Towards Robust Dense Visual Simultaneous Localization and Mapping (SLAM)

by

Juan M. Falquez

M.Sc., The George Washington University, 2009

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2018

This thesis entitled:
Towards Robust Dense Visual Simultaneous Localization and Mapping (SLAM)
written by Juan M. Falquez
has been approved for the Department of Computer Science

_____
Professor Christoffer Heckman

_____
Professor Nisar Ahmed

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Falquez, Juan M. (Ph.D., Computer Science)

Towards Robust Dense Visual Simultaneous Localization and Mapping (SLAM)

Thesis directed by Professor Christoffer Heckman

Long-term autonomy is the dream of many roboticists – and if a robotic system can be split into three main categories: perception, planning and control – then the biggest challenges to achieve this dream are undoubtedly faced in perception. Large scale environments that change with time – due to normal operations or even lighting changes – are typical situations the robot would encounter. As such, a Simultaneous Localization and Mapping (SLAM) system that is robust enough to handle many of these conditions is desired.

The objective of this dissertation is to present components that would lead to a robust dense visual SLAM system. It starts by exploring 3D reconstruction algorithms, showing distinctions between local and global methods and presenting an incremental and adaptive global method designed to create depth maps as a robot navigates in space.

It then introduces the concept of sensor fusion, where multiple sensors are joined to provide a higher degree of tracking accuracy. It compares different visual SLAM systems – dense and semi-dense – and shows how the inclusion of an Inertial Measurement Unit (IMU) aids considerably in tracking. It then uses this localization framework in a large scale volumetric mapping system, and shows results for both indoor and outdoor environments using real world datasets.

Finally, it explores different error metrics used in direct photometric optimization – the foundation of dense tracking systems. It introduces the Normalized Information Distance (NID), an entropy based metric that is shown to achieve high localization success rate and accuracy even in the face of extreme lighting differences.

Dedication

To my family.

Acknowledgements

My doctorate has been a story of a journey more so than a destination, both in the sense of research – changing focus area; and location – moving from DC to Colorado. However, all throughout this journey the people that I have met and left behind have undoubtedly left a mark in me. From all the professors at The George Washington University who first embarked me on this journey, to the professors at the University of Colorado who welcomed me and made me feel at home in Boulder.

I have also been fortunate to have worked with a great group of people in our lab. I am grateful to all the post-docs that not only have I had the privilege to work with but can also count as friends. And of course, to all my colleagues in the lab who are too numerous to list here, but each and everyone have certainly been with me providing guidance and support to carry me through this long journey.

Finally, I would like to thank my advisor Gabe Sibley for guiding me and having created this extraordinary research environment that allowed me to meet and collaborate with amazing people around the world, as well as work on cutting edge projects with top tech companies.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) has been a well-studied topic for the last couple of decades. Even so, researchers around the world are still developing new approaches today. This is in part possible by advancements in technology, not only in terms of computational power but also with the introduction of new sensors like for example, Light Detection and Ranging devices (LIDARs) or structured-light/time-of-flight cameras.

One of the earliest autonomous navigation systems was the Stanford Cart, developed by Hans Moravec [64] in 1979 as completion for his doctoral work. The system utilized a single TV camera that, by sliding it on a 50cm track, created stereoscopic views. These were then used to create a map, extracting 30 features and finding their correlation over multiple views. The features were then triangulated to estimate their 3D position, which were then used to estimate the cart's position. Systems like these, where only a *sparse* set of features are tracked over time, were popular decades ago in particular for their relative low use of processing power.

In recent years, however, a significant shift has been made by the SLAM community towards the use of what are called *dense* methods. Unlike their sparse counterpart, where features were *indirectly* matched through visual codes called descriptors, dense algorithms operate *directly* over the whole image. This has many advantages both in terms of tracking and mapping.

For instance, dense methods have been proven to be robust against extreme motion blur as seen in work published by [70] and [56], and seen in Figure 1.1. Furthermore, while sparse methods are very sensitive to outliers, the use of the full image for tracking makes dense methods particularly

robust against adverse sensing conditions. It also has the extra benefit that, given they require a full 3D reconstruction of the scene, dense systems lead to better and more complete maps as shown by [69], [84], [44] and [106].

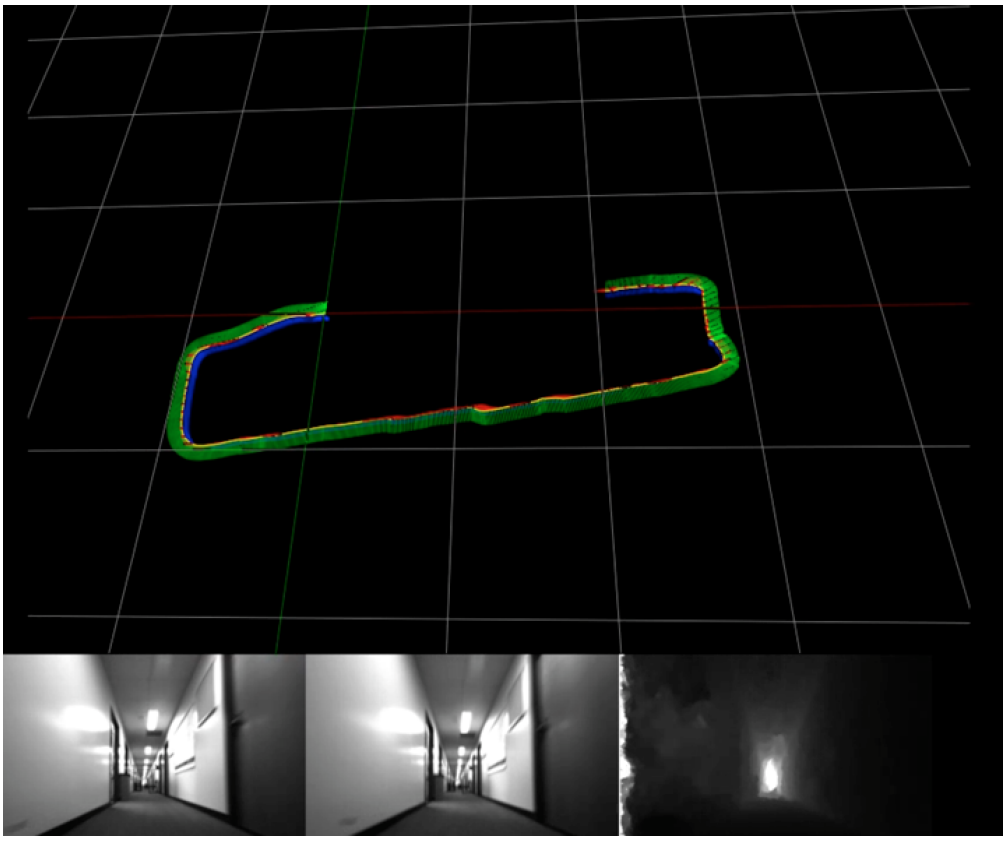


Figure 1.1: Dense techniques are robust against extreme motion blur and specular reflections, as seen in the images on the bottom left and center. The reconstructed depth map using a stereo algorithm is seen on the bottom right. Dense approaches are also globally accurate leading to better overall tracking results.

Dense methods, however, suffer from some drawbacks; the most notable one being that they are computationally very expensive. Sparse methods operate on a few hundred pixels for each image. In contrast, dense methods need to process over 300,000 pixels for a 640x480 resolution image, which quickly becomes intractable as the image resolution is increased.

Nevertheless, the advances in General Purpose Graphics Processing Units (GPGPUs) in the past few years and the highly parallelizable nature of the SLAM problem has allowed researchers

to develop implementations capable of running in real-time like those developed by [105], [104] and [71]. Real-time performance is of great importance in many fields, particularly robotics. Many state of the art algorithms in computer vision, like those used for image segmentation for example, often require hours to process individual frames. Although the results are often impressive, the time requirements of these algorithms prevent their usage in robotic platforms. The trade-off between precision versus performance has always been a compromise roboticists have had to make.

This is evident with a new and recent line of research that focuses on what are called *semi-dense* methods. Like dense, they are a direct method that operate over photometric data. However, the main difference lies in the amount of pixels used: semi-dense only employs pixels with high information (i.e. high gradient areas of the image). The results shown by the work of [22] and [21], and studied in detail in Chapter 4, show that its tracking performance is comparable to fully dense methods. One of the downfalls of semi-dense, however, is that it is not capable of providing a full reconstruction of the scene which is often a necessity in some robotic tasks like manipulation, path planning or obstacle avoidance.

Recently, multi-modal approaches to SLAM have become the norm. In these cases, the system not only operates with visual information from cameras but also fuses other sensor information like Inertial Measurement Units (IMUs), Light Detection and Ranging devices (LIDARs), Global Positioning System (GPS), wheel odometry, etc. These systems not only show higher accuracies but also resilience to failures that would affect a single sensor system if it were exposed to adverse conditions sensitive to that particular sensor [56].

Even more recent, long-term autonomy – both in the areas of mapping and tracking – has been an area of great interest, in particular studying its effects on dense systems. The main limitation of dense algorithms is that they operate directly on photometric information, which means they are extremely sensitive to any change in illumination. Since scene lighting changes with time, this is an intrinsic problem pertinent to long-term autonomy. Recent techniques have been created to mitigate some of these problems, yet this is still an area of active research area.

The objective of this thesis is to go down the journey of dense visual SLAM, starting with its

foundations – the theory behind dense 3D reconstruction, the concept of whole image alignment, the different implementations and metrics used in visual-only systems, and then continue towards more robust versions of dense visual SLAM which make use of multiple sensors jointly optimized in what is often called *sensor fusion*.

This work provides a roadmap for a robust dense visual SLAM system that could be applied to a robotic platform. As such, it is imperative that all the algorithms presented here are capable of operating close to real time and on computing platforms that can be stored on board the robot itself. There is a long journey between theory and practice, especially when attempting to *close the loop*: that is, to have a truly autonomous system capable of sensing, planning and acting. This thesis will only scratch the surface of a fully autonomous robot designed for long-term autonomy, focusing only on the perception component of it – specifically, the mapping and localization.

1.1 Thesis Outline

Chapter 2 provides a theoretical background that covers the basic concepts necessary to better understand subsequent chapters. It will first start by explaining the concepts of dense 3D reconstruction, providing a quick survey between local and global methods and briefly explaining a technique to obtain depth uncertainties for global methods. It will then go into the theory behind dense visual tracking, talking about the image alignment algorithm which is the basis for all direct methods for visual tracking. A quick overview of the forward and inverse compositional approach is presented, and the Efficient Second Order Minimization technique is derived which speeds-up convergence when performing image alignment. Finally, a quick distinction is performed between different mapping representations; specifically, a global versus relative representation. This chapter is particularly important as it familiarizes the reader with the notational conventions and equations commonly used throughout this thesis.

Chapter 3 presents an incremental and adaptive front-end fusion system capable of providing accurate 3D reconstructions of the world using a monocular camera. The algorithm expands on previous volumetric variational approaches for 3D reconstruction by providing two main key

features. The first is a novel incremental method for updating the cost volume which removes the need of keeping hundreds of multi-view comparison images, thus reducing the overall processing time and memory storage of the system. The second feature is a method for dynamically adapting the minimum and maximum depth limits of the cost volume as it adjusts to changes in scene depth, thus achieving optimum resolution in the 3D reconstruction.

Chapter 4 gives an evaluation of different direct methods for computing frame-to-frame motion estimates of the most commonly used multi-sensor rig: a camera capable of providing images and depth maps, and an inertial measurement unit (IMU). In particular, semi-dense and fully dense tracking methods – with and without the aid of an IMU – are compared to see how they perform with respect to changes in image resolution, shutter speed, frame-rates, as well as image and depth noise. This chapter starts to give insight into how the addition of multiple sensors can lead towards robust dense visual SLAM.

Chapter 5 jumps into an actual implementations of multi-sensor dense SLAM. It presents a visual-inertial system capable of mapping large areas using a clever volumetric map representation that only allocates memory when needed. One of the limitations of this representation, however, is that it only stores geometric information and not the actual photometric information necessary for map registration. Crucially, the system is able to leverage inertial measurements for robust tracking when visual measurements do not suffice. Results demonstrate effective operation with simulated and real data, and both indoors and outdoors under varying lighting conditions.

Finally, Chapter 6 tackles the problem of making visual SLAM more robust to extreme illumination changes by comparing tracking algorithms that use different error metrics. A novel algorithm is introduced which speeds-up considerably the computation time of a whole image alignment optimization using the Normalized Information Distance (NID) metric, an entropy based metric proven to be robust to illumination changes.

All of these chapters together create the components necessary to achieve robust dense visual simultaneous localization and mapping.

1.2 Publications

Most of the work described in this thesis appeared in the following publications:

Incremental and Adaptive Front-End Fusion [27]
Juan M. Falquez, Vincent Spinella-Mamo and Gabe Sibley
IEEE International Conference on Robotics and Biomimetics (ROBIO) 2014

Large Scale Dense Visual Inertial SLAM [56]
Lu Ma, Juan M. Falquez, Steve McGuire and Gabe Sibley
International Conference on Field and Service Robotics (FSR) 2015

Inertial Aided Dense & Semi-Dense Methods for Robust Direct Visual Odometry [26]
Juan M. Falquez, Michael Kasper and Gabe Sibley
IEEE International Conference on Intelligent Robots and Systems (IROS) 2016

Robust Dense Visual SLAM Using Spatially Consistent Normalized Information Distance
Juan M. Falquez and Christoffer Heckman
Under submission, International Symposium on Experimental Robotics (ISER) 2018

and was developed in part while collaborating with the following industrial partners:

Toyota - Autonomous Vehicle Research Program
Research grant for robust perception for autonomous driving.

MITRE Corporation
Research grant for uncertainty estimation for joint dense structure from motion.

Google - Advanced Technology and Projects (ATAP)
Project Tango, a mobile platform for indoor navigation, 3D mapping and augmented reality.

Canvas Technology
Autonomous mobile vehicle for end-to-end delivery of goods in industrial settings.

Chapter 2

Preliminaries

This chapter will present the notational conventions and equations commonly used throughout this thesis, as well as briefly provide a background of some core concepts necessary to better understand the work presented here. In particular, two areas will be covered: parametric dense 3D reconstruction and whole image alignment. Both of these concepts go hand-in-hand, since together they form the basis of dense visual tracking.

As it will be seen in subsequent chapters, the quality of depth maps provided by the 3D reconstruction algorithms directly affect the accuracy of the localization system. Furthermore, for robotic applications the accuracy and completeness of the depth maps is of great important since they provide a geometric understanding of the scene the robot is operating on; they are used to figure out how to manipulate objects, how to traverse through a particular area, what are obstacles, etc.

With regards to whole image alignment, a quick survey will be presented starting with template matching, then explaining the forward and inverse compositional/additive approaches for the optimization, going through the efficient second order minimization technique which speeds up converge, and finally presenting particular implementations of systems that use whole image alignment in different modes for visual tracking.

2.1 Parametric Dense 3D Reconstruction

Cameras are great sensors to use in robotic platforms, since they capture a massive amount of data of the scene the robot is operating in. Cameras with millions of pixels are the norm these days, in particular due to their relatively inexpensive price. In contrast, LIDARs can cost up to tens of thousands of dollars and provide a much sparser view of the world – typically 32 to 64 beams.

With this amount of data, however, comes a very high computational cost. Processing millions of pixels is time consuming, though thankfully, with the recent adoption of General Purpose Graphical Processing Units (GPGPUs) and the high parallelizable nature of many computer vision algorithms, it is now possible to process this amount of information in real time.

Figure 2.1: A typical General Purpose Graphical Processing Unit (GPGPU) has thousand of cores, each capable of computing a set of instructions called a *kernel*. One of the largest GPU manufacturers is NVIDIA Corporation, and GPU above is based on their latest *Pascal* architecture: the GTX 1080Ti with 3584 cores. *Image from NVIDIA Corporation, http://www.nvidia.com/*

With GPUs, the algorithm can now be split by the amount of cores available effectively having one core perform computation over one pixel; be that for reconstruction, or state estimation. Comparing this to typical CPUs where the highest core count is 8 or 16 if hyper-threaded, it is clear how the orders of magnitude increase of cores has pushed the boundaries of computer vision in the past years.

This is particularly true for 3D reconstruction algorithms, which allow the estimation of the geometry of the scene – for each pixel – based on multiple camera views. These views can be comprised of a rig that synchronizes multiple cameras separated by a fixed and known baseline (for example, a stereo camera rig), or a single moving camera that creates a baseline through time. This last method of reconstructing a scene is often called Structure from Motion (SfM).

Regardless of the rig used, the basis of all 3D reconstruction algorithms is the same: the correlation between one pixel in one image is found with that of another image from a different viewpoint, and given these two measurements and their baseline, a simple triangulation calculation can be performed to infer depth. Several metrics can be used to find the best match, but the most common one is a direct comparison of the photometric value of each pixel via absolute differences or squared differences. However, because a pixel is such a small discretization of space – especially in high resolution cameras – a *support region* is often used to disambiguate them. For example, a 3x3 "patch" centered around the pixel can then be compared using the Sum of Absolute Differences (SAD), by which all the differences in pixel intensities within the support region are summed. The minimum score is considered the best match for this particular pixel.

To understand better 3D reconstruction algorithms, a fundamental concept in multi-view geometry is required and that is the idea of the *epipolar line*. Figure 2.2 shows images from a stereo rig, in which a feature in 3D can be seen projected on two images. The two focal points of each camera, which is the point at which the light rays meet, and the 3D point in space for the observed feature, create a plane that cuts through both image planes. The intersection of this plane on the image plane is a line, which is the epipolar line. The importance of this line is that: an observed feature in one image is guaranteed to be found on the epipolar line of the other image – assuming the feature is in view.

In general, 3D reconstruction algorithms perform an exhaustive search which makes them very inefficient. By using the concept of the epipolar line, however, the search can be constrained only to this line rather than the whole image. This is still computationally expensive, since the

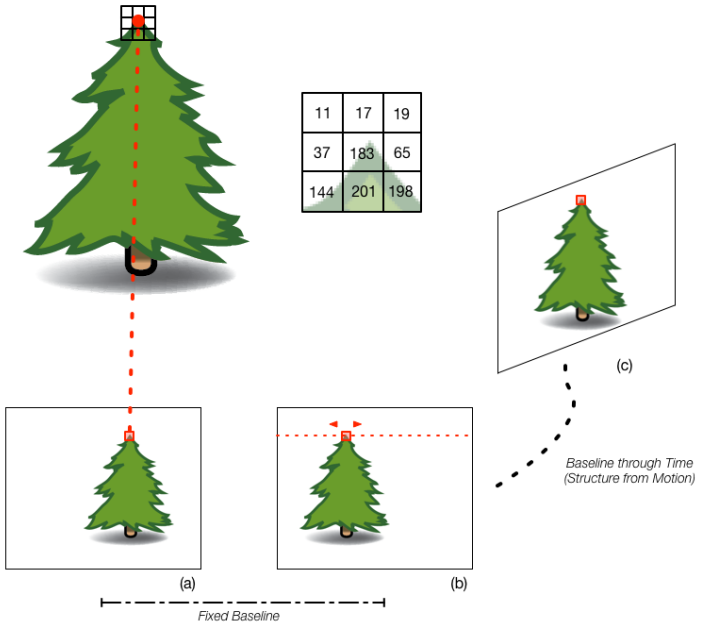


Figure 2.2: Multi-view reprojection of a feature onto a stereo rig. The focal points of the left camera (a) and right camera (b), along with the 3D point of the feature, create a plane that cuts the image planes creating the epipolar line [dotted line in (b)]. The 3x3 patch of the feature is compared with pixels on the epipolar line to find the correspondence that, along with the baseline, is used to calculate depth. With Structure from Motion (SfM), the baseline can be considered as a transform through time (c).

support region patch is slid along this line to find the best match. Higher patch sizes provide better support regions, but at a cost of a higher computational cost. For example, at the most typically used patch size of 3x3, there are still 9 comparison needed to be made *per pixel*.

This is exacerbated even more when not using rigs with a fixed baseline. In a dual camera stereo rig for example, the camera extrinsics – that is, their 6 degree of freedom translation and rotation (pose) – are previously estimated and an image warp is performed via a pre-computed Lookup Table (LUT) such that the epipolar lines fall on the same row for each image. This process is called *scan-line rectification*. Thus, if a feature is observed in a particular row on the left image, the epipolar plane cuts the right camera's image plane on that same row. This has many advantages: the first, the memory is kept contiguous and bounded by maximum one row. The second and most

important advantage is that no linear interpolations for the pixels are necessary. The patch is slid over the other image exactly at the center of each pixel coordinate, thus the differences can be performed over each pixel as is. If, however, the cameras were rotated in such a way that the epipolar line no longer traverses a single row but encompasses multiple rows, an interpolation is required for *all the pixels* in the patch. In the case of the 3x3 patch, a bi-linear interpolation for each of the 9 pixels times the total amount of pixels in the image can quickly become intractable.

Aside from computational limitations, parametric 3D reconstruction algorithms suffer from other challenges. Illumination changes, in particular with moving camera or a rig with an extremely wide baseline, can be very difficult to overcome. Since the Sum of Absolute Difference (SAD) or Sum of Squared Differences (SSD) scores each patch directly by their photometric value, illumination changes due to light variations or multi-path phenomena (like for example glare) can often mis-correlate pixels such that their estimated depth is incorrect.

Several techniques exist to alleviate illumination changes in scenes, like for example applying Zero Normalized Cross Correlation (ZNCC) by which the scene's mean intensity is subtracted to each pixel which in turn is then divided by the standard deviation. This aids in adjusting the brightness of the image due to variations in lighting and exposure conditions.

Another similar technique is called Global Affine Illumination (AI) [49]. In this method, two extra parameters are added to the optimization: a scale α and a bias β parameter that is applied to each pixel $\vec{u}$ with intensity value I such that the new intensities I^* are adjusted as seen in Equation 2.1:

$$I^*(\vec{u}) = (1 + \alpha)I(\vec{u}) + \beta \tag{2.1}$$

Both of these techniques have a major drawback in that they assume that the illumination change is *globally consistent*. It is often the case that lighting conditions affect the scene differently in different areas, depending on the geometry of the scene (e.g. multi-path), the camera position (e.g. non-lambertian reflections) or the material properties of the objects within it (e.g. albedo).

Therefore, adjusting the photometric values globally throughout the image would not capture these intricacies.

A technique that precisely tries to overcome this limitation, and is by far one of the most robust techniques available, was introduced by Zabih and Woodfill in 1994, called the *census transform* [109]. Whereas the Sum of Absolute Difference (SAD), or even the Sum of Squared Differences (SSD) directly operate over the photometric values of pixels, census converts each pixel into a binary signature that encodes whether a pixel's photometric value is lower or not compared to its neighboring pixels. This has the advantage that it encapsulates local consistencies of illumination changes, rather than assuming a global illumination transform.

For example, in the case of a 3x3 patch, the center pixel is compared with its 8 neighboring pixels to create a binary signature as seen in Figure 2.3.

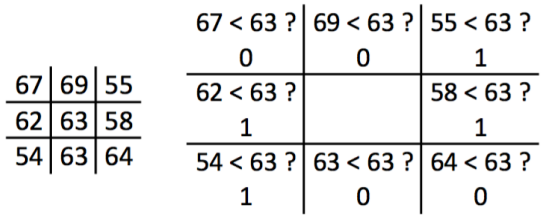


Figure 2.3: Census example: (a) Shows a 3x3 section of the image, while (b) is the output of the census transform for this patch. The center pixel is compared against its neighboring pixels, yielding an 8-bit binary signature of "00111100". The choice of operator { <, <=, >, >= } and the order of the bit encoding does not make a difference, as long as it is consistent throughout its use.

Given the fact that each pixel is now a binary signature, a direct subtraction can no longer be applied to find correlations. This is due to the fact that the position of the bits no longer has any meaning, but rather the actual value of the bit itself. Thus, the Hamming distance is used as a metric of similarity by which the number of matching bits in the binary signature are counted in order to provide the final score.

Even with robust techniques such as census, mismatches can still occur in many cases. This

could be due to ambiguity in the scene (repeated patterns, texture-less areas, etc), or simply by the existence of depth discontinuities at occlusion boundaries. In the latter case, given the movement of the camera, it is often the case that certain features around the occlusion boundary appear in one image but not in the other. Therefore, the mis-correlations are not due to mis-matches but rather the lack of valid data altogether for the feature being matched.

As such, it is important to have methods that can detect or at least mitigate when incorrect depth estimates are performed. One such technique is what is called left-right matching, or sometimes called jealous matching. With this method, not only is the pixel on one image matched against another, but the converse is also applied. In the case of a stereo rig, for example, if the left camera is the reference camera then the pixel on the left image is matched against those in the right. When a match is found, the reverse occurs: the matched pixel on the right image is now used to find a match against the left image. If on both passes, the left-to-right and right-to-left pixel coordinates agree, then the match is considered valid. This is a very reliable method, but sadly incurs a high computation cost since in effect now each correlation search is doubled.

Another technique which is less expensive but similarly effective is filtering out matches that are ambiguous. With this technique, not only is the best score kept but also the second-best score. If the best score is not sufficiently different with respect to the relative error with the second-best score, then the whole match is discarded. This technique is particularly good in scenes where there is a lot of aliasing in the image (for example, repeated patterns often found on carpets), or on the other hand, in cases where there is extremely low texture (like for example, a white wall).

There is a balance between rejecting and accepting matches, since it has a direct effect in the *completeness* of the depth map generated for the scene. Completeness measures how "full" the depth map is: for images of a particular resolution, there is an expectation that the all pixels that reproject onto two or more image views have an estimated depth associated with it. The ratio of actual depth estimates versus total pixels is the completeness of the depth map.

Precision is also an important factor since it measures how good the depth estimates are. Completeness and precision curves are typical metrics used in 3D reconstruction algorithms, since it

is important to have both precise and good coverage of the scene especially for robotic applications. More of these metrics will be seen later in Chapter 3.

2.1.1 Local vs Global Methods

It should also be noted that dense 3D reconstruction algorithms can be broadly classified into two categories: local and global methods. Local methods work over a small support region and typically are faster at the cost of being less accurate. So far, the theory explained in this section is the basis of local methods where correlations are found. For example, an actual implementation of a local method is the Efficient Large-Scale Stereo Matching (ELAS) [35] algorithm.

This algorithm still performs a search over the epipolar line, but it takes advantage of first estimating depth for a sparser subset of highly salient features – that typically are better for correlations – and creating a triangular surface that connect these features in the image. This triangle is used as a prior, restricting the search window for those pixels within it to the depth values of the previously estimated salient features (i.e. the vertices of the triangle). By limiting the search space, the algorithm can perform fast depth estimates, even for high resolution images, in real-time.

In contrast to local methods, global methods minimize a global cost/energy function that combine data and smoothness terms that take into account the whole image. The data term is the information available from the images themselves, and as such is similar in this respect to local methods. While in local methods a patch is used around each pixel, in global methods only the pixel itself is used as data. This alleviates some of the problems at occlusion boundaries, but also loosens the depth estimation by not having a support region. This is where the smoothness term comes in, which is a regularizer term in the optimization and which connects each pixel of the image with each other – thus, each depth estimate is no longer independent, but it is interconnected through this term to the rest of the pixels in the image. As such, global methods are very accurate, but are also more computational and memory intensive than their local counterpart.

Some examples of global methods seen later in this work are Dense Tracking and Mapping

(DTAM) [70] and Semi-Global Matching (SGM) [40].

DTAM works by creating a *cost volume* for a particular scene, and then having a single moving camera capture multiple viewpoints at different baselines. These images are then aggregated in the cost volume, where each slice of the volume corresponds to a depth hypothesis. An optimization is then performed over this cost volume, finding the minimum cost for both the data term (actual pixel difference) and the smoothness term. For DTAM, and for many other 3D reconstruction algorithms, the smoothness term is a constraint added to each pixel such that the depth or disparity value estimated for that pixel should be similar to those pixels around it; in short, it penalizes depth discontinuities. The assumption is that depth in the scene is smooth, and that the values of depth for pixels close to another should be similar. For DTAM in particular, the smoothness term is weighted based on the gradient of the image: this smoothness constraint is loosened if the gradient is high, thus it assumes that neighboring pixels close in intensity should have similar depth values.

