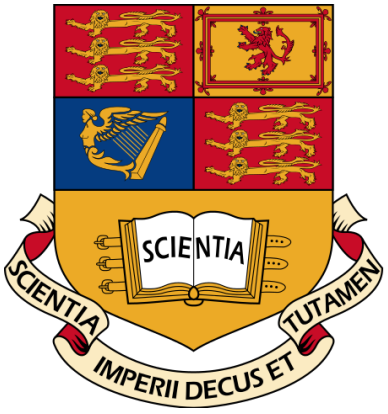
Design-Space Exploration for Dense 3D Scene Understanding: Exploring ElasticFusion

Imperial College of Science, Technology and Medicine

Department of Computing

Author:
Denise Carroll

Supervisor:
Prof. Paul Kelly
Co-Supervisor:
Dr. Luigi Nardi
Second Marker:
Dr. David Birch



June 12, 2016

Abstract

SLAM is the problem of building a map of an unknown environment and locating the robot inside this map. ElasticFusion is a new dense SLAM algorithm, differing from KinectFusion by incorporating photometric tracking and surface reconstruction and not doing pose graph optimisations.

Due to the difficulty of benchmarking SLAM algorithms, evaluation is often qualitative or limited to simple accuracy estimation. We integrate ElasticFusion into SLAMBench, a framework which facilitates the quantitative evaluation of SLAM systems.

We extend tools to explore the design space of ElasticFusion, evaluating algorithmic configurations based on the metrics of trajectory accuracy and execution time. We perform this design space exploration over both synthetic and real-world datasets. This allows us to map the Pareto efficiency surface over ElasticFusion's algorithmic parameters, yielding a 2x performance improvement without sacrificing accuracy, and improving trajectory accuracy results alone by 2x on the synthetic dataset. We are also able to improve the performance on the real-life datasets by up to 1.8x.

Acknowledgments

The dedication of this report is split 7 ways:

- to Prof. Paul Kelly for being a true inspiration and always making time to be involved and supportive through my project,
- to Dr. Luigi Nardi for the many interesting discussions and extensive time put in to help me with understanding concepts and analysing results,
- to Dr. Sajad Saeedi for offering lots of valuable insights throughout the later stages of my project especially on datasets,
- to Emanuele Vespa for the useful talks and helping with some tricky coding problems,
- to Dr. David Birch for providing useful feedback on my project and report,
- to Dr. Anandha Gopalan for being an excellent academic tutor throughout my four years at Imperial,
- and to my amazing family and friends, Lizzy, Charchris, Luke, Joel and especially Zuhayr who has been in my class for 10 years now and somehow still speaks to me.

Contents

Part I

Introduction

Chapter 1

Introduction

1.1 Motivations

Autonomous robotics is a rapidly growing field, and at the center of this is computer vision. For a robot to behave independently, it must have some method of real-time 3D scene understanding. Simultaneous Localisation and Mapping (SLAM) is the problem of building a map of an unknown environment and locating the robot inside this map.

SLAMBench is a 3D vision benchmark software for SLAM algorithms that allows us to investigate how parameter choices affect performance, power consumption, and accuracy. Performance is of high importance to SLAM to prevent the robot from losing track of it's surroundings, by dropping too many frames, and getting lost. Another key aspect of robot autonomy is power consumption, a low battery life means a robot cannot explore and map further distances before needing to be reconnected to power.

Following on from this, design space exploration allows the automated finding of optimal configurations for accuracy, power and performance. It will allow computer vision experts to release their software with ideal optimal default configurations, or release insightful information of different possible default configurations dependant on the dataset type e.g. texture-full datasets, synthetic datasets, loopy datasets, large or small scale datasets. Computer vision experts often don't have the tools to perform these types of software performance optimisation, and SLAMBench is able to facilitate them in benchmarking their algorithm quickly to holistically compare to other implementations and perform diagnostic reasoning.

Currently SLAMBench is based on implementations of two SLAM algorithms: KinectFusion and LSD-SLAM. In this project we integrate and explore an alternative dense SLAM implementation. ElasticFusion is a dense SLAM algorithm which performs SLAM differently to KinectFusion, without pose graph optimisation and utilising photometric tracking alongside geometric tracking.

By integrating ElasticFusion into SLAMBench we can perform a design space exploration of ElasticFusion, so we can identify the key algorithmic parameters which affect the accuracy, performance and power consumption and find better performance configurations.

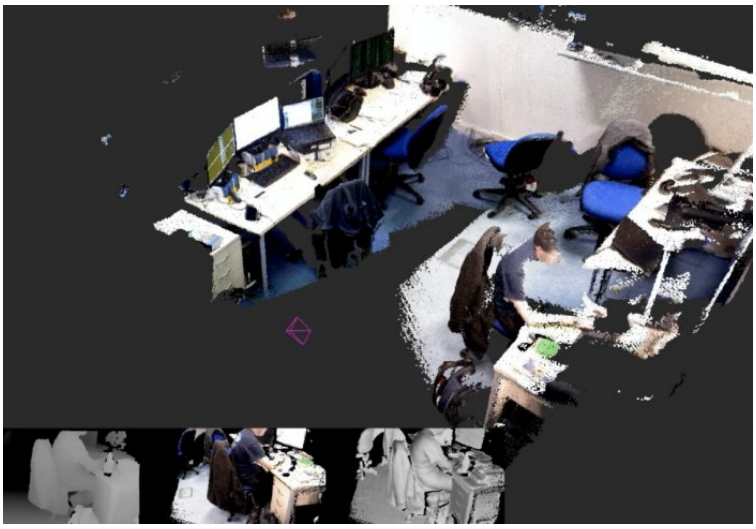


Figure 1.1: An example of ElasticFusion running. Image from Imperial News [5]

1.2 Objectives

1.2.1 Integration of ElasticFusion

Integrate ElasticFusion into SLAMBench, running on the ICL NUIM and TUM RGB-D dataset with functionality for the evaluation of ATE and frame rate.

1.2.2 Identification of algorithmic parameters

Once ElasticFusion is integrated into SLAMBench, the next step is to identify algorithmic parameters for ElasticFusion.

It will involve some preliminary experimental investigation of algorithmic parameters that affect ATE/execution time trade-offs. The aim of this is to find algorithmic parameters which can be used later on when mapping the pareto surface. At this stage it would be good to define the parameters and their potential value ranges, and explain their function and expected effects.

For example, some algorithmic parameters that could be investigated are the number of frames to integrate into the model, amount of the depth frame used for tracking etc.

1.2.3 Design space exploration and the Pareto surface

Map the Pareto surface for frame rate, ATE, reconstruction error for ElasticFusion and compare with KinectFusion. Were interested in optimisation with respect to several different objectives (accuracy, frame rate, power at least). Thus the point is not just to find a single optimum, but to identify configurations that optimally satisfy multiple performance objectives.

There are some recent developments by Dr Luigi Nardi on using machine learning to aid design space exploration, which we will extend to other SLAM implementations and datasets in this project.[18].

1.2.4 Key algorithmic parameters for ElasticFusion

Using the results from our design space explorations, we can gain an understanding of parameters which affect one of the three efficiency metrics of SLAM algorithms. A set of graphs and explanations regarding how each parameter affects the ATE/performance of ElasticFusion would be the output at this stage and the more well defined the better.

1.2.5 Architecture and dataset choice

The results will be fairly dependent on choice of architecture and dataset, therefore we will look at how the Pareto surface differs for different architectures and datasets.

1.3 Contributions

In this project we present the following research contributions:

- We have shown that a machine learning approach to design space explorations generalises to different SLAM applications by performing a design space of ElasticFusion. We have also shown that these design space exploration methods are more efficient than random sampling.
- We have used design space exploration to identify key algorithmic parameters for ElasticFusion.
- Performance and/or accuracy improvements on ElasticFusion

  We have achieved significant performance improvements by performing design space exploration on ElasticFusion. We are able to improve the execution time of ElasticFusion on the ICL datasets by almost 2x with no loss in accuracy, and improving the trajectory accuracy by about 50% whilst also improving execution time. For the TUM RGB-D dataset we can achieve almost a 75% performance increase without losing accuracy, pushing the performance of ElasticFusion from

about real time, to absolutely real time. We analyse and justify these results to draw conclusions in how to best run the ElasticFusion algorithm for certain dataset types.

- Dataset dependence

  We have shown how dependence on the dataset is really important when evaluating a SLAM implementation or performing a design space exploration. We see this as we get different optimal configurations for the ICL Living Room 2 dataset and the TUM RGB-D FR1 Desk dataset.

- Diagnostic SLAMBench

  We have identified potential issues within the ElasticFusion algorithm showing that SLAMBench can be used in a diagnostic way. The issues are related to the photometric and geometric tracking weighting being dependant on dataset for performance, and we suggest various optimisations as a result of our analysis e.g. dynamic frame-to-frame tracking weighting decisions, or reducing tracking pyramid iterations for the lower weighted technique.

To facilitate this we have implemented the following:

- We have integrated ElasticFusion into SLAMBench.
- Tools for ICL-NUIM and TUM RGB-D dataset compatibility with ElasticFusion.
- An extension of the machine learning design space exploration code to allow exploration of another SLAM algorithm - ElasticFusion, and another dataset - TUM RGB-D.

1.4 Organisation

We start by introducing the reader to SLAM, SLAMBench and the concept and methods of design space exploration. We then conclude the background section with a description of ElasticFusion and KinectFusion.

Part III details the integration of ElasticFusion - the initial integration into SLAMBench, and then the process of the evaluation metrics and dataset integrations. Following on from this is part IV which explains the design space exploration methods used for ElasticFusion and explains the algorithmic parameters being explored. Part V is the experimental evaluation - a collection of the findings and analysis of the project.

Finally the report closes with a summary of the most interesting findings of the report, and some directions this project could follow in the future.

Part II

Background

Chapter 2

SLAM

This section introduces the concept of Simultaneous Localisation and Mapping (SLAM), and discusses the typical differences between different SLAM implementations, and how we might evaluate their performance.

2.1 What is SLAM?

Simultaneous localisation and mapping is a computational problem of a mobile robot building a map of an unknown environment and using this map to locate itself. This would make a robot autonomous, eliminating the need for any priori knowledge. This is especially useful when a map cannot be easily constructed before hand, e.g. underwater, underground, or in outer space. The SLAM problem has been solved theoretically but there are still problems with practically implementing them, whether this be related to generality or accuracy [1].

The SLAM problem was first defined in 1986 at the IEEE Robotics and Automation Conference [1]. The Microsoft Kinect camera was a step forward for implementing SLAM methods, which previously required expensive sensors, scanners or stereo vision. General Purpose GPU has also enabled SLAM methods due to higher processing power for algorithms for a relatively affordable expense. [9]

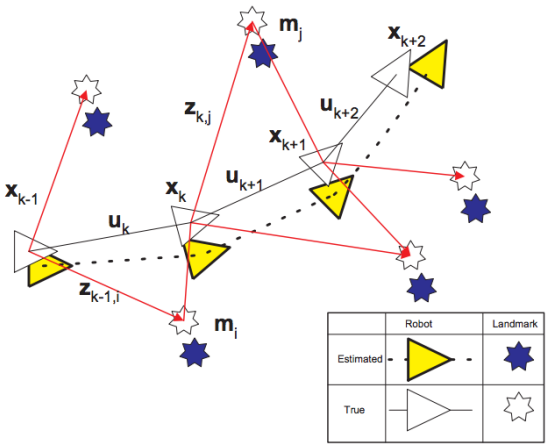


Figure 2.1: In this diagram from the SLAM Tutorial [1] we can see that the robot makes estimates of its position based on predicted landmarks in the map

The SLAM problem can be solved incrementally: Updating the map, then finding

the new camera pose with the updated map, then updating the map again with the new camera pose. These two steps are commonly called tracking and mapping.

2.2 Taxonomy of SLAM algorithms

There are some high-level differences between SLAM implementations, this is ultimately because different algorithms attempt to achieve different things. The needs of a marine robot are very different to that of a drone for example. When we speak about taxonomy, we speak about classification and SLAM algorithms can be fit into different groups.

- **Sparse vs. Dense SLAM**

  Sparse SLAM uses a subset of the possible features in a frame, using intensity information or extracting features using computer vision. Dense SLAM utilizes all of the available information, leading to more detailed 3D models.

- **Scale of environment**

  This can be large e.g. room scale, or small e.g. part of a desk scale.

- **Dynamic interaction**

  Whether or not the SLAM algorithm can deal with dynamic movement in the scene, such as a person walking through it.

- **Open vs Closed Loops**

  This is concerned with whether the algorithm performs loop closures, re-adjusting the model when drift has taken effect.

- **Type of tracking**

  Two examples of this are geometric tracking, which is the most often used tracking technique, and photometric tracking, which we will see is one of the main focuses of the ElasticFusion algorithm.

- **Sequential vs. Parallel**

  The final category we consider is the processing pipeline structure, which could either be parallel or sequential, essentially if the algorithm follows a PTAM like approach or not.

The above are high level concepts which will be explored more thoroughly in Chapter 5 - SLAM Algorithms.

2.3 Evaluation of SLAM algorithms

- **Performance**

  If the robot cannot keep up to real time frame rate the trajectory and environment mapping will become inconsistent and the robot may lose track completely of where it is unable to recover.

- **Power**

  We must measure the energy consumption by the CPU and the camera of the SLAM system. If the robot is to be autonomous, it cannot be connected to a power source/continually need to be recharged. Batteries are a bottleneck due to their low capacity, and improvement of them is slow. Therefore we must optimise the power usage of the devices, whether this be by optimising the algorithms, or turning off components of device that aren't needed at the time avoiding wasted energy.

- **Accuracy**

  The accuracy of the trajectory and environment map must remain at an acceptable level. A threshold of accuracy can be realised to allow space for large improvements in the other two metrics.

2.4 Summary

Having introduced SLAM, and discussed the high level differences between different approaches, we move on to expand more on the evaluation of SLAM algorithms. We have seen that the performance evaluation of SLAM algorithm is important, and automating this process is what we will look at next as we go on to the Chapter 3 - SLAMBench.

Chapter 3

SLAMBench

SLAMBench[11] is a software benchmarking framework, part of the PAMELA project [12], which allows users to investigate the performance, accuracy and energy-consumption of a SLAM system. In this section we will discuss how SLAMBench works, how it evaluates these accuracy metrics, and the performance evaluation methodology.

