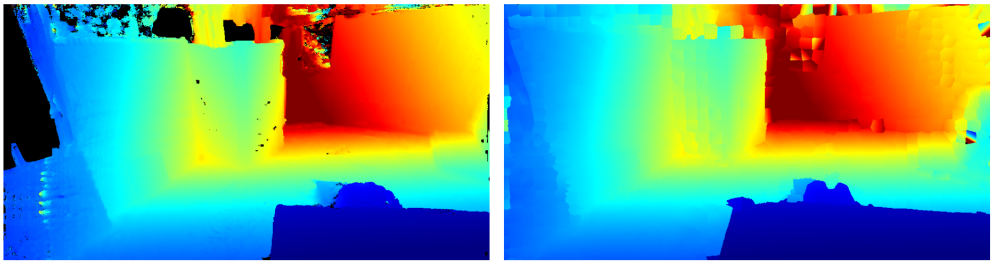

Figure 6-9: Full (left) and interpolated (right) depth-map where the sampling-rate was equal to 2. Filtering is currently not possible in the interpolation scheme which causes several outliers to remain in the final depth-map.

Architecture	F_1	**Precision**	**Recollection**	**Total Time (min)**
PIX4D	**0.7220**	**0.8061**	0.6536	1239.4
COLMAP	0.5643	0.4823	0.6798	1007.2
Proposed: $S = 1$	0.6421	0.5488	**0.7736**	361.6
Proposed: $S = 0.5$	0.5912	0.4832	0.7638	180.5
Proposed: $S = 0.25$	0.6116	0.5057	0.7736	98.8
Proposed: $S = 0.125$	0.6526	0.5705	0.7623	**65.3**

Table 6-4: Dense 3D reconstructions computed on the Vicon Room 1 data set. Showcasing the reconstruction quality and computation times for the current and proposed architecture, computed for different factors S. Comparable reconstructions are obtained at a significant acceleration in computation time.

Now that the performance has been confirmed on the EuRoC MAV Vicon hall data set, a case-study will be performed on a visual-inertial lab recording where the goal is to show the performance of the proposed architecture in a realistic remote inspection scenario.

6-6 Remote inspection case-study

In this section, a case-study will be executed. To this end, Ericsson AB has provided a model cellular tower with corresponding ground-truth measurements. The provided model cellular tower is shown in Figure 6-10.

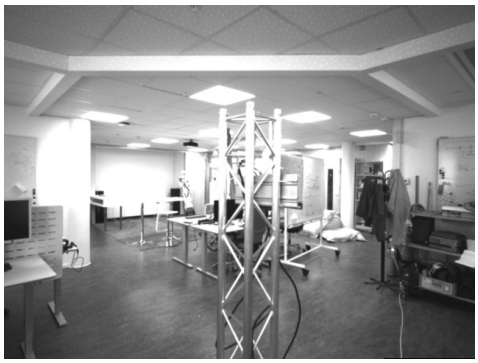

Figure 6-10: Lab set-up showcasing the model cellular tower. Using a synchronized visual-inertial recording, the goal is to compute a dense 3D reconstruction of the model and be able to perform measurements.

This model tower has an antenna mounted on top of it with several cables. Hence, it is a good representation of an actual cellular tower. The goal in this case-study is to compute a dense 3D reconstruction of the model tower with accurate geometric dimensions. There are two criteria that are important in this process. The first aspect is that the reconstruction should run in a time that enables the technician to observe the reconstruction when present at the site. Secondly, the accuracy of the reconstruction should be within the range of the current commercial solution (i.e. PIX4D) which is reported at 2 (cm). Hence, if measurements are performed using the dense 3D point cloud model, the measurements should be accurate within 2 (cm). If this is the case, the performance is comparable to the current architectures.

For this cellular model, no ground-truth point cloud is available. As such, it is not possible to evaluate the reconstruction using the methods outlined in Section 6-4 where point cloud metrics such as F_1 scores could be computed. However, in the context of remote inspection, technicians are typically interested in accurately measuring elements of the tower. For example, one could be interested in measuring the antenna equipment's dimensions. To this end, the following method is outlined to perform and evaluate the case-study. First, a synchronized visual-inertial recording will be obtained from the model tower using a sensor-suite. Next, using the proposed architecture, several reconstructions will be computed. Then, a set of measurements are performed on the reconstruction and compared against measurements obtained by directly measuring the model tower in the lab. In this way, the accuracy of the reconstruction can be determined in the context of a practical remote inspection setting. Within this setting, the goal will be to test whether the presented architecture is capable of computing dense 3D reconstructions at a reduced level of computational complexity without losing the required accuracy of 2 (cm).

6-6-1 Method

To study the reconstruction, several measurements are obtained from the model tower. An overview of the tower is depicted in Figure 6-11.

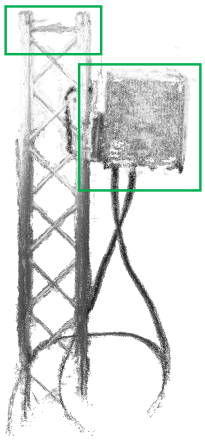

Figure 6-11: Overview of the model tower used for the lab experiment. Two areas are highlighted for which physical measurements are available. These measurements are used as ground-truth dimensional reference.