A limitation of DTAM is that its 3D reconstruction capabilities are bounded to this initial cost volume, which is not resized nor does it move in space. In Chapter 3, a variation of DTAM is presented called Incremental and Adaptive Front-end Fusion (IAFEF) which is capable of both incrementally updating the cost volume as the camera moves through the environment and also adaptively change its minimum and maximum bounds to capture changes in scene depth.

Semi-Global Matching (SGM), unlike DTAM, does not require multiple views of the scene but typically operates only with an image pair from a stereo rig. As such, it can take advantage of the scan-line rectification to quickly correlate pixels and, like other algorithms, uses a smoothness term that penalizes discontinuities – in this particular case, disparity discontinuities. Also, while DTAM's error is directly calculated based on the photometric information of each pixel, SGM uses the census transform and Hamming distance as the error metric. Finally, the novel approach of SGM is the way the algorithm operates: it computes the cost along several paths and then aggregates them at the end. The typical SGM implementation performs eight different sweeps on the reference image, two horizontal ones (left and right), two vertical ones (up and down), and four diagonal ones. Thus, the algorithm can easily be parallelized by running each path independently

in a multi-core architecture.

2.1.2 Depth Uncertainties

Having already generated depth maps, the next step is to find an uncertainty for each depth value. A way to achieve this is to formulate a joint cost function from all images that contribute to the depth estimation, and use a non-linear optimization solver to minimize the total reprojection cost between the reference image and all other images that have seen the same point. The photometric reprojection cost which minimizes the difference between the reference image I_r and support images I_s, is given by:

$$\mathbf{r}_{\mathcal{R}} = \sum_{s} \sum_{i} I_s(\mathcal{F}(\mathbf{T}_{sr}[\mathbf{u}_{ir}\, \rho(\mathbf{u}_{ir})]^T)) - I_r(\mathbf{u}_{ir}) \tag{2.2}$$

where $\mathcal{F}$ is the nonlinear camera projection function going from 4D coordinates to a 2D pixel location, $[\mathbf{u}_i r\, \rho]^T$ is a 4D homogenous vector comprising of the 3D back-projected ray and the inverse depth, $\mathbf{T}_{sr}$ is the transform from the reference frame to the support frame, $\mathbf{u}_{ir}$ is the ith ray of the reference frame and $\rho(\mathbf{u}_{ir})$ is the inverse depth along the ray $\mathbf{u}_{ir}$. $\mathbf{T}_{sr}$ is represented as a 7 degree of freedom SE3 transform: a 4 degree of freedom rotation (quaternion notation) and a 3 degree of freedom translation.

The double sum indicates that the residual is formed by projecting every point of the reference image into all other support images that have seen that point to form the photometric error. In the case of an algorithm that uses a stereo rig (e.g. ELAS or SGM), this would be simply the reference image (left image) reprojected onto only one support image (right image). For DTAM or IAFEF, the number of support images would be much higher (tens to hundreds).

The smoothness term in the global methods add an additional term, the depth map regularizer, which serves to ensure that areas with low gradient information are estimated based on the high gradient areas that surround them. In the case of DTAM, for example, to properly reflect the uncertainties associated with the generated depth map, the denoising cost terms must also

be included in the total residual. The depth map regularization residual for DTAM is defined as follows:

$$\mathbf{r}_{\mathcal{D}} = \sum_{i} g(\mathbf{u}_{ir}) \nabla \rho(\mathbf{u}_{ir}) \tag{2.3}$$

where $\nabla \rho(\mathbf{u}_{ir})$ is the gradient of the depth map evaluated at ray $\mathbf{u}_{ir}$ and $g(\mathbf{u}_{ir})$ is a term that modulates the influence of the regularization based on the image gradient. As explained previously, this modulation term serves to reduce the influence of the regularization term where there are strong image gradients, as they may correspond to edges in the scene.

In order to evaluate $r_{\mathcal{D}}$ in a nonlinear optimization environment, the derivatives of the depth map gradient must be formulated as a function of optimization parameters as seen in Equation 2.4:

$$\nabla \rho(\mathbf{u}) = \frac{\partial \rho}{\partial u} + \frac{\partial \rho}{\partial v} = \rho(\mathbf{u}_{u^+}) - \rho(\mathbf{u}) + \rho(\mathbf{u}_{v^+}) - \rho(\mathbf{u}) = \rho(\mathbf{u}_{u^+}) + \rho(\mathbf{u}_{v^+}) - 2\rho(\mathbf{u}) \tag{2.4}$$

where $\rho(\mathbf{u}_{u^+})$ and $\rho(\mathbf{u}_{v^+})$ are the neighboring rays of the base ray $\rho(\mathbf{u})$ in the u and v image directions respectively, and are used as a finite difference approximation to the depth map gradient, as a functional representation is not available. The depth map values at these rays are also parameters in the optimization which allows the calculation of the Jacobian of $r_{\mathcal{D}}$ in a nonlinear optimization setting. The total cost minimized during the optimization is therefore:

$$\mathbf{r} = \mathbf{r}_{\mathcal{R}} + \mathbf{r}_{\mathcal{D}} \tag{2.5}$$

This equation (2.5) is then solved using a maximum likelihood estimation framework, as no prior information is assumed over the calibration parameters.

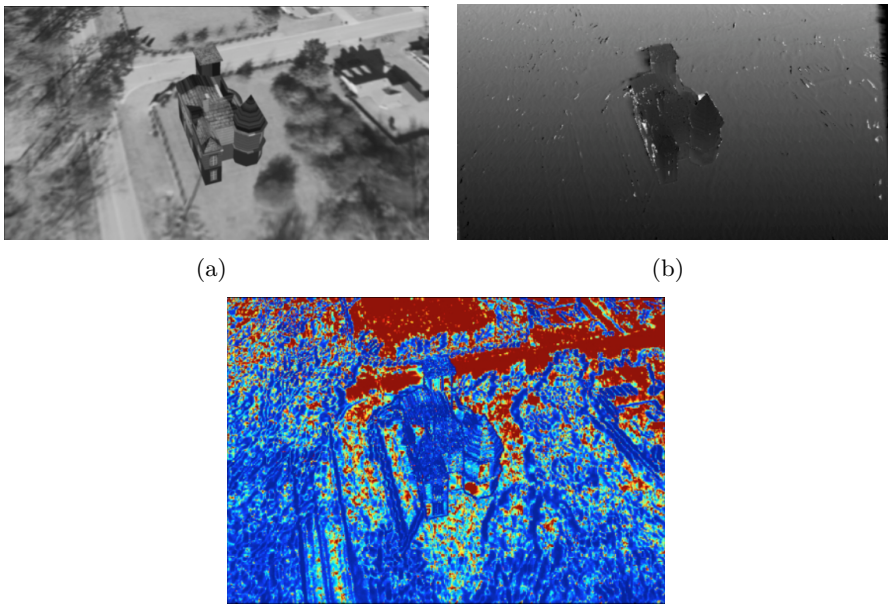
Each term $\mathbf{r}_{\mathcal{R}_i}$ and $\mathbf{r}_{\mathcal{D}_j}$ is computed as a cost term and given to a non-linear solver. The full system Jacobian, $\mathbf{J}$, is calculated and then used to compute the uncertainties. The update to the parameter vector $\Delta\mathbf{x}$ is obtained by solving the equation:

$$\left(\mathbf{J}^T\mathbf{J}\right) \Delta\mathbf{x} = \mathbf{J}^T\mathbf{r} \tag{2.6}$$

Finally, the parameter uncertainty can be estimated from the problem Hessian estimate as follows:

$$\mathbf{C} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1} \tag{2.7}$$

As this is an expensive operation given the number of parameters in the problem, other methods can be used to solve for specific columns of the covariance matrix $\mathbf{C}$ rather than a full uncertainty [25].



(a) (b)

(c)

Figure 2.4: A synthetic dataset (a) generated to simulate data captured from a monocular camera operating on an Unmanned Aerial Vehicle (UAV) flying around a structure. The resulting depth map estimated using DTAM (b), and the covariance extracted using the method above (c). Areas with low texture have higher uncertainty, as expected.

2.2 Dense Visual Tracking

In the previous section, several type of 3D reconstruction algorithms were presented. It is important to understand how these algorithms work, since the depth maps generated by them play a critical part in dense visual tracking.

A typical SLAM system alternates between a mapping and a tracking step. The decoupling of these steps has the effect of compounding errors, since any errors accrued in tracking will be added to the 3D estimation and vice-versa. There are ways to mitigate this, however. In sparse SLAM, for example, depth estimates of features are constantly updated as they are tracked while moving in the scene, or even updated in batch form during a Bundle Adjustment [102, 19] step. Bundle adjustment has been the de facto technique used in past years to reduce error globally, and can be considered as a large but sparse *geometric* parameter estimation problem: the geometric error between the projected 3D landmark and its corresponding 2D measurement is minimized. In short, it is a joint optimization that simultaneously refines the map and poses, and often times, the camera parameters as well.

Dense methods, however, do not have this advantage mostly due to the way the 3D estimation of the scene for dense methods works; e.g. a fully dense reconstruction fully interconnected via the smoothness term. As such, typically only the most current depth map is used at any given time in dense systems. It is, therefore, important that this depth map is as precise as possible in order to obtain a higher quality localization from the tracking system. This is particularly true in the case of monocular SLAM, or even a stereo system that fuses data through time, since any errors in the localization system has a direct impact on the precision of the depth map.

In the next section, it will be shown exactly how the depth map is used on the basis of all dense tracking systems: whole image alignment.

2.2.1 Image Alignment

The first use of image alignment was developed by Bruce Lucas and Takeo Kanade in 1981 for estimating optical flow [55]. Since then, it has been used in many other areas in computer vision including: visual tracking [9, 38], mosaic construction [90], medical image registration [11] and face coding [5, 17].

The basis to image alignment is gradient descent, where the goal is to align a template image to an input image (Figure 2.5). Minor differences in implementations exist, however, depending

if the estimated parameters are added (additive approach) or incrementally warped (compositional approach), if the update is evaluated at the image or template side (forward versus inverse approach), or even what kind of step is performed during the gradient descent: Newton, Gauss-Newton, steepest-descent or Levenberg-Marquardt[6].

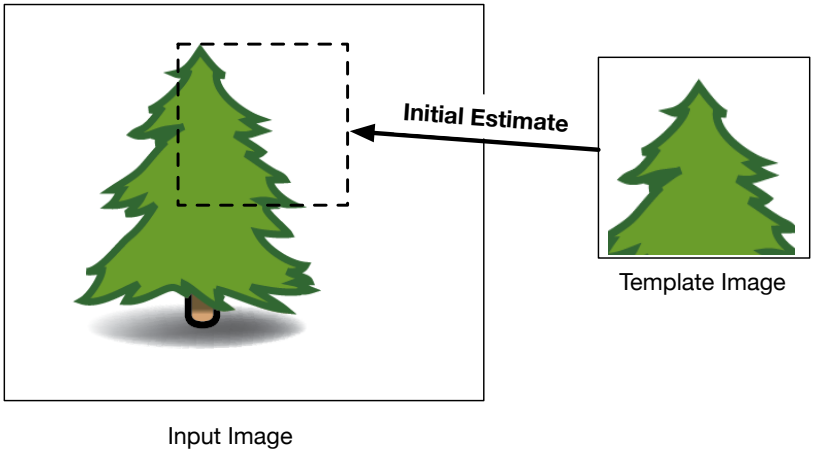


Figure 2.5: The template image is initialized at a particular pixel position in the input image. As the image alignment algorithms iterates, the template image is warped (translated, rotated, scaled, skewed, etc.) over the input image until convergence.

The general equation for image alignment can be written as a least squares optimization:

$$\sum_i [I(\omega(\mathbf{u}_i, \mathbf{p})) - T(\mathbf{u}_i)]^2 \tag{2.8}$$

where I is the input image and T is the template image, ω is a warping function that takes each pixel coordinate $\mathbf{u}$ and warps it by the parameters $\mathbf{p}$ for all pixels i. In short, it is performing the Sum of Squared Differences (SSD) of the photometric values between the input image and the template image.

The warping function $\omega(\mathbf{u}, \mathbf{p})$ maps each pixel coordinate $\mathbf{u}$ from the template image to a sub-pixel coordinate on the input image. The warping parameters $\mathbf{p} = (p_1, p_2, ..., p_n)$ provide different types of warps. For example, a parameter vector $\mathbf{p} = (p_1, p_2)$ can be used for optical flow to capture the adjustments in the pixel's horizontal ($\mathbf{u}_u$) and vertical ($\mathbf{u}_v$) coordinates:

$$\omega(\mathbf{u}, \mathbf{p}) = \begin{pmatrix} \mathbf{u}_u + \mathbf{p}_1 \\ \mathbf{u}_v + \mathbf{p}_2 \end{pmatrix} \tag{2.9}$$

In general, the warping functions can be as complex as desired accepting any number of parameters. They can be applied to 2D problems, like the optical flow example seen above or for estimating homographies, and they can also be applied to 2.5D or 3D problems. For example, when tracking an image patch in 3D an affine warp can be considered:

$$\omega(\mathbf{u}, \mathbf{p}) = \begin{pmatrix} 1 + \mathbf{p}_1 & \mathbf{p}_3 & \mathbf{p}_5 \\ \mathbf{p}_2 & 1 + \mathbf{p}_4 & \mathbf{p}_6 \end{pmatrix} \begin{pmatrix} \mathbf{u}_u \\ \mathbf{u}_v \\ 1 \end{pmatrix} \tag{2.10}$$

The Lucas-Kanade algorithm, as this method has been named, assumes an initial estimate of $\mathbf{p}$ and then solves for the incremental parameters $\Delta\mathbf{p}$, minimizing the following cost function:

$$\mathbf{C}(\Delta\mathbf{p}) = \sum_i \left[I(\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{u}_i) \right]^2 \tag{2.11}$$

with respect to $\Delta\mathbf{p}$ and iteratively incrementing $\mathbf{p}$:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p} \tag{2.12}$$

until an exit condition is found and the optimization is considered converged. These exit conditions typically are a combination of: low relative update $\Delta\mathbf{p}$, total error increasing or simply a maximum number of iterations reached. It should be noted that the '+' symbol in Equation 2.12 does not necessarily imply an addition, but can be any method – for example, a compounding operator – that can update the parameters $\mathbf{p}$.

The cost function is linearized by performing a first order Taylor expansion:

$$\mathbf{C}(\Delta\mathbf{p}) \approx \sum_i \left[\underbrace{I(\omega(\mathbf{u}_i, \mathbf{p})) - T(\mathbf{u}_i)}_{\mathbf{C}(0)} + \underbrace{\nabla I \frac{\partial \omega}{\partial \mathbf{p}}}_{\mathbf{J}(0)} \Delta\mathbf{p} \right]^2 \tag{2.13}$$

where $\mathbf{C}(0)$ is the cost function evaluated at 0, $\mathbf{J}(0)$ is the Jacobian of the cost function evaluated at 0, $\nabla I = \left(\frac{\partial I}{\partial \mathbf{u}_u}, \frac{\partial I}{\partial \mathbf{u}_v}\right)$ is the gradient of the image evaluated at $\omega(\mathbf{u}, \mathbf{p})$ and $\frac{\partial \omega}{\partial \mathbf{p}}$ is the Jacobian of the warp function itself; that is:

$$\frac{\partial \omega}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \omega_u}{\partial \mathbf{p}_1} & \frac{\partial \omega_u}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \omega_u}{\partial \mathbf{p}_n} \\ \frac{\partial \omega_v}{\partial \mathbf{p}_1} & \frac{\partial \omega_v}{\partial \mathbf{p}_2} & \cdots & \frac{\partial \omega_v}{\partial \mathbf{p}_n} \end{pmatrix} \tag{2.14}$$

Minimizing Equation 2.13 is a least square problem and has a close formed solution derived by applying the chain rule as follows:

$$2 \sum_i \left[\nabla I \frac{\partial \omega}{\partial \mathbf{p}}\right]^T \left[I(\omega(\mathbf{u}_i, \mathbf{p})) - T(\mathbf{u}_i) + \nabla I \frac{\partial \omega}{\partial \mathbf{p}} \Delta \mathbf{p}\right] \tag{2.15}$$

Setting Equation 2.15 to zero and solving for $\Delta \mathbf{p}$ gives:

$$\Delta \mathbf{p} = -\mathbf{H}^{-1} \sum_i \left[\nabla I \frac{\partial \omega}{\partial \mathbf{p}}\right]^T [I(\omega(\mathbf{u}_i, \mathbf{p})) - T(\mathbf{u}_i)] \tag{2.16}$$

where $\mathbf{H}$ is the Gauss-Newton approximation to the Hessian matrix:

$$\mathbf{H} = \sum_i \left[\nabla I \frac{\partial \omega}{\partial \mathbf{p}}\right]^T \left[\nabla I \frac{\partial \omega}{\partial \mathbf{p}}\right] \tag{2.17}$$

The estimated update of $\Delta \mathbf{p}$ in Equation 2.16 is applied using Equation 2.12 at every iteration of the Lucas-Kanade algorithm. Since the image gradient is evaluated depending on the warp function, which in turn depends on $\mathbf{p}$, the Jacobian must be re-evaluated at every iteration since $\mathbf{p}$ is being constantly updated by the optimization.

2.2.2 Forward vs Inverse Approach

The cost function in Equation 2.11 has the update term $\Delta \mathbf{p}$ on the input image. This update parameter can be understood as the warping step necessary for the input image to better match the template during the least squares optimization. However, as explained in the previous section, since the image gradient and the warping function both have $\mathbf{p}$ as a dependency, it is necessary to re-calculate their values at each iteration.

The cost function, however, can be re-written with the update term on the template side:

$$\sum_i \left[T(\omega(\mathbf{u}_i, \Delta\mathbf{p})) - I(\omega(\mathbf{u}_i, \mathbf{p}))\right]^2 \tag{2.18}$$

The warping function now appears on both terms, although the derived equations only have the template term since that is where the update parameter $\Delta\mathbf{p}$ appears. This means, in effect, that the update parameters estimated represents the direction to warp the template image in order to better match the input image. Thus, for the parameter update to remain the same – that is, the warp applied only to the input image as seen in Equation 2.12, then the *inverse* of the update needs to be applied. Therefore, instead of applying the update to bring the template image closer to the input image, the inverse update is applied in order to bring the input image to match the template:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}^{-1} \tag{2.19}$$

This is called the inverse approach, in contrast to the forward approach in which both the parameter and the update appear on the same term. The advantage is that, as before, only the input image is warped but the updates are estimated based on the template only. As such, the Jacobian need not be re-calculated as the optimization runs and the warp function only needs to be applied to estimate the final error.

2.2.3 Efficient Second Order Minimization

The Efficient Second Order Minimization (ESM) algorithm was developed for 2D image alignment [7, 58], and can be thought of as a technique that joins both the forward and inverse approaches together. The main idea is to use a second order Taylor expansion of the cost function obtaining:

$$\mathbf{C}(\Delta\mathbf{p}) \approx \mathbf{C}(0) + \mathbf{J}(0)\Delta\mathbf{p} + \frac{1}{2}\Delta\mathbf{p}^T\mathbf{H}(0)\Delta\mathbf{p} \tag{2.20}$$

where, as seen before in Equation 2.13, $\mathbf{C}(0)$ and $\mathbf{J}(0)$ are the cost function and Jacobian evaluated at 0, respectively, and $\mathbf{H}(0)$ is the Hessian (Equation 2.17). Another first order expansion is done, this time on the Jacobian, to obtain:

$$\mathbf{J}(\Delta\mathbf{p}) \approx \mathbf{J}(0) + \mathbf{H}(0)\Delta\mathbf{p} \tag{2.21}$$

By plugging Equation 2.21 into Equation 2.20, the ESM cost function is obtained:

$$\mathbf{C}(\Delta\mathbf{p}) \approx \mathbf{C}(0) + \mathbf{J}(0)\Delta\mathbf{p} + \frac{1}{2}\left[\mathbf{J}(\Delta\mathbf{p}) - \mathbf{J}(0)\right]\Delta\mathbf{p} \tag{2.22}$$

$$= \mathbf{C}(0) + \frac{1}{2}\left[\mathbf{J}(\Delta\mathbf{p}) + \mathbf{J}(0)\right]\Delta\mathbf{p} \tag{2.23}$$

The problem with Equation 2.23 is that to evaluate $\mathbf{J}(\Delta\mathbf{p})$, the value of $\Delta\mathbf{p}$ needs to be known – which is precisely the parameter being estimated. However, one advantage that the image alignment problem has is that: even though the solution is not known, *how* the image looks at the solution is known; it is precisely the template image being matched to.

Furthermore, this can be extended to assume that the *gradient* of the warped input image at the solution is the same as the gradient of the template image. Thus, the Jacobian of ESM is comprised of a mixture of the gradients of the two images: the input image and the template.

$$\mathbf{J}_{ESM} = \frac{1}{2}\left(\nabla I + \nabla T\right)\frac{\partial\omega}{\partial\mathbf{p}} \tag{2.24}$$

By having access to this extra source of information, ESM has shown to converge faster than by using either the forward or the inverse approaches individually.

2.2.4 Direct Visual Odometry

Image alignment is great for template matching, but what happens when the image being matched is not a sub-set of the original image but rather the same size? This is the basis for whole image alignment, and the foundation of direct visual odometry. By applying the Lucas-Kanade

algorithm to two consecutive images moving through time, a relative transform (i.e. pose) can be estimated between these two frames. The warping function would be written such that its parameters are the 6 degree of freedom parameters of a pose in 3D space: (x, y, z) for translation and $(\theta_x, \theta_y, \theta_z)$ for roll, pitch and yaw angles.

As mentioned before, the images used can be two consecutive images in time: the current image (I_{cur}) compared to the previous image (I_{pre}), in which case frame-to-frame tracking is performed. The problem can also be generalized so that a particular reference image (I_{ref} or I_r) – like for example, a keyframe – can be used as a basis of comparison against the most recent live image (I_{live} or I_l). This notation will be used in subsequent chapters, always with the assumption that the most recent image is labeled "live" or "current".

The cost function for whole image alignment is based off the Lucas-Kanade Equation 2.11, the difference lying on the particulars of the warp function and the fact that the parameter $\mathbf{p}$ now is the 6-DOF pose being estimated:

$$\mathbf{C}(\Delta\mathbf{p}) = \sum_i \left[I_l(\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p})) - I_r(\mathbf{u}_i)\right]^2 \tag{2.25}$$

where I_l is the live image, I_r is the reference image. The warping function is defined as follows:

$$\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p}) = \pi^{-1}\left(\mathbf{K}\, T(\mathbf{p})\, T(\Delta\mathbf{p})\, \mathbf{K}^{-1}\, \overset{\circ}{\mathbf{u}}_i\, \mathbf{d}_{\mathbf{u}_i}\right) \tag{2.26}$$

Homogeneous coordinates are used which increase the pixel vector coordinate size by one, such that $\overset{\circ}{\mathbf{u}}_i = \begin{pmatrix} \mathbf{u}_i \\ 1 \end{pmatrix}$. The reverse operation, the de-homogenization function, converts back homogeneous coordinates to 2D pixels and is depicted by π^{-1}:

$$\pi^{-1}\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \frac{a}{c} \\ \frac{b}{c} \end{pmatrix} \tag{2.27}$$

$\mathbf{K}$ is the $3x3$ camera intrinsics matrix which describes the mapping between 3D points in the

world to 2D image coordinates, T is the function that converts the 6-DOF Cartesian coordinates $\mathbf{p}$ into a homogeneous transform matrix ($4x4$ or $3x4$), and $\mathbf{d}_{\mathbf{u}_i}$ is the depth of pixel $\mathbf{u}_i$ obtained from a previously estimated depth map.

The derivative of the cost function of Equation 2.25 is:

$$\left.\frac{\partial I_l(a)}{\partial a}\right|_{a=\pi^{-1}\left(\mathbf{K}\,T(\mathbf{p})\,T(0)\,\mathbf{K}^{-1}\,\mathring{\mathbf{u}}_i\,\mathbf{d}_{\mathbf{u}_i}\right)} \left.\frac{\partial \pi^{-1}(b)}{\partial b}\right|_{b=\mathbf{K}\,T(\mathbf{p})\,T(0)\,\mathbf{K}^{-1}\,\mathring{\mathbf{u}}_i\,\mathbf{d}_{\mathbf{u}_i}} \mathbf{K}\,T(\mathbf{p}) \left.\frac{\partial T(\Delta\mathbf{p})}{\partial \Delta\mathbf{p}}\right|_{\Delta\mathbf{p}=0} \mathbf{K}^{-1}\,\mathring{\mathbf{u}}_i\,\mathbf{d}_{\mathbf{u}_i} \tag{2.28}$$

where $T(0)$ is the $4x4$ identity matrix I_4, ∂I_l is the gradient of the live image, $\partial \pi^{-1}$ is the derivative of the de-homogenization function of Equation 2.27 given by:

$$\partial \pi^{-1} = \begin{pmatrix} \frac{1}{c} & 0 & -\frac{a}{c^2} \\ 0 & \frac{1}{c} & -\frac{b}{c^2} \end{pmatrix} \tag{2.29}$$

and $\partial T(\Delta\mathbf{p})$ is the derivative of the homogeneous transform around its $\mathbf{0}$ provided by the Lie Group generator:

$$\left.\frac{\partial T(\Delta\mathbf{p})}{\partial \Delta\mathbf{p_i}}\right|_{\Delta\mathbf{p_i}=0} = gen_i\ \mathbb{SE}(3) \tag{2.30}$$

The Lie group is heavily used in robotics and computer vision, providing a series of transformation matrices in 3D: rotation matrices belonging to the *Special Orthogonal* Lie group $\mathbb{SO}(3)$ and rigid body transformation matrices belonging to the *Special Euclidean* Lie group $\mathbb{SE}(3)$. The derivative of these group elements around its $\mathbf{0}$ are trivially formed which facilitates calculations. To find the complete list of the generators used in this work, please refer to Appendix A.

2.2.4.1 Iteratively Reweighted Least Squares

It is often the case that the data for dense image alignment contains noisy data that can influence the optimization negatively. This could be due to incorrect data from the depth map estimates either due to mis-correlations or occlusion boundaries, or directly on the image itself

from pixel noise introduced by the image sensor, lens aberrations like distortion and vignetting, or even movements of the objects in the scene, which would skew the estimated pose of the camera.

Therefore, a way to down-weight or even reject completely outliers is highly desirable. Sparse visual odometry systems typically use the Random Sample Consensus (RANSAC) [30], which is a non-deterministic algorithm that fits a model to a random subset of the measurements and tests all other sample points labeling them as inliers and outliers if they fit the model or not.

Doing RANSAC on thousands and millions of sample points, however, would be too computationally expensive, so dense methods typically rely on robust norms which quantify the amount by which the predicted value deviates from the actual value. These type of estimators are called *M-Estimators*, and the dense cost function of Equation 2.25 in this form would look as follows:

$$\mathbf{C}(\Delta\mathbf{p}) = \sum_i \rho\left(I_l(\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p})) - I_r(\mathbf{u}_i)\right) \tag{2.31}$$

where ρ is the robust norm function. Instead of minimizing this new cost function directly, an iteratively reweighted least squares problem can be solved instead where each pixel's influence is weighted by their residual $\mathbf{r}$ [112]:

$$\mathbf{C}(\Delta\mathbf{p}) = \frac{1}{2}\sum_i \vartheta\left(\left[I_l(\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p})) - I_r(\mathbf{u}_i)\right]^2\right) \tag{2.32}$$

where ϑ is the weight function defined by an influence function ψ as follows:

$$\vartheta(\mathbf{r}) = \frac{\psi(\mathbf{r})}{\mathbf{r}}, \qquad \psi(\mathbf{r}) = \frac{\partial\rho(\mathbf{r})}{\partial\mathbf{r}} \tag{2.33}$$

For the typical least squares optimization, the influence function $\psi(\mathbf{r}) = \mathbf{r}$ which equates to the influence value increasing as the residual increases. Thus, by using this influence on equation 2.33, it can be seen that the weight for each pixel for the regular least squares is $\vartheta(\mathbf{r}) = 1$. Thus, the regular least squares optimization can be seen as a special case of the iteratively reweighted least squares with an L_2 norm influence. Table 2.1 shows the most commonly used robust norms

and their corresponding weight and influence functions [112]. It should be noted that some robust norms accept a parameter $\mathbf{c}$ which controls the influence factor.