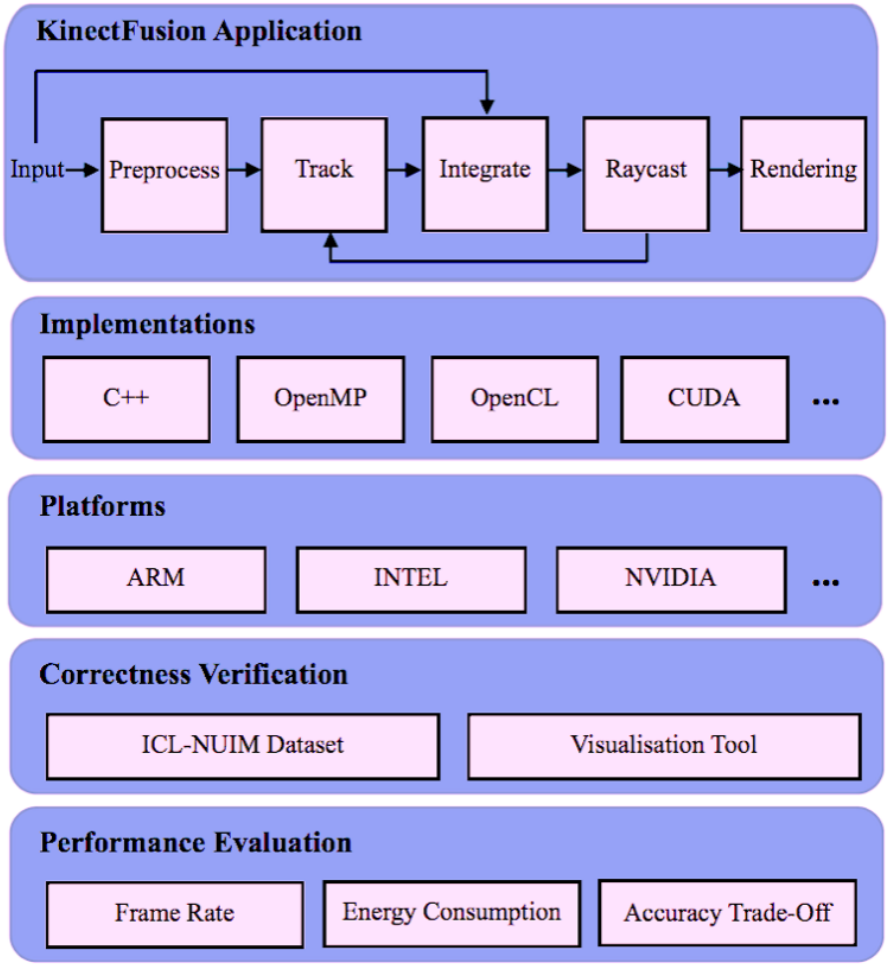


Figure 3.1: SLAMBench architecture diagram taken from the SLAMBench paper[11]

3.1 Performance Evaluation

As we can see in Figure 3.1 and from the previous chapter, we have three main metrics for evaluating performance as one of the layers of SLAMBench. These are execution time, accuracy, and energy consumption. The need for consistent metrics in the performance evaluation methodology ensures that we have comparable results between different SLAM implementations.

3.2 Accuracy measurement

In this section we deliver a brief introduction to the accuracy measurement in SLAMBench, delving deeper in the later section on performance metrics in the integration chapters of the report.

3.2.1 Trajectory

Absolute Trajectory Error (ATE)

At each frame the poses must be evaluated against the ground truth value. The absolute trajectory error (ATE) compares trajectories at each frame, recording the absolute difference between these two poses. The error for the whole trajectory can be calculate using an averaging of the per-pose absolute errors, for example root mean square error (RMSE) or arithmetic mean. The difference between these two means is that the RMSE is skewed more by outliers.

Relative Pose Error (RPE)

Another measure is the relative pose error (RPE) which compares correlation between two trajectories over time [26]. The relative pose error measures the local accuracy of the trajectory over a fixed time interval delta.

3.2.2 Mapping accuracy

Mapping accuracy

Currently it is assumed that a good trajectory will lead to an accurate map. However this is not guaranteed, and even small inconsistencies in the reconstructed map can cause issues for a robot moving within it.

Getting a ground-truth map of an environment for comparison against is a non-trivial task, and is an area still being researched.

Along with this difficulty, is that of estimating the accuracy of the map if a ground truth is available. THe most simple idea is to use surface registration, which we will

cover in more detail when we go on to SLAM metrics in the Integration chapters of the report. Emanuele Vespa, in the software performance optimisation team, has done a lot of work on estimating the accuracy of the map [2].

3.3 SLAMBench Performance Evaluation Methodology

SLAM implementations are iterative and involve approximations, therefore evaluating accuracy is not as simple as an equality test. Algorithms may be used in different contexts, so it must be taken into account that it may place stress on different areas of the system each time. It is therefore necessary to have some kind of methodology in place for evaluation.

SLAMBench's methodology has a number of components which ensure consistent and reproducible experiments, which is something we will have to take care to comply to when integrating ElasticFusion:

1. Deterministic datasets

   The same pre-recorded data set is used for evaluation, making the tests reproducible. However this prevents data retrieval being taken into account for the performance.

2. Process-every-frame

   If a system cannot keep up with real-time then some frames will be dropped. Therefore SLAMBench removes time stamps and evaluates all of the frames in order. This removes bias due to platform performance or camera speed.

3.4 Datasets

ICL-NUIM Living Room Datasets

This is a synthetic living room model, containing 4 different camera trajectories. The datasets provides simple ground truth files, well aligned camera poses and ground truth maps, making it an easy dataset to work with and integrate into new SLAM implementations, but lacking in being realistic.

TUM RGB-D Datasets

These are real scene datasets, captured with a Kinect depth camera [26]. The RGB images in the ICL-NUIM dataset aren't as realistic, lacking texture and colour in comparison to the real-world datasets of TUM. The benefit of a real world dataset

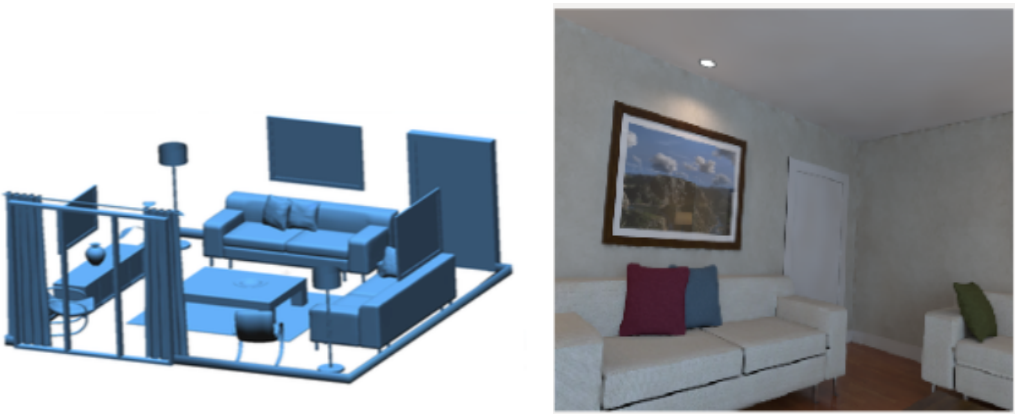


Figure 3.2: A ground truth map of the ICL NUIM Living Room scene, and an example RGB frame from the Living Room scene.

is that the noise in depth measurements its more accurate and will be similar to a SLAM system which is running live, which is of course the ultimate use of a SLAM system.

The TUM RBG-D scenes have a lot more colour and texture than the ICL NUIM scenes, an important factor in evaluating SLAM implementations that use photometric tracking.

Figure 3.3: An example scene from the TUM RGB-D FR2 Warehouse scene.

3.5 Summary

Having discussed SLAMBench, it's ideas and methodologies, we move on next in our background research to design space exploration methods, which are facilitated by SLAMBench and allow us to find optimal configurations to improve the performance of SLAM implementations.

Chapter 4

Design Space exploration

Finding good performance configurations for SLAM algorithms is a difficult task as an exhaustive search over an entire design space is so infeasible. Here we introduce methods of design space exploration that we could use to facilitate design space exploration of ElasticFusion.

4.1 Design space Exploration

Design space exploration (DSE) is the activity of exploring design alternatives before implementation. We can see methods of design space exploration dating back to the 90s, using response surface models to provide a quick and accurate method of exploring the entire design space [13]. To find this Pareto surface a design space exploration can be performed. SLAMBench facilities design space exploration by allowing one to alter parameters, whether it be high or low level e.g. TSDF distance of the KinectFusion algorithm, or compiler optimisation parameters. This allows us to assess the effect of many combinations of parameters to see how they affect the three different criteria of efficiency.

4.1.1 Pareto surface

Multi-objective optimisation involves mapping the optimisation space over multiple objectives. When there are three objectives to simultaneously optimise, this will result in a pareto surface. In Figure 4.1 the points on the Pareto front represent parameter configurations which result in a maximum for some optimisation combination. Points behind the line (in light pink) represent configurations which are outperformed on all metrics by another configuration.

4.2 Design Space Exploration methods

This section discusses various methods for performing design space exploration that we might be able to use in this project.

4.2.1 Brute force

Design spaces grow exponentially as we include algorithmic parameters and increase their possible range of values. These design space sizes can be in the magnitude

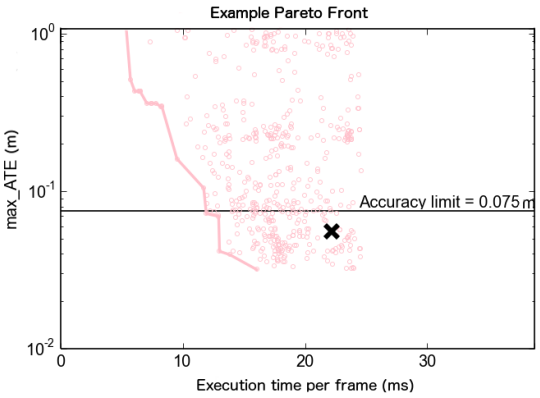


Figure 4.1: An example of a pareto front curve. The points on the curve represent varying levels of optimisation for each objective.

of millions, or essentially infinite. It is practically infeasible to explore all of these configurations as execution time may not be within the limits of a human life time. However this is the only true and accurate way of exploring all possible configurations and getting the very best performance result, as all other methods explore only a subset of possible configurations - and whilst they may be clever and have high probability of success, cannot be exhaustive.

4.2.2 Design of experiments

The first approximation we will introduce is Design of Experiments.

DOE[16] would make informed decisions on varying input parameter values, to try to gain maximum amounts of information on the results on the output. It is a systematic method which selects the best possible parameter choices early on with the statistical knowledge than they will cover a large part of the design space. Later in the report we touch on a small experiment we did using DOE techniques.

4.2.3 Sensitivity analysis

Sensitivity analysis studies use statistical analysis to evaluate cost functions over the inputs of an algorithm[15]. It is all about how the study of input parameters may affect the output. The theory that uncertainty in the output of a model can be explained by the uncertainty in its inputs. When we perform an senstivity analysis, we are concerned with finding the parameters, which when fixed to their optimal value would cause the greatest reduction in variance of the output [14]. To find optimal configurations without performing an exhaustive search, attempts to quickly converge to estimated optimal solutions are iterative search algorithms based on defined cost functions.

4.2.4 Random Sampling

A set of random configurations is a good way of achieving a smaller subset of the whole design space without bias. However, it may not be as efficient as more refined methods. The chance of hitting interesting parameter configurations shrinks with the sample size, so more direct

4.2.5 Active learning

The active learning paradigm in machine learning is concerned with generating new samples based on an internal predictor, running this samples and then retraining the predictor. It is an iterative semi-supervised machine learning algorithm.

This can be applied to design space exploration as we can train a predictor which should predict the best configurations for a SLAM algorithm. Each iteration we predict new optimal configurations, benchmarking them and then predict new points which would optimise our output.

The actual predictor used in machine learning can be a regressor or classifier of choice. In the work by Luigi Nardi [18], a randomized decision forest is used which is ideal for a configuration type design space as it is essentially a multi-level decision tree. We will revisit this in more detail later in the report.

4.2.6 Summary

Table 4.2 summaries design space exploration techniques in terms of comparative efficiency. The four main methods we have seen that improve the efficiency of the process sacrifice accuracy but still result in a good accuracy of result - however the results will of course be application dependant.

Random sampling has been rated as having the lowest efficiency after exhaustive search as there is no attempt to filter out pathological results. However, random sampling - not suffering from possible bias as the other techniques may, tends to generate really good results even if this is not the most efficient path towards them.

Method	Brute Force	DOE	Sensitivity Analysis	Random Sampling	Active Learning
Efficiency	Lowest	Good	Good	Low	Good

Figure 4.2: The efficiency of the design space exploration techniques

In this project we attempt a small experiment using Design of Experiments but don't find it has very good results, and move quickly onto a focus of random sampling and active learning design space exploration techniques.

Chapter 5

SLAM Algorithms

Now that we have introduced the concept of SLAM, and researched our tools for evaluating SLAM and performing a design space exploration, it is time to delve into the SLAM algorithms that we are interested in for this project. The first is ElasticFusion, which is the primary focus of the report. We also introduce KinectFusion, which is the first SLAM implementation to be introduced into SLAMBench, and we use it to contrast some of our findings from ElasticFusion in the evaluation chapters.

5.1 Preliminaries

5.1.1 Mathematical Preliminaries

Homogeneous coordinates

Homogeneous coordinates are a tool used in computer vision to represent 2D points as 3D points. The additional point is called the focal length.

Transformations can be represented as one matrix, the three parts rotation, scaling and translation all in one, and can then be applied using simple matrix multiplication.

Camera intrinsics matrix, K

The internal camera pixels may be different e.g. skew factor, lens distortion, so the camera intrinsics matrix, K, is used to correct this during projection.

Perspective Projection

Projection is the process used in computer vision to convert a scene or model from 3D space to 2D space (i.e. for viewing on a 2D screen).

Perspective projection has a focal length, allowing for realistic projections as a human eye would see them. It has depth.

Alternatively, orthographic projection has no depth. Objects close and far have the same dimensions.

Given a 3D point $p = [x, y, z]^T$, the perspective projection is $u = \pi(Kp)$

Back-projection

The back projection of a point u, with depth map D is defined as p(u, D) = $K^{-1}ud(u)$

Dehomogenisation

This is the process of converting from a 3D homogeneous coordinate back to a 2D coordinate. In this paper it will be represented by the function $\pi(p)$

Given a point $p = (x, y, z)$, $\pi(p) = (x/z, y/z)$

therefore the points $(2, 4, 1)$, and $(4, 8, 2)$ represent the same point in homogeneous space.

5.1.2 Theoretical Preliminaries

Loop Closures

A loop closure resolves drift that appears from a robot revisiting an area on a map where the current frame is inconsistent with the existing map. Loop closures detect this drift, and then we can apply deformation or some other correction to the map to tie everything together. Loop closures can be done on large and small scales, especially important are global loop closures to keep larger scenes consistent and free of large scale drift.

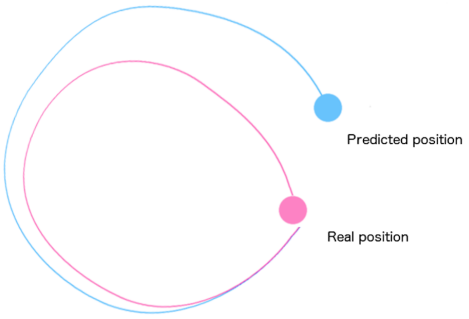


Figure 5.1: An example of a loop closure. The predicted position has drifted from the real position, and can be corrected with a loop closure deformation to fit the real pose.

Types of motion

- *Loopy*: Painting type motion, lots of back and forth over a small area. This is the kind of motion KinectFusion can expect. This can be dealt with using joint probabilistic filtering or joint pose and feature optimisation in sparse SLAM, but in dense SLAM there are far too many features for this to be computationally viable.
- *Corridor-like*: Not many small loops, but drift can occur on a larger scale.

5.2 Elastic Fusion

Elastic Fusion is a incremental, dense slam algorithm, without a pose graph. ElasticFusion introduces a more map-centric approach, taking the focus away from pose optimisations. Most SLAM algorithms don't target for both very loopy motion and large scale loop closures.