In Figure 6-11, two areas where the measurements are obtained are highlighted (green rectangles). Both areas are shown in greater detail in Figure 6-12 and 6-13.

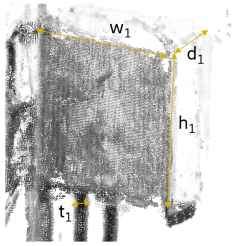

Figure 6-12: Antenna equipment of the model tower. The width w_1, height h_1, depth d_1 and connector thickness t_1 are measured.

As shown in Figure 6-12, in the antenna, a set of four measurements are obtained. Specifically, the width w_1, height h_1, depth d_1 and the thickness of the cable t_1 is measured.

From Figure 6-13, it can be concluded that a set of four measurements are extracted - the width of the tower w_2, depth of the tower d_2, the thicknesses of the main bar element t_2 and thin bar element t_3. The complete set of measurements is summarized in Table 6-5.

Element	w_1	h_1	d_1	t_1	w_2	d_2	t_2	t_3
Dimension (cm)	30	38.5	12.5	26.5	26.5	1.5	4.5	4.5

Table 6-5: Obtained measurements of different elements of the model tower.

From the reconstructed dense 3D models, these measurements will be extracted as well by measuring the point cloud models and compared against the actual measured distances. The

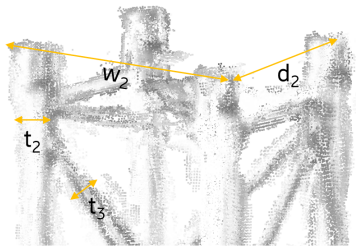

Figure 6-13: Top structure of the model tower. The width w_2, depth d_2, bar thicknesses t_2 and t_3 are measured, respectively.

goal will be to accurately extract the correct dimensions within 2 (cm). In this case, the measurement satisfies the industry requirements and make the dense 3D model usable. All reconstructions are computed using the hardware set-up that is described in Appendix B.

To acquire the visual-inertial recording, the Intel Realsense D435i sensor-suite is used. For this experiment, Ericsson AB captured a series of visual-inertial recordings where the visual sensor was set to capture 640×480 grayscale images using the global shutter camera. The visual recordings are synchronized with the Inertial Measurement Unit (IMU). The complete Realsense sensor-suite is then calibrated using Kalibr [14, 15, 28]. To acquire the data, the Realsense camera is moved around the model tower in a circular pattern thereby capturing it from different angles. Then, this visual-inertial data is recorded and processed using the proposed visual-inertial architecture.

Using the recordings, for different downscale factors S, several 3D reconstructions are computed. Specifically, four reconstructions are computed where $S = \{1, 0.5, 0.25, 0.125\}$. To choose the required number of super-pixels, the minimum pixel size p_{min} has to be determined. For the lab-scene, similar to the EuRoC data set, 17 pixels capture the majority of the fine details. This is visualized in Figure 6-14.

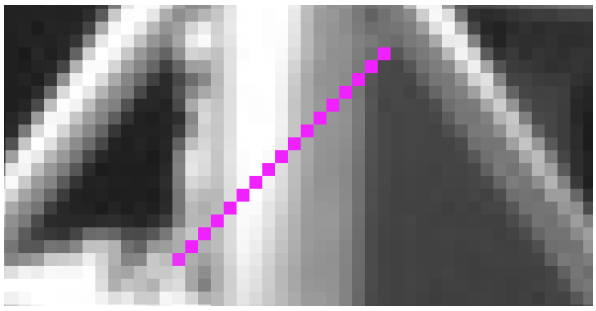

Figure 6-14: Choosing p_{min} on the model-tower data set recorded in the lab. Using 17 pixels seem to capture the majority of the fine details in an image.

Filling in Equation (5-7) yields

$$K_{min} = \frac{w \cdot h}{p_{min}^2} = \frac{640 \cdot 480}{17^2} = 1062. \tag{6-4}$$

This result is rounded to $K = 1000$ super-pixels. Using this amount of super-pixels and requiring at least a single sample in each cluster, the maximum sampling-rate can be calculated

using Equation (5-22) as

$$k_{max} = \max\left(1, \sqrt{\frac{p_{min}^2}{k_{min}}}\right) = \max\left(1, \sqrt{\frac{17^2}{1}}\right) = 17, \tag{6-5}$$

which is sufficient for the maximum sampling-rate of $k = 8$ which corresponds to the case $S = 0.125$. The reconstructed dimensions at the reference locations depicted in Figures 6-12 and 6-13 are extracted and compared against the values of Table 6-5. The goal will be to assess within which accuracy the dimensions are reconstructed. Specifically, the accuracy of the dimensions computed using the super-pixel based acceleration scheme for different sampling-rates will be studied.

6-6-2 Results

The set of reconstructions that are obtained using the proposed architecture are depicted in Figure 6-15.