Table 2.1: Commonly Used M-Estimators

$Type$	$\rho(\mathbf{r})$	$\vartheta(\mathbf{r})$	$\psi(\mathbf{r})$
L_2	$\mathbf{r}^2/2$	$\mathbf{r}$	1
L_1	$\lvert\mathbf{r}\rvert$	$sgn(\mathbf{r})$	$\frac{1}{\mathbf{r}}$
$Cauchy$	$\frac{\mathbf{c}^2}{2} log\left(1+(\mathbf{r}/\mathbf{c})^2\right)$	$\frac{\mathbf{r}}{1+(\mathbf{r}/\mathbf{c})^2}$	$\frac{1}{1+(\mathbf{r}/\mathbf{c})^2}$
$Huber \begin{cases} if \lvert\mathbf{r}\rvert < \mathbf{c} \\ if \lvert\mathbf{r}\rvert >= \mathbf{c} \end{cases}$	$\begin{cases} \mathbf{r}^2/2 \\ \mathbf{c}(\lvert\mathbf{r}\rvert - \mathbf{c}/2) \end{cases}$	$\begin{cases} \mathbf{r} \\ \mathbf{c}\, sgn(\mathbf{r}) \end{cases}$	$\begin{cases} 1 \\ \mathbf{c}/\lvert\mathbf{r}\rvert \end{cases}$
$Tukey \begin{cases} if \lvert\mathbf{r}\rvert <= \mathbf{c} \\ if \lvert\mathbf{r}\rvert > \mathbf{c} \end{cases}$	$\begin{cases} \frac{\mathbf{c}^2}{6}\left(1-\left[1-(\mathbf{r}/\mathbf{c})^2\right]^3\right) \\ \frac{\mathbf{c}^2}{6} \end{cases}$	$\begin{cases} \mathbf{r}\left[1-(\mathbf{r}/\mathbf{c})^2\right]^2 \\ 0 \end{cases}$	$\begin{cases} \left[1-(\mathbf{r}/\mathbf{c})^2\right]^2 \\ 0 \end{cases}$

Which influence function to use depends on the application, as each has its advantages and disadvantages. The L_1 norm provides a linear influence with respect to the residual error, while L_2 tends to be aggressive as the residual increases. Huber is a popular one in computer vision as it is a mixture of L_2 if the error is below the parameter $\mathbf{c}$ and L_1 if above. Finally, Tukey is particularly attractive at completely rejecting certain measurements since its influence goes to zero if the residual is higher than the parameter $\mathbf{c}$. Throughout this dissertation, Tukey is the norm typically applied for dense visual tracking unless otherwise noted.

2.2.4.2 Error Metrics

In the beginning of this section, it was shown that direct visual odometry works by minimizing the photometric error of two images since it is assumed that the illumination changes between images close in time remains the same – this is what is called the *brightness constancy assumption*. This works well when doing visual odometry, which estimates the pose of the camera frame-to-frame, since the changes in the scene's illumination are minimal between two captured images – in particular with a high frame rate camera.

Frame to frame tracking, however, is not ideal since it drifts over time and thus it is desired to localize the camera with respect to a map instead. That is the essence of SLAM: not only to map an environment but also to localize against it. However, the brightness constancy assumption does not hold with long-term maps since illumination changes continually from day to day (morning, afternoon, evening) and is dependent on certain external conditions like weather (clouds, rain, snow) or even seasons (summer vs winter). As such, a direct photometric minimization is often inadequate in these kind of situations.

There are ways to mitigate this, however. Some recent work have changed the error metric from a photometric error to one which is more robust to illumination changes. The most obvious choice is using the census transform. However, using census directly is not easy since as mentioned in Section 2.1, the value of the whole census transform itself does not have any meaning but rather the individual bit values themselves. And using a typical bit comparison metric, like for example the Hamming distance, is not ideal since it is not continuous. Thus, what the authors of the work in [2] proposed is to use each bit independently as a different channel or "plane". Furthermore, each bit is interpolated during the reprojection in order to find the error per bit which is then aggregated with the rest of the census bits to calculate the final error for that pixel. The downside to using this technique, however, is that the computation per pixel has now been incremented by the number of bits in the census transform used. Thus, for the typical 3x3 census window, 8 additional comparisons would need to be made per pixel.

Another technique proposed was using the Normalised Information Distance (NID) metric [53, 75, 98]. This technique makes use of mutual information [57, 103] which has been shown to be robust in the alignment of multi-modal images, in particular for medical applications. The images are converted into histograms, which are then aligned using an entropy metric. This method, however, is extremely computationally expensive and can be sensitive to noise in the depth map.

As it has been shown in this section, whole image alignment is the foundation of dense methods for visual odometry and even though the base cost function is rather simple, the complexity lies in the type of warping function used, how robust it is against noise and outliers and what type

of error metric is used. The pipeline for the localization component in SLAM is first the depth map generation and then the image alignment algorithm. In the next section, a brief overview will be given on SLAM's second component: mapping.

2.3 Relative vs Absolute Maps

One of the biggest questions in SLAM is what type of map to use. A relative map is a map that is *locally* consistent, and is typically represented as a pose graph where each frame is connected with an edge that contains the estimated relative transform between that edge and its neighboring edges. The advantage of this representation is that the map is malleable: changes in the measurements between these frames, either from new observations or a batch optimization, can easily be applied and the frames would all be updated accordingly. Depth estimates attached to this frame – be that in the form of sparse depth/inverse depth points, a full depth map or point cloud – will naturally move in 3D space as the reference frame is adjusted.

For example, Figure 2.6 shows a visualization of a relative map from the KITTI dataset [34] using stereo DTAM as the depth map reconstruction algorithm. The system slowly drifts with time, however when using techniques like loop closure detection and pose graph relaxation (PGR), the global error is drastically reduced (Figure 2.7).

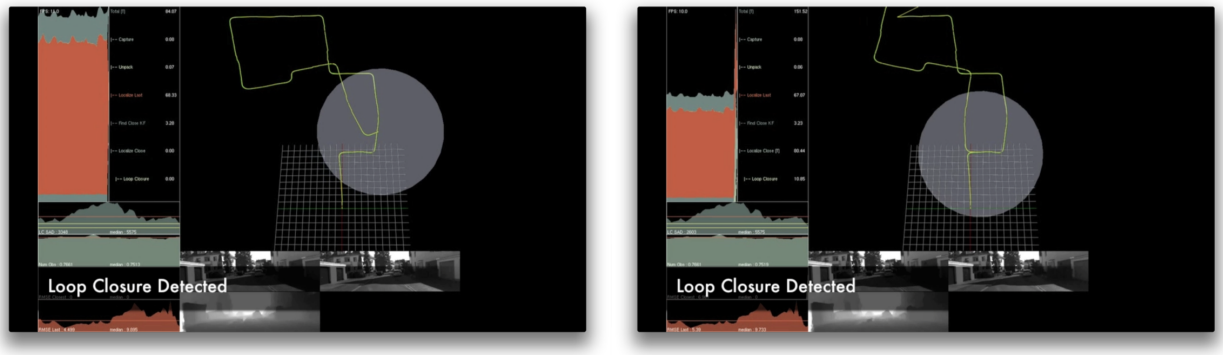


Figure 2.6: The relative map is comprised of frames connected via edges (yellow line on the map). During normal operation, the system accumulates drift (left) that is quite noticeable during a loop closure event (right).

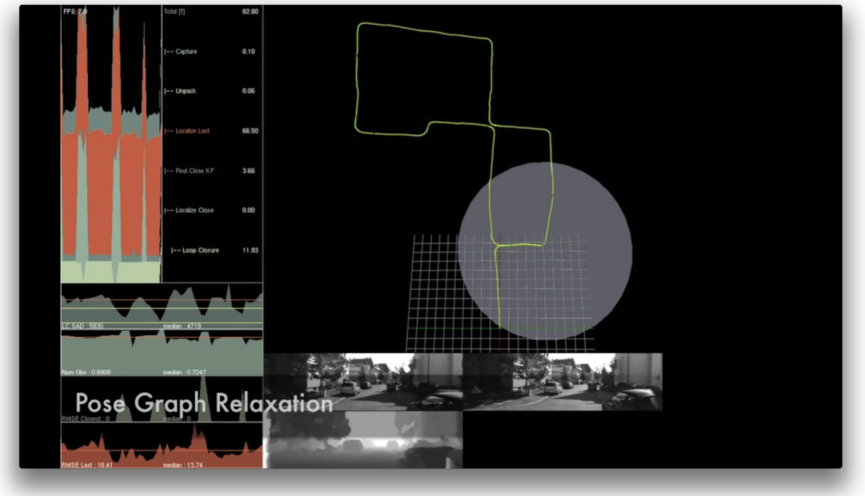


Figure 2.7: After a loop closure is detected, the system performs a pose graph relaxation which globally reduces the error in the map.

The disadvantage of relative maps, as can be seen, is that errors are compounded when the map is *lifted* to an absolute representation. So even though it is locally consistent, the map is not globally consistent without some extra optimizations.

This is in contrast to the absolute representation of a map, in which it is globally consistent. Such a map is typically represented in a volumetric data structure like octomap [42], which is a type of occupancy octree map, a voxel based signed distance function (SDF) [56, 44] or a large pointcloud with millions of vertices [104, 107, 108]. However, these maps have a large memory footprint and are typically extremely computationally intensive if any updates are made – like for example, during loop closures.

Chapter 3

Incremental and Adaptive Front-End Fusion

In the previous chapter, a brief survey of 3D reconstructions algorithms was provided and in particular two methods were explained: local methods, which operate on pixels independently typically by employing a support region (i.e. a patch), and global methods which use the information in the whole image to estimate the depth for each pixel. Specifically, the DTAM algorithm was introduced and the use of a cost volume to estimate depth of a scene. This chapter presents a modification of the DTAM algorithm, targeted in particular to moving robotic platforms, that is capable of both incrementally updating the cost volume as the camera moves through the environment and also adaptively change its minimum and maximum bounds to capture changes in scene depth.

3.1 Introduction

Sparse algorithms were the de facto localization and reconstruction methods back in the 80s. And even though 3D reconstruction has been a very well studied area in computer vision for the past decades, it is only in the past few years that **dense** 3D reconstruction has been possible given the advances in computational power with the introduction of GPUs as seen in Chapter 2.

This has been instrumental in the recent proliferation of robotic research, since robots require full spatial information about an environment in order to interact with it. For example, a robot moving in a retirement home would need to have an accurate reconstruction of the scene it is looking at in order to navigate safely and assist individuals in everyday tasks. Enabling this on

mobile robotics, however, requires fast algorithms with low memory requirements.

Previously, 3D reconstruction of a scene from multiple images using a monocular camera - known as Structure from Motion - has been shown using both local and global methods [88]. In particular, global methods using variational techniques over a cost volume have shown impressive results for indoor environments [70]. Despite their ability to reconstruct the scene with a high level of fidelity, however, these methods are generally both computationally and memory intensive which makes them hard to implement in mobile robotic platforms.

In this section, a global method which provides good scene reconstructions of indoor environments while minimizing computational and memory requirements is presented[27]. The algorithm provides dense reconstruction of scenes by incrementally adding new frames into a moving cost volume that is constantly being updated, and, adaptively changing the cost volume's boundaries in order to adjust to changes in scene depth.

3.2 Background

Perhaps the first work showing how duality methods could be applied to variational problems is presented in [8]. By representing the problem as an inequality, the Primal-Dual Hybrid Gradient (PDHG) from linear programming can be applied to optimizations involving regularization. One of the first applications of this method to vision problems was the formulation of the ROF (Rudin, Osher, and Fatemi) model for denoising [82]. These duality methods have been applied to more generic computer vision problems as well [4]. In order to increase the speed of convergence of these algorithms, adaptive step sizes can be used [10]. Further work in speeding up primal-dual hybrid gradients is shown in [36] by creating highly adaptive step sizes in performing the iterations.

The authors of Dense Tracking and Mapping in Real-Time (DTAM) [70] have shown that the data term used in these techniques can be represented volumetrically; that is, the cost volume stores a sum of photometric errors in a volumetric representation, where each voxel represents a sum of photometric errors for a set of comparison images at a specific depth and pixel coordinate. This volumetric method, however, introduces a non-convex component in the global optimization.

In general, the primal-dual hybrid gradient method for global optimization is confined to convex functions [10]. To overcome this limitation, DTAM proposes alternating the primal-dual update steps with a finite search over the cost volume to determine the minimum. The optimization is also augmented by a quadratic relaxation term, as described in [96]. However, this volumetric approach has two major limitations.

The first limitation is the discretization of space. Increasing the number of voxels representing depth increases the precision of the system, i.e. a finer discretization in depth. This finer level of discretization, however, incurs a higher computation cost. Every additional level of discretization requires not only an additional calculation of the photometric error when constructing the cost volume, but also an additional step in the search through the cost volume performed at every iteration.

The second limitation is that the boundaries set on the cost volume, namely the minimum and maximum depths, can alter the quality of reconstruction if not properly set. For example, a far scene with short boundaries will have poor reconstruction. This is impractical in robotic applications, since it is known that depth ranges may change greatly as a robot navigates through the environment. For example, a robot scanning a desk at close range would require different depth ranges to one that is navigating down a long corridor. A static boundary on the cost volume results in incorrect estimates for depth with the changing scene. It is possible, however, to adaptively expand and contract the volume by sampling the scene and obtaining a rough distribution of depth values. By limiting the volume to only visible depth areas, the system achieves optimum depth resolution.

In addition to the above problems, the reconstruction is performed over a set of images with an associated set of relative poses, selecting one image to serve as a reference image and the others serving as comparison images. To be useful for mobile robotics, however, the depth needs to be calculated with respect to the most current frame. The cost volume must then be constructed based on this reference image. This means that as the robot moves forward the cost volume needs to be recomputed at every new frame. This computation, and storage, becomes prohibitive as the

number of frames fused is increased.

The next section shows that an iterative and adaptive method for forming the cost volume can be used to overcome the above mentioned problems.

3.3 Methodology

The same global formulation for depth optimization as the one proposed in [70] is used, parameterizing in inverse depth, namely:

$$\mathbf{E} = \int_{\Omega} g(\vec{u}) \left|\left|\nabla \xi(\vec{u})\right|\right|_{\epsilon} + \frac{1}{2\theta}(\xi(\vec{u}) - \alpha(\vec{u}))^2 + \lambda C(\vec{u}, \alpha(\vec{u}))dx \tag{3.1}$$

where $\mathbf{E}$ is the total cost over the image domain and $\vec{\mathbf{u}}$ is the pixel coordinate, $\vec{u} : \Omega \to R^2$.

The first term in (3.1) is a regularizer which enforces second order smoothness over the inverse depth, ξ. The regularizer is scaled by a weighting function which serves to reduce the regularization where there is a large image gradient. The weighting function $\mathbf{g}(\vec{u})$ is defined by:

$$g(\vec{u}) = e^{-\alpha |\nabla I_r(\vec{u})|_2^{\beta}} \tag{3.2}$$

Here, α and β are constants selected to vary how much the image gradient, $\nabla I_r(\vec{u})$, impacts the weighting of the regularizer. The regularizer selected is a Huber norm of the gradient of inverse depth at a pixel coordinate, $\left|\left|\nabla \xi(\vec{u})\right|\right|_{\epsilon}$.

The last term in (3.1) is the data term, which is the value of the cost volume at a specific inverse depth and pixel coordinate scaled by a factor λ. The cost at a specific pixel and inverse depth location is the sum of photometric errors between a reference image and a set of comparison images, as defined below.

$$C\left(\vec{u}, \xi(\vec{u})\right) = \frac{1}{\mathcal{I}_m} \Sigma_m \left| I_r(\vec{u}) - I_m(\vec{W}(\vec{u}, \xi(\vec{u}))) \right| \tag{3.3}$$

where $\vec{W}$ warps the pixel coordinate from the reference image $\mathbf{I_r}$ into each of m comparison images

$\mathbf{I_m}$, assuming some estimated inverse depth value $\xi(\vec{u})$. $\vec{W}$ is defined as:

$$\vec{W}(\vec{u}, \xi) = \Pi \left(K T_{mr} \frac{1}{\xi(\vec{u})} K^{-1} \begin{pmatrix} \vec{u} \\ 1 \end{pmatrix} \right) \tag{3.4}$$

where Π is a de-homogenization function, K is the camera matrix and T_{mr} is the estimated pose between the comparison image and the reference image.

Finally, as described in [96], the original cost C and the regularizer are decoupled via an auxiliary variable, $\alpha(\vec{u})$. This appears as the second term in (3.1), which shows the coupling of the estimated inverse depth $\xi(\vec{u})$ and the auxiliary variable $\alpha(\vec{u})$. The variable θ enforces the amount of coupling, with a smaller θ enforcing stricter coupling. During a PDHG optimization, θ is reduced at every iteration, thereby driving the original inverse depth term and auxiliary depth variables together.

The volumetric representation described above assumes some discretization of inverse depth, starting from a minimum inverse depth ξ_{min} (furthest scene depth) up to a maximum inverse depth ξ_{max} (closest scene depth).

The system alternates between dense tracking and depth estimation. Dense tracking is performed using a 2.5D Lucas-Kanade style minimization of photometric errors [6, 13] using the depth maps estimated at each step. Since the system starts without any depth map, a semi-dense monocular estimation pipeline similar to [31] is used to bootstrap the dense reconstruction algorithm. After the system is initialized, a sparser version of the algorithm continues to run in the background tracking fewer points. These points are then used to adapt to scene depth, where the minimum and maximum depth estimates from the sparse tracker are used to set the bounds on the new cost volume. This new volume is then populated by a linear interpolation of the old volume. The transformation of the cost volume and rescaling is done in a single step. After transformation, depth estimation is then performed by an optimization in accordance with a pixel-wise gradient ascent/descent in the dual/primal spaces.

3.3.1 Incremental

The novel component of this work involves how the cost volume is computed from frame to frame. To remove the requirement of keeping multiple images and multiple transforms between the images, a single cost volume is used which is incrementally transformed into the most current reference frame. This assumption is valid as long as the relative motion from frame to frame is small and is a common situation encountered in indoor environments, especially for cameras running at 30 frames per second or more.

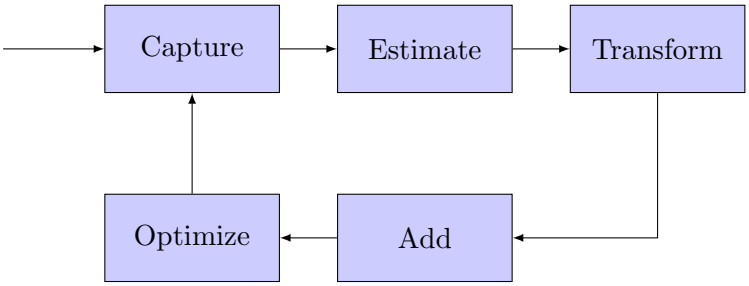
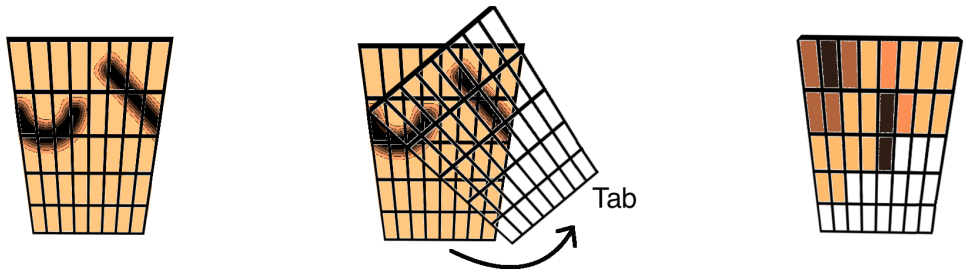


Figure 3.1: Block diagram of the system. An image is first captured, and then the relative pose is estimated using whole image alignment. Given this pose, the cost volumes are transformed and the photometric error with the current image is added to the volume. Finally, scene depth values are estimated through the optimization.

Incrementally refining the cost volume from frame to frame is illustrated as a multistep process, shown in Figure 3.1. In the first step, a new image is captured. In the estimation step, the newly acquired image and the previous estimate of depth with its associated intensity image are used to determine the relative pose of the camera using an RGBD optimization based on the Efficient Second Order Minimization (ESM) technique as described in [49]. In the transformation step, the estimated relative pose is used to reinterpret the cost volume from the perspective of the current frame. As illustrated in Figure 3.2, a new cost volume is placed on top of the old cost volume. The values of the new cost volume are calculated via a trilinear interpolation. Any locations which do not overlap are marked as invalid and are not taken into consideration during the optimization. Once the cost volume is transformed into the new frame, data from the current



(a) (b) (c)

Figure 3.2: A top down view of the cost volume at frame N (a) and a visual depiction of how the cost volume is transformed to a new cost volume at frame N+1 (b). Darker areas correspond to smaller photometric errors, indicating a surface. These values are mapped via a trilinear interpolation onto the new cost volume in (c). Any voxels of the cost volume in frame N+1 which are not contained in the previous cost volume are marked as invalid.

image is added.

In computing the traditional cost volume, the sum of all the photometric errors are stored in an *Error Volume (EV)* which is normalized by the total number of images used, maintained by the *Frame Count Volume (FV)*. That is, separate volumes are used to track the photometric errors and the number of images used to calculate that error. A *Normalized Cost Volume (NCV)* is then created by dividing the error by the number of frames that have reprojected for every pixel and inverse depth location, as shown in (3.5). The *NCV* is calculated on-demand by the depth estimator and is no longer needed after the optimization ends. The only volumes that are carried throughout the scene are the EV and FV.

$$NCV(\vec{u}, \xi(\vec{u})) = \frac{EV(\vec{u}, \xi(\vec{u}))}{FV(\vec{u}, \xi(\vec{u}))} \tag{3.5}$$

Once the EV and FV are transformed to the new frame, the photometric error of the last image with respect to the current image, $C(\vec{u}, \xi(\vec{u}))$, is calculated and then added into the corresponding pixel and inverse depth location in the cost volume. The frame count for that location is then incremented by one. For any points that do not reproject, there will be neither an additional cost term nor any addition to the corresponding frame count volume. In order to overcome problems associated with occlusions, we down-weight the previous data exponentially, as

shown in the equations below.

$$\begin{aligned} EV_{new}(\vec{u}, \xi(\vec{u})) &= C(\vec{u}, \xi(\vec{u})) + \omega EV_{prev}(\vec{u}, \xi(\vec{u})) \\ FV_{new}(\vec{u}, \xi(\vec{u})) &= 1.0 + \omega FV_{prev}(\vec{u}, \xi(\vec{u})) \end{aligned} \tag{3.6}$$

Further details regarding the weight parameter ω and its influence in the cost volume fusion process can be seen in Section 3.4.

3.3.2 Adaptive

As mentioned before, previous volumetric approaches use a constant sized cost volume. If improper bounds are chosen for discretization of the cost volume along inverse depth, either a significant portion of the information will be disregarded, or the cost volume will be larger than the scene to be reconstructed. In either case, this leads to a poor scene reconstruction. In other words, the bounds of the cost volume are subject to the following two problems:

(1) **Over Sizing**: A lack of data in the near or far region of the cost volume indicates that there is no reprojection for that choice of inverse depth and that the boundaries of the cost volume should be narrowed. This is done by increasing ξ_{min} and/or decreasing ξ_{max}.

(2) **Under Sizing**: A large percentage of the data in the near or far regions of the cost volume indicates that there is potentially more scene information outside of the bounds as set and that ξ_{max} should be increased or ξ_{min} decreased, respectively.

The bounds of the volume, namely ξ_{min} and ξ_{max}, can be dynamically altered in order to increase the resolution and accuracy based on the data provided by the sparse system running in parallel introduced in Section 3.3. The reason for relying on a secondary system is that it not possible for any volumetric representation to correctly infer depth values if the volume bounds are set incorrectly in the first place. The minimum and maximum depth estimates from the sparse tracker are used to set the scene maximum inverse depth and minimum inverse depth, respectively, as seen in Figure 3.3.

Once the desired range of the cost volume is estimated, $\xi_{min} \rightarrow \xi_{max}$, it is now possible to use the same transform function to warp the cost volume to fit the new space. Any areas of the cost volume that do not fall within the old volume are marked as invalid.

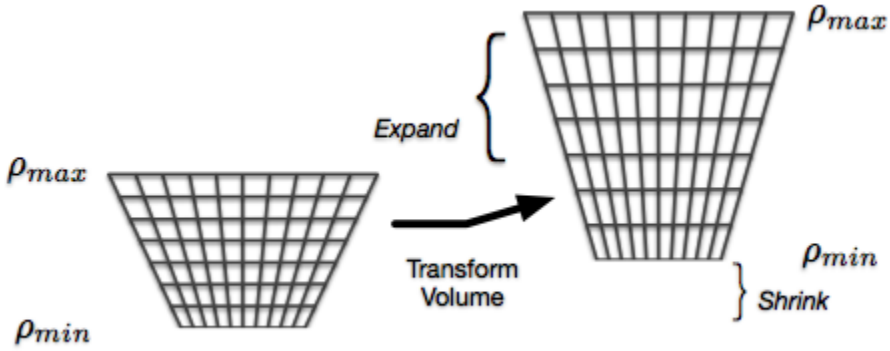


Figure 3.3: Example of how the cost volume is expanded or contracted depending on estimated inverse depth ranges from the sparse tracker. A trilinear interpolation is used to expand/reduce the cost volume.

3.4 Results

The algorithm was tested with both simulated and real data. For the simulated runs, the Tsukuba dataset [59] was used, a very popular dataset that provides ground truth depth and has a high variance of depth throughout the sequence. For the live data, a PointGrey Bumblebee camera set in single image (non-stereo) mode was used and a living room scene was captured. As an error metric, both the average absolute difference between the estimated depth and the ground truth, as well as the completeness and precision curves, are provided.

In order to test how well the reconstruction algorithm alone performs, any errors associated with pose estimation and scale ambiguities were removed by using the ground truth poses. The algorithm parameters from Table 3.1 were used when evaluating the reconstruction method. To test the adaptability of the algorithm to depth changes in the scene, a simplified version that only performs incremental volume updates, but does not dynamically adapt to changes in depth, was used. The incremental only algorithm is designated by Incremental Front-End Fusion (IFEF) to

differentiate from the full system that is both Incremental and Adaptive (IAFEF).

Table 3.1: Algorithm Parameters

α	β	ϵ	λ_{start}	θ_{start}	θ_{end}	ω
100	1.6	10^{-4}	1.0	1.0	10^{-5}	0.5

The final parameter in Table 3.1, ω, controls the contribution to the incremental update of the cost volume as shown in (3.6). This value is directly related to the amount of potential occlusions in the scene given viewpoint changes. Ideally, the value should be as high as possible in order to obtain the maximum contribution from previous estimates into the new estimate. The mean error with different values of ω for a section of the dataset that experiences high viewpoint changes was plotted as seen in Figure 3.4.

Although a value of 0.7 seems to give the best performance for this particular test sequence, for the general case a value of 0.5 was selected as it is believed to be a conservative value of ω for *any* type of scene.

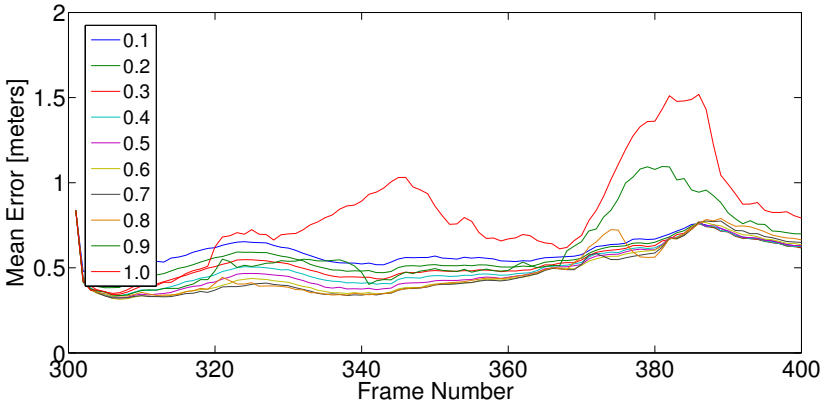


Figure 3.4: Mean Error versus ω as it contributes to the incremental update of the cost volume. A value of 1.0 would yield the maximum contribution.

3.4.1 Timings

The time that it takes to transform the cost volume is the same as the time to add an image to the cost volume. This is expected, as the operations are essentially the same: they both require an evaluation at every pixel and inverse depth position in the cost volume. In other words, the time it takes to construct the cost volume for DTAM is dependent on the number of comparison frames to be fused, $\mathcal{O}(M)$. On the other hand, since there is only a single comparison image in this method, the time to transform and add the most recent image remains constant, $\mathcal{O}(1)$.