The algorithm creates a surfed-based model of the environment. Each frame attempts to perform local loop closures - to stay close to the mode of the map distribution, and also global loop closures - to avoid drift and maintain global consistency.

The explanations below are inspired from the ElasticFusion paper [3].

5.2.1 Summary

- Real-time
- Dense map
- Room-scale size

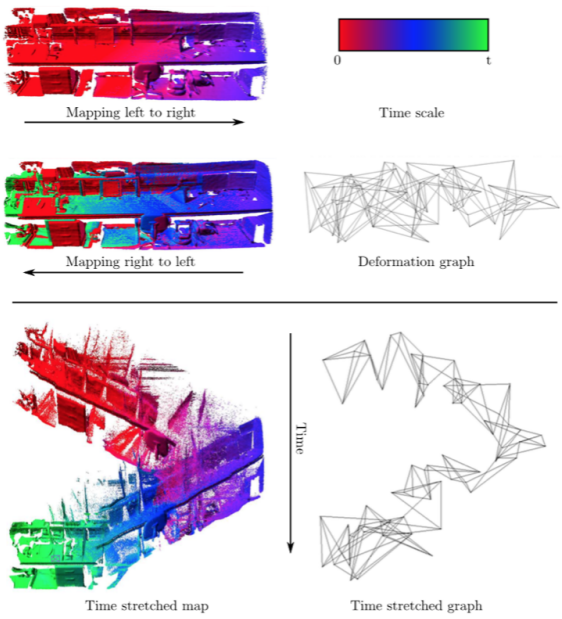
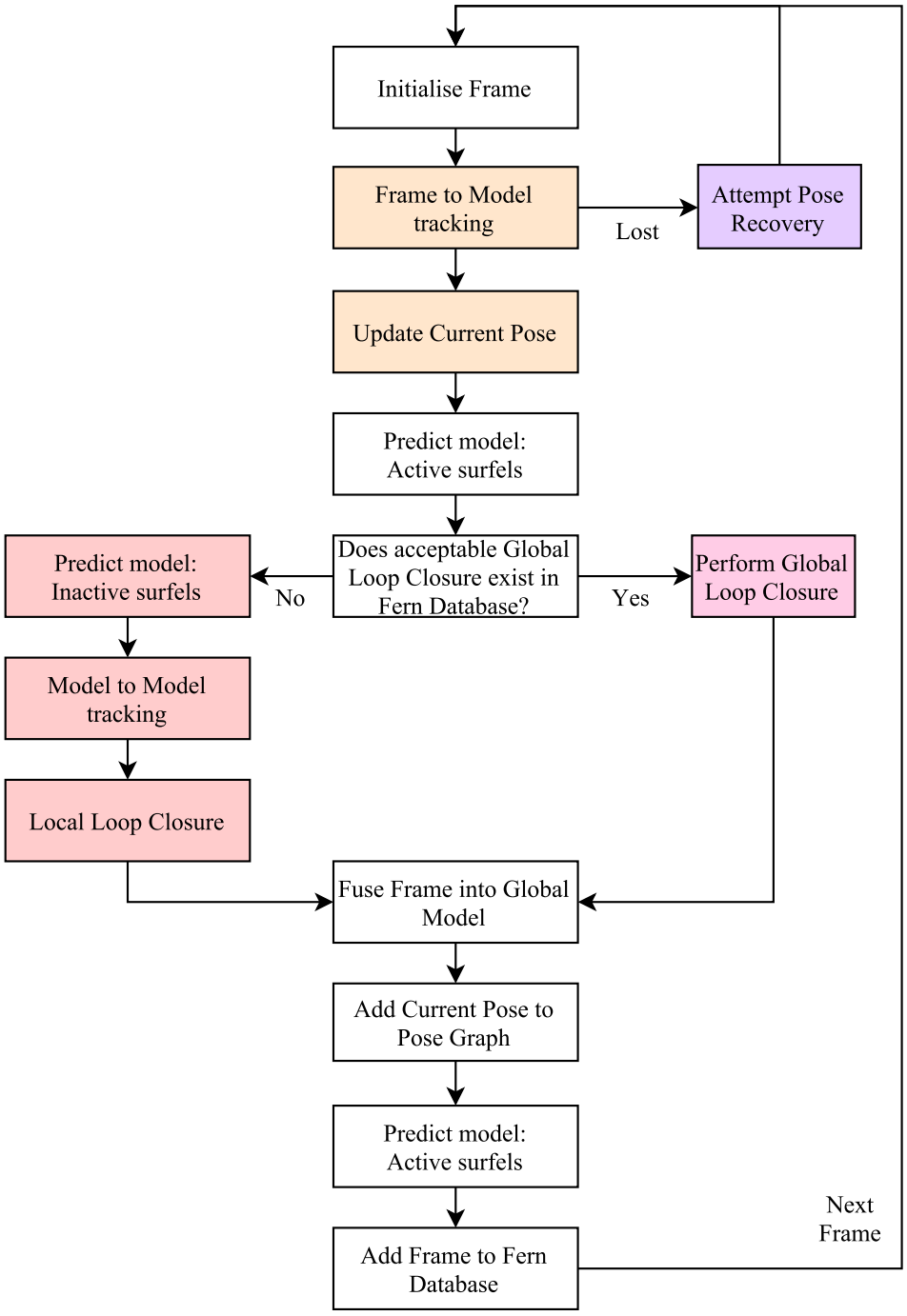


Figure 5.2: This diagram, taken from the ElastsicFusion paper, shows how temporally close surfels are connected in the deformation graph, rather than being connected spatially

5.2.2 Block diagram

The following figure is a block diagram of ElasticFusion, showing the main components involved in processing a frame:



5.2.3 Preliminaries

Surfel-based model

The scene is respresented by an unordered list of surfels. A surfel is a geometric representation for rendering which consists of a tuple of values: a position, a normal, a colour, a weight, a radius (surface area around a point), an initialisation timestamp, and a current timestamp.

Active and Inactive areas

There is a time window threshold, dt, which separates the list of surfels into active and inactive areas. The inactive area is not used for tracking and fusion until there has been a loop closure which makes it active again.

RGB-D frame

Each RGB-D frame has two main components:

Depth and Normal Map
Each frame has a depth map, with each pixel representing the depth from the camera. The normal for each pixel is calculated from the depth using the central difference method to create a normal map.

Colour Image
Each frame also has a colour image which has a colour for each pixel captured. Each pixel has a colour, c, with three colour values. The intensity of the pixel is the arithmetic mean of these three colour values.

5.2.4 Tracking

Pose estimation

Tracking is concerned with estimating the pose of the camera. The global pose of the camera is estimated on each new frame, by registering the current depth map and colour image with the predicted maps.

This results in a transformation matrix of the form:

$$P_t = \begin{pmatrix} & R_t & & T_t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where R and T are the rotation and translations required to register the two poses.

The approach uses splatted rendering to predict both a depth map and a full colour surfel model.

Geometric Pose Optimisation

Geometric pose optimisation is concerned with finding the registration which minimise the point-to-plane error.

The cost is defined as the squared sum of distances between the transformed vertexes from the previous frame and the current vertexes.

Photometric Pose Optimisation

Photometric pose optimisation aims to minimise the intensity difference after registration between pixels.

The cost is defined as the squared difference in intensity between the transformed pixels from the previous colour image and the live colour image.

In corridor-like environments there are not as many features, so relying on depth alone for the camera pose is not sensible, so we can match on intensity combined. To facilitate the photometric pose optimisation a colour surfel rendering must be predicted alongside the predicted depth map.

Combined cost function

Total cost = Geometric cost + 0.1Photometric cost.

The cost function is minimised to find an optimal transformation matrix to transform the current pose to align with the active model. Once the current pose is aligned, it means the pixels in the live data are aligned with the surfels in the current map - i.e. it is ready for fusion.

5.2.5 Deformation graph

A deformation is a set of nodes and edges which are to be deformed. Each node consists of a timestamp, a position, a set of k neighbour nodes (edges), and an affine transformation (a rotation and a translation). The rotation and translation are optimised according to surface constrains from the loop closures.

Construction

On each new frame a new deformation graph is created which is computationally less expensive than updating an existing one. The nodes, G, are sampled from the list of surfels such that the number of deformation nodes is much less than the number of surfels. The nodes follow a uniform distribution to preserve spatial density of the surfels.

This set of nodes, G, is ordered and connected by time, to avoid active and inactive areas influencing each other (Can be seen in diagram in Figure 5).

Deformation Application

This algorithm outlines the process of deforming the surface on a per-surfel basis. A set of influencing nodes are found and a weighted transformation is calculated for the surfel based on these influencing nodes.

1. Find the set of influencing nodes for a surfel
   (a) Find the temporally closest node, c, to the surfel
   (b) Select another α nodes close in time: [c - $\alpha/2$ to c + $\alpha/2$]
   e.g. $\alpha = 4$, c = 7, close nodes = 5, 6, 7, 8, 9
   (c) Take the closest k nodes as the set of influencing nodes
2. Calculate weight of each influencing node
   (a) Calculate dmax, the distance between the furthest influencing node and the surfel.
   (b) Calculate the weight for each influencing node, as (1− distance to surfel / $dmax)^2$
   (c) h becomes the sum of the distances for all the nodes (to scale them to 1 later)
3. Apply the transformations to the surfel vertex and normal
   The deformed surfel is calculated by the sum of the following two matricies for each influencing node:

- *Vertex*: Apply the node rotation matrix to the surfel-node vector, then translate with the node translation matrix. Multiply by influencing weight.
- *Normal*: Rotate the surfel normal by the inverse-transpose of the node's rotation matrix. Multiply by influencing weight.

Optimisation

This set of points Q represents the source and destination points, and their corresponding timestamps.

- **Rigidity**

  The rotation cost function is defined as the absolute squared difference between the rotation matrix * inverse-transpose rotation matrix and the identity matrix. The reason being so that items not on the leading diagonal will be optimised towards 0 i.e. there is no scaling, a more rigid transform.

- **Regularisation**

  The transformation between a surfel and its neighbouring nodes should be minimal.

- **Constraint**

  The squared distance between the transformed source node position and the destination node position. In other words, the source position should become the destination position after a transformation has been applied to it.

- **Pin**

  The destination position of inactive areas should not move, meaning the active areas are being merged into the inactive areas, and not the inactive areas being merged. This is computed as the squared distance between the destination node and the transformed destination node.

There are a set of four optimisation functions to: maximise rigidity (rot), regularisation (reg), position error (con), inactive pinning (pin).

A total cost is calculated from these 4 cost functions:

$$E = w_{rot}E_{rot} + w_{reg}E_{reg} + w_{con}E_{con} + w_{con}E_{pin}$$

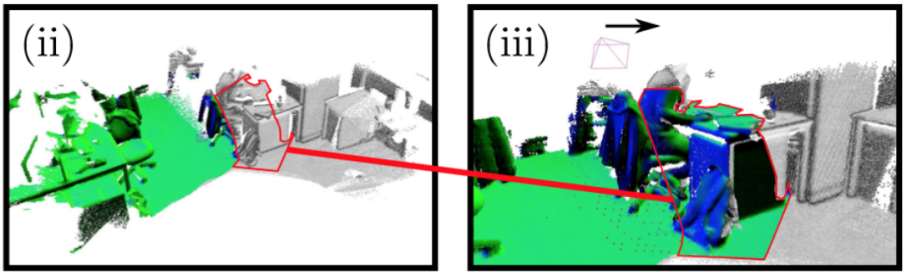
which is minimised using the iterative Gauss-Newton algorithm.

Loop Closures

To ensure surface consistency successful loop closures are applied as non-rigid surface deformations.

5.2.6 Local Loop Closures

When areas of the map get revisited, the existing map is reviewed to find local loop closures that be be deformed to align together.



Areas of the map that haven't been visited recently are set to inactive. When these areas are revisited, it is attempted to register the active part of the frame and the inactive part of the frame together. The inactive areas become active again after being revisited.

If there has been no global loop closure then try to find a loop closure between the active and inactive set of surfels.

The registration is computed using the same method as the tracking stage for the global camera pose, resulting in a transformation matrix output, H. The cost of alignment for this matrix must be small enough for the deformation to take place.

Surface constraints

The transformation matrix calculated by registering the active and inactive areas is used to generate the surface constrains for the deformation.

The destination position is calculated as:

$$(HP_t)p(u, D_t^a)$$

which is the transformed current pose estimate multiplied by the back projection of u given depth map D.

The source position is simply:

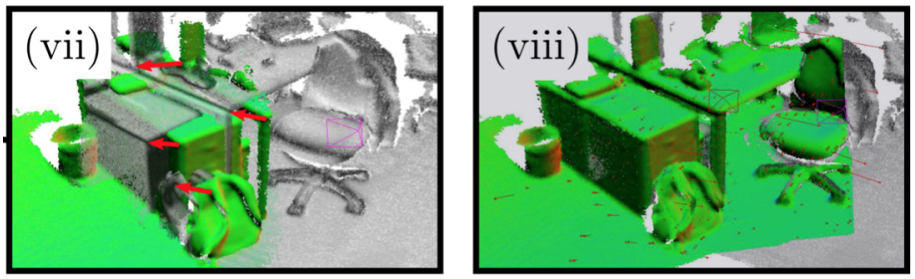
$$(P_t)p(u, D_t^a)$$

which is just the current pose estimate multiplied by the back projection of u given depth map D.

Updates

The current pose is updated by transforming by H.The depth map for each surfel must be updated to reflect the deformations that have happened.

5.2.7 Global Loop Closures



As we can see in the diagram above, the map is inconsistent and misaligned. To solve this a global loop closure is performed to match up the active and inactive areas of the map. This solves the problem of large scale corridor-like motion drift.

Fern Encoding Database

Predicted views of the surface map are encoded into the fern database once they are aligned and fused with the live camera view. If there are parts of the predicted view which dont have any mapped surface then these gaps are filled from the live depth camera.

There is a fern encoded frame database, each element has a fern encoding string, a depth map, a colour image, a source camera pose, and an initialisation time.

Each frame, the predicted active model depth and colour image after fusion is added to the database. The database is queried to see if there is a global loop closure required.

Global Loop Closure Constraints

Attempt to register the current model with the fern frame. If this is successful, it will result in a transformation matrix, H, like in the local loop closure method.

The destination position here will be the transformed fern camera pose times the back projected fern-pixels, and the source will be the current camera pose times the back-projected pixels.

The tracking cost functions from the deformation algorithm are much more tightly controlled here because a global loop closure will have a larger effect on the global maps.

Updates

The current pose estimate is updated to the transformed fern camera pose. No other updates are done to the surfels, because they will be sorted by local loop closures in the next frame.

5.3 KinectFusion

Each depth frame from the RGB-D camera is registered for pose and fused into the environment map. The following explanations are based on the KinectFusion algorithm in [8].