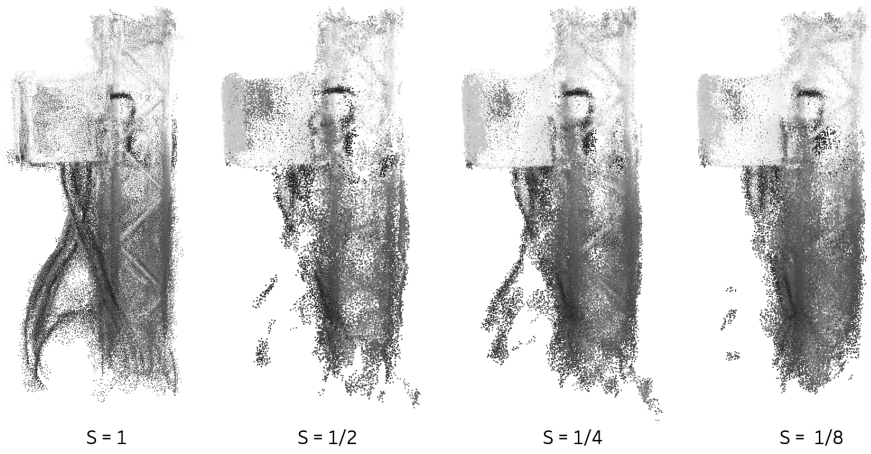

Figure 6-15: Reconstructions computed using the proposed visual-inertial pipeline for different downscale factors S. The reconstructions have been cleaned by removing the surrounding environment of the lab.

Unfortunately, the resolution of 640×480 was clearly not high enough to produce crisp point cloud models from the tower. This can be seen in Figure 6-15 due to the presence of noise between the bar elements of the tower. Due to the low resolution, the dense 3D modeling pipeline is unable to properly differentiate the background pixels from the pixels that belong to the bar elements. This stems from the fact that, in the PatchMatch scheme, the image patches that are used to compute the photo-consistency scores between the reference and source images are too large with respect to the fine details in the scene. This effect causes ambiguity in the matching step resulting in faulty points between the thin bar elements. Although the point cloud models are noisy, they can still be used to measure the dimensions of the tower and the equipment. This is achieved by directly measuring the point cloud models. Comparing the extracted dimensions with the measured ground-truth dimensions yields a set of errors for each point cloud model. These errors are shown for the four reconstructions in Figure 6-16.

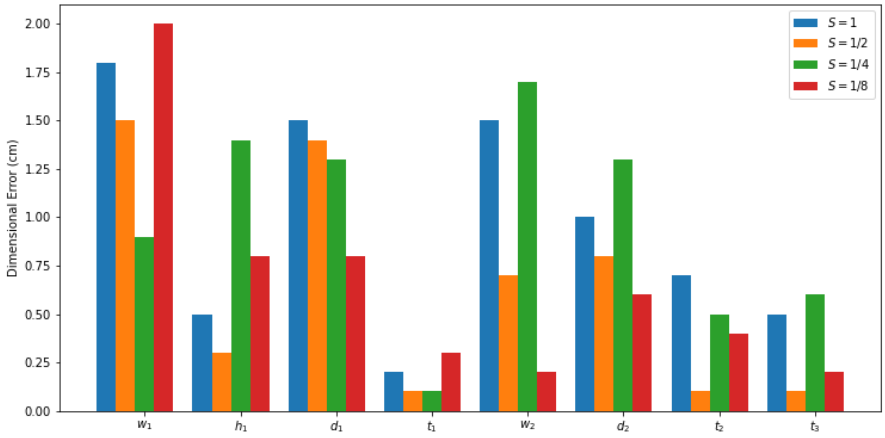

Figure 6-16: Errors showcasing the difference between the measured and reconstructed elements of the model tower. The errors are shown for different reconstructions computed at different downscale factors S.

From Figure 6-16 it can be concluded that across all downscale factors S, the dimensions are reconstructed within the 2 (cm) accuracy bound. This means that, even at the sparsest level, the dimensions of the model-tower were reconstructed within the current industrial standard. Now that the accuracy has been evaluated, it will be interesting to compare the computation times. These are depicted in Table 6-6.

S	Total Time (min) Including Interpolation	Total Time (min) Excluding Interpolation
1	121.8	121.8
0.5	142.7	56.1
0.25	72.5	29.7
0.125	**21.6**	**16.3**

Table 6-6: Total computation time excluding and including the interpolation time for different downscale factors S on the lab-recording.

Whilst all reconstructions yielded dimensions within 2 (cm) accuracy, the computation times were significantly different. This can be seen in Table 6-6. For $S = 0.125$, the total computation time, excluding interpolation, was equal to 16.3 (min). This is a factor 7.5 faster than then current approach, which is identical to the scenario of $S = 1$. These results indicate the advantages of the presented approach, which reduced the computation time whilst delivering dense 3D models with an accuracy that is within the required specifications.