Therefore, the speed-up in this implementation is most noticeable as the number of comparison images is increased. Regular DTAM uses hundreds of comparison images, but even with a basic 30 window implementation it can be seen that there is a 15x speed-up in the construction of the cost volume: 30 additions to a cost volume as opposed to the time it takes to transform the volume and add a single image.

3.4.2 Iterative Method

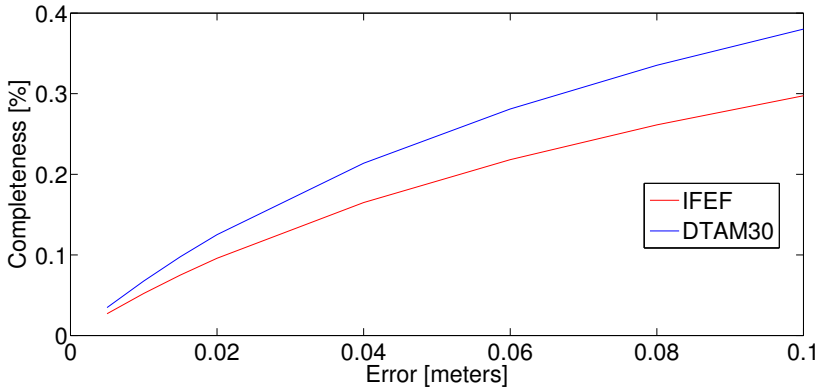
An initial qualitative depth map reconstruction is shown in Figure 3.5. For actual error measurements, completeness and precision curves are computed by comparing a pure incremental approach (IFEF) with a 30 windowed DTAM implementation. This means that the algorithm does not store any additional frames, but rather transforms the volume and fuses the current frame at every step. As seen in Figure 3.6, IFEF slightly under-performs compared to a full windowed version of DTAM, but at a 15x speed-up gain. This is not unexpected, as the pure incremental algorithm only fuses small baseline images, whereas DTAM has both small and wide baselines in the comparison window.

To perform a fair comparison against DTAM, a windowed implementation that contains a mixture of small to large baseline images was used. The algorithm is inherently capable of supporting this hybrid approach, as the cost volume aggregation can happen whenever frames drop out of the selected window. This still has the advantage of a speed-up, as long as the window

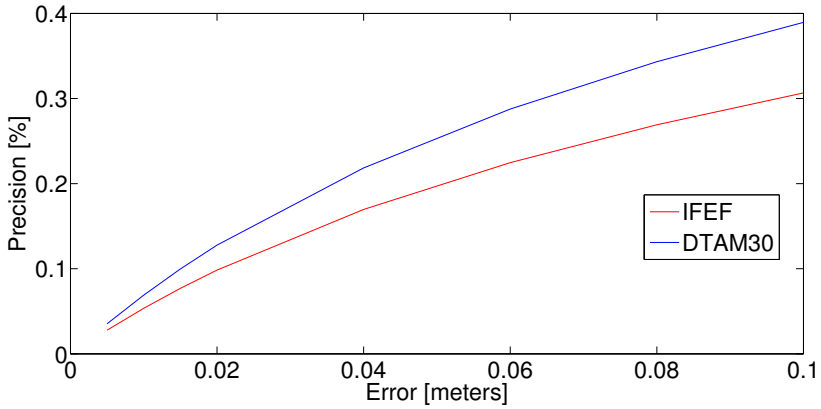
Figure 3.5: Images from the Tsukuba sequence with their corresponding depth maps generated with this approach. The first depth map is of inferior quality, as it corresponds to the start of the sequence where there is limited data and little movement.

selected is kept small. This feature in the algorithm has the extra-benefit of giving fine control to the user, choosing performance over accuracy depending on the situation. The result can be seen in Figure 3.7, where the above procedure was tested by adjusting the comparison window and seeing how the error changes with different window sizes.

Similarly, a comparison was performed whereby both algorithms were constrained to run



(a)



(b)

Figure 3.6: Precision and completeness curves of the algorithm versus a 30 frame windowed version of DTAM. Precision is the percentage of estimates that are within certain error from the ground truth. Completeness is the percentage of ground truth measurements that are within certain error of estimated depth values.

under the same time limit. To do this, DTAM was run with the number of comparison selected to yield the equivalent time it takes this algorithm to run; that is, two frames. This comparison is shown in Figure 3.8 and illustrates that the incremental cost volume aggregates more information than the two frame window set of DTAM. As can be seen, the algorithm consistently outperforms DTAM from frame to frame. This is an indication of the advantage of using the incremental method, which aggregates all previous data.

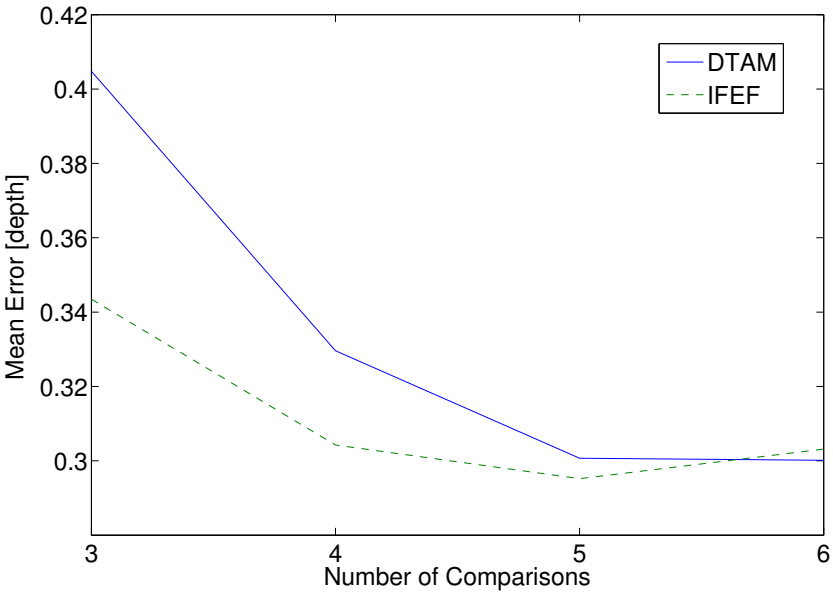


Figure 3.7: Mean depth error of DTAM versus this method with respect to the number of comparisons performed, where the aggregation of the incremental cost volume counts as an extra comparison.

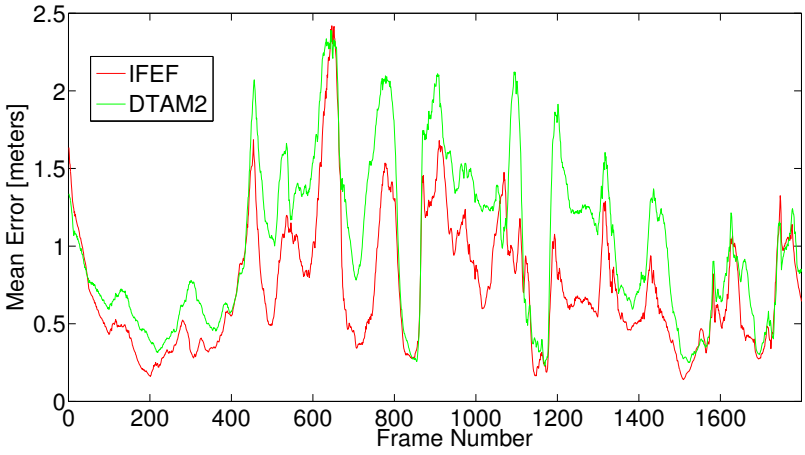
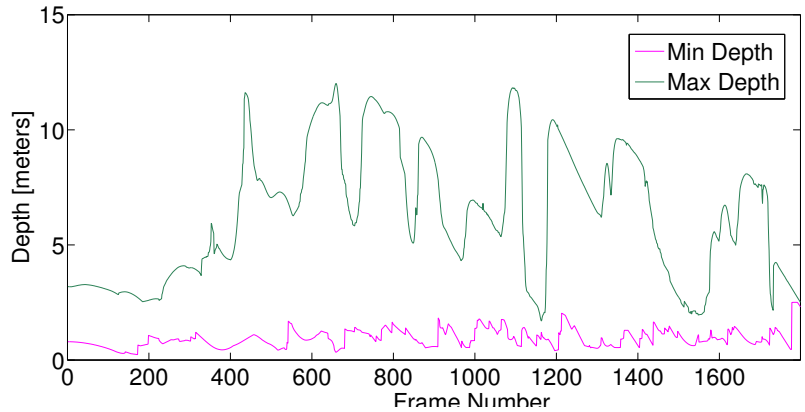


Figure 3.8: Mean depth error for this algorithm (red) and DTAM (green) running under the same time constraint; the equivalent of using two image comparisons.

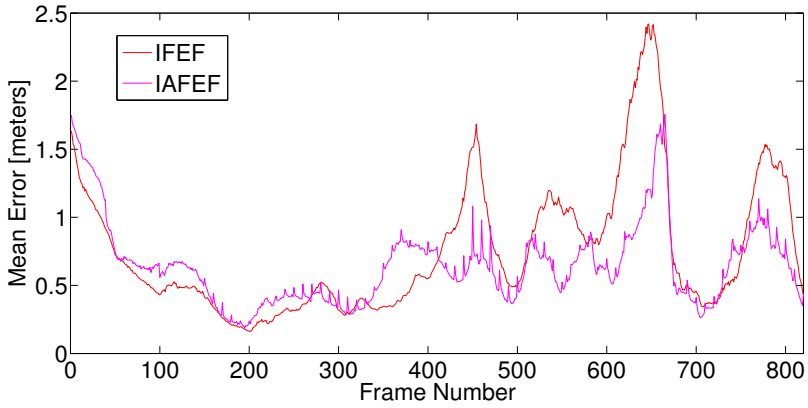
3.4.3 Adaptive Window

In order to test the adaptability of this algorithm, the window size was changed on every tenth frame using the information from the sparse tracker running in the background. As can be

seen in Figure 3.9, the Tsukuba dataset is a particularly challenging set for a fixed cost volume size implementation, such as DTAM, given the large variations in scene depth. Using the estimated depths from the sparse features, a marked improvement in the general depth estimates can be seen.



(a)



(b)

Figure 3.9: (a) shows scene minimum and maximum depths per frame in the Tsukuba dataset. (b) shows how the mean error is improved by using an adaptive window.

3.5 Conclusion

This work has shown two improvements in volumetric approaches for depth estimation. The first is that it is possible to incrementally update the cost volume representation from frame to frame. This incremental approach reduces the memory and computational time required to recon-

Figure 3.10: Point cloud (a), depth map (b), created from images (c) in a live data sequence using this algorithm. The scene depicts a camera scanning a bookshelf in a living room environment. High levels of fidelity, as those provided by volumetric approaches to 3D reconstruction, are important for robotic interaction in indoor environments.

struct a 3D scene from a multi-view stereo and provides depth estimates at the most current frame. In order to maintain a constant memory footprint and computation cost during reconstruction, it is shown that for small displacements it is possible to maintain a single cost volume and simply transform it into the most current frame. This re-calculation allows to continually estimate depth in the most recent frame. Results, like those seen in a live sequence in Figure 3.10, show that this approach is comparable, and in most cases, surpasses traditional volumetric approaches where hundreds of frames need to be stored and re-calculated at every step.

It has also been shown that it is possible to dynamically change the bounds of the cost volume to produce better estimates of the scene geometry. This enables mobile robots to operate in indoor environments with wide ranges of depth values, by only selecting the minimum and maximum depth bounds required to fit the scene in view.

A shortcoming of this algorithm is that it only fuses small baseline images. The original DTAM implementation uses a combination of small and large baselines, starting by first fusing the small baselines to provide an initial but less accurate depth estimate and finalizing by adding the larger baseline images to refine the estimates. A modification of this algorithm can be to carry a small subset of images from different baselines and add them in reverse order: closest to farthest.

Other improvements can be done to this algorithm which would speed up the depth estimation process itself, which is the biggest hurdle now that the time required to build the cost volume has been reduced. Some work has been done with Signed Distance Functions (SDFs), which makes it easy to raycast the depth seen from a particular viewpoint. After the pose is estimated by the tracker, it is possible to use this pose estimate to generate an initial depth map of the scene given the viewpoint change. This depth map can therefore be used as an initial value, effectively seeding the depth optimization by reducing the need to iterate through all the slices of the cost volume to find the minimum energy. This has the advantage of allowing to increase the number of slices of the cost volume, which is desirable in order to achieve a finer granularity for the depth estimates, without incurring the penalty of a higher computation time.

Chapter 4

Inertial Aided Direct Methods for Robust Visual Tracking

As explained in Chapter 2, the depth map plays a critical part in the localization component of a SLAM system. In this chapter, a more in depth study is done on the issues that affect tracking: not only external factors like image or depth map noise related to sensor, lens aberrations or calibration issues, but also internal camera settings like image resolution and frame rate. Furthermore, a new sensor is introduced to aid the visual tracking: the Inertial Measurement Unit (IMU). This is the next step towards robust dense visual SLAM.

4.1 Introduction

Over the past few years there has been a proliferation of different varieties of SLAM methods for robotic platforms. Recently, many approaches have been developed for direct visual odometry ranging from fully dense methods [70, 107, 47] to semi-dense[22, 87], and even hybrid approaches like semi-direct [31]. Even more recent, methods that combine cameras and Inertial Measurement Units (IMUs) to some degree – either tightly coupled or loosely coupled – have been produced[56, 50]. These have been shown to provide higher accuracy compared to visual-only methods. An in-depth analysis, however, has not been made which can shed light on how robust these systems are and under what circumstances they provide better performance.

To this end, a framework to compare visual odometry systems in an unbiased way that also permits exact repeatability is desired. Previous work [39] performs a similar study for cameras and uses a fully dense localization pipeline. The authors conduct an in-depth characterization of

camera physics, light and even noise, and analyze how this affects image formation at different frame-rates. Their experiments lead to quantitative conclusions about frame-rate selection when taking tracking performance into consideration.

This section will present a method similar in spirit, though focused on studying the latest techniques in direct visual odometry including those that make use of inertial measurement units. In particular, a comparison between fully dense versus semi-dense methods and study how the addition of an Inertial Measurement Unit (IMU) in the sensor rig contributes to the estimation result. This information is very useful in many applications, particularly in robotics, when it is often unclear what is the best sensor to use for a particular situation.

4.2 Methodology

To perform an accurate and unbiased evaluation of any tracking system, a framework that allows repeatability must be created, especially one that allows to easily modify multiple parameters. This is particularly true when using an Inertial Measurement Unit (IMU) because the rig must follow the exact same trajectory, at the exact same acceleration, every time.

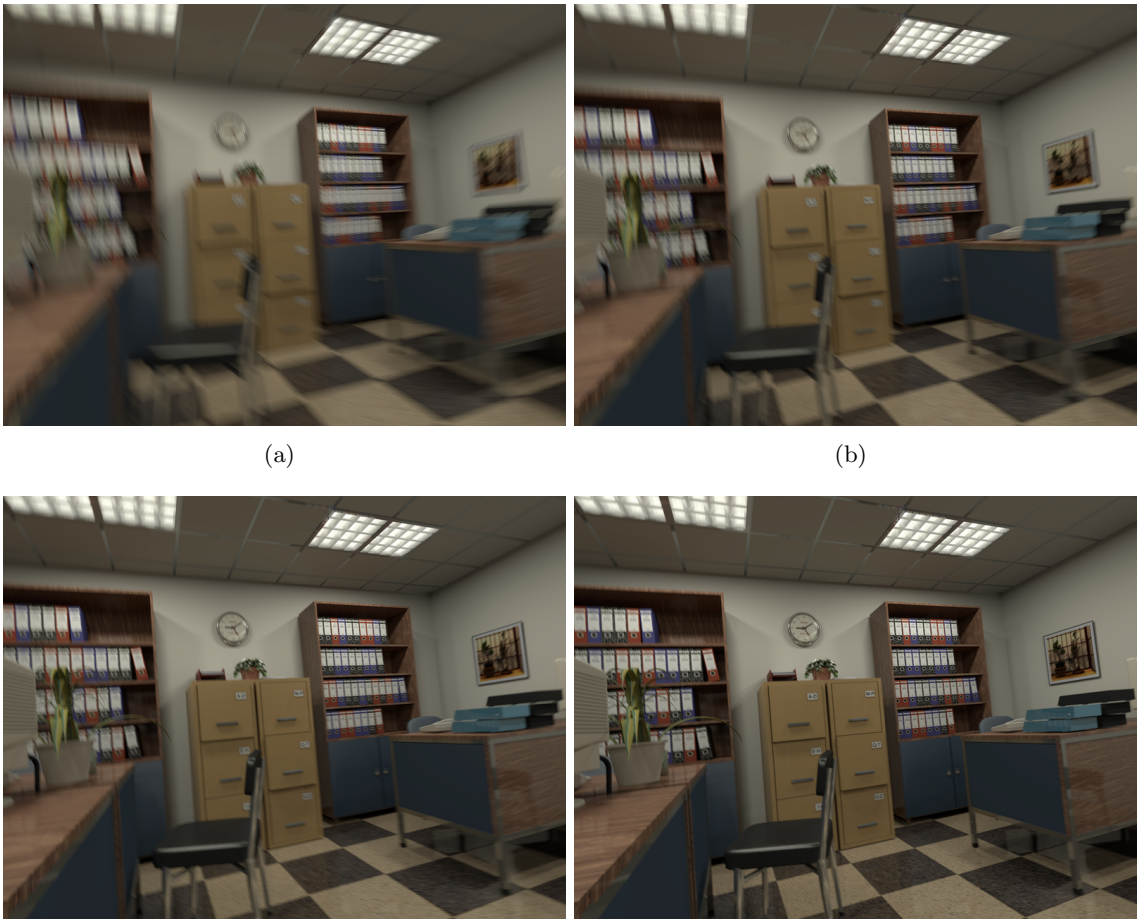
Furthermore, having complete ground truth data in order to compare results is highly desirable. This includes camera and IMU positions, velocities, sensor biases and noise characterization. High quality depth maps are also desirable, since having very accurate 3D reconstructions of the scene removes any unwanted bias from using a particular 3D reconstruction algorithm. Finally, as it will be seen in the evaluation method in Section 4.2.3, in order to fairly compare different experiments the system should be capable of synchronizing image capture at exact pre-determined positions in the trajectory. To guarantee this in a real life setting, the camera capture would have to be synchronized with a robotic arm or pan-tilt unit in order to obtain the same trajectory segments for all experiments.

Given all of these requirements, a synthetically generated dataset that models real life sensors as closely as possible seems to be the clear solution. The challenge now lies in how to produce a sensor rig trajectory that can generate valid IMU measurements and a way to render photo-realistic

images.

It first starts by recreating a rig based on sensors available in real life and that are commonly used: The camera is a 2.2MP monochrome camera, with a maximum frame rate of 170FPS, and is based on the Ximea MQ022MG-CM. The inertial measurement unit with sample rates of 200Hz is based on a MicroStrain 3DM-GX3.

4.2.1 Generating Photo-Realistic Visual-Inertial Synthetic Data



(a) (b)

(c) (d)

Figure 4.1: Examples of photo-realistic images generated by POV-Ray using the University of Colorado's Janus supercomputer at different frame-rates: (a) 15 FPS, (b) 30 FPS, (c) 60 FPS and (d) 120 FPS. To model motion blur, the luminance integration is done in irradiance space and then transformed back to intensities.

To generate simulated IMU and camera data, a spline must first be defined representing the motion of the sensor rig. For this study, a simple path that resembles a figure-8 pattern is used such that it excites all 6 degrees of freedom. The total duration of the motion is $7.07s$, in which the IMU travels a total of $1.76m$ with a maximum linear speed of $0.36m/s$. As the spline is C^2-continuous, the second derivative of the rig's position can be used to compute its acceleration data, which is sampled at $200Hz$.

Each simulated camera captures images at a resolution of 800x600, which are rendered using a simple pinhole camera model with a horizontal field of view of 80°, and an aspect ratio of 0.75. Additionally the camera is offset from the IMU by the translation vector $t = \begin{pmatrix} 0.1 & -0.05 & 0.05 \end{pmatrix}$ and rotated $+0.03$ radians about each axis to provide a non-trivial and more realistic rig configuration. POV-Ray[73], an open-source path-tracing tool, is used to render the camera images. The scene to be rendered is a small office environment, created by artist Jaime Vives Piqueres[77], consisting of several desks, chairs, and bookshelves. Images are rendered with the highest POV-Ray quality-level at 3200x2400 and then down-sampled to 800x600.

In order to simulate motion blur, the keyframes are rendered at each camera position sampled at $120Hz$. Keyframes are rendered with a modified version of POV-Ray, as used in [39], which additionally outputs depth values. With these depth values, each pixel location is reprojected at the next keyframe to compute the distance traveled in pixel-space. A new sample rate is then selected such that the greatest observed motion over the entire path is no greater than 0.5 pixels. This ensures combined images yield smooth, continuous streaks of motion-blur. For the selected scene and camera path, this results in a sample rate at just over $1400Hz$.

Using the computed path and sample rate a total of 10,188 rendered images are required. A single image at the selected resolution takes 15-17 minutes to render on a Intel i7-2.5GHz computer. This would require an estimated 120 days to render all images in the sequence. For this reason, 50 nodes from the University of Colorado's Janus supercomputer were allocated to render all 10,188 images in 3 days time.

Once all the images are rendered,a sequence of blurred images for each camera is created

as dictated by their dimensions, frame-rate and shutter speed. All images that would have been captured while the simulated camera's shutter is open will be blurred together into a single image. Instead of directly averaging images in intensity space, the average is done in irradiance space, having converted from the rendered intensities using a common image sensor. In this case, the the *agfacolor-futura-100CD* camera response function was used as defined in [37].

To maintain a relative constant illumination at different exposure times, the camera gain parameter was simulated to compensate for dim images captured at low shutter speeds. Image noise was therefore added in order to emulate the effects of gain variations.

4.2.2 Visual-Inertial Tracking

Tracking is performed in a loosely coupled sliding windowed dense visual-inertial bundle adjuster [45]. The bundle adjuster receives as input relative constraints between poses from the visual odometry engine as well as IMU measurements to form visual and inertial residuals.

4.2.2.1 Visual Residuals (e_v)

Visual only frame-to-frame tracking is performed in a 2.5D Lucas-Kanade[6] style photometric minimization using the Efficient Second Order minimization [49] technique:

$$e_v = \sum_i \rho \left(I_{cur}(\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p})) - I_{prev}(\mathbf{u}_i), \mathbf{c} \right) \tag{4.1}$$

where ρ is the Tukey robust error norm function with parameter $\mathbf{c}$. The previous image I_{pre}, and its corresponding depth map, are used to compute the photometric error e_v of a re-projected point onto the current live image I_{cur} through a warping function ω that takes camera calibration parameters (intrinsics), the pixel position $\mathbf{u}_i$, depth $\mathbf{d}_{\mathbf{u}_i}$ and an $\mathbb{SE}(3)$ transform parametrized with $\mathbf{p}$ as seen in (4.1). The relative transform between the previous and current frame, $T_{cp} = T(\mathbf{p})$, is the transform to be estimated and is initialized either as identity I_4, which corresponds to the previous camera position, or is seeded with the integration of the IMU measurements received between the last captured frame and the current frame.

If the visual estimation engine is initialized at the previous camera location, a coarse-to-fine approach must be used in order to guide the optimization to the basin of convergence. This is done by creating an image pyramid, where each level corresponds to a decimated image (half the resolution) of the image in the level below. This effectively acts as a smoother in the cost function. For these experiments, an image pyramid of 4 levels was used.

The optimization iterates over all pixels through each pyramid level, updating the pose estimate at each step. The optimizer performs a maximum of 50 iterations at each pyramid level, or can exit early if either of two conditions are met: the norm of the full pose update is less than $1e^{-5}$ or the error is increasing. Except for situations with extreme image or depth noise, as seen in Section 4.3, it is very unlikely that the optimization exits due to reaching the upper limit of 50 iterations.

Finally, the visual frame-to-frame relative transform T_{cp} estimated by the optimization is transferred into the IMU's reference frame – which is the privileged frame – and added into the bundle adjuster as a binary constraint.

4.2.2.2 Inertial Residuals (e_I)

Inertial measurements between frames are integrated forming residuals against the estimated poses as seen in Figure 4.2. Each pose in the bundle adjuster contains world poses comprised of a 3-DOF translation and 4-DOF rotation vector (quaternion parametrization). It also stores a 3-dimensional velocity vector and two 3-dimensional vectors for the accelerometer and gyroscope biases. The residuals are formed between the estimated parameters and the integrated state as seen in (4.2).

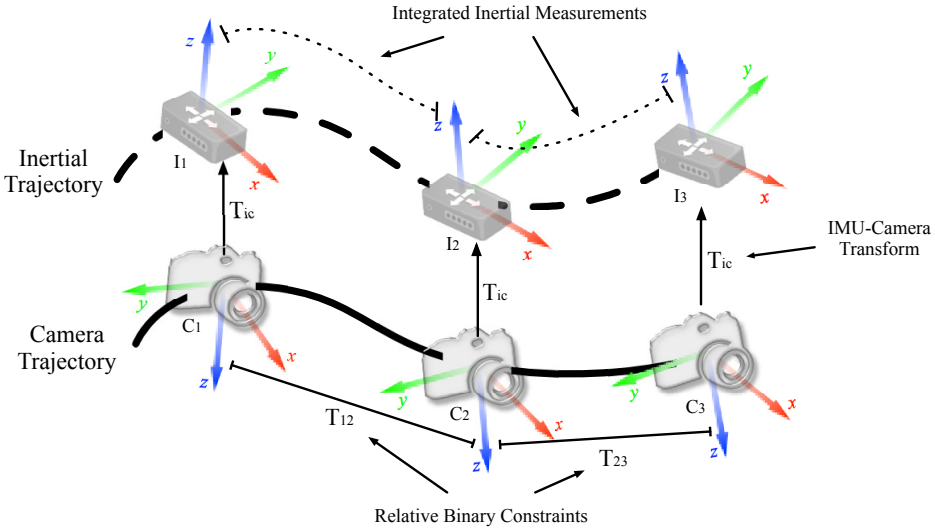


Figure 4.2: A windowed bundle adjuster takes frame-to-frame relative estimates, and its corresponding covariance, as binary constraints and jointly optimizes poses with integrated inertial measurements. The sensor rig calibration file, which contains camera intrinsics as well as the camera-to-IMU transform, is also provided.

$$e_I = \left\| \begin{bmatrix} p_{wp} - \hat{p} \\ log\left(q_{wp}^{-1} \otimes \hat{q}\right) \\ v_w - \hat{v} \\ b_g - \hat{b_g} \\ b_a - \hat{b_a} \end{bmatrix} \right\|^2 \tag{4.2}$$

where $(p_{wp} - \hat{p})$ is the translation residual, $log\left(q_{wp}^{-1} \otimes \hat{q}\right) \in R^3$ calculates the rotation residual in $so(3)$, $(v_w - \hat{v})$ is the velocity residual, and $(b_g - \hat{b_g})$ and $(b_a - \hat{b_a})$ are the gyro and accelerometer bias residuals respectively.

The rig trajectory described in Section 4.2.1 was generated to excite at least 2 axes at any given time. Given the size of the sliding window and the unambiguity of scale from using metric depth maps, no marginalization or conditioning is performed as all parameters are observable.

The bundle adjuster requires an initialization phase in which the initial velocities of all the poses, as well as the accelerometer and gyroscope biases, must be estimated prior to making use of

the IMU. This includes any inertial measurement integrations used for seeding the visual odometry engine. As such, the system relies on visual only estimations for the first few frames. For the experiments, an initial window of 30-60 frames was used to bootstrap the bundle adjustment, which then maintains a sliding window of 15 frames.

4.2.3 Evaluation Method

There are different ways of evaluating the accuracy of SLAM systems. Typical methods include open loop accuracy, where a trajectory loop is performed in such a way that the initial and final poses are the same. The error, in this case, is the difference between the initial and final poses. This clearly has a major drawback since it does not take in consideration the full trajectory, and a system that serendipitously ends up in the same place it started, or one that never estimated any movement, will show low error when in reality the *trajectories* might not match.