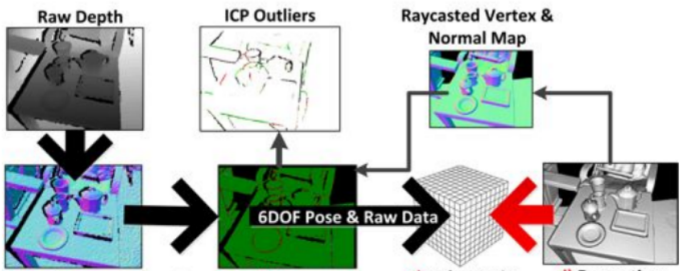
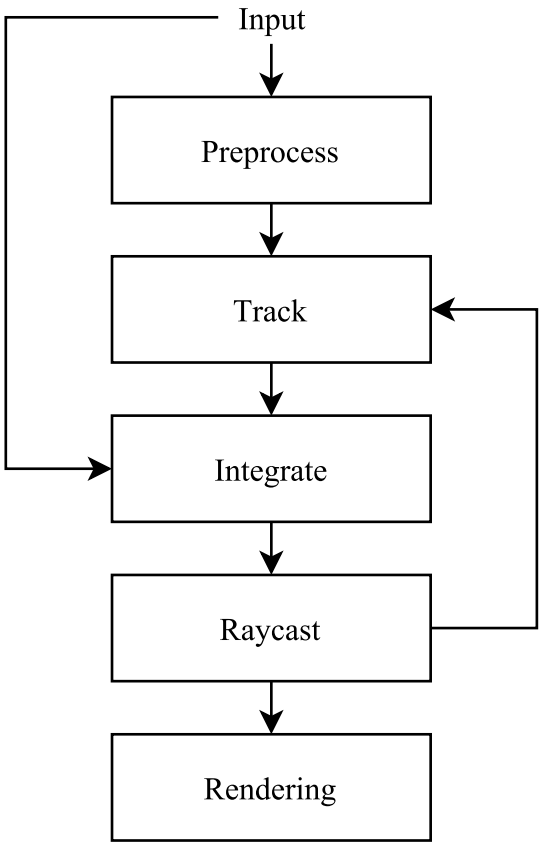


Figure 5.3: Overview of the KinectFusion system

5.3.1 Summary

- Real-time
- Dense map
- Room scale
- Can handle short-term dynamic objects

5.3.2 Block diagram



5.3.3 Preprocessing

Depth Map Conversion

Each GPU thread re-projects a specific depth measurement as a 3D vertex in the cameras coordinate space by multiplying the depth of the pixel with the inverse intrinsic calibration matrix, resulting in a vertex map, V. A normal vector map, N, is also created.

Filtering

A bilateral filter is applied to the depth frame to reduce noise.

5.3.4 Tracking

This stage involves registering the current point cloud with the global map using Iterative Closest Points, which finds the camera pose.

Camera pose matrix

The transformation matrix has 6 degrees of freedom, 3 for a rotation and 3 for a translation.

$$T_k = \begin{pmatrix} & R_k & & t_k \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Iterative Closest Points

Find correspondences between current points and previous points. This is done using a method called projective point-plane data association. Corresponding points which have a distance over a certain threshold are discarded as outliers (because there is a high frame-rate, there should be little distance between feature points between two temporally adjacent frames.

The next step is to find the transformation matrix that minimises the equation below. This is the sum of squared differences between the feature points between the current transformed frame and previous frame:

$$argmin \sum \left|\left|(Tv_i(u) - v^g_{i-1}(u)) * n^g_{i-1}(u)\right|\right|^2$$

where u is the current pixel, v the vertex depth map value at u, n the normal depth map value at u, and T the iterative transformation matrix.

The ICP algorithm is dense, it uses all the available points, and is therefore safe from sudden movements because there are enough other points for the points to register. However this will not be robust to long term continuous movement.

5.3.5 Fusion

The depth map is fused into the current model after the tracking stage. This model is represented using a voxel grid which uses a truncated signed distance function (TSDF).

Truncated Signed Distance Function

Each voxel in the voxel grid has a pair of TSDF value and a weight. The values are positive in front of the surface, and negative behind the surface edge. The surface can be recovered using ray casting which is explained in section section.

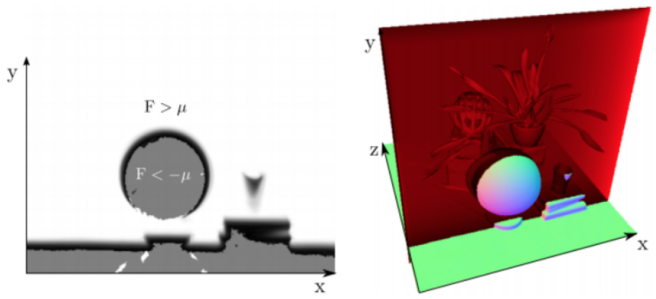


Figure 5.4: An example of the TSDF. The voxels not able to be seen behind the sphere are negative.

The surface is ready to extract by finding the zero-crossings, rather than having to calculate the mode of a probability distribution. Speed is really important to make this algorithm real-time. It also deals with uncertainty in range data, and fills in holes when new data is incorporated.

The TSDF weight function introduces a tradeoff between fast updates and increased noise if a higher weight is given to the newer values.

5.3.6 Volumetric integration

This section involves fusing the new frame into the TSDF environment map model.

For each voxel in the volume:

1. Calculate distance from camera center to vertex
2. Find actual depth measurement by looking up the image coordinates of vertex (back-project)

3. Calculate difference between calculated distance and measured distance as the SDF value

4. Normalise SDF value to a TSDF value, by dividing the SDF by the maximum truncation and capping at bounds [-1, 1]

5. Calculate a running average with the stored TSDF value and the new TSDF value

5.3.7 Ray casting

Each thread walks over a ray, testing each voxel until the zero-crossing is found. The surface point is calculated using 3D linear interpolation between the voxels on either side of the crossing. Rendering the view allows it to be used for tracking in ICP and for visualization.

When the camera pose equals the raycast camera transform, the depth and normal maps are the same because they are from the same perspective without the noise problems of the raw Kinect frames, i.e. tracking frame-to-model rather than just frame-to-frame.

Part III

Integration

Prelude

At this point, from the background section we have covered an overview of SLAM, the SLAMBench framework and design space exploration techniques, and have seen two major SLAM algorithms, KinectFusion, and Elastic Fusion - the focus of our investigation. From here we can begin our investigation, starting with the integration stages which are required to run experiments and perform our analysis.

In this section we first talk about the integration of ElasticFusion into SLAMBench, facilitating our later design space exploration (Chapter 6). We discuss how we run ElasticFusion to comply to SLAMBench's operational requirements, and various issues we encountered along the way.

We then go on to a chapter about the evaluation metrics we use, where we discuss the work we did to evaluate the three metrics of power, performance and accuracy for ElasticFusion (Chapter 7).

Finally, we supply and explain tools to integrate the datasets of ICL and TUM into ElasticFusion (Chapter 8).

Chapter 6

Integration of ElasticFusion into Slambench

6.1 Integration

Currently there are two SLAM algorithms integrated into SLAMBench. These are KinectFusion and LSD-SLAM.

6.2 ElasticFusion

An open source implementation of ElasticFusion is available from Whelan et al. on GitHub[29]. This is the version which we integrate into SLAMBench. It should be easy to add changes and updates to SLAM implementations in SLAMBench, so that improvements, updates and new versions can be easily benchmarked easily by developers. Using an open source implementation such as this will allow users simply to pull down any changes and rebuild SLAMBench.

6.3 Dependencies

6.3.1 Software dependencies

Additional required software required to build ElasticFusion

- OpenGL
- CUDA >=7.0
- Pangolin
- Other Packages: SuiteSparse, Eigen, zlib, libjpeg

6.3.2 CUDA and Hardware Dependencies

ElasticFusion is written using CUDA, so it is a requirement to run on an NVidia GPU. Whelan et al recommend the use of a very fast NVidia GPU (3.5TFLOPS+)[29], and to run on a fast CPU as well.

To run on a different GPU, the tracking code can be rewritten into OpenCL, or could be ported to plain C++ to run on CPU. The rest of the pipeline is written in OpenGL which is portable.

6.4 Benchmarking mode and OpenGL

Ideally, we would run ElasticFusion in benchmark mode. This is essentially running it without a GUI. The benefits of running in benchmark mode mean that there will be no GUI rendering times effecting the performance results.

ElasticFusion relies on an OpenGL context being available in the current thread - which is simple when a Pangolin GUI window is open because this provides an OpenGL context. A window isn't necessarily required, just a context. However removing this window is not so simple, meaning that it is difficult to run ElasticFusion Core headlessly.

Due to the fact that it was taking a long time to try and solve this issue and it didn't seem to have a simple solution to a non-expert, we decided the best way forward was to run on with the project with the GUI window active. It would require hacking in something like GLUT or directly interface X11, which was not really the focus of my project and not worth wasting too much time over. During the course of the project, a member of of the SLAMBench group, Bruno Brodin, was working on finding a solution to this problem, having submitted a question to Whelan et al concerning it.

To mitigate this situation, we calculate the total performance results as execution time per frame, with the tickers for these times not including the GUI rendering code. However, the on screen rendering may still cause some effect on performance, but this is not really an issue as all of the results we generate include the GUI code so the comparative results between ElasticFusion configurations are still valid.

A side effect of this point was the benefit of the GUI - we were able to gain a lot of visual insight and confirmation into how the parameters affect the ElasticFusion algorithm and where the camera was completely losing track or generating poor accuracy maps.

6.5 Process-every-frame mode

Process-every-frame mode enforces a sequential scheduling of all the frames in a log to ensure applications are processing the same frames.

ElasticFusion allows the skipping of frames if processing a log to simulate real-time. This is facilitated through the use of the frame skip parameter (-fs). If this flag is set to false, i.e. the parameter is not passed when running ElasticFusion, then all of the frames in a log are processed in sequential order from the log file.

Provided this flag is false, the frames processed are therefore deterministic and comply to SLAMBench's operational requirements. For safety, we actually removed this parameter completely during the experiments, but alternatively we could have set all the defaults to make sure it is false when running any experiments (the default in the ElasticFusion is set to true to mimic real time performance on lower specification hardware)

6.6 Tracking failures

To identify if tracking has been successful through the whole sequence a flag can be added to the output called "tracked" which states on a per-frame basis whether or not the frame has been tracked in to the model.

This type of data is only available if the re-localisation parameter is set, more on this later in Chapter 9. This mode is enabled by default in ElasticFusion. A brief summary however, essentially the algorithm finds that it is `lost` then it attemps to relocate the position by using recovery frames. On a tracking failure, the log reader keeps reading new frames, but the model is only concerned with the frame it got lost on, constantly trying to get back on track.

During a design space exploration, the tracked flag can then be checked to see if any tracking failure has occurred. This can be strict, e.g. considering even one frame not tracked a failure. However, considering ElasticFusion can recover from a lost pose using the re-localisation code and can utilise local and global deforms to try to get back on track, it may be more wise to weaken this threshold slightly to allow fast recoveries through.

6.7 Camera Calibration

The Camera Calibration matrices must be in the correct format to work with ElasticFusion. This requires a change of representation of the KinectFusion camera calibration files in SLAMBench, and this is listed below for the ICL-NUIM and TUM RGB-D datasets for reference and clarity.

The format of the camera intrinsics matrix in ElasticFusion is in the OpenCV camera intrinsics format.

$$M = \begin{vmatrix} fx & 0 & 0 \\ 0 & fy & 0 \\ cx & cy & 1 \end{vmatrix}$$

- (cx,cy) is a principal point that is usually at the image center
- fx,fy are the focal lengths expressed in pixel units

6.7.1 ICL-NUIM

https://www.doc.ic.ac.uk/ ahanda/VaFRIC/codes.html

The ICL-NUIM datasets use camera calibration specified below:

$$M = \begin{vmatrix} 481.20 & 0 & 319.50 \\ 0 & -480.00 & 239.50 \\ 0 & 0 & 1 \end{vmatrix}$$

Which results in a camera calibration file of "481.2 480 320 240", fx fy cx cy respectively.

6.7.2 TUM RGB-D

The TUM RGB-D datasets use the camera calibration specified below:

$$M = \begin{vmatrix} 580.80 & 0 & 308.80 \\ 0 & -581.80 & 253.00 \\ 0 & 0 & 1 \end{vmatrix}$$

Which results in a camera calibration file of "580.8 581.8 308.8 253.0", fx fy cx cy respectively.

Chapter 7

Evaluation of SLAM algorithms

SLAM algorithms are evaluated on three main metrics of efficiency: accuracy, power and execution time. We discuss the evaluation of these metrics in depth, and explain why we are interested in these metrics and why we chose them.

We begin by discussing accuracy. We argue between ATE vs RPE, and integrate RPE measurements into the trajectory accuracy scripts of SLAMBench. We also discuss the differences between evaluating accuracy on the ICL and TUM datasets, which we need to understand well to be able to integrate the same scripts into the design space exploration code later.

This is followed by a breakdown of ElasticFusion into "building blocks" so that a more thorough analysis of its performance in various sections of the code can be analysed during our analysis of the algorithm.

Finally, we discuss power readings and the problems we encountered during this project related to this metric.

7.1 Accuracy

7.1.1 Trajectory accuracy

Evaluating accuracy for the ICL-NUIM dataset

The ground truth trajectory for the ICL-NUIM dataset needs to have a constant offset removed, and the y axis flipped to align the ground truth and estimated trajectories allowing for an accurate

Evaluating accuracy for the TUM-RGBD dataset

The ground truth TUM datasets are recorded using a real depth camera, and the time stamps are asynchronously captured. Matching depth and RGB frames may not have the same timestamp like they do in the ICL-NUIM datasets. They are in fact approximately 20ms apart. To solve this problem they provide a script to associate the RGB and depth frames, which matches up RGB and depth frames which are a time offset apart.

The ground truth and estimated trajectory are then aligned because the TUM datasets are defined in an arbitrary coordinate frame making it hard to compare them without alignment (ICL-NUIM can simply rotate and remove offsets from theirs because the datasets are defined on origin coordinate frame). The trajectories are

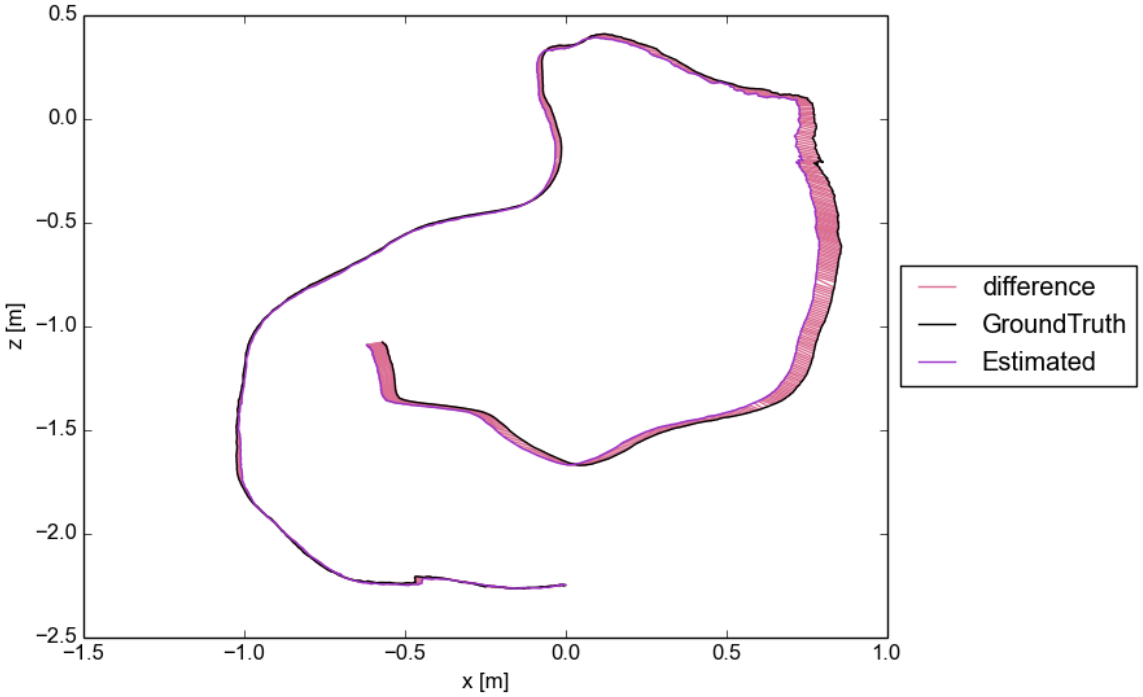


Figure 7.1: This is an example trajectory, with poses mapped down to the X and Z coordinate axes to 2D viewing. The different between the ground truth and estimated trajectory is the accuracy error, and can be evaluated in different ways.

aligned using the method of Horn described below:

Horn transform

The Horn transform is an image registration technique which here finds the rigid transform between two trajectories which may be in arbitrary coordinate frames.