Chapter 7

Conclusions and Future Work

7-1 Conclusions

In this thesis, a novel monocular 3D modeling architecture was presented. The presented architecture is capable of modeling scenes with accurate geometric scales in times that are unparalleled in the current state-of-the-art approaches. In this thesis, the state-of-the-art systems were chosen to be PIX4D and COLMAP, the industrial and academic standard in monocular 3D modeling, respectively. The presented architecture achieves a significant increase in performance over the current state-of-the-art systems in two steps.

First, instead of using only monocular visual data, inertial data was included in the architecture. The inclusion of inertial information made it possible to leverage efficient frameworks developed for real-time robotics purposes which have not been widely used in the context of dense, monocular 3D reconstruction. To this end, the Maplab framework was chosen for this use-case which offers a state-of-the-art visual-inertial sparse mapping architecture that has been widely tested on several robotics systems such as micro aerial vehicles. This architecture demonstrated to be capable of, under controlled flight patterns, compute more accurate sparse 3D reconstructions and poses of a scene when compared to PIX4D and COLMAP. In benchmark experiments, the proposed architecture was shown to achieve a speed up of a factor 49 and 37 in the sparse 3D reconstruction process when compared to PIX4D and COLMAP, respectively.

Secondly, a novel dense 3D modeling system was presented. The presented system was capable of reconstructing dense 3D models at unprecedented computation times while keeping an acceptable loss of model quality. To achieve this, the computational complexity of the classical Multi View Stereo (MVS) problem was reduced by leveraging local depth-color correlations and assuming a planar surface within these regions. By using these assumptions, local similar colored image regions were extracted by computing a set of super-pixels. The super-pixels were capable of providing a compact, powerful representation of the image at only a fraction of the original image size. The problem in MVS to compute a depth-normal value on a pixel level was then reduced to finding a depth-normal value for at least one pixel within a super-pixel due to the planar surface approximation. To leverage the reduction in computational

complexity, in this thesis, the main code of COLMAP was adjusted. Specifically, a sampling-rate k was introduced which defined that computations were only performed on a uniform grid of pixels by processing every k^{th} pixel. This yielded a sparse depth-normal image which was densified by robustly estimating a set of planes for each super-pixel. Effectively, by setting a larger sampling-rate k, the number of processed pixels is reduced by a factor k^2. The presented densification system was tested on aMVS benchmark and the results were significant. At a sampling-rate of 8, the densification step was accelerated by a factor 8 and the number of processed pixels was reduced by a factor 64 - from 100 (%) to 1.56 (%). The dense 3D reconstruction quality, defined by the F_1 score, decreased from 0.77 to 0.70.

Both Maplab's visual-inertial framework and the novel dense 3D modeling system were then combined to yield the final presented system which was shown in Figure 5-16. The presented system was tested on the EuRoC Micro Aerial Vehicle (MAV) benchmark and a remote inspection case-study. On the EuRoC MAV Vicon Hall 1 data set 01, COLMAP computed a dense 3D reconstruction in 1007.1 (min), PIX4D in 1239.4 (min) and the presented system, using a sampling-rate of 8, in 65.3 (min) and 135.5 (min) by excluding and including the plane-estimation step, respectively. Furthermore, it was able to compute 3D reconstructions with accurate geometric scale without relying on external metadata such as a Global Positioning System (GPS). Although an increase in robustness was expected by including the Inertial Measurement Unit (IMU) as input to the system, no experiments were performed to validate this assumption. Furthermore, in the challenging EuRoC MAV Vicon Hall 1 data set 03, the proposed system reconstructed a dense 3D model with a F_1 score of 0.44. This was lower compared to PIX4D and COLMAP which produced reconstructions with a score of 0.51 and 0.62, respectively. The decrease in performance was observed due to the fact that Maplab's front-end is running online. In this specific data set, the drone followed a fast and uncontrolled trajectory with many sudden movements. Since both PIX4D and COLMAP are running offline, the performance is less influenced in such a scenario. However, in practical reconstruction scenarios, the MAV would fly a slow and controlled trajectory around the scene. This scenario would closely resemble the data set 01 where the presented system managed to outperform PIX4D and COLMAP on all metrics. Besides the benchmark experiment, a remote-inspection case-study was performed. From this case-study, it was concluded that the proposed system was able to decrease the computation time with a factor 5.6, from 121.8 (minutes) to 21.6 (minutes) whilst reconstructing a dense 3D point cloud model with accurate geometric scale that was accurate within 2 (cm).

From this thesis, it can be concluded that the presented visual-inertial system is capable of mitigating the typical issues in the current state-of-the-art approaches since it is capable of directly observing geometric scale and scales better with the input data in terms of computational complexity.

7-2 Future work

For future work it will be interesting to study the dense 3D modeling system in its minimal representation: only computing pixels that correspond to the centroids of super-pixels. This is the minimal case since one sample needs to be processed in each super-pixel. Furthermore, throughout all the experiments, the maximum sampling-rate was equal to 8. In this thesis, the code of COLMAP was adjusted and used to compute sparse depth-normal

images. Although the results are promising, due to COLMAP's parallel implementation, the complexity reduction is not fully leveraged. Since COLMAP process one image row at a time, the observed speed-up was proportional to k even though the total number of computations was reduced by a factor k^2. Other MVS approaches, such as Gipuma [18], have different computation schemes that could fully leverage the complexity reduction. It is expected that for these approaches, the speed-up will be proportional to a factor k^2 which will improve the results even more.