An alternate metric is to measure poses from a world coordinate frame, and the final error is the summation of all the errors between the estimated poses and the ground truth poses. This also has a some drawbacks since estimation systems usually provide relative and not absolute global poses. To obtain global poses, the relative estimates must be compounded together in a chain. This means, however, that errors are also compounded: a system that performs a poor estimation in the beginning of the chain will continue to show high errors even if the subsequent estimates were very accurate.

Finally, another metric would be to use the mean relative pose error. All the relative pose errors are added together, and divided by the total number of estimates. This also has some problems, in particular if a comparison between cameras operating at different frame-rates is desired. It is clear that a high frame-rate camera will incur lower mean relative pose errors compared to a slower frame-rate camera, simply by virtue that the slower camera will have to estimate larger transforms at each step.

A method that overcomes all of these shortcomings, and is the metric used in this work, is to compute errors by considering only the relative transformations between certain *key* poses,

as introduced in [52]. This is particularly useful in rigs with different sensor modalities, since all error calculations are based on the corrected trajectory of the robot. Since the robot will follow the exact same path for every experiment, segments of the trajectory which are common in all experiments can be chosen. In this way, a total trajectory error can be computed based on the common segments all the experiments share.

In particular for all of the experiments performed in this work, the segments of the slowest camera – that is, 15 Frames Per Second – were chosen to be the comparison segments given that all other camera frame-rates are divisible by this rate. Thus, for the 15 FPS camera each single segment corresponds to a frame-to-frame transformation, for 30 FPS it is the compounding of 2 frames, for 60 FPS the compounding of 4 frames and for 120 FPS the compounding of 8 frames (Figure 4.3). The final error, therefore, is the accumulation of these individual segment errors over the whole trajectory.

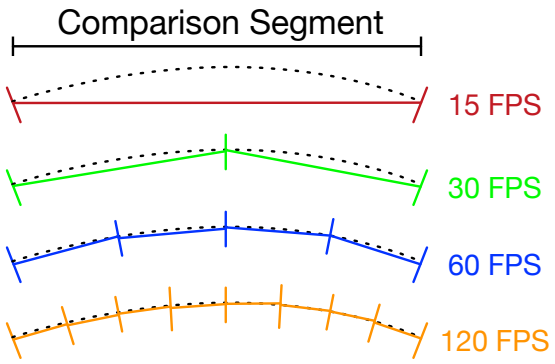


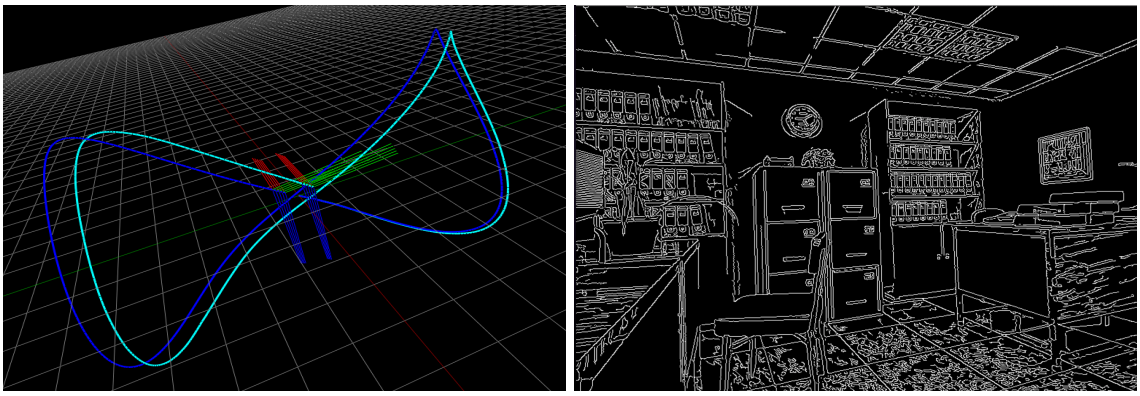
Figure 4.3: A comparison segment is the metric used to compare different camera frame rates, with the minimum size being set by the slowest frame rate: 15 FPS. Thus, at 15 FPS the comparison segment is comprised of only 1 odometry segment, for 30 FPS it is composed of 2, for 60 FPS it is composed of 4, up to 8 odometry segments for the 120 FPS case.

4.3 Results

The following are the results of the experiments that were conducted by permuting different parameters in the camera and the optimization engine. This yielded hundreds of experiments

which were all identically replicated and over the exact same trajectory, executed on fully dense and semi-dense tracking algorithm implementations.

Both of these methods run the same optimization code, with the only difference being in the number of pixels that contribute to the solution. Whereas in the fully dense *all* pixels are used in the pose estimate, for semi-dense only the edges are utilized (Figure 4.4). The experiments are further expand by analyzing the influence of the IMU in the estimation results.



(a) (b)

Figure 4.4: Figure (a) shows the figure-8 trajectory used in all experiments, with blue being the ground truth trajectory and light blue being the estimated trajectory. Figure (b) shows the edges of the scene in Figure 4.1 which are the only pixels used in the semi-dense implementation.

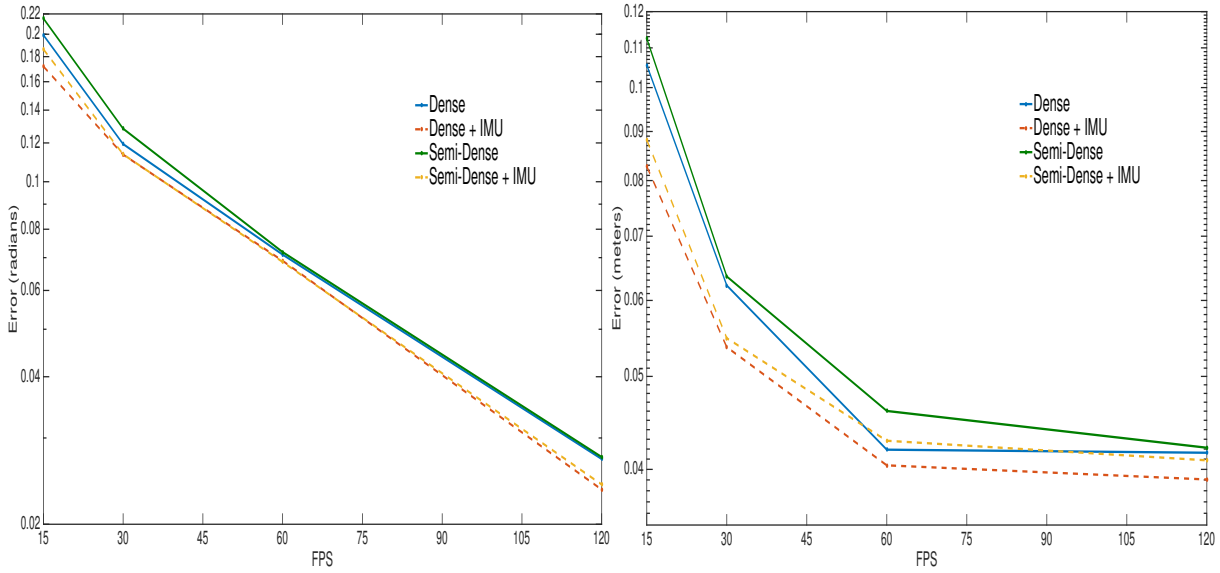
It should be noted that when using the IMU, the estimates are those obtained from the bundle adjuster after it has been initialized. Furthermore, the initialization of the visual odometry engine when using the IMU is done by integrating the inertial measurements between the last and the current frame instead of using an image pyramid.

Finally, even though the errors seem relatively small in some experiments, it should be noted that these errors are the accumulated *per trajectory*, which is approximately 1.8 meters. For a robotic platform traveling hundreds and thousands of meters at a time, a difference of millimeters or centimeters for every couple of meters traveled adds up to considerable error.

4.3.1 Frame-Rate

In this experiment, images are generated at different frame-rates (15, 30, 60 and 120 FPS) as seen before in Figure 4.1. The rotation error (in radians) and the translation error (in meters) can be seen in Figure 4.5. As expected, all methods incur high error at slower frame-rates; the highest being semi-dense. The use of an IMU undoubtedly helps throughout the different frame-rates, more so on the rotation component rather than the translation. This is not surprising since, unlike the gyroscope, the accelerometer requires a double integration.

Another interesting observation is that as the frame-rate increases the difference between fully dense and semi-dense, and even both methods aided with an IMU, decreases considerably to become almost indistinguishable.



(a) (b)

Figure 4.5: Frame-Rate vs Error: Rotation (a) and translation (b) errors at different frame-rates. *(Logarithmic scale in the Y-axis.)*

4.3.2 Shutter Speed

The shutter speed of the camera is now adjusted. This is done by simulating a camera at a particular frame-rate and adjusting the exposure time between the capture interval accordingly. This allows generating images at different shutter speeds, and the effects of blur in the pose estimation, while maintaining a fixed frame-rate, can be analyzed.

For this particular experiment, a frame-rate of 15 FPS was selected. The capture interval at this frame-rate is 0.0667 seconds. As explained in Section 4.2.1, blur is simulated by averaging images in irradiance space; the longer the exposure time, the more images are averaged. Different exposure times were selected, in particular those that correspond to the higher frame-rates studied here: From the initial 15 FPS, through 30 FPS (0.0333s), 60 FPS (0.0167s), 90 FPS (0.0111s) up to 120 FPS (0.0083s).

This experiment allows to better discern the nature of the error reduction in the previous section, where the frame-rate was adjusted. By maintaining a constant frame-rate, any other factors implicit in cameras that operate at different rates – the most obvious being the distance traveled between two image captures – can be factored out.

Figure 4.6 shows the result of the experiments assuming constant illumination. It can be seen that as the exposure time is decreased, and thus also image blur, the estimation error decreases down almost comparable to the camera operating at 120 FPS as seen in the previous section.

4.3.3 Image Resolution

For the next experiments, the image resolution is adjusted from the nominal 800x600 down to 400x300, 200x150 and 100x75. Since the blur observed by down-sampling a blurry image is not the same as the blur observed in an image from a camera natively capturing at a lower resolution, the blurred images have to be re-generated with these new dimensions. That is, first downsampled to the correct image size and then averaged in irradiance space at this resolution.

Figure 4.7 shows the results of this experiment. It can be observed that as the image resolution

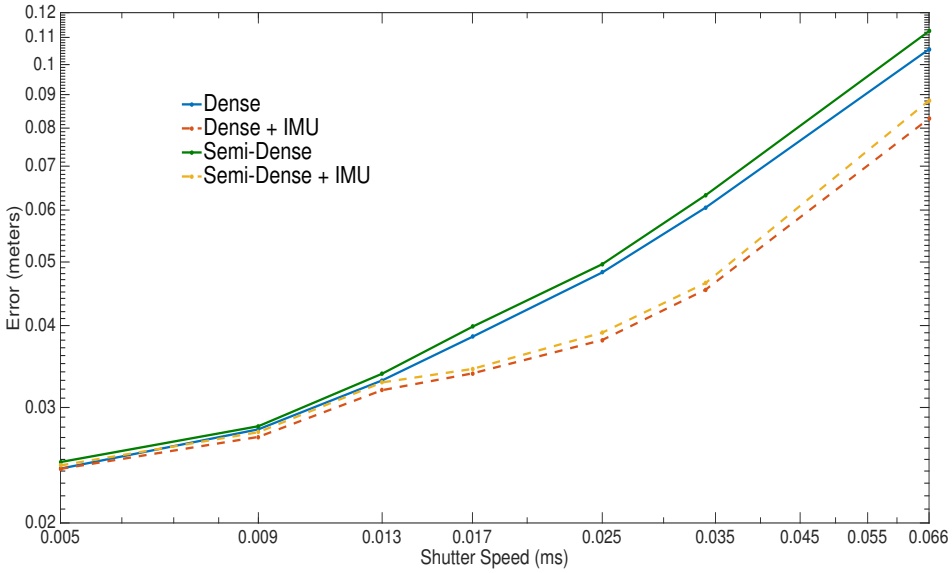


Figure 4.6: Shutter Speed vs Error: Shutter speeds/exposure times are adjusted for a high frame-rate camera in order to effectively lower its capture rate and simulate blur. *(Logarithmic scale on both axes.)*

is increased, so does the accuracy of all methods. Furthermore, given the shape of this graph it can be seen that further higher resolutions would continue to achieve better results.

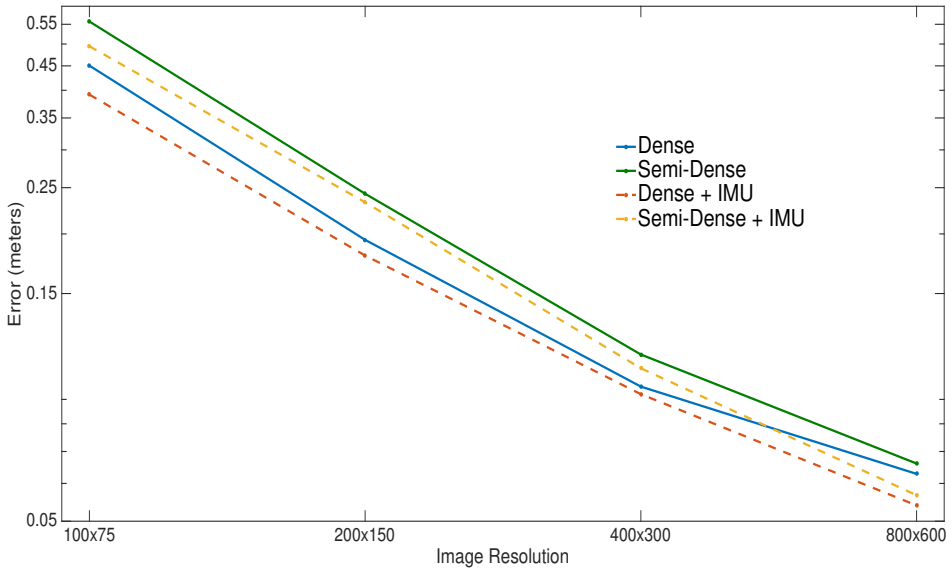


Figure 4.7: Image Resolution vs Error: Trajectory error at different image resolutions. (*Logarithmic scale on the Y-axis.)*

4.3.4 Pixel Count

The objective of this experiment is to analyze how the percentage of pixel contributions affects estimation errors. For this, the estimation starts from a semi-dense implementation (which contains approximately 5-10% of image pixels for this particular scene) and slowly increase the number of pixels contributing on the visual engine's optimization up to 100% which corresponds to a fully dense method. The pixels are chosen starting from those with the highest gradient, which are the ones where semi-dense operates, up to non-edge pixels which are randomly selected until the specified percentage of pixels is reached.

All the runs at different frame-rates (15, 30, 60 and 120 FPS) are averaged to obtain the graph in Figure 4.8. It is very interesting to see how the pose error quickly tapers down after only 30% of pixel contributions, and overall as have been seen throughout all the experiments, the difference between fully dense (100%) and semi-dense (5-10%), within each plot, is not very considerable.

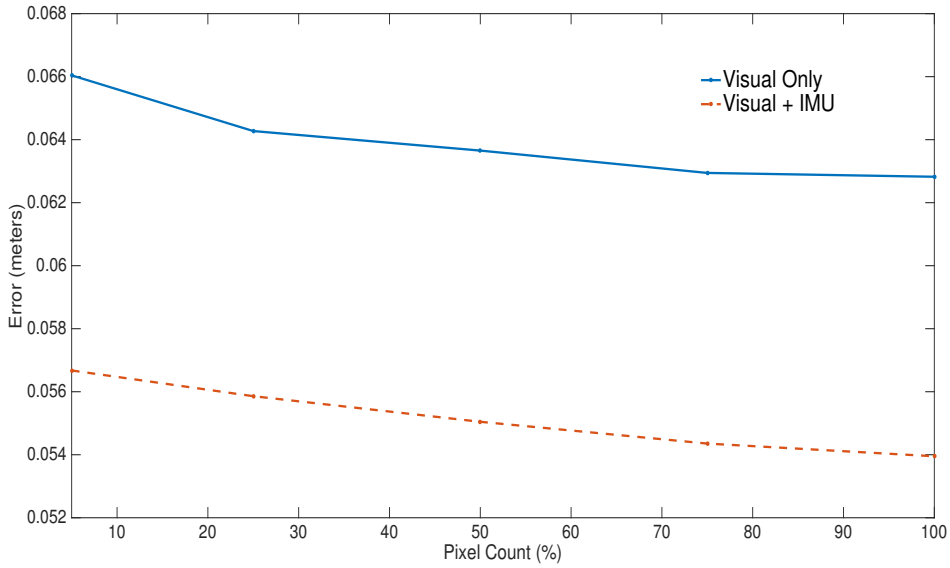


Figure 4.8: Pixel Count vs Error: The graph shows the pixel contribution for the visual odometry engine's optimization against the trajectory error.

4.3.5 Image and Depth Noise

This final experiment studies how noise in the image and the depth map affect accuracy throughout the different methods being studied here. Image noise, which corresponds in particular to camera and shot noise, is typically seen when using a cheap camera sensor, setting a high gain value or capturing images with very low illumination. 3D reconstruction, on the other hand, is not only correlated directly to image quality and as such affected by image noise, but it also has its own noise characteristics: systemic errors in the 3D reconstruction algorithms (for instance, the use of a smoothing function), bad camera calibration or improper baseline for a given scene, all affect the quality of the reconstruction which in turn affect localization accuracy.

In all the runs so far, noise has been added to the data, including inertial measurements, to simulate measurement noise and thus make the experiments more realistic. For the IMU, the noise was modeled from the MicroStrain 3DM-GX3 sensor and was added to the ground truth measurements obtained by sampling the C^2-continuous curve. This noise was then propagated as sensor uncertainties through the estimation model.

For this particular experiment, the image noise σ was increased in piecemeal fashion from an intensity value of 1 up to 35 (for the 8-bit, greyscale image with maximum value of 255) and the depth noise σ from 0.01 meters up to 0.5 meters. The results can be seen in Figure 4.9.

From the image noise graph, in sub-figure (a), it can be seen how the use of the IMU considerably helps in maintaining a low pose error and overall, the fully dense method surpasses the semi-dense in accuracy. Interestingly enough the reverse is seen in sub-figure (b). Throughout all of these experiments, the tendency has been for the fully dense method to have overall a higher accuracy compared to the semi-dense method. However, in the case of a noisy depth map, the semi-dense method seems more robust.

At closer inspection, it can be seen that using only edges for the semi-dense method makes the whole system more robust against depth noise. For semi-dense, a noisy depth measurement would reproject in an area off the edge which would incur in a very high photometric error that

would be down-weighted by the robust norm. This is also shared with a fully dense method. The difference, however, lies in the other pixels that are contributing to the fully dense estimation. Those pixels are in texture-less areas of the image, so a noisy depth value will most likely still fall in the same texture-less area, producing a small photometric error. Thus, the estimate will not be down-weighted by the robust norm and will be erroneously taken in consideration by the optimizer. It is the addition of these erroneous measurements that eventually diverge the fully dense method from the correct solution.

4.3.6 Number of Iterations and Computation Time

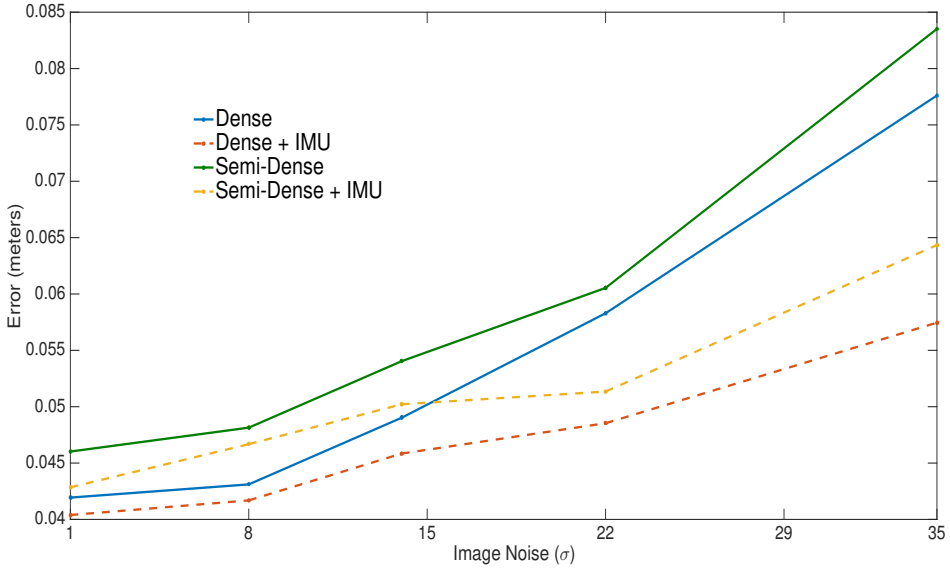
One of the most interesting results throughout these experiments is related to the number of iterations required for the visual odometry engine, and in particular, the per frame estimation time.

The number of iterations and estimation times were averaged throughout the different frame-rates, obtaining an average count of iterations and computation time required per frame as seen in Tables 4.1 and 4.2. Computation times were calculated by using a multi-threaded CPU implementation of the dense/semi-dense visual odometry engine and bundle adjuster, running on an Intel i7 2.5GHz computer. As such, the performance is not real time as expected if the implementations were written for a GPU, but it does guarantee consistency throughout the experiments.

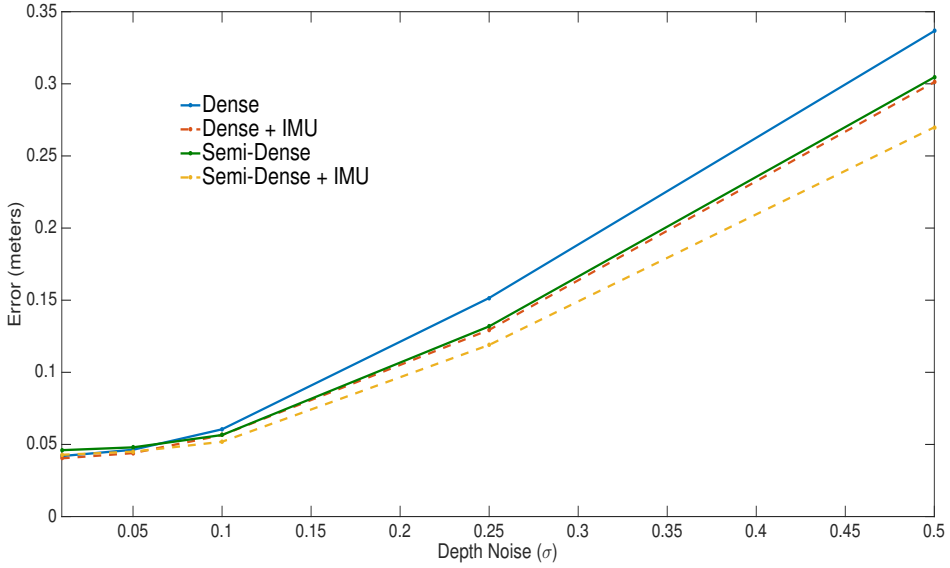
In Table 4.1 several interesting points stand out. First, the number of iterations decreases as the frame-rate increases, regardless of the method used. This is not surprising, since the estimation engine does less work due to the fact that the images are closer in time.

Another interesting point is that the number of iterations for semi-dense is higher than that of a fully dense method. Since fully dense methods make use of more information, it can be inferred that its estimation steps are "better" in that they get closer to the solution faster at each iteration compared to semi-dense.

Finally, a very noticeable point is how the number of iterations is reduced drastically by



(a)



(b)

Figure 4.9: Noise vs Error: Pose errors as noise is added to the image (a) and depth map (b). Unlike in most of the previous experiments, fully dense methods seem to perform poorly when sufficient noise ($\sigma > 0.06$) is added to the depth map.

taking advantage of the IMU seeding. Since the visual odometry engine is no longer required to perform a coarse-to-fine approach via an image pyramid, the number of iterations overall is reduced by more than half.

The use of an IMU to initialize visual odometry also reflects in lower estimation times, as

seen in Table 4.2. This is more noticeable when comparing visual only dense with inertial aided dense, since each iteration in a fully dense method is computationally very expensive. This is due to the fact that it has to iterate through the whole image, which for a 800x600 resolution image consists of 480,000 pixels. In contrast, semi-dense methods only make use of around 5-10% of an image. Interestingly, even though the number of iterations of semi-dense was higher than that of a fully dense method, as seen in the previous table, the estimation time is still considerably less.

Table 4.1: Average Number of Iterations Per Frame

Method	FPS			
	15	30	60	120
Dense	14.6	14.1	13.9	13.8
Dense + IMU	6.2	5.0	3.1	2.4
Semi-Dense	16.2	15.5	15.3	15.0
Semi-Dense + IMU	6.3	5.2	3.5	2.8

Table 4.2: Average Estimation Time Per Frame

Method	Time (ms)
Dense	73.09
Dense + IMU	54.30
Semi-Dense	22.97
Semi-Dense + IMU	20.84

4.4 Conclusion

A series of experiments were conducted in order to gain a better understanding of the different direct visual odometry methods, particularly those that make use of inertial measurement units. Throughout these experiments it has been seen that, as suspected, fully dense methods perform better under many situations. The exception occurred in the case when noise was added to the depth map.

It was also seen that the addition of an inertial measurement unit undoubtedly helps in

maintaining low pose errors, especially in the presence of extreme motion blur incurred by low frame-rate cameras or long exposure times. However, throughout the different experiments a common trend could be seen: with a high enough frame-rate, and therefore low image blur, all methods seem to perform relatively similar.

This is particularly important in cases when computation time is a big factor. Compared to fully dense methods, semi-dense takes almost 3x less time while achieving comparable results under low blur conditions. This relatively high accuracy at considerably low pixel count is possible due to the fact that semi-dense operates only on edges, which are the pixels with the highest information in an image.

Similarly, the inclusion of an IMU increases the accuracy of the system especially in areas with high camera movement. An additional advantage of using an IMU is that it removes the need to use an image pyramid, saving considerable computation cycles by guiding the visual odometry engine to the convergence basin. The optimizer would only need to perform a couple of iterations to refine the integrated pose provided by the IMU.

It should be noted that the way the IMU was used in this paper, specifically through a sliding window bundle adjuster where visual odometry constraints are added as binary constraints, is only one particular implementation of a visual inertial tracker and it would not be surprising that a different implementation – for instance, a filter based tracker – would yield different results. It would be hard to see, however, those differences being too significant.

While these early findings are certainly insightful, it should be noted that not all the possible properties found in real-life settings were modeled. The work chose to focus instead on the properties that are **intrinsic** to the sensor rig (frame rate, resolution, etc.), rather than visual phenomenon that are typically corrected prior to any estimation such as vignetting and lens distortion.

In short, the results can be summarized as follows: For applications that require very high accuracy, an inertial aided fully dense method is the best choice. It provides the highest accuracy, especially in the presence of extreme motion blur, while at the same time being faster than a vision-only fully dense method by making use of the IMU seeding. The elbow in the curve, especially

when considering computation time, is without a doubt an inertial aided semi-dense algorithm. It still provides a good degree of accuracy while being extremely efficient. Finally, for applications that will not encounter extreme camera motions or alternatively if a camera that supports very high frame-rates is used, a vision only approach would suffice for most general situations.

Chapter 5

Large Scale Dense Visual-Inertial SLAM

The previous chapter presented a study between different direct methods for tracking, in particular a comparison between fully dense and semi-dense methods was presented and how the camera characteristics like image resolution and frame rate affected localization accuracy. Also, a secondary sensor was introduced into the mix: the Inertial Measurement Unit (IMU) which aided in tracking especially during fast camera motions with high amount of blur. The experiments presented were conducted over synthetic data, due to the necessity of having repeatability and ground-truth information.

Furthermore, with previous systems, the maps tended to be relatively small in size and limited to a single scene (a desk or a room) up to a full floor of an office building. Large scale mapping, especially with dense maps, are rare and typically an area left only for autonomous vehicle researchers. However, its application in any robotic platform is fundamental. The next section will therefore present a large scale dense SLAM system for a robotic platform, capable of operating in large environments and even outdoors in adverse weather conditions due to the resilience of the multi-sensor approach of visual plus inertial localization. The system uses a global representation of a map, based on a voxel hashing technique [56], and stores only *geometric* information of the scene – that is, 3D data – rather than *photometric* information which would be necessary for a full map registration using a dense system.