However, it is debatable whether aligning the trajectories will skew the results and hide problems with global trajectory accuracy regarding drift. Therefore in this project we will be comparing the results of aligning and not aligning.

Absolute Trajectory Error

The absolute trajectory error (ATE) compares trajectories at each frame, recording the absolute difference between these two poses. The error for the whole trajectory can be calculate using an averaging of the per-pose absolute errors, for example root mean square error (RMSE) or arithmetic mean. The difference between these two means is that the RMSE is skewed more by outliers.

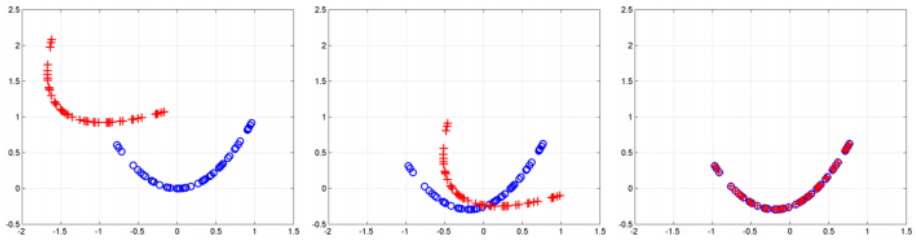
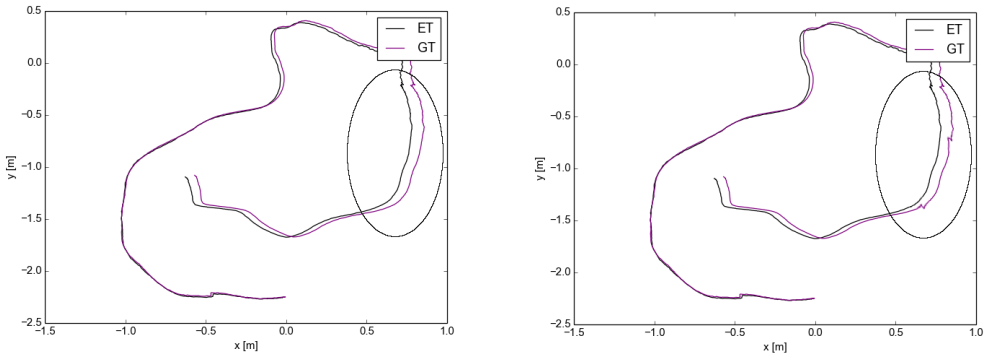


Figure 7.2: An example of an image registration technique aligning two datasets. Diagram by Dr. Ben Glocker [19]

The drift problem

Accuracy results can be masked by drift that results in an accumulative problem. Drift in a trajectory is an accumulative problem. A constant drift of a fraction of a millimetre per frame can build to centimetres of overall drift by the end of a trajectory.



Drift can hide poor spikes in the trajectory. In the first graph, the trajectory is very smooth and follows the ground truth very well, but has some drift. However in the second graph, there are spikes in the estimated trajectory, small blips in the pose graph. Arguably the second trajectory is less accurate, yet it would achieve a lower ATE because the spikes happen to be closer to the ground truth.

Selecting a suitable averaging statistic

When calculating the absolute trajectory error care must be taken into picking a correct average statistic.

- Mean and RMSE ATE: The Root Mean Square Error (RMSE) gives more weight to outliers because the error is squared - which is a more suitable measurement than the simple arithmetic mean here.
- Max ATE: The maximum error is useful in that it will not hide poor results within a mean. In the case of a SLAM algorithm that has poor tracking recovery,

the max would be an ideal metric as it would correlate well with the mean. However, this metric is potentially problematic in the case that an algorithm recovers well from a mishap. A large spike in an otherwise perfect trajectory could score worse than a heavily small spiked trajectory.

Relative Pose Error

The relative pose error (RPE) is a different way of evaluating the accuracy of an estimated trajectory. It calculates the rigid transform between two points on a trajectory, and compares this to the transform between the same two points on the other trajectory. The benefit of RPE is that because the two trajectories can be evaluated independently, they don't have to be aligned and the ground truth values can be pre-calculated.

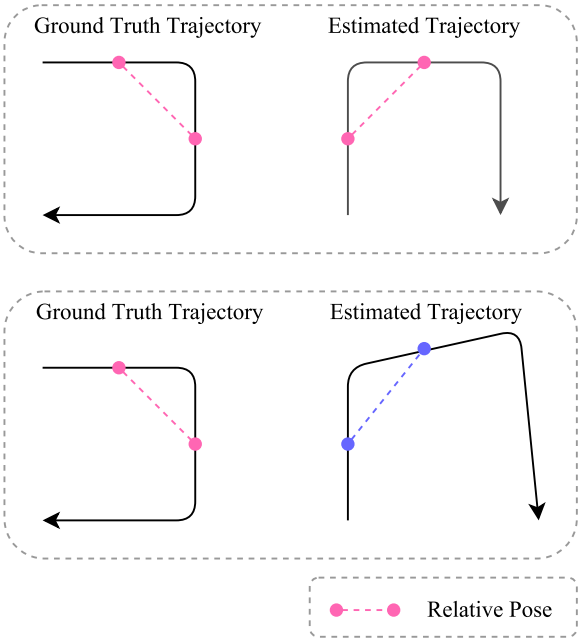


Figure 7.3: An example of RPE calculation. In the top diagram we can see that although the two datasets are not aligned, the relative pose between two identical points is the same. In the bottom section, we can see that the different in relative pose between two points is not the same.

The transform between points is calculated as $A^{-1}.B$ where A and B are the relative pose points on the trajectory.

Often the RPE calculation value will be higher than the ATE. This is because the RPE metric factors in a whole rigid transform - this is a translation and rotational error. The ATE is purely a translation error (rotational error would be somewhat encapsulated by the translation error if the trajectory was not aligned).

Fixed delta : Choosing the points to calculate the transform between is important for the result. The RPE can be calculated between every possible pair of points on the trajectory. Alternatively, a fixed delta can be used which discretises the calculation to pairs only with a fixed delta distance between them.

7.1.2 Map accuracy

There is a correlation between a good trajectory and a good map, however a good trajectory does not necessarily prove a good map - a small error in the map can cause problems for objects moving within it, e.g. tripping over something, or walking into a table leg.

Especially in the case of ElasticFusion, which employs a "map-centric" approach, it would be useful to have a metric evaluating this map accuracy as well as the trajectory accuracy.

The most simple way to do this conceptually would be similar to calculating the absolute trajectory error - simply identifying key points on the ground truth map and estimated map, registering the points and then calculating the absolute difference between each of them.

There is some automatic surface registration code from Handa et al. [27] [28] to do this with the ICL NUIM datasets. The problems with this are generating the estimated map in a compatible format to work with this tool; which could be of varying difficulty and required separately for each different SLAM algorithm.

However, obtaining a ground truth map is difficult, especially so in real-life datasets. Ground truth maps are not available for commonly used datasets e.g. the TUM RGB-D datasets. They do exist for the ICL-NUIM datasets, however this would force the map evaluation of SLAM algorithms to be limited to only this synthetic dataset (which we will see later in the report is a bad idea as design space exploration is both dataset dependant and synthetic and real-life datasets can perform very differently for SLAM algorithms).

Automating this process is difficult, and outside the scope of this project - however it will be something useful to do in the future of analysing ElasticFusion.

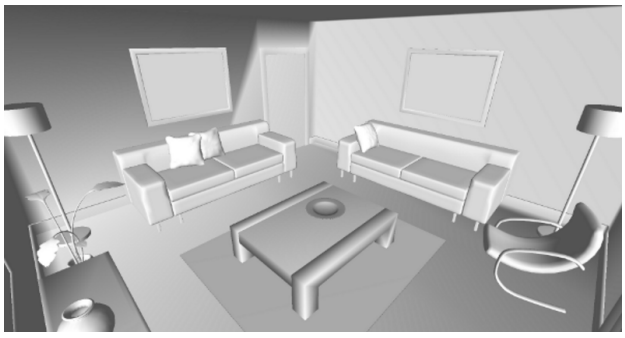


Figure 7.4: An example of a ground truth surface model. [27]

7.2 Performance

Performance here refers to the execution time of the algorithm. The performance results can be evaluated both coarse and fine grain. The benefit of a coarse grain representation is that it is easy to digest, and results can be evaluated quickly. The benefit of a finer grain profiling is that it can be seen which parts of ElasticFusion dominate the execution time, allowing more detailed analysis to take place.

7.2.1 Time per frame

The median of the total execution time per frame is the base performance result to be tested with. This value is in milliseconds (ms).

7.2.2 Fine grain

In KinectFusion, the code has been split into separate kernels which represent the different processes that the algorithm completes. A similar thing can be done for ElasticFusion. The dominant part of the code is the tracking code, and the model fusion code. These are the values we will pull out here to see how the parameters affect separate parts of the algorithm.

7.2.3 Blocks

Preprocess

This part is concerned with filtering the input depth frames. We will expect to see a different here when using the depth parameter.

Odometry

This is the total time for tracking of the current frame into the model. This includes the initialisation for the odometry, `odomInit` and the tracking itself `odom`. The initialisation sets up the RGB and ICP models.

FernOdometry

Similar to the normal odometry stage, this section is concerned with tracking a frame from the fern database into the current model. (This step is concerned with the global loop closures)

Fuse global mode

These are the three steps make up the code for fusing the current frame into the global model, Copy, Update and Data.

Tracking steps

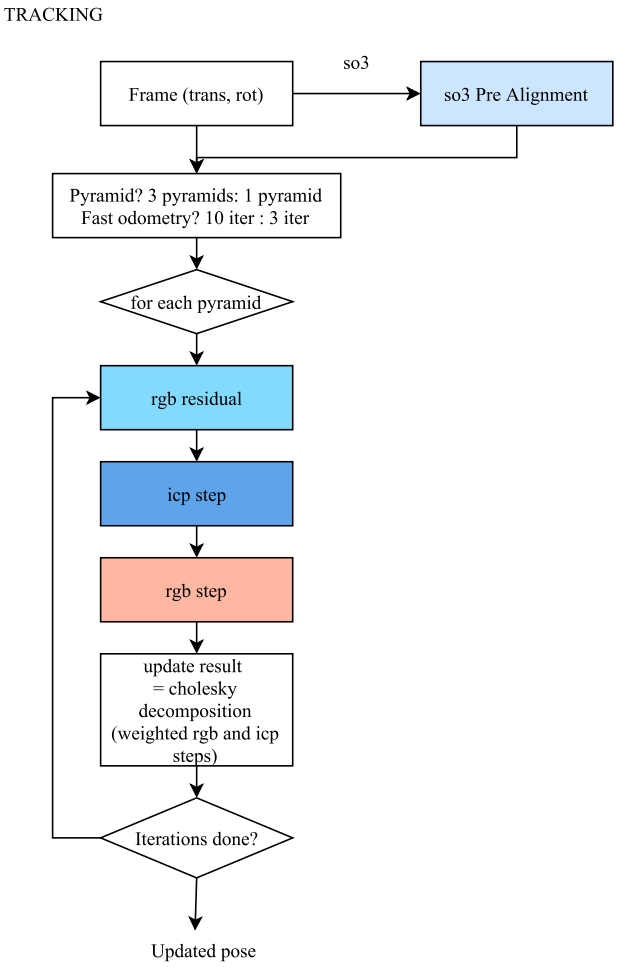


Figure 7.5: The four tracking steps - so3 pre alignment, compute RGB residual, icpStep (The time taken to complete the geometric tracking), rgbStep (The time taken to complete the photometric tracking) These four parts are highlighted in the tracking diagram, and constitute to the bulk of the processing for tracking.

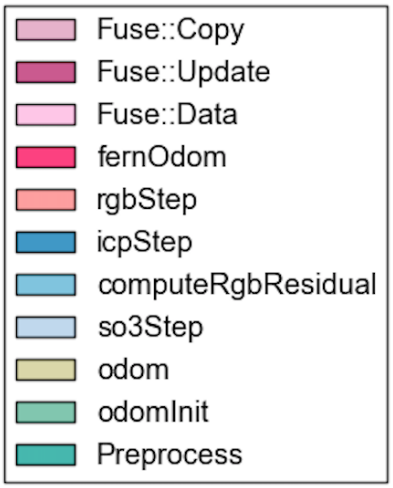


Figure 7.6: A summary of the parts of the code under analysis - using the colours that they will appear later in, in performance graphs.

7.3 Power measurement

Power readings are the next important metric to be considered. This is especially important on embedded mobile devices where power is limited by practicality and portability concerns and not just expense.

7.3.1 RAPL and PAPI

Currently in SLAMBench power is monitored by accessing the Intel Running Average Power Limit (RAPL) [23] driver which exposes power readings from the Intel Model Specific Registers (MSR). This driver is accessed via the Performance Application Programming Interface (PAPI).

Below is a summary of the power measurements that can be obtained using the Intel RAPL:

- PACKAGE_ENERGY: Total energy used by entire CPU unit i.e. PP0 + PP1
- PPO_ENERGY: "Power Plane 0" which includes all core CPU components
- PP1_ENERGY: The non-core components e.g. on original Sandy Bridge this includes the on-chip Intel GPU
- DRAM_energy: DRAM controller energy usage

ElasticFusion's tracking code is written using CUDA, which requires an NVidia GPU to run on. A bulk of the processing for this algorithm will be performed on the GPU. The PAPI API is used to monitor the RAPL register which is Intel, therefore the only power data we can get for ElasticFusion using PAPI is CPU based. The most interesting power statistics would be GPU based.

7.3.2 nVidia GPU power and NVML

To measure the power usage on nVidia GPUs we can utilise the NVML library. This is the library used by the `nvidia-smi` tool which reports statistics of the installed GPU. The GPU in my project machine (used for my experiments) is an nVidia GeForce 780 Ti. Unfortunately, the GeForce series does not support power measurements using NVML. Power measurements are available only on certain GPU series: Quadra and Tesla.

Having done some research into the problem, it became apparent that the nVidia drivers conceal this data and hacks are available on early nVidia drivers to access the power usage and utilisation. However they may be good reason that this data is unsupported for the GeForce series, perhaps because the measurements may be incorrect.