Besides fully leveraging the possible complexity reduction by either further limiting the number of processed pixels or using a different parallel computation scheme, another interesting subject will be to change the matching window. The matching window, or support region in the current implementation was defined as a rectangular patch of pixels surrounding the pixel of interest. This patch was matched across different images to find a depth-normal hypothesis that maximized the photo-consistency across the image sets. It will be interesting to study whether using the super-pixels directly as one patch will improve the results by removing much of the ambiguity in this patch-matching process. The removal of ambiguity is expected since an envisioned super-pixel patch will account for naturally present boundaries and edges which rectangular patches do not have.

Intuitively, it can be thought of as instead of performing a sparse set of computations on a pixel level with rectangular support regions and collecting those results within each super-pixel in a post-processing step, to directly perform the photo-consistency computations on a super-pixel level. At the most fundamental level, it can be seen as changing the classical objective of MVS. Instead of computing a set of depth-normal values for each pixel, the goal becomes to compute a set of planar super-pixels. Then, it becomes obvious that one could move away from using rectangular patches as support regions for super-pixels but instead, directly use the super-pixel itself. Moving in this direction, it can be envisioned that the complexity reduction and accuracy can be increased even further, when compared to using a post-processing step. The fundamental assumption in the PatchMatch stereo algorithm is that of structured region information, the idea that local clusters of pixels in an image typically share similar depth-normal values. The key-insight in this thesis is that this characteristic is not properly leveraged in all state-of-the-art MVS algorithms since they operate on the pixel level using standard support regions. There is a significant gain in computational complexity reduction to be made by more effectively leveraging structured region information in images where this work provides a first step in this direction.

Finally, the presented system was only tested in ideal lightening conditions or in indoor settings. As the goal is to compute reconstructions in outdoor settings, the weather may also play an important role. For example, bright sunlight, fog or heavy rain may influence the 3D model quality. In this work, these effects were not studied properly. Although an increase in robustness in challenging visual situations was expected due to the integration of an IMU, it was not studied in depth in this work. It will be interesting to test the presented system in these realistic settings and study the performance. Essentially, to further study the differences in robustness of the presented system with the baseline architectures.

Appendix A

Appendix A: The Projection Model

The projection model describes how 3D observations are related to 2D image observations. In essence, it describes the measurement model of a camera which is the cornerstone of every vision based reconstruction algorithm. Every projection model consists of two elements - a camera model and a distortion model. These will be treated sequentially. The goal of this appendix is to provide the interested reader with an overview of the most commonly used projection models and point towards more complete references describing the variety of different models.

A-1 The Camera Model

The camera model, assuming no distortion, relates how 3D points are projected on the image plane in the form of observations. Specifically, it is of interest to obtain a camera model $f(\cdot)$ that maps a scene point to the image plane. In digital cameras, observations are represented as pixels where each pixel is a discrete point in the image plane. Such pixel observations typically contain greyscale intensity values or three dimensional color values, for example, RGB or LAB colors. The 2D image plane describes the location of the observations with a coordinate system that is aligned with the image plane. The convention is to place the center of this coordinate system at the top-left of the image plane. This is depicted in Figure A-1.

To relate 3D points to the pixel locations in the image plane, typically, a pinhole camera model is used. The pinhole camera model has a center point defined as the projection center. The camera's coordinate system is then defined at this location with the z-axis pointing towards the image plane. A 3D point defined in the world coordinate system is then projected to the image plane in two steps. First, it is transformed to the camera's coordinate system and secondly it is projected on the image plane via the pinhole model. This is depicted in Figure A-2.

The first transformation maps a point from the world's coordinate system, defined by the superscript $^{(w)}$, to the camera's coordinate system, denoted by the superscript $^{(c)}$, via a homogeneous transformation defined as the pose $\mathbf{T}_j \in \mathrm{SE}(3)$ of camera j

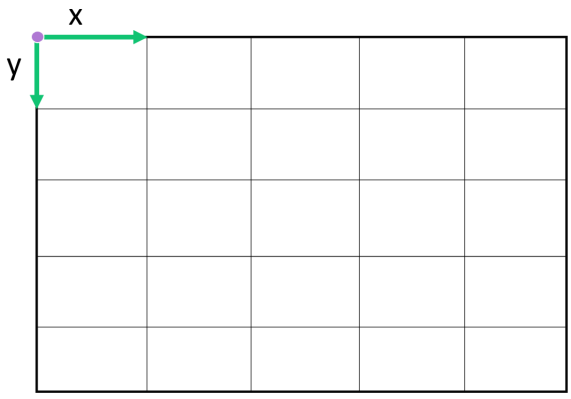

Figure A-1: The image plane of a digital camera. The coordinate system is defined from the top-left of the image plane. In this example, the image plane consists of 25 pixels.