5.1 Introduction

Large scale SLAM is an important research area in robotics and computer vision, since for a robot to be useful it needs to be able to operate autonomously long-term in a potentially large environment. Perhaps the most popular approaches to large scale SLAM are point cloud based [72, 29, 100]. Normally, such approaches use a point cloud to reconstruct the scene and cannot reconstruct connected surfaces. These approaches fuse the individual point clouds from different views and present the final reconstruction result as a point cloud as well. However, this has the drawback that it requires much more memory for storage and also it is incapable of providing the connected surface component which is important for planning and control of robotic platforms.

Dense SLAM with volumetric representation have been popular in recent years [69, 43, 46]. Such techniques use a Truncated Signed Distance Function (TSDF) to represent the scene surface and incrementally refine it with the registered depth frames. Meanwhile, similar approaches have also been proposed in monocular SLAM [70, 15]. Usually, these approaches use a fixed amount of GPU memory for tracking and reconstruction; this hard constraint limits the size of the reconstructed scene and cannot be used for large scale dense SLAM.

Several systems have been proposed in order to reconstruct large scale scenes with volumetric approaches. [110, 97] proposed an octree based approach for indoor dense SLAM. [80, 104, 28] used a fixed bounded volume to represent portions of the scene and incrementally reconstruct it with a rolling scheme. However, these approaches mostly focus on the indoor scene and uses RGB-D sensors, which cannot perform outdoor SLAM due to their sensitivity to sunlight. They also rely heavily on the Iterative Closest Point (ICP) algorithm for tracking – that is, in short, point cloud alignment – which is not suitable for outdoor environments due to the lower quality of the depth maps from the stereo sensors when operating outside. Besides, a combined ICP + photometric tracking approach [80] may also fail if the scene only contains simple geometric or color information.

In this section a new large scale dense visual inertial SLAM system is presented that does not rely on active depth sensing but uses a stereo rig to generate 3D depth maps. The system uses a

rolling grid fusion scheme which effectively manages GPU memory and is capable of reconstructing a fully dense scene online.

The system obtains depth maps using the Efficient Large Scale stereo matching [35] described in Chapter 2, and simultaneously localizes the camera based on whole image alignment and inertial data while reconstructing the scene with SDF fusion. The system automatically saves and loads data from device, host memory and hard disk, and generates a mesh (.obj, .dae, .ply formats) of the large scene (e.g. 20 millions vertices) in seconds. Given these components, a wide range of applications can be developed, especially in robotics where the proposed system is capable of providing high fidelity meshes of any outdoor environment for use in path planning and control.

The most similar approaches to the one presented in this section are [104, 71, 89, 79]. There are, however, key methodological differences:

(1) This approach focuses on outdoor scenes and uses stereo data while [104, 71, 79] use an active sensor (i.e. Kinect) and mainly focus on indoor scenes.

(2) The system uses a dense visual inertial stereo system for tracking while others rely solely on cameras, either via an ICP or photometric alignment approach.

(3) This approach uses a simple rolling grid SDF pipeline for reconstruction while [71, 79] used a hashing scheme, [104] used a rolling SDF scheme and [89] uses a fix grid volume scheme.

5.2 Rolling Grid Based Volumetric Map

This section will briefly cover the map representation of this method in order to better understand the tracking system described in the next section. The system uses a rolling grid based volumetric representation, namely the *Grid SDF*, to reconstruct a 3D model of the scene in the current camera view.

An SDF, or Signed Distance Function, is a function that determines the distance of a given point to a particular boundary (i.e. a surface), with the sign determining if the point is inside or outside the boundary. It is a frequently used data representation for 3D spaces, and can be used

as a global map that stores geometric information of a scene. In voxel form, each voxel stores the sign and distance to the closest surface. It typically requires allocating memory for each voxel, regardless of how close it is to the surface.

The Grid SDF proposed here, however, can be considered as a hybrid: each cell in the grid may or may not contain data, and if it does, then memory is allocated for a smaller SDF voxel cube (Figure 5.1).

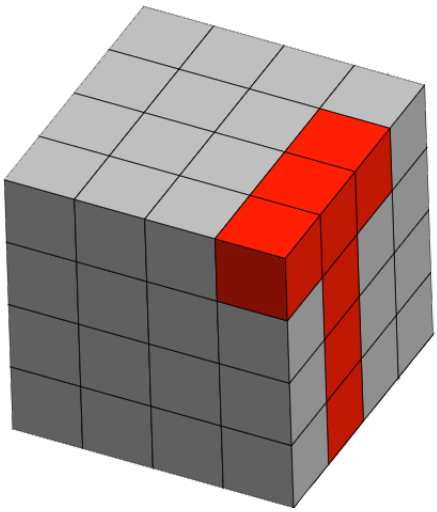


Figure 5.1: An example of the *Grid SDF*. In this example, the grid is comprised of $(8 * 8 * 8)$ cells. The GPU memory of a cell is not initialized (gray cells) until there is actual information available, in which case a small voxel sub-cube is initialized (red cells). This allows for an efficient use of memory to only areas where there is actual depth information.

Each cell in the Grid SDF is a small $NxNxN$ dimensional TSDF (Truncated Signed Distance Function) volume and contains a pointer to GPU memory. The Grid SDF contains (x_g, y_g, z_g) cells in each dimension. Assuming that the resolution of each voxel is r_v, the maximum size of the scene in each dimension is the number of cells in that dimension times the size of the SDF. For example, for the x dimension this means:

$$r_x = r_v * N * x_g. \tag{5.1}$$

The values of x_g and y_g are usually selected depending on the horizontal and vertical field

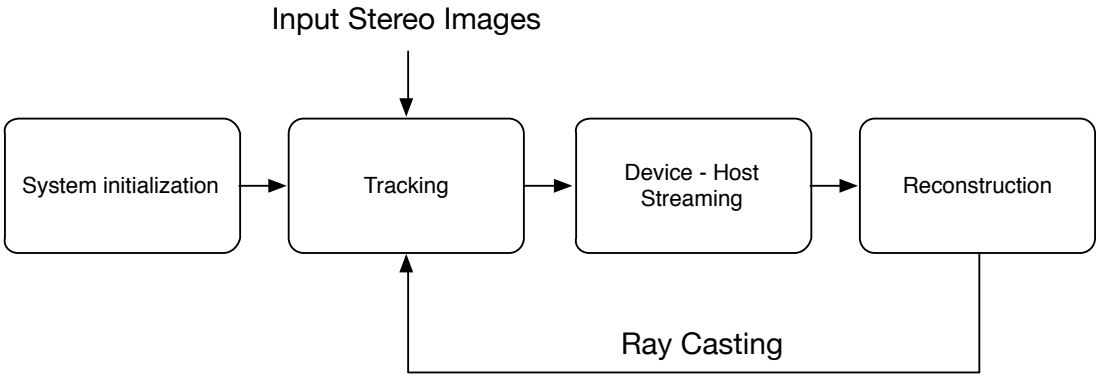


Figure 5.2: After system initialization, the proposed system localizes the pose of cameras and incrementally reconstructs the scene with a rolling SDF scheme. Portions of the scene that are out of the camera view will be streamed from the GPU memory to the CPU memory (or the hard disk) directly.

of view of the camera, and z_g is based on the maximum depth measurement desired. This can be selected dependent on the maximum expected scene depth, or ideally, thresholded by the maximum depth uncertainty desired given the rig's stereo baseline. Notice that when initializing the grid, the system does not allocate any GPU memory for any cell. For a more detailed description of the rolling grid SDF representation, please refer to the original paper [56].

5.3 Visual-Inertial Tracking

Tracking is performed in a windowed dense visual inertial joint optimizer similar to a visual inertial bundle adjuster [45]. Visual-only frame-to-frame constraints are transformed into the IMU frame and added into the joint optimizer as binary constraints. Inertial measurements between frames are integrated forming residuals against the estimated poses. Velocities and IMU biases are also estimated, and are carried through in the sliding window. The camera to IMU transform T_{ic} is previously calibrated offline and does not change during the run as the sensors are rigidly attached to the rig.

As seen before, visual tracking is performed by a Lucas-Kanade [6] style whole-image alignment algorithm via the Efficient Second Order Minimization (ESM) technique [49], and a 6-DOF

camera transform is estimated by minimizing the photometric error (e_v) between a reference image and the current live image:

$$e_V = \sum_i \rho \left(I_{live}(\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p})) - I_{ref}(\mathbf{u}_i), \mathbf{c} \right) \tag{5.2}$$

where ω is the warping function as defined in Chapter 2:

$$\omega(\mathbf{u}_i, \mathbf{p} + \Delta\mathbf{p}) = \pi^{-1} \left(\mathbf{K}\, T(\mathbf{p})\, T(\Delta\mathbf{p})\, \mathbf{K}^{-1}\, \mathring{\mathbf{u}}_i\, \mathbf{d}_{\mathbf{u}_i} \right) \tag{5.3}$$

The pixel $\mathbf{u}_i$ in the reference frame is back-projected using the camera matrix $\mathbf{K^{-1}}$ and the associated depth value $\mathbf{d}_{\mathbf{u}_i}$ obtained by the ELAS stereo reconstruction algorithm. The 3D point is then transferred into the live frame via the estimated transform, $T_{lr} = T(\mathbf{p})$, and projected through $\mathbf{K}$ onto the live camera.

The pose covariances from the visual tracking system are then added into the joint optimizer, which runs once a sufficient number of frames and inertial measurements are obtained. The covariance of the inertial residual between two consecutive frames is dependent on the number of measurements between images, and as such must be carried forward during the integration process (Figure 5.3). Details about inertial integration and error propagation can be found in [45].

Inertial residuals e_I between the parameters and the integrated state take the form of:

$$e_I = \left\| \begin{bmatrix} p_{wp} - \hat{p} \\ log\left(q_{wp}^{-1} \otimes \hat{q}\right) \\ v_w - \hat{v} \\ b_g - \hat{b_g} \\ b_a - \hat{b_a} \end{bmatrix} \right\|^2 \tag{5.4}$$

where $(p_{wp} - \hat{p})$ is the translation residual, $log\left(q_{wp}^{-1} \otimes \hat{q}\right) \in R^3$ calculates the rotation residual in $\mathfrak{so}(3)$, $(v_w - \hat{v})$ is the velocity residual, and $(b_g - \hat{b_g})$ and $(b_a - \hat{b_a})$ are the gyro and accelerometer bias residuals respectively.

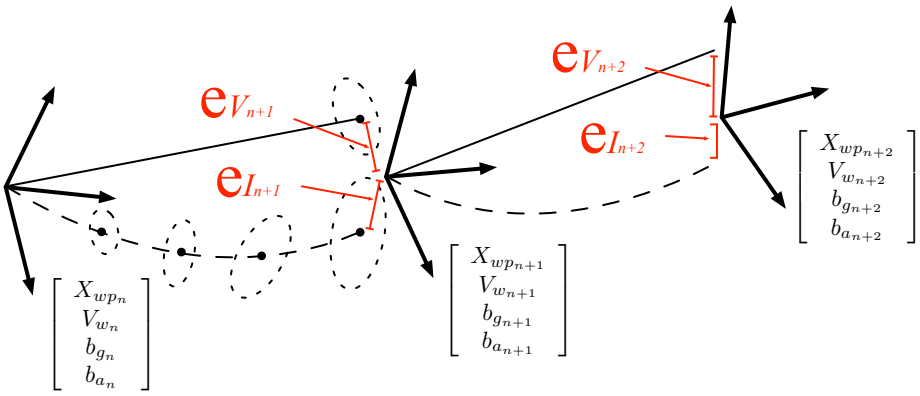


Figure 5.3: Errors from the vision system (e_v) are formed by compounding the estimated relative transforms with world poses. Similarly, inertial errors (e_I) are formed by integrating inertial measurements. Uncertainties (shown as ellipsoids) are used to weigh in residuals for the estimation of the state parameters: world poses comprised of a translation (p) and rotation (q) vector ($X_{wp} = [p_{wp} \;\; q_{wp}]^T$), velocities ($V_w$), accelerometer biases (b_a) and gyroscope biases (b_g).

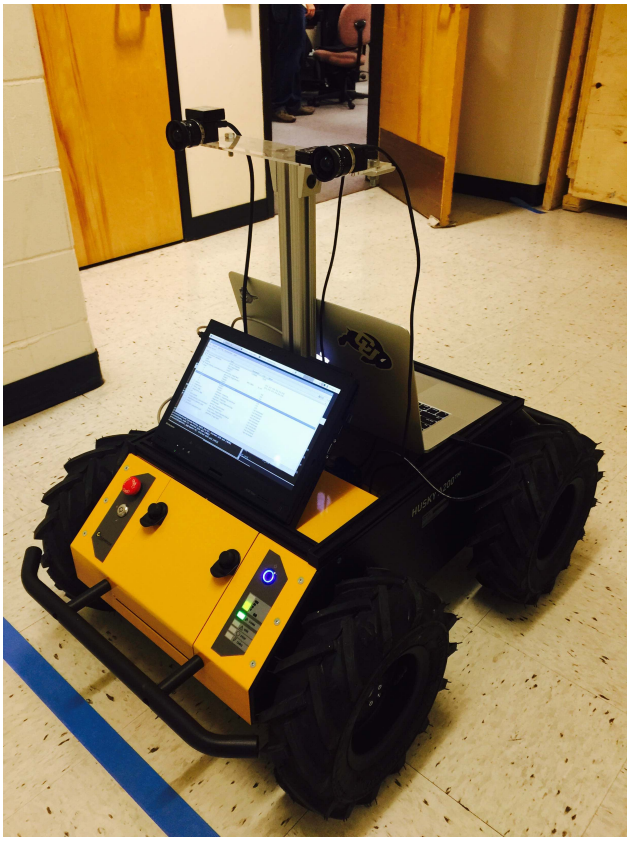
A total of 15 parameters per frame are estimated during the sliding window optimization: 6 for pose parameters, 3 for velocities, 3 for accelerometer biases and 3 for gyroscope biases. Initial velocities, as well as the biases, are estimated and kept up to date as the sliding window shifts during execution. Given the size of the sliding window and the unambiguity of scale from the stereo vision system, no marginalization or conditioning is done on the sliding window as all parameters are observable.

The dense visual inertial tracker initially performs visual odometry only using a coarse to fine approach via an image pyramid. After a minimum number of image frames are acquired, the sliding window is used and the image pyramid is no longer required, since the IMU is capable of seeding the visual odometry optimization by providing a hint of the camera pose. The window size used for all experiments was 15, with the minimum number of frames being 10.

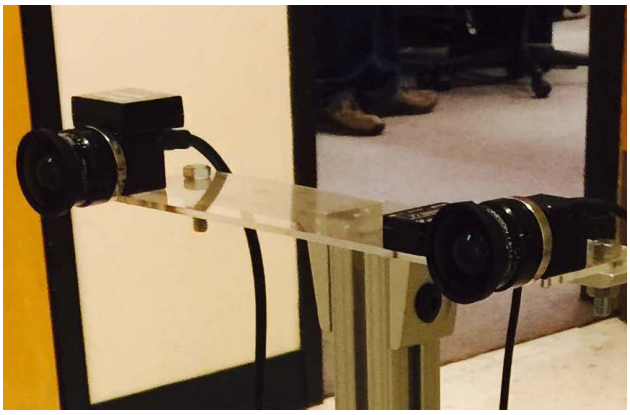
5.4 Results

The proposed system is tested by a hand held camera and a ClearPath Robotics Husky robot (Figure5.4) with two Ximea (MQ013MG-ON) gray scale cameras and a Microstrain 3DM-GX3-35 Inertial Measurement Unit (IMU). The camera intrinsics as well as sensor extrinsics are calibrated

offline with a method similar to [54], and the rigid sensor rig is attached to the robot via a T-mount.



(a)



(b)

Figure 5.4: The stereo camera pair and IMU sensor head on top of the Clearpath Husky robotic platform. This particular system is not autonomous, but is operated remotely via joystick.

The inclusion of inertial data enhances visual tracking in general, and in particular during fast

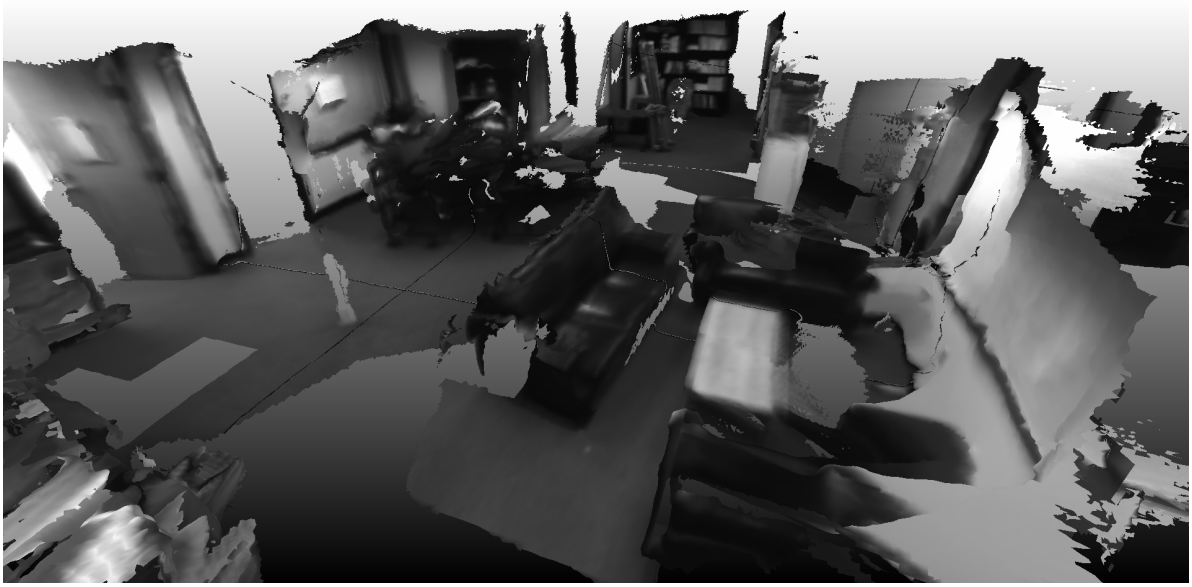
camera movements and low textured areas. The addition of the IMU also speeds up visual tracking, since the typical coarse-to-fine pyramid scheme used in visual odometry is no longer required once the system has been bootstrapped. Instead, the visual tracking is initialized with an estimated pose given by the integration of inertial measurements from the last frame up to the point where a new image is captured. In this way, only a refinement in the form of a few iterations at full image resolution is required for the final pose estimate.

5.5 Conclusion

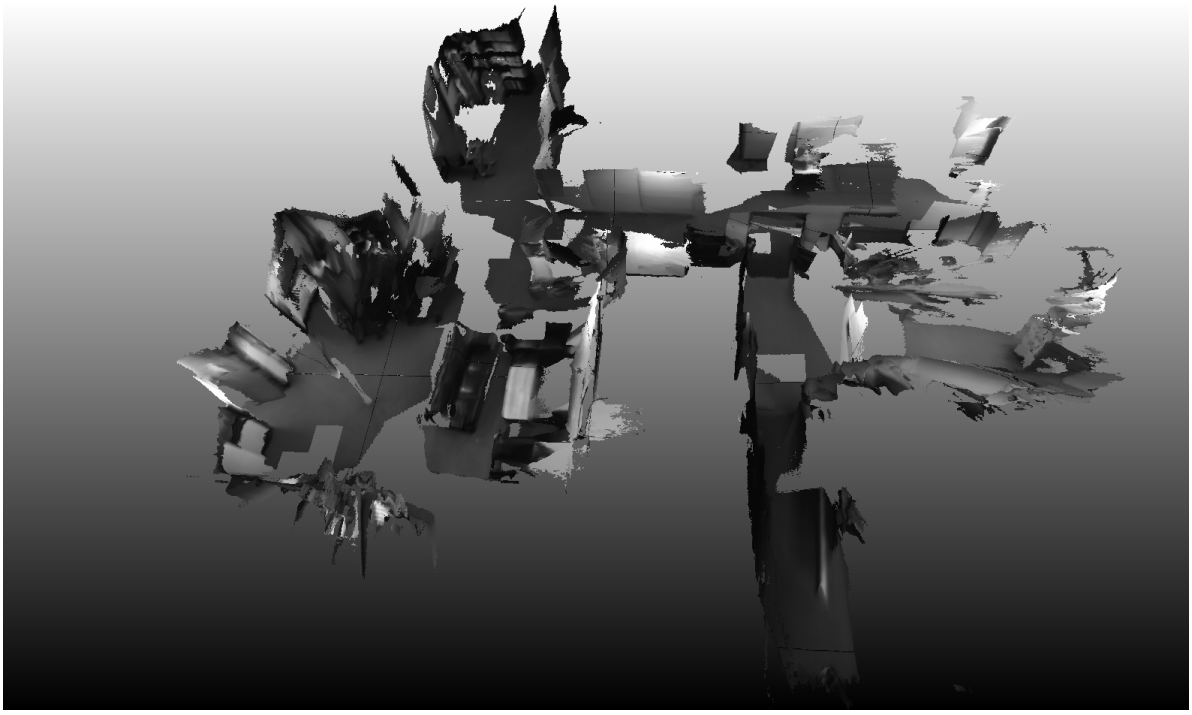
A large scale dense visual inertial SLAM system based on a rolling grid fusion scheme was presented. The proposed system manages the space into small volume grids and only allocates GPU memory for cells if data exists. A large scale dense mapping solution is obtained via a rolling grid scheme with simple index computation while the device and the host memory automatically stream between each other in order to reuse the GPU memory. Depending on the requirements of an actual application, the system utilizes stereo cameras in both indoor and outdoor scenes.

The system is tested in several outdoor and indoor scenes under different lighting (illumination changes), weather (e.g. snow, sunny), and motion conditions and shows promising results. The main contributions of the paper are: a new large scale outdoor dense mapping system based on stereo data and a new dense visual inertial dense tracking pipeline.

Although the system is robust to many real-world conditions, it has a main limitation: the final reconstruction and tracking results depend heavily on the quality of the depth images which can be improved by using global methods which have been proven to produce better reconstructions [40, 41]. The system also does not handle loop closures, which would increase considerably the tracking and therefore the final reconstruction.

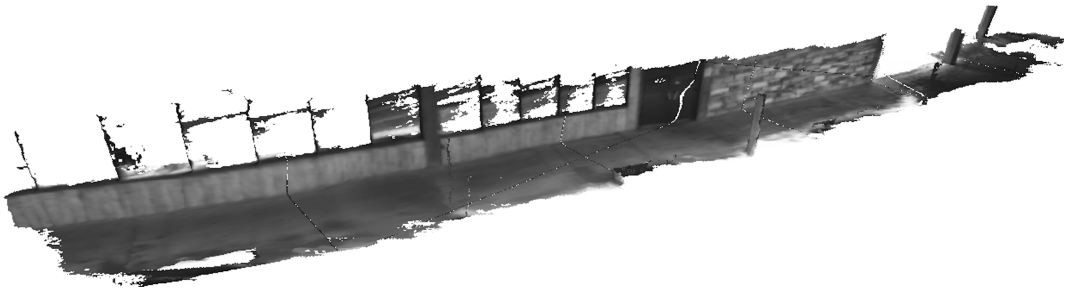


(a)



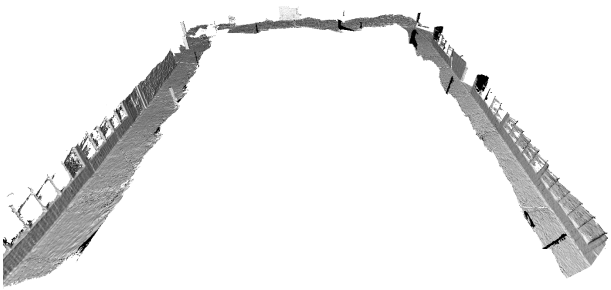
(b)

Figure 5.5: An example of the reconstruction result for an indoor office scene (a) and a top-down view of the final map of the trajectory (b).



(a)

(b)



(c)

Figure 5.6: An example of the reconstruction result for an outdoor scene from 7000 stereo frames (approximately 75 million vertices). a - b) Reconstruction detail of a scene with both shadow and harsh illumination, and snow on the ground. c) An overview of the camera path.

Chapter 6

Towards Robust Dense Visual SLAM

So far throughout this dissertation, all the tracking systems presented have been operating as visual-only or visual-inertial *odometry*: that is, the system performs frame to frame or frame to keyframe localization but registering against a map – be that pre-mapped at a different time or even registering against a recently traversed area – has not been accomplished.

There is a considerable amount of effort to register against a map, especially one that changes constantly either due to changes in illumination (time of day, seasons, weather, etc.) or simply by the dynamic nature of the environment. This last is particularly challenging, and is one of the currently active research areas in computer vision in order to accomplish true long-term autonomy.

This next chapter introduces the final component towards robust dense visual SLAM, and a key component in long-term autonomy: a system capable of registering against a map, even in the event of extreme illumination changes.

6.1 Introduction

Direct visual odometry[70, 49] works by minimizing the photometric error of two images since it is assumed that the illumination changes between images close in time is small—this is what is called the *brightness constancy assumption*. This works well for visual odometry, which estimates the 3D pose of the camera frame-to-frame, since the changes in the scene's lighting is minimal within a short time interval, especially with a high frame rate camera.

Frame-to-frame tracking, however is not ideal due to natural drift over time. Instead it is

desirable to localize the camera with respect to a prior map. This is the essence of simultaneous localization and mapping (SLAM): not only to map an environment but also to localize against it. However, the brightness constancy assumption does not hold with long-term maps since illumination changes continually from day to day (morning, afternoon, evening) and is dependent on certain external conditions like weather (clouds, rain, snow) or even seasons (summer vs. winter). As such, a direct photometric minimization is often inadequate in these kind of situations.

While some work exists to create maps that handle well these changes, like for example experienced based mapping [12] or localizing against a navigation sequence [63], these techniques mitigate the problem by simply using redundant information and storing all the different conditions in maps that are co-registered.

A far better approach is for the map to store a generative model of the scene, in particular light sources and material properties. Most of this line of work, however, only consider distant lights and Lambertian surfaces [111] and are extremely computationally intensive which make them unsuitable for real time applications.

By far, the most common techniques involve folding in the illumination robustness directly into the image registration optimization. This is done by either adding additional constraints that model lighting variance, or by simply using robust photometric error functions. For a comprehensive study of the different error functions used for whole image alignment, the reader is referred to [74].

An example of such a technique is applying Zero Normalized Cross Correlation (ZNCC) [20] by which the scene's mean intensity is subtracted to each pixel which in turn is then divided by the standard deviation. This aids in adjusting the brightness of the image due to variations in lighting and exposure conditions. Another similar technique is the Global Affine Illumination (AI) [49]. In this method, two extra parameters are added to the tracking optimization: a scale α and a bias β parameter that is applied to each pixel's intensity when calculating the photometric error.

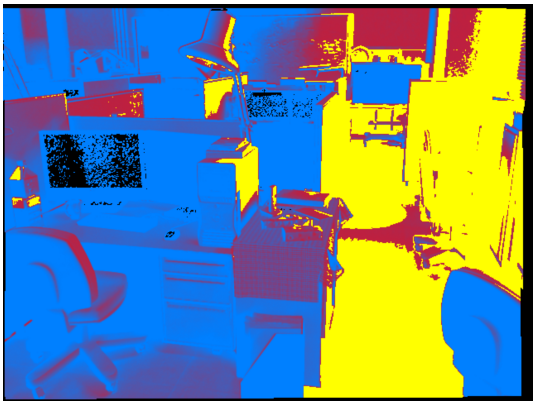
These techniques have a major drawback in that they assume that the illumination change is *globally consistent*. It is often the case that lighting conditions affect the scene differently in different areas, depending on the geometry of the scene (e.g. multi-path), the camera position

(e.g. non-Lambertian reflections) or the material properties of the objects within it (e.g. albedo). Therefore, adjusting the photometric values globally throughout the image would not capture these intricacies (Figure 6.1).

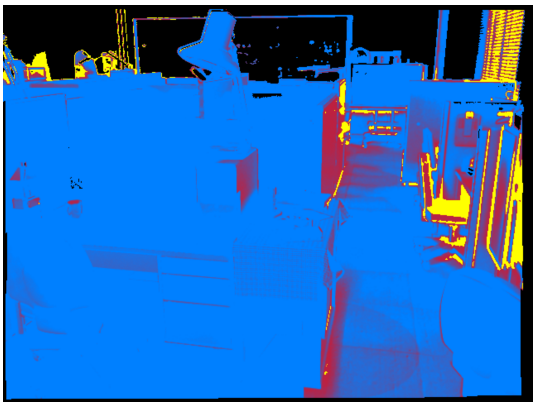
Figure 6.1: Example images of the same scene with two illumination profiles: an indoor office fluorescent lighting, and one with bright sunlight coming from the window (Tsukuba dataset).