7.3.3 Possible solutions

- Only using CPU power readings

  This is not a good solution. The GPU dominates in power consumption for ElasticFusion. After some initial investigation of CPU power readings there was very little difference in power readings, a range of barely 5Watts. A small increase in power consumption on the CPU could correspond to a large saving in power on the GPU for example. There is too little correlation between CPU and GPU power usage to be able to use these results to produce any meaningful results.

- Running only on Quadra or Tesla GPUs

  The problem could be solved by benchmarking ElasticFusion only on specific nVidia hardware. The benefit is that the NVML software can be easily written to feed power measurements into SLAMBench. The negative is that obtaining specific GPU hardware is difficult without a high budget, and the exploration would be constrained further (and already contained to running on nVidia hardware).

- Porting to OpenCL/C++

  One option for solving this problem is to port the CUDA code in ElasticFusion to OpenCL. The benefit is that this would increase the scope of hardware able to be tested such as Intel GPUs and even integrated graphics units on laptops. The code would still run on the GPU so the performance results would be comparable to the CUDA version. Another option would be to port to C++ and run on the CPU. The tracking code is not designed to run on the CPU so the results here would not be comparable to the GPU versions, however it would be more portable and a design space exploration could still be performed on it in isolation.

- Wall power readings

  The power measurements could be taken using a wall reader. The pros of this solution are that the method would be architecture independent. The cons are that it measures all the energy of all the components at once - affecting the accuracy of the analysis. However, it could be argued that for algorithms which run primarily on the GPU, the GPU power readings would dominate other components anyway.

Chapter 8

Datasets

In this chapter we explain how we got the datasets of ICL-NUIM [27] and TUM RGB-D [26] to work with ElasticFusion in SLAMBench. We explain our decision to convert the datasets rather than rewrite the ElasticFusion log reading code, and introduce the tools we wrote to facilitate this. We then introduce the datasets we converted and used in this project, the four ICL-NUIM Living Room datasets, and two of the TUM RGB-D datasets (one which is a test dataset and moves minimally, and another that moves rapidly across a desk scene). As suggested in the background, and as we will see during our experimental evaluation, we need to use a variety of datasets to factor in the dataset dependence of benchmarking SLAM algorithms.

8.1 KLG file format

ElasticFusion runs using KLG log files. To run ElasticFusion on other datasets besides the one they provide, raw image and depth frames must be converted into the KLG file format.

We chose to convert the scene to KLG format, rather than rewrite the log reading code for ElasticFusion to accept RAW data files, which is the file format used by KinectFusion. This was to avoid editing too much of the log-reading logic of ElasticFusion.

The KLG file format is a log file type, the contents of this log are:

```
double:   timestamp
int32_t: depthSize
int32_t: imageSize
depthSize * unsigned char: depth_compress_buf
imageSize * unsigned char: encodedImage->data.ptr
```

8.2 Converting scenes to KLG

The creators of ElasticFusion provide two pieces of software for generating KLG files from live input data using OpenNI (Logger1) [30] or OpenNI2 (Logger2) [31], but not from a pre-recorded sequence. However this code was useful to find out the required format of the KLG log file.

Depth encoding

- **ICL-NUIM**

  The ICL-NUIM dataset depth files represent the depth as the euclidian distance to the camera. This has to be converted to a z coordinate to use for tracking in SLAM algorithms[27] (See Figure 8.1 for more details).

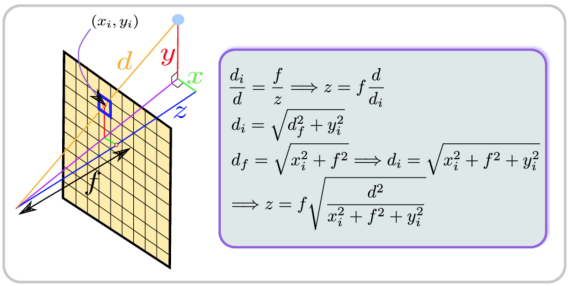


Figure 8.1: An explanation of the mathematics of the camera calibration conversion. Diagram taken from the ICL-NUIM dataset website. [27]

- **TUM-RGBD**

  The precision of the TUM datasets timestamp required us to change the precision of the KLG timestamp. The files are structured differently than ICL as provided by TUM [25], so we use a flag in this tool to separate the methods of reading for ICL and TUM.

  The TUM RGB-D dataset was taken asynchronously - so, like the conversion to the RAW log file format, the frames are inserted into the KLG file chronologically, using associate to find matches between the rgb and depth frames.

RGB encoding

The required image compression for ElasticFusion is jpeg. The conversion tool decodes the png files and compresses them to jpeg format.

Pseudocode to create KLG:

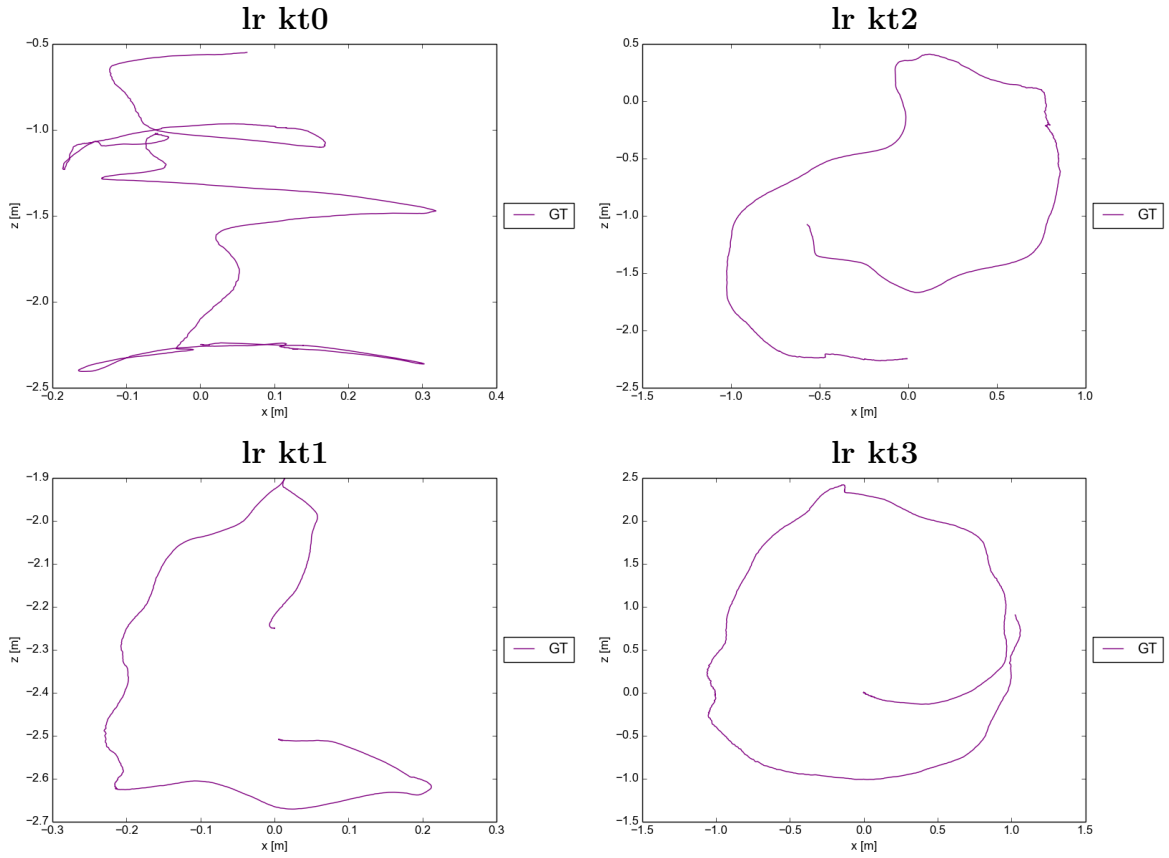
```
for frame_no, image, depth in all images and depths:

    if icl:
      distortDepthImage(depth, camera_calibration)
    if tum:
      compressDepth(depth)
    compressed_depth = compress(depth)
    compressed_image = encodeJpeg(image)
    append(frame_no, depth.size, image.size, compressed_depth, compressed_image)
```

8.3 Synthetic Datasets

Statistics on the datasets:

Living Room	'lr kt0'	'lr kt1'	'lr kt2'	'lr kt3'
Number of Images	1510	967	882	1242
Size (G)	3.3	1.9	1.9	2.6
Frame Rate (Hz)	30	30	30	30
Total Time (s)	51	33	30	42

The ground truth trajectories mapped down to the x and z planes, for the ICL NUIM Living Room datasets:



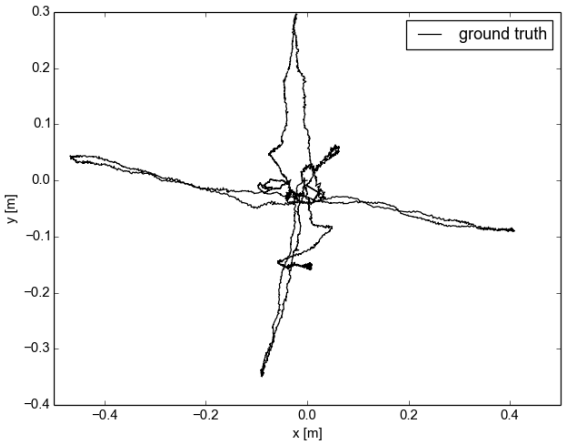
8.4 Real-Life Datasets

Statistics on the dataset:

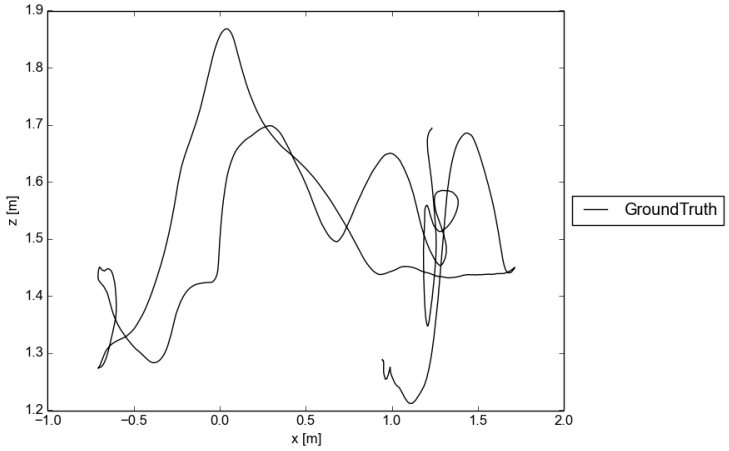
TUM	'fr2/xyz'	'fr1/desk'
Number of Images	3670	614
Size (G)	2.05	0.32
Frame Rate (Hz)	30	26.24
Total Time (s)	122.74	23.40

The ground truth trajectories mapped down to the x and z planes, for two TUM RGB-D datasets:

TUM RGB-D fr2 xyz



TUM RGB-D fr1 desk



Part IV

Design Space Exploration

Prelude

In this part of the report, we discuss the algorithmic parameters of ElasticFusion, identify suitable ranges for them and explain what they do at an algorithmic level and which parts of the code they affect (Chapter 9). We then discuss the various design space exploration techniques we attempted throughout this project, which include a design of experiments method, random sampling and an active learning approach designed by Luigi Nardi of the software performance optimisation team (Chapter 10). We also talk about the techniques we use to evaluate the design space exploration techniques and analyse these results.

Chapter 9

ElasticFusion parameters

Here we begin by introducing a summary of the algorithmic and configuration parameters of ElasticFusion. We then go into more depth explaining what the algorithmic parameters do and which parts of the ElasticFusion code they affect, hence predicting their affect later in the experimental evaluation stage. We predict the size of the design space using suitable discrete ranges, as the true size of the design space is infinite.

9.1 Parameters summary

9.1.1 Configuration parameters

Flag	Parameter	Description	Default
-s	Start cut-off frame	Frames to skip at start of log.	0
-e	End cut-off frame	Cut off frame of log.	maxint
-q	Quit	Quit when finished a log.	True
-cal	Camera Calibration File	Loads a camera calibration file specified as fx fy cx cy.	.
-l	logfile	Processes the specified .klg log file.	None
-fs	Frame Skip Mode	Frame skip if processing a log to simulate real-time.	False
-out	Ouput file	Output estimated trajectory file	logfile + ".freiburg"
-part	Partitioning mode	"reporting updates mode"	False

9.1.2 Algorithmic parameters

Flag	Parameter	Description	Default
-d	Depth cut off	Cutoff distance for depth processing	3m
-i	ICP/RGB weight	Relative ICP/RGB tracking weight	10
Thresholds			
-c	Confidence threshold	Surfel confidence threshold	10
-ie	ICP error threshold	Local loop closure residual threshold	5e-05
-ic	ICP count threshold	Local loop closure inlier threshold	35000
-cv	Covariance threshold	Local loop closure covariance threshold	1e-05
-pt	Photometric threshold	Global loop closure photometric threshold	115
-ft	Fern threshold	Fern encoding threshold	0.3095
Modes			
-o	Open loop	Open loop mode.	False
-rl	Relocalisation	Enable relocalisation.	False
-fo	Fast odometry	Fast odometry	False
-py	Pyramid	Use a pyramid for tracking in the RGB odometry	True
-nso	Disable pre alignment	Disables SO(3) pre-alignment in tracking.	False
-ftf	Frame to frame RGB	Do frame-to-frame RGB tracking.	False

9.2 Size of the design space

Total design space size

The total design space is actually infinite due to the continuous nature of the range of the majority of the parameters. Suitable discrete values have been placed on the parameters to allow a more realistic representation of the design space size and allow realistic analysis to happen during design space exploration.

Total design space size with discretisation

Total design space size with discretisation of parameter ranges:

= 11 * 13 * 13 * (2^5) * (5^5)
= 1,430,000,000
= 1.43 million possible configurations

Total design space without thresholds included :

= size(depth) * size (icp) * size(confidence) * size(boolean)^5
= 11 * 13 * 13 * (2^5)
= 457,600
= 450 thousand possible configurations

9.3 Parameter details

9.3.1 Depth cutoff

Cutoff distance for depth processing

Range: (0 - 10] m
Default : 3m

This parameter cuts the raw depth input off at n metres. The default is 3m. This parameter will be very dependant on the scale of the dataset or live input space used.

In a real or live dataset, the accuracy of raw depth measurement will decrease with distance from the camera. This is due to hardware limitations of the cameras. In this case, whilst a large enough cut off is needed to allow enough points for tracking, it must also not be too large to introduce too many camera inaccuracies.

For ICL-NUIM it can be expected that the depth will be equally accurate at any distance as it is in artificially measured dataset - therefore an infinite depth cut-off

would be ideal for accuracy in this dataset.

The range is defined to be 0 to 10m for this design space exploration, due to the datasets being used having scale lower that this. This parameter would need to be altered if much larger scenes were being considered.

The range for this dataset should also be continuous, because the most interesting points for consideration are likely to congregate around the absolute distance from camera to the edge of the scene.

9.3.2 ICP / RGB tracking weight

Relative ICP/RGB tracking weight

Range: [0 - 100%] (0.0 - 1.0 in paper)
Default : 10 % (0.1 in paper)

This parameter is the weighting between geometric and photometric error for calculating tracking error. A weight value of 0 indicates that only photometric (RGB) tracking should be performed, whereas a value of 1 indicates that only geometric tracking should be performed (ICP). Values between these two boundaries represents the joint cost function: RGB+ weight * ICP.