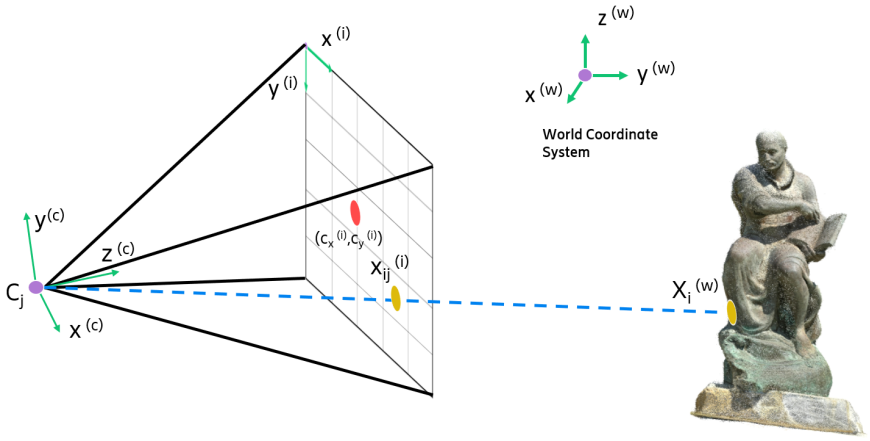

Figure A-2: Overview of the pinhole camera model of camera C_j showcasing the camera's projection center (purple) and the principal point (red). Coordinate systems are represented by superscripts. A scene point $\mathbf{X}_i^{(w)}$ (yelllow) represented in the world coordinate system is projected in the image plane where it is observed as $\mathbf{x}_{ij}^{(i)}$ in two steps.

$$\begin{pmatrix} \mathbf{R}_j & \mathbf{t}_j \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}, \tag{A-1}$$

where $\mathbf{R}_j \in \mathrm{SO}(3)$ and $\mathbf{t}_j \in \mathbb{R}^3$ are the rotation matrix and translation vector, respectively. Next, the scene point defined in the camera's coordinate system is mapped to the image plane, denoted by the superscript $^{(i)}$, via the pinhole camera projection matrix $\mathbf{K}_c \in \mathbb{R}^{3\times 4}$

$$\mathbf{K}_c = \begin{pmatrix} f_x & 0 & \bar{c}_x & 0 \\ 0 & f_y & \bar{c}_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{A-2}$$

where f_x and f_y are the focal lengths described in pixel units over the x- and y-axis, respectively and $\bar{c}_x$ and $\bar{c}_y$ are defined as the principal point location. The parameters $\bar{c}_x$ and $\bar{c}_y$

describe the distance between the camera projection center and the image coordinate system (located at the top-left of the image-plane). Combining the transformations $\mathbf{T}_j$ and $\mathbf{K}_c$ yields a complete camera model

$$\mathbf{x}_{ij}^{(i)} = f\left(\mathbf{K}_c, \mathbf{T}_j, \mathbf{X}_i^{(w)}\right), \tag{A-3}$$

which describes how 3D observations are observed in the image plane. However, this model assumes that there exists not distortions in the camera. In practice, every camera has some form of distortions. Such distortions change the way that 3D points are observed in the image plane and need to be modeled properly, depending on the used camera model.

A-2 The Distortion Model

As mentioned before, the pinhole camera is an ideal model that assumes no form of distortion or other nonlinearities. In practice, there are many forms of distortions but most camera's can be adequately described with two - radial and tangential distortion.

Radial distortion is visualized in Figure A-3.

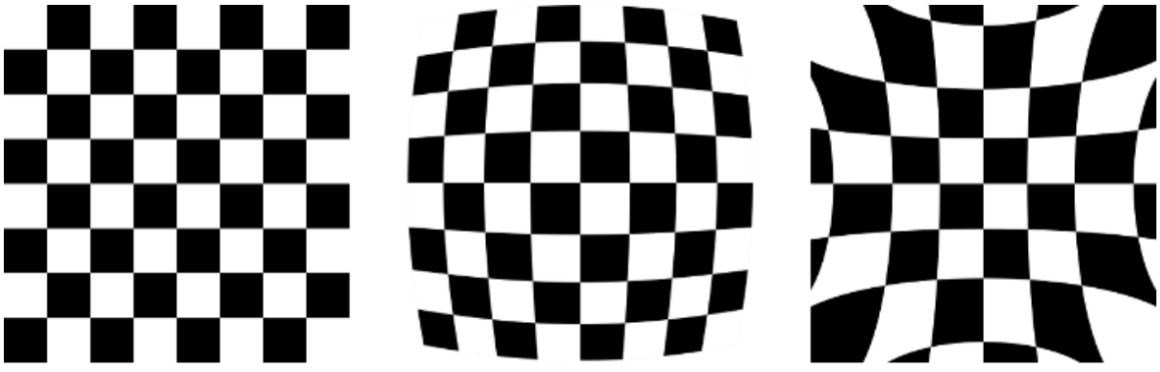

Figure A-3: No distortion (left), positive radial or barrel distortion (middle) and negative radial or pincushion distortion (right). The figure was obtained from [7].