Figure 6.2 shows the resulting error images for the same scene when lighting conditions change. In this case, the glare from sunlight coming from the office window to the right is modeled, which is not unlike real life day/night lighting changes in a typical office environment. The difference between the error images, when comparing a pure photometric minimization versus when a robust lighting technique like ZNCC or GAI is applied, is very noticeable. For the pure photometric minimization, most of the right side of the image is being rejected by the robust norm (pixels labeled yellow) with areas of high error (pixels labeled red). Zero normalized cross correlation performs slightly better, yet when the mean and standard deviation offsets are applied, the image quickly over-saturates (pixels labeled black, top area). Finally, global affine illumination handles the lighting difference better, under-saturating a small area of the image (the computer monitor) and rejecting only half of the top section due to the robust norm.

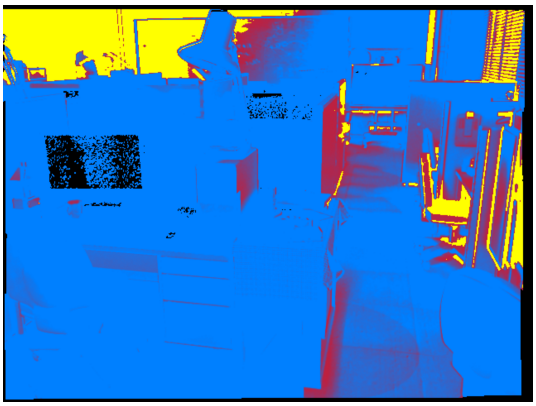
One error function that is robust against this non-uniform lighting limitation, and is one of the most popular techniques used in computer vision, is called the *Census transform* [109]. Whereas



(a)



(b)



(c)

Figure 6.2: The resulting error images of the photometric optimization: (a) pure direct photometric image alignment with no robust lighting technique, (b) with zero-normalized cross correlation and (c) with global affine illumination. The blue-red heatmap highlight areas with high (red) and low (blue) errors, with pixels rejected by the robust norm marked as yellow, and pixels discarded due to under/over saturation marked as black.

the Sum of Absolute Difference (SAD), or even the Sum of Squared Differences (SSD) directly operate over the photometric values of pixels, Census converts each pixel into a binary signature that encodes whether a pixel's photometric value is lower or not compared to its neighboring pixels. This has the advantage that it encapsulates local consistencies of illumination changes, rather than assuming a global illumination transform.

However, given the fact that each pixel is now a binary signature, a direct subtraction can no longer be applied to find correlations. This is due to the fact that the position of the bits no longer has any meaning, but rather the actual value of the bit itself. Thus, the Hamming distance is used as a metric of similarity by which the number of matching bits in the binary signature are counted in order to provide the final score.

A major disadvantage of using the Hamming distance in optimizations is that it is not continuous. Thus, what the authors of the work in [2] proposed is to use each bit independently as a different channel or *plane*. Furthermore, each bit is interpolated during the reprojection in order to find the error per bit which is then aggregated with the rest of the Census bits to calculate the final error for that pixel. The downside to using this technique, however, is that the computation per pixel has now been incremented by the number of bits in the Census transform used. Thus, for the typical 3x3 Census window, eight additional comparisons would need to be made per pixel.

A hybrid technique that combines whole image alignment with a generative model was presented in [62], in which model-based tracking parameters were optimized along with the visual odometry parameters. The system relied in *augmented* reference images, which are synthetic views generated from multiple keyframes. This requires, however, that the scene geometry between keyframes does not change which is unrealistic for long-term autonomy where a map is constantly in flux.

More recently, the Normalized Information Distance (NID) metric [53, 98, 75] was proposed. This technique makes use of mutual information [57, 103] which has been shown to be robust in the alignment of multi-modal images, in particular for medical applications. The images are converted into histograms, which are then aligned using an entropy metric. This method, although

far superior to others, is extremely computationally intensive and can take up to three orders of magnitude more than the typical photometric minimizations; even those with robust lighting techniques.

In this section, a novel algorithm is introduced which speeds-up considerably the computation time of a whole image alignment optimization using the Normalized Information Distance (NID) metric. The algorithm converts the typical NID problem into a least squares type optimization, using the conventional Gauss-Newton method to solve. This has many advantages: the rate of convergence has now become near quadratic, which allows to a much faster convergence; a covariance matrix for the estimate can be easily extracted, which is extremely useful in other SLAM algorithms like pose graph relaxations or sensor fusion; and finally, techniques to characterize the solution can be easily applied to validate the estimate without the need to run it through a complex classifier. Furthermore, unlike other implementations that make use of NID, our algorithm is capable of coping with errors in reprojection due to the spatial distribution of cells in the image: a bad reprojection in a cell can be down-weighted or even rejected without affecting the total error.

6.2 Background

6.2.1 Information Entropy

To better understand the Normalized Information Distance, the concept of entropy and mutual information must first be explained.

Entropy can be thought of as a measure of uncertainty or randomness in a system. Conversely, it can be thought of as the amount of information in a system. An event that has a very high probability of occurring conveys no information, whereas one that is very unlikely conveys a lot of information – when it occurs.

Consider the case of tossing a coin. If the coin is fair, the probability of obtaining heads or tails is equal: exactly 1/2. This coin has the maximum entropy, and as such any event – be that heads or tails – will provide the maximum amount of information. However, if the coin had two

heads and no tails, then its entropy would be zero since the outcome can be predicted perfectly: always heads. As such, the outcome of a toss with this coin provides no information.

Entropy H of a random variable X is defined as:

$$H(X) = \sum_{x \in X} P_X(x) I_X(x) \tag{6.1}$$

where $P_X(x)$ is the probability mass function and $I_X(x)$ is the information content defined as:

$$I_X(x) = log\left(\frac{1}{P_X(x)}\right). \tag{6.2}$$

Information entropy is typically measured in bits, sometimes called *shannons*, although it can also be measured in natural units, or *nats*, or even in decimal digits called *dits*. The unit of the measurement depends on the base of the logarithm that is used to define the entropy.

It is important to note that, in general, entropy is not a function of the values the events represent but rather a function of their probabilities. In the case of the coin example described above, the entropy quantity remains the same irrespective of the label assigned to each event (heads or tails) as long as the probabilities remain the same. This has the nice side-effect that entropy is independent of the amount of data used to generate the distribution.

6.2.2 Mutual Information

Entropy can also be extended to two random variables to measure the amount of information obtained about one random variable, through the other random variable. This is what is called mutual information. Figure 6.3 shows a graphical representation of two random variables and their entropy.

$H(X)$ and $H(Y)$ are the individual entropies for the correlated random variables X and Y, represented by the red and blue circles. $H(X,Y)$ is the joint entropy and is the area contained by both circles, while $H(X|Y)$ and $H(Y|X)$ are the conditional entropies. The purple is the mutual

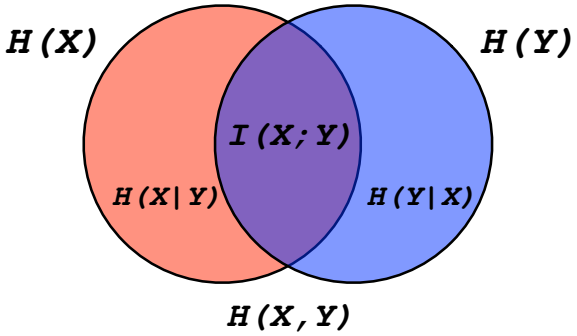


Figure 6.3: Entropy diagrams of two non independent random variables. The information common between the two random variables is called mutual information, and is denoted as $I(X;Y)$.

information $I(X;Y)$, and can be considered as a measure of similarity between the two random variables. Formally it is defined as:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{6.3}$$

where,

$$H(X,Y) = \sum_{x \in X,\, y \in Y} P_{XY}(x,y) \quad log\left(\frac{1}{P_{XY}(x,y)}\right) \tag{6.4}$$

As can be seen from the previous equation, mutual information is maximized as the two circles overlap completely with each other. In contrast, variables that are independent have circles that do not overlap and as such convey no mutual information.

6.2.3 Normalized Information Distance

One of the biggest issues with mutual information is that it is not a true metric as it does not satisfy the triangle inequality.

Consider the case of Figure 6.4. In this example, both cases have the same amount of mutual information $I(X_1;Y_1) = I(X_2;Y_2)$, yet with different joint entropies $H(X_2,Y_2) < H(X_1,Y_1)$. It is desirable to disambiguate between the two cases, and have a distance that captures $\mathcal{D}(X_1,Y_1) < \mathcal{D}(X_2,Y_2)$.

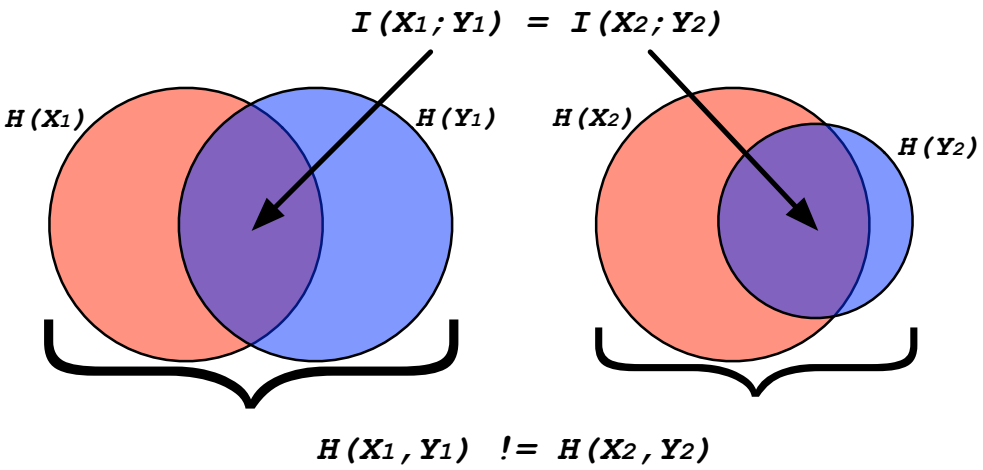


Figure 6.4: Two cases which have the same mutual information $I(X_1;Y_1) = I(X_2;Y_2)$ (purple areas), yet with different joint entropies $H(X_1,Y_1) \neq H(X_2,Y_2)$ (captured by the brackets).

The Normalized Information Distance (NID) precisely tries to overcome this limitation by normalizing the variation of information by the joint entropy; and unlike mutual information, it is a true metric $\in [0,1]$. Formally, NID is defined as:

$$NID(X,Y) = \frac{H(X,Y) - I(X;Y)}{H(X,Y)} \tag{6.5}$$

and obtaining the final NID cost by using Equation 6.3 on Equation 6.5 above:

$$NID(X,Y) = \frac{2H(X,Y) - H(X) - H(Y)}{H(X,Y)} \tag{6.6}$$

6.3 Methodology

Pascoe et al. [76, 75] were one of the first to propose a dense tracking system that directly minimizes a single global NID cost. The optimization uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which is a quasi-newton method by which the Hessian is iteratively approximated from the gradient and provides robustness by using a line search. Although the algorithm shows promising results for scenes with extreme illumination changes, the convergence rate of BFGS is extremely slow. Furthermore, since the images are converted into a single histogram,

inconsistencies due to outliers cannot be handled like typical direct photometric methods could.

Thus, instead of minimizing a single cost, this work proposes to use NID as an image distance measure instead that provides multiple residuals, and therefore the problem can be rewritten to use the conventional Lucas-Kanade[6] whole image alignment formulation (see Section 2.2.1). As such, a second-order optimization method like Gauss-Newton can be used to yield near quadratic convergence rates. Furthermore, the optimization is now capable of providing its true covariance matrix, which is important for other SLAM algorithms like pose graph relaxation and sensor fusion.

To achieve this, the image is now split into cells, for each of which a NID cost is computed. The optimization is looking to minimize the *summation* of the NID costs for all cells in the image (Figure 6.5).

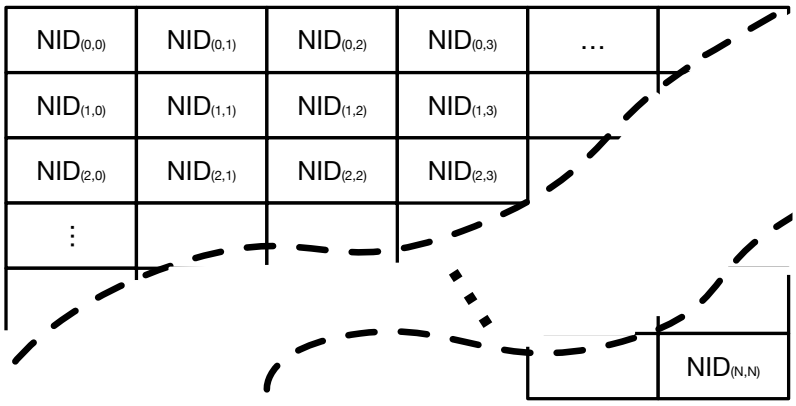


Figure 6.5: The image is split into cells, for each of which the NID cost is computed. This grid approach also has the advantage that due to the cell spatial distribution, inconsistencies can be down-weighted or rejected since the histograms generated are restricted to only a particular area of the image.

A typical Lucas-Kanade whole image alignment formulation is written as follows:

$$\arg\min_{\mathbf{p}} \mathcal{D}\left(\omega\left(I_{cur}, \mathbf{p}\right), I_{ref}\right), \tag{6.7}$$

where $\mathbf{p} \in \mathbb{SE}(3)$ is the six degree-of-freedom pose being estimated, $I_{cur} : \Omega \to \mathbb{R}^+$ and $I_{ref} : \Omega \to$

$\mathbb{R}^+$ are the current and reference images respectively, and ω is the warping function that transforms the current image using the estimated parameters $\mathbf{p}$.

Finally, $\mathcal{D}$ in Equation (6.7) is a measure of distance and as such must be designed to meet certain conditions:

- Non-negativity: $\mathcal{D}(X, Y) \geq 0$
- Equivalence: $\mathcal{D}(X, Y) = 0 \iff X = Y$
- Symmetry: $\mathcal{D}(X, Y) = \mathcal{D}(Y, X)$
- Triangle Inequality: $\mathcal{D}(X, Y) + \mathcal{D}(Y, Z) \geq \mathcal{D}(X, Z)$

For the conventional photometric image alignment algorithm, the distance metric is the summation of the squared errors for each pixel in the image. This work posits that the NID operation defined in Equation (6.6) provides a robust result to brightness inconstancy:

$$\mathcal{D}_{NID}(X, Y) = \frac{2H(X, Y) - H(X) - H(Y)}{H(X, Y)} \tag{6.8}$$

where, $H(X)$ and $H(Y)$ are the individual entropies of the images X and Y whose distributions are calculated by a sampling method using bins. $H(X, Y)$ is the joint entropy.

The NID cost only requires computing the joint distribution, since the individual distributions can be calculated by marginalization. A sampling method is performed to construct each distribution, represented as a K-bin histogram, with the final joint distribution being of size KxK.

Finally, the spatial distribution of cells in the image provides additional benefits with regards to histogram consistency. For the single residual NID case, it is possible to construct a problem by which the cost is minimal over the entire image regardless of how the image is warped due to the fact that all reprojections fall under a single histogram. For example, imagine the simple case of blocking the camera such that a third of the image is black. Regardless of how the image is warped, the joint entropy will be the minimum at some incorrect transform since the bad observations were already added to the histogram. However, with the cell based approach the position of the black

area of the image will only affect the cells it reprojects into, and as such, the histograms of the other cells will not be affected. Thus, a technique to down-weight or reject high errors could mitigate the influence of these erroneous cells yet still achieve the same original histogram (Figure 6.6).

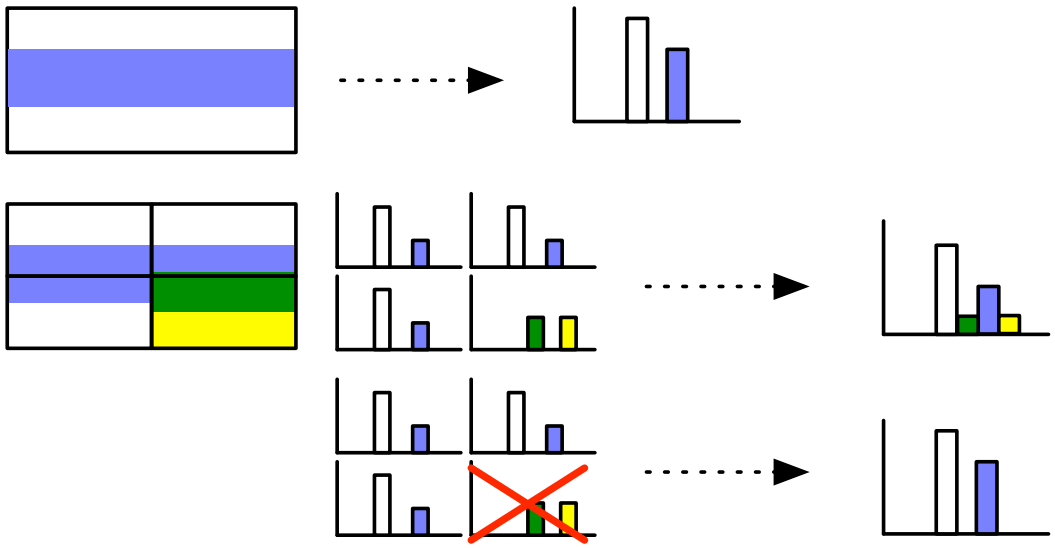


Figure 6.6: The grid cell approach preserves the histogram consistency regardless of reprojection errors due to incorrect depth estimates or outliers, like for example new objects in scene. For the case of the whole image NID, the final histograms between the top and middle row will not match correctly. With the grid cells, however, the cells that are affected can easily be down-weighted or rejected and as such preserve the original histogram (bottom row).

6.4 Evaluation Method

To evaluate the algorithm, experiments were conducted on the Tsukuba dataset [59] (Figure 6.1). This dataset is comprised of 1800 image pairs, ground truth depth maps, disparity and occlusion maps, as well as their corresponding camera poses. The images are colored images, having a 640x480 resolution. The dataset simulates a 30 FPS camera trajectory moving for one minute inside an indoor office environment. It provides four different lighting conditions: fluorescent, which models a typical office environment with homogeneous lights and no external light sources; daylight, similar to the previous but in addition it provides an external source of light from the windows simulating direct sunlight; lamp, which simulates the office with all ceiling lights turned off with

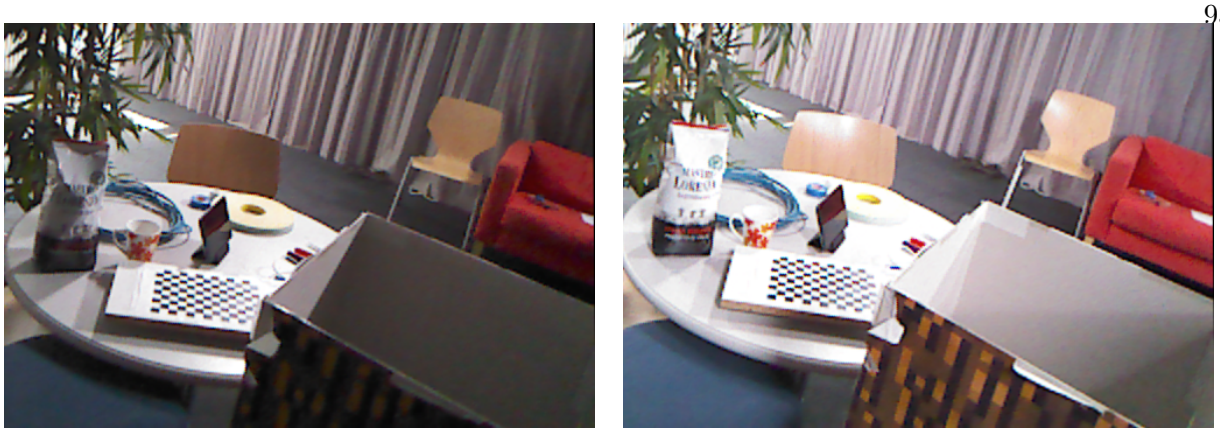
the exception of desk lamps; and finally flashlight, which simulates a completely dark environment while having a flashlight on top of the rig as it moves through the scene.

Since the last two lighting conditions (flashlight and lamps) are not common environments seen in a typical robotic application, the experiments were only conducted between daylight and fluorescent lighting. A *reference* trajectory is selected with one lighting condition, and a *live* trajectory with the opposite lighting condition. Frame to frame estimates are performed such that frame N from the reference trajectory is used to estimate the pose of frame N+1 of the live trajectory frame. When choosing which lighting environment to use as the reference trajectory, either daylight or fluorescent lighting, it was noted that the results were almost equivalent and as such the average of the two was used instead when showing these results.

In order to asses the validity of the algorithm in real world settings, experiments were conducted on images from the ETH Zurich - Computer Vision and Geometry Group's dataset [74] (Figure 6.7). This dataset contains 640x480 resolution images of the same scene in multiple lighting conditions. It is comprised of RGB-D data obtained from a color camera and depth information from a Kinect sensor, with ground truth poses obtained from a motion capture system. Unlike the Tsukuba dataset where identical camera trajectories were captured with different lighting conditions, the dataset has a single camera trajectory and instead the lights are turned on and off at different points. 35 such transitions were detected, and the experiments were conducted on both transitions, $on \rightarrow off$ and $off \rightarrow on$, thus yielding 70 test cases.

Pose errors were estimated independently for the translation and rotation components: the Euclidean distance (ℓ^2-norm), was used to estimate the translation error while the Infinity norm (ℓ^∞-norm) was used to estimate the rotation error. In average, the camera moves approximately 2.8 centimeters in translation and 0.015 radians in rotation between each frame; some areas exclusively with high amount of camera rotation while others with only translation.

Finally, since in some experiments a comparison is performed between pure photometric methods and those using normalized information distance, a *pass* criteria shared by all of them



(a) (b)

Figure 6.7: Example images from the ETH Zurich - CVG dataset. Light transitions (on/off) occur throughout the camera trajectory, and for each such pair the reference and live images are alternated, thus yielding 2 test cases for every image pair.

must be defined. Thus, a frame is considered to have been successfully estimated if the result of the optimization marks it as such and if the final relative pose error is less than 0.5.

For the algorithm presented here, the experiments were performed by adjusting the number of cells the image was split into. As such, the original NID algorithm which makes use of the BFGS optimization is labeled as NID-BFGS, while the technique used here is labeled as NID-GN based on the Gauss-Newton optimization. Cell sizes were estimated based on the number of splits applied to each dimension in a recursive manner. Thus, 1 split means the image is divided by two, 2 splits means the image is divided twice – one to obtain halves, and then another to obtain quarters, and so on.

Thus, NID-GN3 is the label for the configuration of the algorithm which contains three splits. Table 6.1 shows the number of splits for each configuration, and the corresponding number of cells and cell size.

It should be noted that NID-GN1 is not tested since this would yield an under-determined system, having only 4 residuals. As the number of splits is increased, the cell resolution is lowered up to the point that no clean split can be performed after 5. Thus, all experiments were performed between 2 to 5 splits, yielding 16 to 1024 residuals.

Table 6.1: NID-GN Configurations

Configuration	Splits	Num Cells per Dimension	Total Num Cells	Resolution of Cell
NID-GN1	1	2	4	320x240
NID-GN2	2	4	16	160x120
NID-GN3	3	8	64	80x60
NID-GN4	4	16	256	40x30
NID-GN5	5	32	1024	20x15

Finally, for all NID versions of the algorithm (both BFGS and GN) the number of bins used to create the histograms that represent the distributions was $K = 16$. As such, the joint distribution is a matrix of size 16x16.

6.5 Results

For the Tsukuba dataset, when choosing which lighting environment to use as the reference trajectory, either daylight or fluorescent lighting, it was seen that the results were similar and as such the average of the two runs was used yielding a total of 3600 test cases.

Table 6.2 shows the percent of successful estimates of the different algorithms used with different number of cell splits (see Table 6.1). Overall the amount of successful estimates for all algorithms is still low, barely half, which shows that the underlying brightness constancy assumption for dense whole image alignment is not valid with extreme illumination changes. This is partly due to how challenging the dataset is, with some sections of the trajectory showing extremely over-saturated pixels which lead to loss of information (Figure 6.8).

Table 6.2: Successful Estimates for Tsukuba Dataset

NID-BFGS	NID-GN2	NID-GN3	NID-GN4	NID-GN5
45.00%	19.44%	36.11%	45.56%	49.44%

As a comparison, pure photometric only yields 28.33% success rate for the whole trajectory so it is clear that the Normalized Information Distance metric is more robust to these extreme

Figure 6.8: Sample images from the Tsukuba daylight sequence that experience extreme over saturation, and thus, data loss.

illumination changes – yielding more than 20% higher success rate.

From Table 6.2 it is also interesting to see how the success percentage of NID-GN increases as more splits are done. This is not unexpected, as the least squares optimization has a higher number of measurements to work with. With NID-GN2, having only 16 measurements, the percent of success is lower than even photometric or bitplanes. NID-GN3 surpasses the regular photometric variants, although it still is lower than the NID baseline using BFGS. However, NID-GN4 quickly levels the playing field and NID-GN5 even surpasses the baseline.

Focusing on the NID variants, Figure 6.9 shows the average/max/min times for the whole trajectory. The baseline NID-BFGS takes in average 11.70 seconds per frame, whereas the NID-GN variants take in average 4.75 seconds, with the slowest (NID-GN4) taking 5.24 seconds. Overall this technique achieves a speedup of almost 2.5x compared to the baseline NID implementation.

Finally, Figure 6.10 show the average/max/min translation and rotation errors of the full trajectory. As expected, NID-GN5 achieves the least amount of error for both translation (0.524 centimeters) and rotation (0.0025 radians), compared to the other cell based variants since it uses a much larger number of observations (1024). NID-BFGS, however, yields the smallest error: 0.456 centimeters in translation and 0.0016 radians in rotation. The belief is that this is due to the way BFGS works, in that it dynamically adjusts the step size at each iteration and as such is able to achieve a higher resolution at the convergence basin.

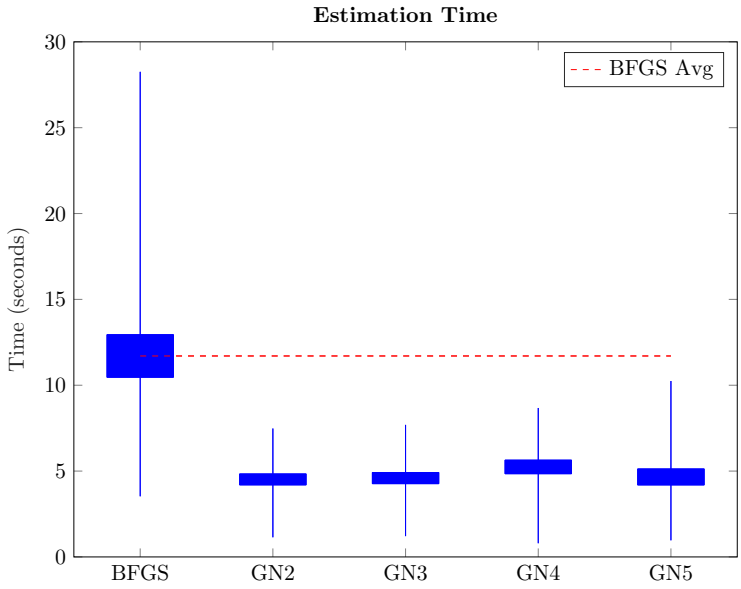


Figure 6.9: Average/max/min estimation time (in seconds) between the different NID algorithms for the Tsukuba dataset. The red dotted line represents the BFGS average.