9.3.3 Confidence threshold

Surfel confidence threshold

Range: [0 - 12]
Default : 10

This parameter is used by the splatted rendering shaders during fusion to cut off surfels beyond the confidence threshold. It is used:

- When predicting the model
- When fusing a frame into global model
- When synthesizing depth

9.3.4 Thresholds

These threshold values correspond to thresholds for tracking transform acceptance or fern database acceptance thresholds.

We chose not to delve too deep into these parameters to make analysis of the results of the design space exploration easier i.e. to avoid the "wild goose chase".

- **-cv** : Local loop closure covariance threshold

  Default : 1e-05

  This is the covariance threshold to decide whether to accept a model-to-model incremental transform i.e. the transform covariance when calculating a local loop closure.

- **-ie** : Local loop closure residual threshold and -ic : Local loop closure inlier threshold

  Default ie: 5e-05
  Default ic: 35000

  ICP error must be below the residual error threshold, and the ICP inlier count must be above the inlier residual threshold during geometric tracking.

- **-pt** : Global loop closure photometric threshold

  Default : 115

  When matching to a frame in the fern database for a global loop closure, the matched frame photometric error must be lower than this threshold value.

- **-ft** : Fern encoding threshold

  Default : 0.3095

  This threshold decides whether or not to add a frame to the fern database.

9.3.5 Modes

Fast Odometry, Pyramid Levels, and so3 Pre-Alignment

Range: [True, False]
Default Fast Odometry: False
Default Pyramid Levels: True
Default so3: True

These three parameters are all involved in the tracking steps of ElasticFusion (the incremental transform step), and are best explained with the help of a diagram:

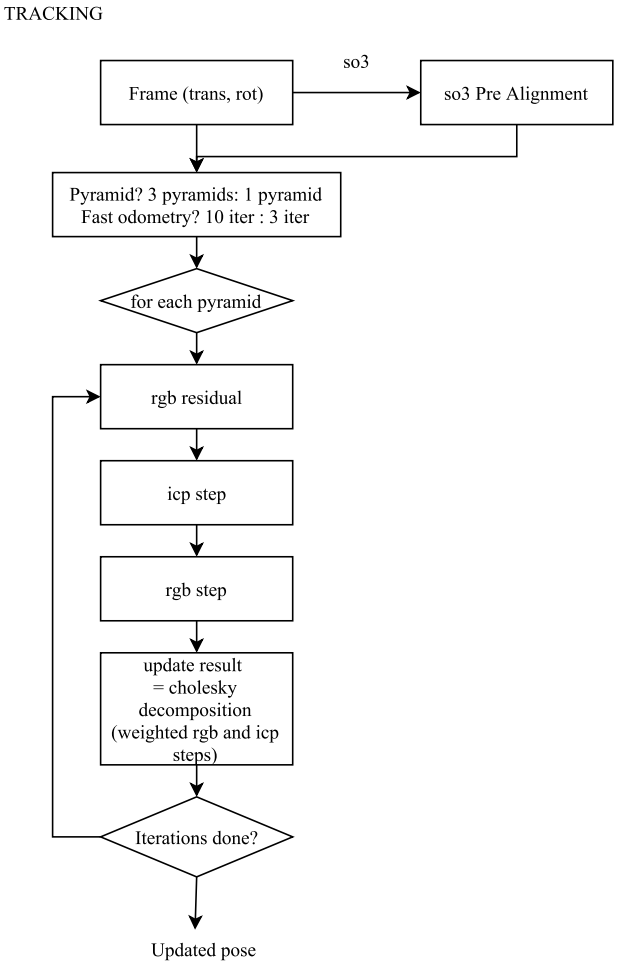


Figure 9.1: A block diagram of the ElasticFusion tracking algorithm

Figure 9.1 outlines the key steps to find the incremental transform between two views of the scene e.g. in current-frame-to-model, or model-to-model tracking.

The first parameter we see is the so3 Pre-Alignment step, which is performed before the pyramid stages of the transform. If the parameter so3 is disabled, this step won't be performed.

We then see the next two parameters, the fast odometry option and the pyramid option.

The pyramid parameter controls whether or not to use a pyramid for tracking. If this is set to false, then a one level pyramid is used. If it is set to true, then three pyramid levels are used. Three is the magic value used in many vision applications. This parameter could be opened up to make it more configurable, but to avoid over complicating the analysis this will be done later if the results indicate that interesting values could lie between, or beyond, 3.

Similarly, the fast odometry parameter alters the number of transformation iterations per pyramid level. If fast odometry is enabled, the current translation and rotation is iteratively re-calculated only 3 times instead of the default 10 times. These values are not explained by Whelan et al., and whilst we run with them for now, it again may be interesting to generalise this parameter as well to see if more interesting values are in between these two hard coded values.

Open loops

Range: [True, False]
Default : False

Open loop mode is the absence of closed loop mode, i.e. it disables the local loop closure code in ElasticFusion.

Re-localisation mode

Range: [True, False]
Default : False

If this mode is enabled then ElasticFusion attempts to relocate its pose, i.e. 'get back on track' if it is lost.

Each frame if the camera is not lost, it will check the covariance of the frame to the current model. If this is over some threshold for 10 frames in a row, then it declares it lost. If it's already lost, and if the current pose has been set to the recovery pose, then the covariance is checked to see if this recovery fixes the covariance and brings it back on track.

Each frame a recovery pose is calculated as an alternative to the current pose. This is set to the current pose if the camera is lost. No more actions will happen for each frame aside from recovery if the pose is lost.

N.B. the camera cannot be 'lost' without re-localisation mode enabled. Therefore, frames will continue to try to be integrated and the pose will go haywire.

Chapter 10

Design space exploration

10.1 Finding interesting parameters

The first step to finding interesting configurations is to find parameters that have some effect on a metric. There are three main metrics we are considering here; accuracy, power and time duration. A parameter may affect only one of these, and have little effect on the others. In many cases, changing a parameter may have a positive effect on one metric, and a negative effect on another.

10.2 First attempt

This section outlines a small experiment that was tried before moving onto more accurate design space exploration methods.

The first idea was to try was changing only one parameter at a time to find interesting values for parameters. The aim would be to then be to run experiments on combinations of the values found in the previous step. The design space reduction would come from losing any bad parameter values early - but good ones may be lost too.

Advantages of this method:

- Can quickly identify strong values for parameters

Negatives of this method:

- Lots of bias
- This method doesn't allow parameters to interact. Parameters may have interesting values only when used in conjunction with each other, and this method would wipe these out early.

Results

Although this method was not great for an overall design space exploration because it doesn't scale well, it was useful for defining good parameter ranges before moving onto more sophisticated methods of design space exploration.

An example of this is the value of the ICP-RGB weight parameter. We could see from these experiments that a very low or very high ICP-RGB weight value was preferential for a higher accuracy in ElasticFusion, i.e. fully geometric tracking or a small amount of photometric tracking.

10.3 Machine learning

10.3.1 Random sampling

Initial random data is needed to feed into the machine learning stage. This is generated by picking a random value from the defined range for each parameter to create a random configuration. This can be used as a method of design space exploration itself, where one would hope that with enough samples that good configurations may be found. This is used quite heavily in the later section involving experimental evaluation, as a benchmark to compare machine learning against and is evaluated on its own merit.

10.3.2 Active Learning

In active learning, points are added each iteration to improve an internal estimator.

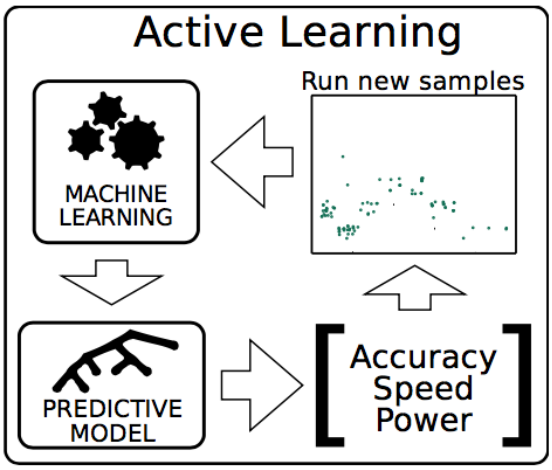


Figure 10.1: The active learning paradigm predicts new samples, runs these samples and retrains using them

Figure 10.1 shows the process the active learning takes - random forest regressor is used to create a predictive model of the pareto front, it then predicts new configurations which should have a good performance, runs these samples and the process repeats.

The active learning iterations of the design space exploration code are based on a random forest regressor. A random forest regressor is essentially a multi-level decision tree. Randomisation is also added here to avoid over-fitting in the training stages.

Predict all the possible points using the random forest which was trained from the last set of samples. The results are predicted with the metrics of trajectory accuracy

and total execution time. We then run these samples to see how well they performed, and repeat the training of the regressor.

New samples are created using a variance reduction technique or by choosing points that are close to the pareto front. Using a variance reduction technique is useful to create a model which we can then later use to predict points that will be on the pareto front with high certainty. This is most useful for creating a very accurate model of the pareto front, otherwise we will not be able to easily predict points that will lie up on it. Variance minimisation will cause us to lose the outlier points which have very high performance and very high accuracy if they don't minimise variance on the pareto curve, but we are interested in these points so we can see which parameters interact to cause this high performance. Therefore machine learning whilst optimising for a really low ATE and really low time by choosing points close to the pareto front is best from a software performance optimisation.

10.3.3 Integrating ElasticFusion

To perform a design space exploration of ElasticFusion the design space must be defined.

In the current work by Luigi Nardi, the design space is KinectFusion. This defines various algorithmic parameters such as the volume resolution or the compute size ratio. To allow exploration of ElasticFusion, the code must be altered to allow different SLAM algorithms.

This integration involves redefining the parameters of the SLAM algorithm, defining the ranges that these parameters can take, and modifying the SLAM code to generate correctly formatted output logs. To allow for the design space exploration of the TUM datasets, we integrate code to calculate the ATE for asynchronous ground truth data files.

ElasticFusion output

To allow ElasticFusion to work with the active learning code, we added a parameter to ElasticFusion to allow the optional printing of parameters and statistics. This format of log file is required to correctly create the .data format used in the rest of the DSE code.

The parameter is called partitioning mode, with flag -part, to represent the work on the `reporting_updates` branch of the `peach` project which outputs KinectFusion in `partitioning` mode. Having the parameter for ElasticFusion allows the log file to be outputted normally for other uses without any changes from the default.

The format of the expected log file output:

- Title: `SLAMBENCH Report %time`
- Scene properties: camera, input-file etc.
- Algorithmic properties: parameter values
- Compilation properties:
- Hardware properties:
- Statistics: pose graph, profiling data, power readings etc.

10.4 Design space exploration results

The main aim of design space exploration is to find better results.

Therefore we must evaluate how efficient the design space exploration method is, i.e. how quickly can it get the best performance results? We want to find good results, but we also want to find them efficiently. We also want to find out how good the results we are achieving are.

We must also look at the results, find good configurations and analyse why these configurations perform best. To analyse these results we must find patterns and correlations, and use reasoning from a computer vision perspective to gain insight into why the results perform best so we can achieve good performance on new datasets.

10.4.1 Analysing a single configuration

When running design space exploration we need to think about what a success actually is. What is a good result?

The first place to look is points on the pareto front. These are the multi-objective optimal points. This is ideal when we are looking for the *best* results. However, there will be a lot of points that are almost on the pareto curve which should be considered when we evaluate the efficiency of the design space exploration technique. These are 'good' points.

A simple way of defining 'good' is those results which fall underneath certain threshold. We could set a hard limit, e.g. ATE $<$ 5cm or execution time per frame $<$ 20ms. This means the SLAM algorithm is performing to a high standard.

Another way of defining 'good' is any results that are better than the performance of the default configuration. If the default configuration performs as an ATE of 6cm, and the execution time per frame is 22ms - any results that perform better than this are interesting because they can become new defaults, which is important as non-expert users would not touch the default configuration themselves.

10.4.2 Evaluating the method

Things we are interested in :

- **Efficiency of finding good points**

  We can compare different methods to each-other as long as we make the evaluation fair.

  A benchmark to evaluate efficiency against:

  To analyse the performance of a new design space exploration technique, we can compare it to random sampling.

The machine learning technique uses the active learning paradigm, which uses random data as an input. So to fairly compare random sampling with machine learning, we must compare the results of:

- Random Sampling + Machine Learning (with random sampling input)
- Random Sampling + More Random Sampling

We would then hope to see that the machine learning technique performs better in comparison to the additional random sampling.

- **How good are the points we are finding**

  This is concerned with finding the best points. It can be comparative (i.e. which method gets the very best points for accuracy or performance). It can also be of merit as a comparison against the algorithm itself, analysing the speed-up or accuracy improvement against the default configuration.

Methods for the analysis of efficiency and success of design space exploration techniques:

- Percentage of good/bad results found

  We can measure the proportion of good results found from the technique. This will give a good idea of how many good configurations we can expect to get, and how many bad configurations we can expect to avoid.

- The number of pareto points

  This is not really a good metric. Having lots of pareto points is not necessarily a measure of success. One really good configuration can dominate n other pareto points.

- The number of good points over time

  A better method's graph would grow faster than a poorer method. Both metrics will continue to grow

  However, whilst this graph will show efficiency of the method, it will hide the best results achieved. We wont see if the method

- The best accuracy/performance achieved over time

  Here we can plot the best result found so far for a specific metric, over time (or number of experiments). E.g. We can plot the best ATE result found at each new configuration. For 1000 random samples, we would plot the best result we have found after 1, 2 ... 1000 samples.

- If we see one technique getting to good results faster than the other - then we show its more efficient
- If we see one technique rise higher than the other - we show we can get better results

However this technique is not without problem; it is pretty dependant on the data. If we get lucky on the random sampling it will look like the active learning doesn't perform that well in comparison, therefore we would ideally plot this over multiple different design space explorations to see a more general trend.

10.4.3 Finding the good configurations

Although the actual configurations are easy to see, as they are output on the pareto front, they must be understood to gain any insight into optimising the algorithm for different datasets and improving the algorithm. However, this is not necessary if the user simply wants to find the best possible performance parameters quickly without understanding why they work so well.

Potential Challenges

1. Magnitude of results

   Too few good results make it hard to identify any patterns at all, whereas too many results can hide trends. The first issue here can be resolved by taking a good amount of random samples, e.g. 2000+, and using efficient design space exploration methods such as an active learning approach. The second issue can be resolved by using statistical analysis methods as we will see below.

2. Finding patterns in the data

   It might be hard to see coherent patterns in the results, for example if all of the pareto front configurations do not obviously correlate to the human eye. Techniques such as PCA could be employed to identify key parameters which make the strongest effects on the data. An example of the usage can be seen in Figure 10.2. Using PCA is outside the scope of this project, however we discussed the merits of using it and is a useful tool to highlight to the reader.

3. Convex / Non-Convex Optimisation Space

   There may be multiple different ways to get to the best performance configurations. This means the space is non-convex, as opposed to convex where there is one optimal solution, making it more complicated to map out the directions to the best performance. This is made especially significant when we consider the design space exploration holistically, as if we are to use different architectures, datasets, or number of cameras, we will have different optimal configurations here too; adding another dimension to the complexity of the problem.