The second distortion is defined as tangential distortion. This distortion is caused by the misalignment of the camera's lens with the image plane [8]. Tangential distortion is visualized in Figure A-4.

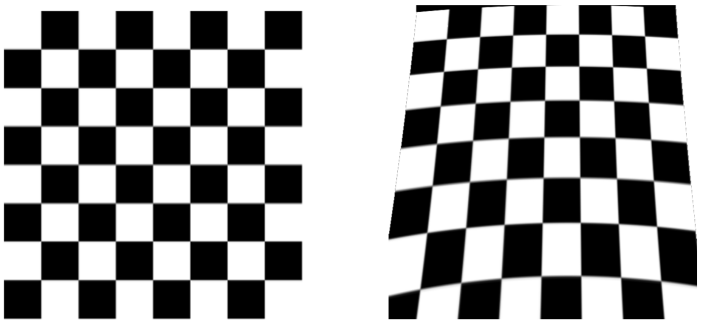

Figure A-4: No distortion (left), tangential distortion (right).

Most camera models combine radial-tangential distortion models. This was first introduced by D. C. Brown et al. [8, 10] and has proven to be a highly effective way of modeling the distortion effects that are typically observed. Although the radial-tangential model is the most widely used, other variants exists depending the specific use-case. For example, for fish-eye lens camera's, the radial-tangential model fails to properly capture the complete distortion and more suitable alternatives exist [23]. For a more complete overview of different distortion models the reader is referred to [42].

When the radial-tangential distortion model is assumed, it becomes possible to define the projection model with the inclusion of distortions. This will move a projected, undistorted scene point $\mathbf{x}_{ij} = (x, y)^T$ to its distorted location $\mathbf{x}'_{ij} = (x', y')^T$

$$\begin{aligned} x' &= x\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + 2p_1xy + p_2\left(r^2 + 2x^2\right), \\ y' &= y\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + p_1\left(r^2 + 2y^2\right) + 2p_2xy, \end{aligned} \tag{A-4}$$

where $r^2 = x^2 + y^2$, k_1,k_2,k_3,k_4,k_5, and k_6 are the radial distortion coefficients and p_1 and p_2 are the tangential distortion coefficients.

Using (A-3) and (A-4), it becomes possible to define a mapping that will project a scene point $\mathbf{X}_i^{(w)}$ to an observation $(\mathbf{x}'_{ij})^{(i)}$. This is achieved by substituting (A-3) in (A-4).

Appendix B

Appendix B: Hardware Set-up

This appendix describes the hardware platform that was used throughout this thesis as described in Table B-1.

Lenovo Legion T730-28ICO	
RAM Memory	31.3 GiB
CPU	Intel Core i7-9700K CPU@ 3.60GHz × 8
GPU	NVIDIA GeForce RTX 2080 SUPER/PCIe/SSE2
GNOME	3.28.2
OS	Ubuntu 18.04 LTS
OS Type	64 bit

Table B-1: Hard- and software-specifications of the computing platform used throughout this work.

Bibliography

[1] PIX4D. https://www.pix4d.com.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a Day. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 72–79, 2009.

[4] S. Bianco, G. Ciocca, and D. Marelli. Evaluating the Performance of Structure from Motion Pipelines. *Journal of Imaging*, 4(8):98, 2018.

[5] M. Bleyer, C. Rhemann, and C. Rother. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *Proceedings of the British Machine Vision Conference*, pages 14.1–14.11, 2011.

[6] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust Visual Inertial Odometry Using a Direct EKF-based Approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 298–304, 2015.

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[8] D. C. Brown. Close-range Camera Calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[9] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC Micro Aerial Vehicle Datasets. *The International Journal of Robotics Research*, 35:1157–1163, 2015.

[10] T. A. Clarke and J. G. Fryer. The Development of Camera Calibration Methods and Models. *The Photogrammetric Record*, 16(91):51–66, 1998.

[11] Z. Cui, L. Heng, Y. C. Yeo, A. Geiger, M. Pollefeys, and T. Sattler. Real-Time Dense Mapping for Self-Driving Vehicles using Fisheye Cameras. In *Proceedings of the International Conference on Robotics and Automation*, pages 6087–6093, 2019.

[12] J. Delmerico and D. Scaramuzza. A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2502–2509, 2018.

[13] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017.

[14] P. Furgale, T. D. Barfoot, and G. Sibley. Continuous-time Batch Estimation Using Temporal Basis Functions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2088–2095, 2012.

[15] P. Furgale, J. Rehder, and R. Siegwart. Unified Temporal and Spatial Calibration for Multi-sensor Systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286, 2013.

[16] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards Internet-scale Multi-view Stereo. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1434–1441, 2010.

[17] Y. Furukawa and C. Hernández. *Multi-View Stereo: A Tutorial*. Now Foundations and Trends, 2015.

[18] S. Galliani, K. Lasinger, and K. Schindler. Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015.

[19] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (4th Edition)*. Pearson Education (US), 2017.

[20] M. Grupp. EVO: Python Package For The Evaluation Of Odometry And SLAM. https://github.com/MichaelGrupp/evo, 2017.