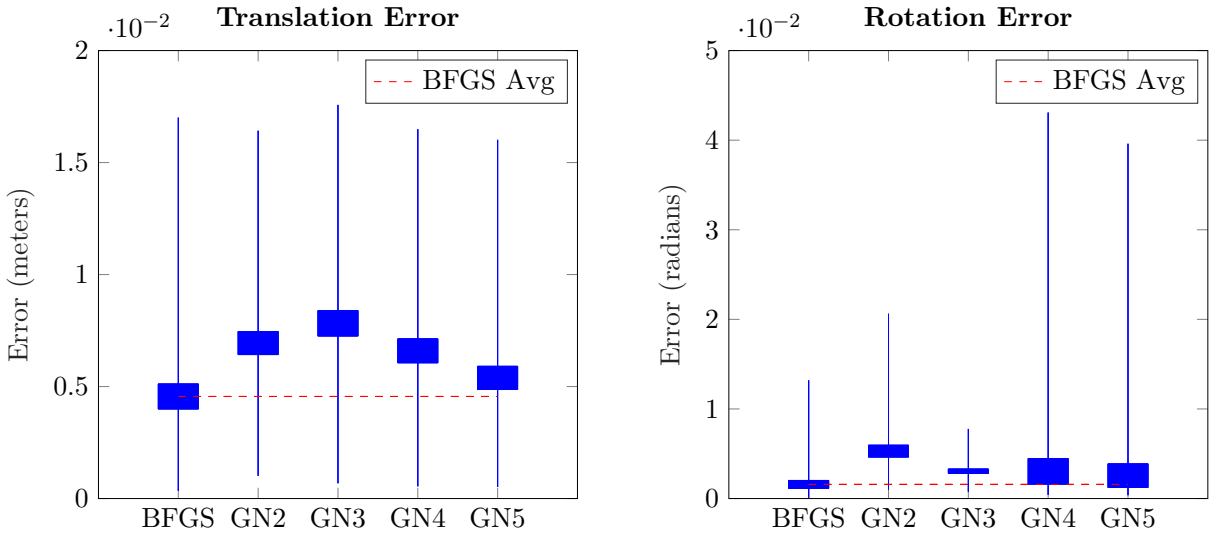


Figure 6.10: Average/max/min translation error (in meters) and rotation error (in radians) between the different NID algorithms for the Tsukuba dataset.

Interestingly, the expectation would be for the errors to decrease as the number of cells increases yet this is not the case for all configurations. The reasoning for this behavior is that configurations with more cells end up estimating more images correctly; many of which could not even be estimated by previous configurations. It is fair to assume that these frames failed in other

methods because they were very challenging and as such, even when passing the acceptance criteria described above, the estimate would still incur a high amount of error, skewing the average.

This is particularly visible with the rotation errors, where the maximum errors for NID-GN4 and NID-GN5 are clearly outliers compared to the other configurations. As such, it is not easy to perform a fair comparison of errors between all of these configurations.

With regards to the ETH Zurich - CVG real world dataset, the experiments show similar results to those seen from the Tsukuba dataset.

Table 6.3: Successful Estimates for ETHZ-CVG Dataset

NID-BFGS	NID-GN2	NID-GN3	NID-GN4	NID-GN5
57.14%	61.43%	62.86%	75.71%	57.14%

Table 6.3 shows the percent of successful estimates for the different algorithms. It is interesting to see that, unlike the Tsukuba dataset, the number of successful estimates does not always increase as the number of cells is increased. Upon further investigation, the belief is that – similar to optical flow – the algorithm is very sensitive to initialization especially as the cell size is decreased. This was corroborated when the optimization was initialized to some percentage of the solution, rather than at identity, in which case NID-GN5 reaches the same level as NID-GN4 with regards to number of successful estimates.

The average time for NID-BFGS was similar as applied to Tsukuba, given that they both have the same image resolution and camera movements: 10.48 seconds. The NID-GN variants take in average 5.66 seconds, with the slowest taking 6.21 seconds (Figure 6.11). Overall this technique achieves a speedup of almost 1.8x compared to the BFGS implementation.

Finally, Figure 6.12 show the average/max/min translation and rotation for the real world dataset. As with the Tsukuba dataset, BFGS achieves the highest precision: 0.591 centimeters of error in translation and 0.0089 radians in rotation. In contrast, the best NID-GN implementation achieved 0.70 centimeters of error in translation, and 0.0092 radians in rotation. This is an error

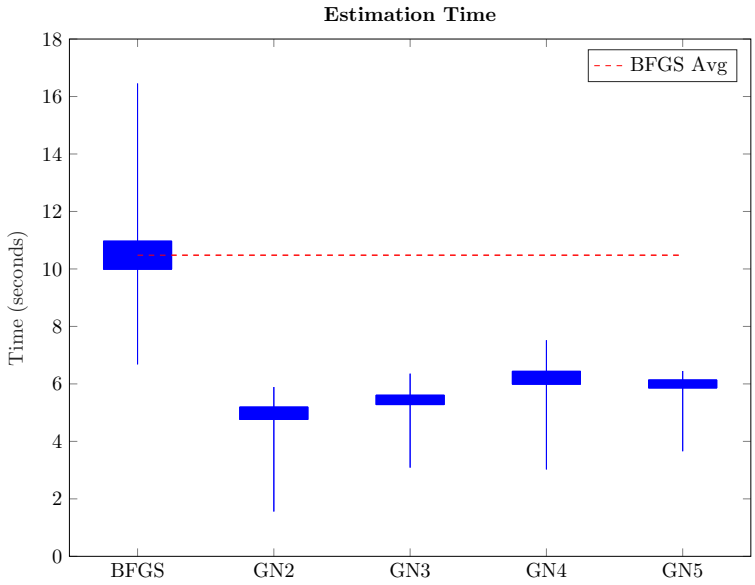


Figure 6.11: Average/max/min estimation time (in seconds) between the different NID algorithms for the real world dataset. The red dotted line represents the BFGS average.

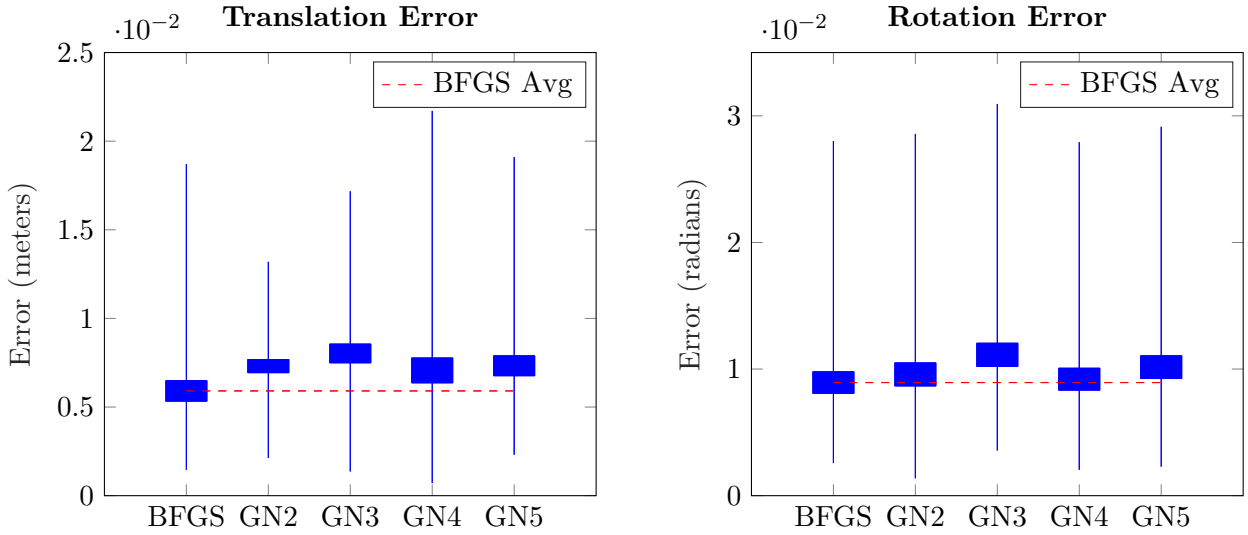


Figure 6.12: Average/max/min translation error (in meters) and rotation error (in radians) between the different NID algorithms for the real world dataset.

difference of only 0.11 centimeters and 0.0003 radians.

6.6 Conclusion

In this chapter, an algorithm that makes use of the Normalized Information Distance (NID) metric for whole image alignment was introduced. The algorithm splits the image into cells, each counting as an observation for a least squares style Gauss-Newton optimization. The NID metric was shown to be robust to extreme illumination changes, achieving almost 20% higher estimation successes when compared to conventional photometric algorithms.

The algorithm also achieved an average 2x speedup compared to a baseline NID implementation that made use of the BFGS optimization. Multiple configurations of the cell-based algorithm were tested, each achieving comparable errors to the baseline NID-BFGS implementation.

Although the BFGS variant still had the highest accuracy, the difference is negligible: a translation error in average of 0.085 centimeters and a rotation error of 0.0006 radians. Thus, the compromise seems to be a loss of 3-6% of precision for a 1.8-2.5x increase in performance and a higher estimation success percentage for the whole trajectory.

The spatial distribution of cells over the whole image also provides additional benefits with regards to histogram consistency, since for the case of a single NID cost any erroneous observations – either due to depth errors, occlusion boundaries or new objects in the scene – will affect the joint histogram. By having cells distributed through the whole image, these inconsistencies can be down-weighted or even rejected guaranteeing that the cost will only be minimal if *all* the non-rejected cell costs are minimal.

An interesting observation from the results obtained is that the amount of successful estimates for all algorithms is low. This shows that – even though NID performs well under extreme light conditions – the underlying brightness constancy assumption for dense whole image alignment has its limits. It is very likely that as generative models mature and are capable of jointly estimating light sources, material properties and scene geometry in real time, that approaches that incorporate model parameters into the tracking optimization will yield far better results and would be capable of handling all type of light conditions.

It is left as future work to test if the performance trend – precision and success percentage – continues to increases as more cells are used. The biggest limitation is image resolution, in which case a dataset with much higher resolution would have to be generated. To mitigate the initialization problems seen with the ETHZ-CVG dataset, a coarse-to-fine approach could be implemented by which initial estimates are done with a small number of cells and as the optimization converges the number of cells is increased.

Also, the algorithm presented here did not make use of a robust norm since the scenes in datasets tested were static. However, it should be trivial to add a robust norm to the optimization such that cells with high cost would be down-weighted or even rejected. This situation can occur in highly dynamic scenes, where objects have moved between frames and as such the appearance of each cell is different. In such cases, the inclusion of a robust norm would mitigate estimation inaccuracies.

Finally, the algorithm performance could also be improved by the way it is implemented: the joint distribution and cost computations are done sequentially. The cell based approach easily separates the construction of the problem into logical units that can be easily parallelized: each cell can be launched on its own CUDA thread, for example.

Chapter 7

Conclusion

The focus of this thesis has been to show a journey towards robust dense visual Simultaneous Localization and Mapping (SLAM). The goal is for any researcher to read this work and be able to obtain the necessary tools and concepts to implement their own robust dense visual SLAM system.

It first starts with 3D reconstruction, which plays a vital part not only in the accuracy of the tracking system but it is also necessary in robotic applications in order to interact with the environment, avoid obstacles and plan. A general survey of different parametric 3D reconstruction algorithms was provided, and in particular an incremental and adaptive front-end fusion system was presented. This method was capable capable of providing accurate depth maps of the world and expanded on previous volumetric variational approaches for 3D reconstruction by providing two main key features. The first was a novel incremental method for updating the cost volume which removed the need of keeping hundreds of multi-view comparison images, thus reducing the overall processing time and memory storage of the system. The second feature was a method for dynamically adapting the minimum and maximum depth limits of the cost volume as it adjusts to changes in scene depth, thus achieving optimum resolution in the 3D reconstruction.

Finally, a technique that can provide depth uncertainties was briefly described. Having metric uncertainties for the depth map is invaluable, as it allows the propagation of error through the SLAM system to obtain the correct pose covariance. This also makes it easier when fusing data from multiple sensors, by keeping the uncertainties in meaningful units.

After, the second component of SLAM was studied: tracking, in particular a survey of dif-

ferent direct methods for localization was presented. Direct methods are the foundation of dense systems, since they operate directly over image data rather than indirectly through the use of features and descriptors like their sparse counterpart. Many hybrid systems have been proposed lately; for example, fully dense, semi-dense and semi-direct. The study compared the tracking performance of these methods for computing frame-to-frame motion estimates. It also introduced the most commonly used multi-sensor rig: a camera capable of providing images and depth maps, and an inertial measurement unit (IMU). The different direct tracking methods – with and without the aid of an IMU – were compared to see how they performed with respect to changes in image resolution, shutter speed, frame-rates, as well as image and depth noise. The results of this study gave insight into how the addition of multiple sensors can lead towards a more robust dense visual SLAM.

As for mapping, a robust dense visual SLAM system should be able to operate in large scale environments. A method was presented, which made use of a volumetric map representation and visual-inertial tracking that showed promising results – both in terms of 3D reconstruction quality and tracking accuracy – for both indoor and outdoor environments.

Overall different map representations were presented, and the distinctions between absolute and relative maps were discussed. The choice of map representation is important when building a SLAM system, as each representations has its strengths and weaknesses. Some are more malleable, able to more easily adjust to corrections when performing global techniques like loop closures and pose graph relaxations. Others provide richer geometric reconstructions of the world.

Regardless of the representation used, it is important for the system to be able to localize against this map. Since dense visual SLAM systems rely on **photometric** data, it is therefore imperative that the map is able to store this information while being robust against illumination changes due to time of day (morning, afternoon, evening) or seasons (summer, winter). It should also be capable of being adaptive and being robust against dynamic objects in the environment as things move (people, pallets, shelving, etc).

To this end, several metrics were presented that showed robustness to lighting changes. In

particular, the Normalized Information Distance (NID) showed impressive results in extreme lighting conditions with over-saturated components. A least square style optimization using an image cell approach was presented, which achieved high localization recall in comparison to the conventional direct photometric methods. It also achieved very high accuracy both in translation and rotation error. Although the optimization is not fast enough yet to be used in a front-end system, it can be used as a much slower asynchronous back-end that performs map co-registrations and loop closures.

To conclude, this work provides all of the necessary components to create a robust dense visual simultaneous localization and mapping system – yet the road is still a long one. Although new techniques are available to provide some resilience to lighting changes – a common situation seen in long-term autonomy – the biggest advancement in photometric optimizations is just on the horizon: light estimation and material properties of the scene.

With this new information, a map that truly captures light sources, intensities, the geometric information of the scene as well as the material properties of objects, not only would provide high fidelity in tracking and reconstruction, but also move robotics a step closer to true scene understanding.

Bibliography

[1] Sameer Agarwal, Noah Snavely, Steven M. Seitz, and Richard Szeliski. Bundle adjustment in the large. In Proceedings of the 11th European Conference on Computer Vision: Part II, ECCV'10, pages 29–42, Berlin, Heidelberg, 2010. Springer-Verlag.

[2] Hatem Alismail, Brett Browning, and Simon Lucey. Bit-planes: Dense subpixel alignment of binary descriptors. arXiv preprint arXiv:1602.00307, 2016.

[3] Hatem Alismail, Brett Browning, and Simon Lucey. Direct visual odometry using bit-planes. arXiv preprint arXiv:1604.00990, 2016.

[4] Jean-François Aujol. Some first-order algorithms for total variation based image restoration. Journal of Mathematical Imaging and Vision, 34(3):307–327, 2009.

[5] Simon Baker and I Matthews. Lucas-kande 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Carnegie Mellon University Robotics Institute, 2002.

[6] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. International journal of computer vision, 56(3):221–255, 2004.

[7] Selim Benhimane and Ezio Malis. Real-time image-based tracking of planes using efficient second-order minimization. In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 1, pages 943–948. IEEE, 2004.

[8] A Bermúdez and C Moreno. Duality methods for solving variational inequalities. Computers & Mathematics with Applications, 7(1):43–58, 1981.

[9] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. International Journal of Computer Vision, 26(1):63–84, 1998.

[10] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision, 40(1):120–145, 2011.

[11] Gary E Christensen and Hans J Johnson. Consistent image registration. IEEE transactions on medical imaging, 20(7):568–582, 2001.

[12] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. The International Journal of Robotics Research, 32(14):1645–1661, 2013.

[13] Andrew I Comport, Ezio Malis, and Patrick Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In Robotics and Automation, 2007 IEEE International Conference on, pages 40–45. IEEE, 2007.

[14] Alejo Concha and Javier Civera. Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 5686–5693. IEEE, 2015.

[15] Alejo Concha, Wajahat Hussain, Luis Montano, and Javier Civera. Manhattan and piecewise-planar constraints for dense monocular mapping.

[16] Alejo Concha, Giuseppe Loianno, Vijay Kumar, and Javier Civera. Visual-inertial direct slam. In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 1331–1338. IEEE, 2016.

[17] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. IEEE Transactions on pattern analysis and machine intelligence, 23(6):681–685, 2001.

[18] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research, 27(6):647–665, 2008.

[19] Amaël Delaunoy and Marc Pollefeys. Photometric bundle adjustment for dense multi-view 3d modeling. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014), pages 1486–1493. IEEE, 2014.

[20] Luigi Di Stefano, Stefano Mattoccia, and Federico Tombari. Zncc-based template matching using bounded partial correlation. Pattern recognition letters, 26(14):2129–2134, 2005.

[21] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In European Conference on Computer Vision (ECCV 2014), September 2014.

[22] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In IEEE International Conference on Computer Vision (ICCV 2013), Sydney, Australia, December 2013.

[23] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[24] C. Engels, H. Stewenius, and D. Nister. Bundle adjustment rules. In Photogrammetric Computer Vision, 2006.

[25] Ryan Eustice, Hanumant Singh, John Leonard, Matthew Walter, and Robert Ballard. Visually navigating the rms titanic with slam information filters. In Proceedings of Robotics: Science and Systems, Cambridge, USA, June 2005.

[26] J.M. Falquez, M. Kasper, and G. Sibley. Inertial aided dense & semi-dense methods for robust direct visual odometry. In International Conference on Intelligent Robots and Systems (IROS 2016), 2016.

[27] J.M. Falquez, V. Spinella-Mamo, and G. Sibley. Incremental and adaptive front-end fusion. In IEEE International Conference on Robotics and Biomimetics (ROBIO), 2014.

[28] Ross Finman, Thomas Whelan, Michael Kaess, and John J Leonard. Efficient incremental map segmentation in dense rgb-d maps. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 5488–5494. IEEE, 2014.

[29] Nicola Fioraio and Kurt Konolige. Realtime visual and point cloud slam. In Proc. of the RGB-D workshop on advanced reasoning with depth cameras at robotics: Science and Systems Conf.(RSS), volume 27, 2011.

[30] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM, 24(6):381–395, June 1981.

[31] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In Proc. IEEE Intl. Conf. on Robotics and Automation, 2014.

[32] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. IEEE Trans. Pattern Anal. Mach. Intell., 2010.

[33] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 28(5):1188–1197, 2012.

[34] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR), 2013.

[35] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In Asian Conference on Computer Vision (ACCV), 2010.

[36] Tom Goldstein, Ernie Esser, and Richard Baraniuk. Adaptive primal-dual hybrid gradient methods for saddle-point problems. arXiv preprint arXiv:1305.0546, 2013.

[37] M.D. Grossberg and S.K. Nayar. What is the Space of Camera Response Functions? In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume II, pages 602–609, Jun 2003.

[38] Gregory D Hager and Peter N Belhumeur. Efficient region tracking with parametric models of geometry and illumination. IEEE transactions on pattern analysis and machine intelligence, 20(10):1025–1039, 1998.

[39] Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison. Real-time camera tracking: When is high frame-rate best? In Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII, pages 222–235, 2012.

[40] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 2, pages 807–814. IEEE, 2005.

[41] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 30(2):328–341, 2008.

[42] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, 2013.

[43] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In Proceedings of the 24th annual ACM symposium on User interface software and technology, pages 559–568. ACM, 2011.

[44] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S Torr, and D. W. Murray. Very high frame rate volumetric integration of depth images on mobile device. IEEE Transactions on Visualization and Computer Graphics (Proceedings International Symposium on Mixed and Augmented Reality 2015), 22(11), 2015.

[45] N. Keivan and G. Sibley. Realtime simulation-in-the-loop control for agile ground vehicles. In Towards Autonomous Robotic Systems (TAROS 2014), pages 276–287. Springer, 2014.

[46] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In 3D Vision-3DV 2013, 2013 International Conference on, pages 1–8. IEEE, 2013.

[47] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 2100–2106. IEEE, 2013.

[48] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '07, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.

[49] Sebastian Klose, Philipp Heise, and Alois Knoll. Efficient Compositional Approaches for Real-Time Robust Direct Visual Odometry from RGB-D Data. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), November 2013.

[50] Laurent Kneip, Margarita Chli, Roland Siegwart, Roland Yves Siegwart, and Roland Yves Siegwart. Robust real-time visual odometry with a single camera and an imu. In BMVC, pages 1–11, 2011.

[51] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pages 3607–3613. IEEE, 2011.

[52] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of slam algorithms. Auton. Robots, 27(4):387–407, 2009.

[53] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. The similarity metric. IEEE transactions on Information Theory, 50(12):3250–3264, 2004.

[54] Steven Lovegrove, Alonso Patron-Perez, and Gabe Sibley. Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. In BMVC, 2013.

[55] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[56] L. Ma, J.M. Falquez, S. McGuire, and G. Sibley. Large scale dense visual inertial slam. In Field and Service Robotics Conference (FSR 2015), 2015.

[57] Frederik Maes, Dirk Vandermeulen, and Paul Suetens. Medical image registration using mutual information. Proceedings of the IEEE, 91(10):1699–1722, 2003.

[58] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, volume 2, pages 1843–1848. IEEE, 2004.

[59] Sarah Martull, Martin Peris, and Kazuhiro Fukui. Realistic cg stereo image dataset with ground truth disparity maps. ICPR workshop TrakMark2012, 111(430):117–118, 2012.

[60] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. Rslam: A system for large-scale mapping in constant-time using stereo. International journal of computer vision, 94(2):198–214, 2011.

[61] Christopher Mei, Gabe Sibley, Mark Cummins, Paul M Newman, and Ian D Reid. A constant-time efficient stereo slam system. In BMVC, pages 1–11, 2009.

[62] Maxime Meilland, A Comport, Patrick Rives, and INRIA Sophia Antipolis Méditerranée. Real-time dense visual tracking under large lighting variations. In British Machine Vision Conference, University of Dundee, volume 29, 2011.

[63] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 1643–1649. IEEE, 2012.

[64] Hans P. Moravec. The Stanford cart and the CMU rover. Springer, 1990.

[65] Jack Morrison, Dorian Gálvez-López, and Gabe Sibley. Scalable multi-device slam. 2014.

[66] John G Morrison, Dorian Gavez-Lopez, and Gabe Sibley. Scalable multirobot localization and mapping with relative maps: Introducing moarslam. IEEE Control Systems, 36(2):75–85, 2016.

[67] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. IEEE Transactions on Robotics, 31(5):1147–1163, 2015.

[68] Richard A Newcombe and Andrew J Davison. Live dense reconstruction with a single moving camera. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1498–1505. IEEE, 2010.

[69] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11, pages 127–136, Washington, DC, USA, 2011. IEEE Computer Society.

[70] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, Washington, DC, USA, 2011. IEEE Computer Society.

[71] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (TOG), 32(6):169, 2013.

[72] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6d slam3d mapping outdoor environments. Journal of Field Robotics, 24(8-9):699–722, 2007.

[73] POV-Ray: The Persistence of Vision Raytracer. http://www.povray.org/, 2013.

[74] Seonwook Park, Thomas Schöps, and Marc Pollefeys. Illumination change robustness in direct visual slam. In ICRA, 2017.

[75] Geoffrey Pascoe, Will Maddern, Michael Tanner, Pedro Piniés, and Paul Newman. Nid-slam: Robust monocular slam using normalised information distance.

[76] Geoffrey Pascoe, William P Maddern, and Paul Newman. Robust direct visual localisation using normalised information distance. In BMVC, pages 70–1, 2015.

[77] Jaime Vives Piqueres. http://www.ignorancia.org/en/, 2013.

[78] M. Pizzoli, C. Forster, and D. Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In International Conference on Robotics and Automation (ICRA 2014), pages 2609–2616. IEEE, 2014.

[79] Victor Adrian Prisacariu, Olaf Kähler, Ming Ming Cheng, Julien Valentin, Philip HS Torr, Ian D Reid, and David W Murray. A framework for the volumetric integration of depth images. arXiv preprint arXiv:1410.0925, 2014.

[80] Henry Roth and Marsette Vona. Moving volume kinectfusion. In BMVC, pages 1–11, 2012.

[81] Cyril Roussillon, Aurélien Gonzalez, Joan Solà, Jean-Marie Codol, Nicolas Mansard, Simon Lacroix, and Michel Devy. Rt-slam: a generic and real-time visual slam implementation. In International Conference on Computer Vision Systems, pages 31–40. Springer, 2011.

[82] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60(1):259–268, 1992.

[83] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1352–1359, 2013.

[84] R.F. Salas-Moreno, B. Glocken, P.H.J. Kelly, and A.J. Davison. Dense planar slam. In IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2014), 2014.

[85] Keir Mierle Sameer Agarwal. Ceres Solver: Tutorial & Reference. Google, November 2012.

[86] Davide Scaramuzza and F Fraundorfer. Tutorial: visual odometry. IEEE Robotics and Automation Magazine, 18(4):80–92, 2011.

[87] Thomas Schops, Jakob Engel, and Daniel Cremers. Semi-dense visual odometry for ar on a smartphone. In Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on, pages 145–150. IEEE, 2014.

[88] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 519–528. IEEE, 2006.

[89] Sunando Sengupta, Eric Greveson, Ali Shahrokni, and Philip HS Torr. Urban 3d semantic modelling using stereo vision. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 580–585. IEEE, 2013.

[90] H-Y Shum and Richard Szeliski. Construction of panoramic image mosaics with global and local alignment. In Panoramic vision, pages 227–268. Springer, 2001.

[91] G. Sibley, C. Mei, I. Ried, and P. Newman. Adaptive relative bundle adjustment. In Robotics: Science and Systems, 2009.

[92] Gabe Sibley. Relative bundle adjustment. Department of Engineering Science, Oxford University, Tech. Rep, 2307(09), 2009.

[93] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. Journal of Field Robotics, 27(5):587–608, 2010.

[94] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Planes, trains and automobilesautonomy for the modern robot. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 285–292. IEEE, 2010.

[95] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. The International Journal of Robotics Research, 29(8):958–980, 2010.

[96] Frank Steinbrucker, Thomas Pock, and Daniel Cremers. Large displacement optical flow computation withoutwarping. In Computer Vision, 2009 IEEE 12th International Conference on, pages 1609–1614. IEEE, 2009.

[97] Frank Steinbrucker, Jurgen Sturm, and Daniel Cremers. Volumetric 3d mapping in real-time on a cpu. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 2021–2028. IEEE, 2014.

[98] Alex Stewart. Localisation using the Appearance of Prior Structure. PhD thesis, University of Oxford - New College, 2014.

[99] Hauke Strasdat. Local Accuracy and Global Consistency for Efficient Visual SLAM. PhD thesis, Imperial College London, October 2012.

[100] Hauke Strasdat, Andrew J Davison, JMM Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 2352–2359. IEEE, 2011.

[101] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam: Why filter? In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 2657–2664. IEEE, 2010.

[102] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis. In Vision Algorithms: Theory and Practice, LNCS, pages 298–375. Springer Verlag, 2000.

[103] Paul Viola and William M Wells III. Alignment by maximization of mutual information. International journal of computer vision, 24(2):137–154, 1997.

[104] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust tracking for real-time dense rgb-d mapping with kintinuous. 2012.

[105] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially extended KinectFusion. In RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Sydney, Australia, Jul 2012.

[106] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM without a pose graph. In Robotics: Science and Systems (RSS), Rome, Italy, July 2015.

[107] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust real-time visual odometry for dense rgb-d mapping. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 5724–5731. IEEE, 2013.

[108] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. The International Journal of Robotics Research, 34(4-5):598–626, 2015.

[109] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In European conference on computer vision, pages 151–158. Springer, 1994.

[110] Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. Octree-based fusion for realtime 3d reconstruction. Graphical Models, 75(3):126–136, 2013.

[111] Li Zhang et al. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multiview stereo. In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pages 618–625. IEEE, 2003.

[112] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Image and vision Computing, 15(1):59–76, 1997.

Appendix A

Lie Group Generators

A.1 Special Orthogonal Group $\mathbb{SO}(3)$

$$gen_0 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad gen_1 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad gen_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

A.2 Special Euclidean Group $\mathbb{SE}(3)$

$$gen_0 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad gen_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad gen_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$gen_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad gen_4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad gen_5 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$