[21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[22] P. J. Huber. Robust Estimation Of A Location Parameter. In *Breakthroughs in Statistics*, volume II. Springer, 1992.

[23] C. Hughes, M. Glavin, E. Jones, and P. Denny. Review of Geometric Distortion Compensation in Fish-eye Cameras. In *Proceedings of the IET Irish Signals and Systems Conference*, pages 162–167, 2008.

[24] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. SurfaceNet: An End-to-End 3D Neural Network for Multiview Stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2326–2334, 2017.

[25] A. Knapitsch, J. Park, Q. Zhou, and V. Koltun. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017.

[26] C. Leng, H. Zhang, B. Li, G. Cai, Z. Pei, and L. He. Local Feature Descriptor for Image Matching: A Survey. *IEEE Access*, 7:6424–6434, 2019.

[27] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[28] J. Maye, P. Furgale, and R. Siegwart. Self-supervised Calibration For Robotic Systems. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 473–480, 2013.

[29] O. Özyesil, V. Voroninski, R. Basri, and A. Singer. A Survey Of Structure-from-Motion. *Acta Numerica*, 26:305–364, 2017.

[30] A. Poms, C. Wu, S. Yu, and Y. Sheikh. Learning Patch Reconstructability for Accelerating Multi-view Stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3041–3050, 2018.

[31] K. Prokopetc and R. Dupont. Towards Dense 3D Reconstruction for Mixed Reality in Healthcare: Classical Multi-View Stereo vs. Deep Learning. In *IEEE/CVF International Conference on Computer Vision Workshop*, pages 2061–2069, 2019.

[32] Ericsson Mobility Report. This Is 5G. https://www.ericsson.com/49df43/assets/local/newsroom/media-kits/5g/doc/ericsson_this-is-5g_pdf_2019.pdf, 2019.

[33] I. Rey-Otero and M. Delbracio. Anatomy Of The SIFT Method. *Image Processing On Line*, 4:370–396, 2014.

[34] P. J. Rousseeuw and C. Croux. Alternatives To The Median Absolute Deviation. *American Statistical Association*, 88(424):1273–1283, 1993.

[35] E. Salahat and M. Qasaimeh. Recent Advances in Features Extraction and Description Algorithms: A Comprehensive Survey. In *Proceedings of the IEEE International Conference on Industrial Technology*, pages 1059–1063, 2017.

[36] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth. In *Proceedings of the Pattern Recognition Conference*, pages 31–42, 2014.

[37] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. Maplab: An Open Framework for Research in Visual-Inertial Mapping and Localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018.

[38] J. L. Schönberger and J. M. Frahm. Structure-from-Motion Revisited. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[39] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys. 3D Modeling on the Go: Interactive 3D Reconstruction of Large-Scale Scenes on Mobile Devices. In *Proceedings of the International Conference on 3D Vision*, pages 291–299, 2015.

[40] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2538–2547, 2017.

[41] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2^{nd} edition, 2010.

[42] Z. Tang, R. Grompone von Gioi, P. Monasse, and J. Morel. A Precision Analysis of Camera Distortion Models. *IEEE Transactions on Image Processing*, 26(6):2694–2704, 2017.

[43] A. Torres. Afraid of heights? Drones, AI and Digitalization To The Rescue! https://www.ericsson.com/en/blog/2020/3/intelligent-site-engineering, March 2020.

[44] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372. Springer Berlin Heidelberg, 2000.

[45] S. Umeyama. Least-squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(76):380, 1991.

[46] Q. Xu and D. Ma. Applications Of Lie Groups And Lie Algebra To Computer Vision: A Brief Survey. In *Proceedings of the International Conference on Systems and Informatics*, pages 2024–2029, 2012.

[47] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust Monocular Epipolar Flow Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1862–1869, 2013.

[48] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. Recurrent MVSNet for High-Resolution Multi-View Stereo Depth Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5525–5534, 2019.

[49] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan. BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1790–1799, 2020.

[50] Y. Yuan, J. Fang, and Q. Wang. Robust Superpixel Tracking via Depth Fusion. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(1):15–26, 2014.

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering
APE	Absolute Pose Error
GPS	Global Positioning System
GPU	Graphical Processing Unit
LiDAR	Light Detection And Ranging
MP	Mega Pixels
DCSC	Delft Center for Systems and Control
DOF	Degrees Of Freedom
EKF	Extended Kalman Filter
EXIF	EXchangeable Image File-format
IMU	Inertial Measurement Unit
IRLSQ	Iteratively Reweighted Least Squares
MAD	Median of Absolute Deviations
MAV	Micro Aerial Vehicle
MVS	Multi View Stereo
NCC	Normalized Cross Correlation
RMSE	Root Mean Squared Error
RPE	Relative Pose Error
SfM	Structure from Motion

SLAM Simultaneous Localization and Mapping

SLIC Simple Linear Iterative Clustering

SE Special Euclidean

SO Special Orthogonal

TU Delft Delft University of Technology

UAV Unmanned Aerial Vehicle

XR EXtended Reality