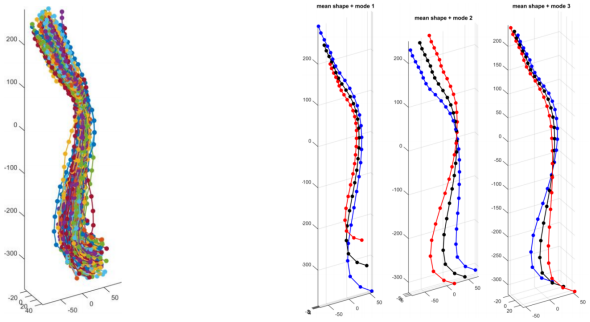


Figure 10.2: Example of PCA used on spine shape data to identify the key principal directions in spine curvature. This could be employed to identify the key algorithmic parameters in SLAM. Diagram by Dr. Ben Glocker [19]

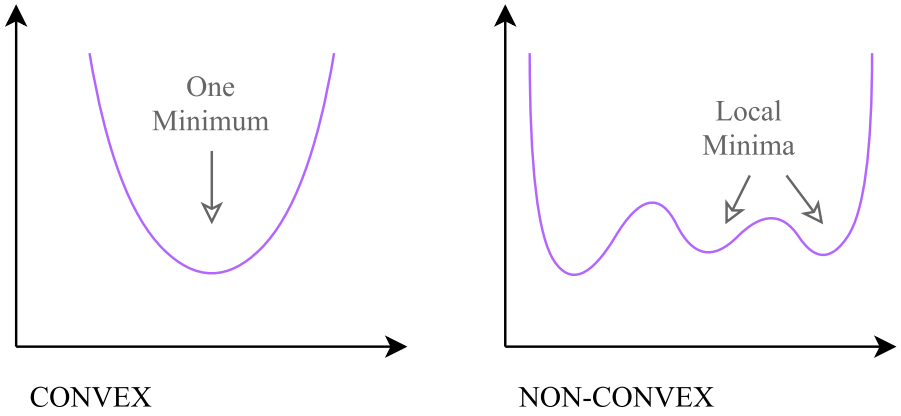


Figure 10.3: The first image shows a 2D convex space - it has one optimal solution. The second image shows a non-convex space - it has multiple optimal solutions and local minima. In 3D, these curves would simply be surfaces, the idea is the same.

Once the configurations are understood we must delve into what this means for an algorithm. The analysis is multi-objective - therefore there does not exist one unique optimum point. However we can look for the trade-offs between parameters and how they affect points on the pareto front.

Part V

Experimental Evaluation

Prelude

In this evaluation section I report the findings of my project, and analyse the results of the design space exploration. We start by analysing the default configuration of ElasticFusion to gain insight into where the bulk of the execution time is spent (Chapter 11). We identify the key algorithmic parameters of ElasticFusion, explaining why they make such an effect and explaining why the non-key parameters don't (Chapter 12).

We then present the results from performing a random sampling design space exploration on multiple datasets, evaluate this merits and downfalls of this method and suggest some simple improvements that can be made to random sampling to greatly improve the results if there exists some understanding of the design space (Chapter 13).

We then present the results of performing the active learning design space exploration method. We show how this method is more efficient than the random sampling method, and perform a design space exploration on both a synthetic dataset and real-world dataset (Chapter 14).

We map the Pareto efficiency front over the metrics of trajectory accuracy and execution time, generating better results than the default configuration of ElasticFusion. We can achieve almost 2x better performance on both the synthetic and real-life datasets, and can improve the trajectory accuracy on the synthetic datasets by 2x also. We explain the results, showing why these speed-ups and accuracy gains are possible, identifying possible issues in the ElasticFusion code leading to these poor results.

We identify some trade-offs that can be made between the ElasticFusion parameters, compromising either on performance or trajectory accuracy for a gain in the other.

10.5 Platform

These experiments are run on a desktop with 8 CPU cores using a very fast nVidia GPU. For a more detailed specification of the experiment environment see Figure 10.4.

Machine name	**graphic08**
Machine type	Desktop
CPU	Intel E5-1620 v2 - Ivy Bridge
CPU cores	8
CPU GHz	3.70
GPU	nVidia GeForce 780 Ti
GPU MHz	875 - 928
CUDA Cores	2880
Language	C++, CUDA, OpenGL
gcc version	4.8.4
OpenGL version	1.4
CUDA Toolkit version	7.5.18
Ubuntu OS (kernel)	14.04.4

Figure 10.4: Evaluation Machine

10.6 Datasets

In the following experiments the default dataset is the ICL-NUIM Living Room 2 dataset. If another dataset is used, it will be specified. The first 400 frames of the ICL Living Room 2 trajectory are used in the following experiments. This is because half of the sequence is representative enough to be used[18] - cutting down the experiment time in half.

Chapter 11

Default Configuration Results

ElasticFusion provides a set of default parameter configuration values to use. To have a good base point to evaluate the design space exploration findings against, an analysis must be performed of the performance metrics under the default configuration of ElasticFusion.

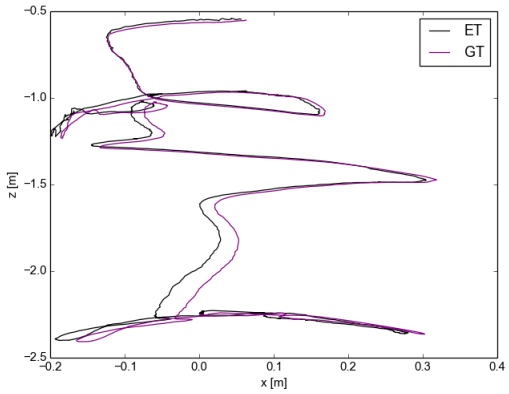
```
Default Configuration

Parameter              Value

confidence            : 10.0
depth                 : 3.0
icp                   : 10.0

so3 pre alignment     : 1.0
enable_relocalisation : 0.0
close loops           : 1.0
frame_to_frame_rgb    : 0.0
fast_odometry         : 0.0
```

11.1 Accuracy

Firstly, we investigate the accuracy of the datasets using the default ElasticFusion configuration. The trajectory accuracy is evaluated using the ATE, and the mean used is RMSE.

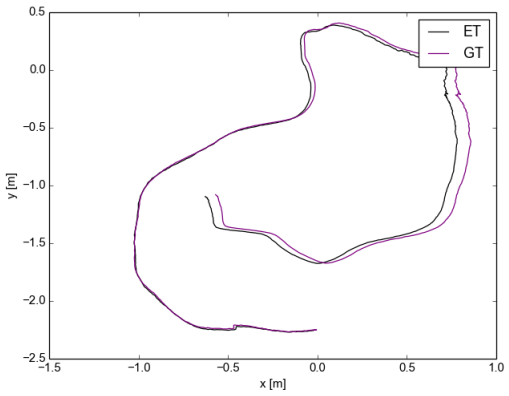
The following are the results for evaluating accuracy on the ICL-NUIM and TUM RGB-D datasets.

ICL-NUIM Living room 0



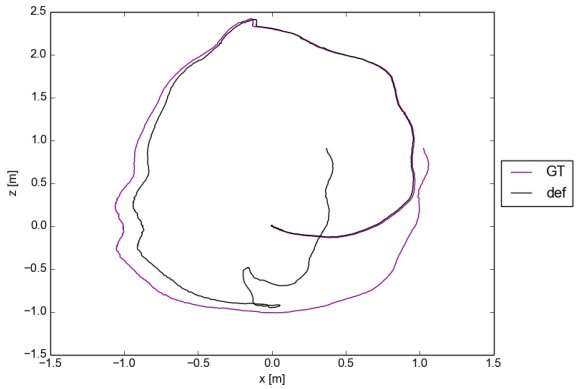
ATE RMSE: 0.0304m

ICL-NUIM Living room 2



ATE RMSE: 0.05415m

ICL-NUIM Living room 3



ATE RMSE: 0.2897m

ICL-NUIM Living room 3

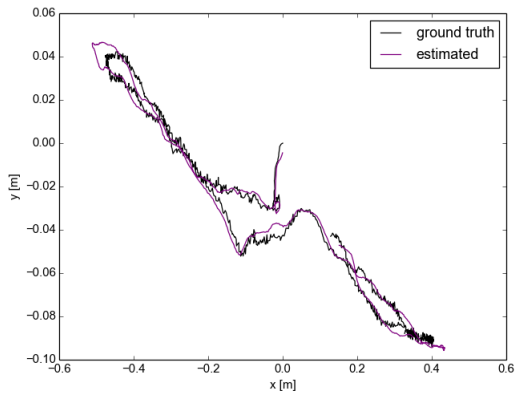
ElasticFusion performs poorly on this dataset; which is consistent with the experimental findings from their paper. This result is actually good. As we can see from Figure 11.1 from the ElasticFusion paper many SLAM implementations it compared against perform badly on this dataset.

Algorithm	ATE
DVO SLAM	0.152m
RGB-D SLAM	0.433m
MRSMap	1.090m
Kintinuous	0.355m
Frame-to-model	0.243m

Figure 11.1: Performance of other SLAM algorithms on the ICL Living Room 3 dataset reported from the ElasticFusion paper [**?**]

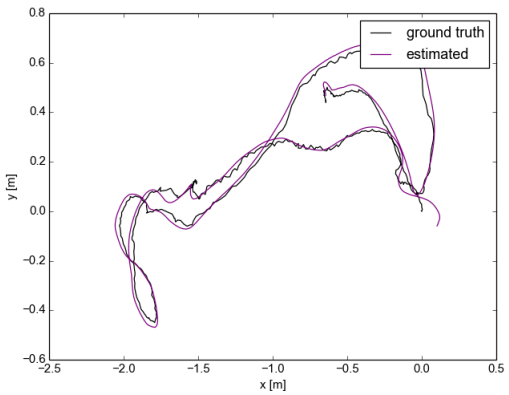
Initially I thought this result was strange, because there is a loop in the dataset, and ElasticFusion is meant to perform well for loop closures. However, it performs better because it does do loop closures - the difficultly of this dataset arises from the closeness to the wall and sofa, not because of the loop.

TUM RGB-D FR2 XYZ



ATE RMSE: 0.01744 m

TUM RGB-D FR1 DESK



ATE RMSE: 0.0406m

11.2 Performance

Total time per frame (ms): 22.5

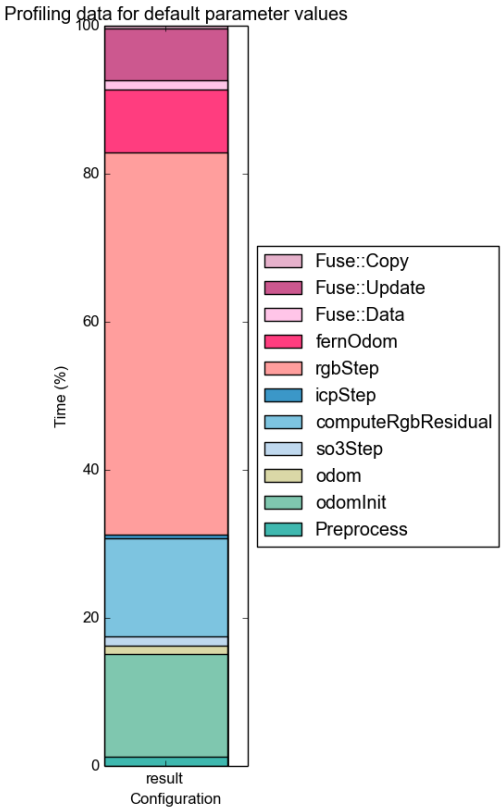


Figure 11.2: This graph shows what proportion of time ElasticFusion spends in each part of the code

Figure 11.2 shows the percentage of time spent in each of the kernels.

ElasticFusion spends a total of 12ms in these parts of the code, which makes up a fair proportion of the total time per frame (as not all parts of the code are tracked for performance times).

From the graph, the dominating times come from the tracking code, mainly the photometric tracking (rgbStep), with the geometric tracking (icpStep) and the odometry initialisation also quite significant. The fusion of the frame into the global model also takes a significant proportion of the time.

Seeing this data allows us to know which parts of the code dominate execution for different configurations, so we can make better decisions about what parameters we can configure to improve our performance.

Chapter 12

Algorithmic parameters

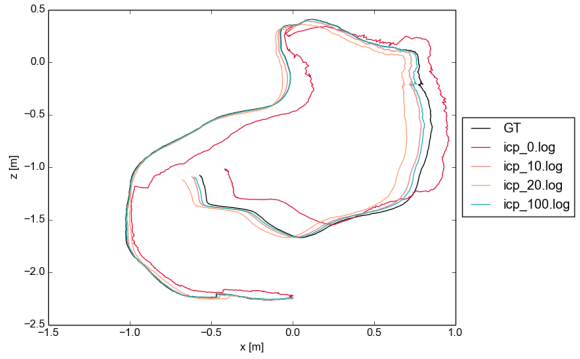
In this chapter, we discuss the key algorithmic parameters and explain why they have an effect on the performance and accuracy of ElasticFusion. Primarily, these are the parameters of ICP-RGB weight, depth cut off point and the parameters related to tracking pyramid iteration amounts. We also explain why some parameters have little or no effect and when we might expect to see these making more of a difference.

The following experimental data is on the ICL Living Room 2 dataset primarily, with results from the two TUM RGB-D datasets also.

12.0.1 Effect of ICP_RGB weight parameter

Accuracy

Close ICP-RGB weight values:



Mid level ICP-RGB weight:

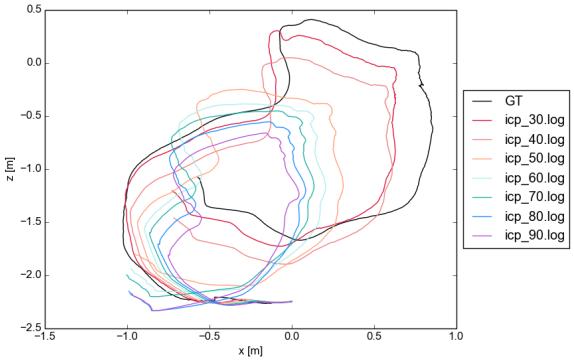


Figure 12.1: Plots of the trajectory for different values of the parameter ICP-RGB weight using the ICL Living Room 2 dataset. At 0 we have fully photometric tracking, and at 100 we have fully geometric tracking. Values in between are increasing amounts of geometric tracking approaching 50:50.

This parameter corresponds to:

- ICP 0 Fully Photometric tracking (because 0% ICP weight means full RGB)
- Other Increasing weight to geometric tracking finalising at about 50:50
- ICP 100 Fully Geometric tracking (because 100% weight given to ICP)

N.B: It is confusing in the paper as the weight is expressed in terms of RGB, whereas in the code it is expressed in terms of ICP. The parameter is called `'icp'` *in the ElasticFusion GUI and code, but it actually would be more suitably named as* `'icp weight'`*, and in the paper we see the parameter to referred to as* `'rgb weight'`*.*

For the ICL-NUIM Living Room 2 dataset the ICP-RGB weight of 0 performs okay on ATE but the accuracy is poor - we have a very spiky estimated trajectory. The weight of 0 corresponds to a fully photometric tracking, no geometric tracking is performed here.

The ICP-RGB weight of 100 corresponds to full geometric tracking, which actually performs best on this dataset. We believe that this is due to the living room dataset being uniformly coloured with minimal texture, so the photometric registration is less accurate.

The default value of weight 10 (corresponding to photometric tracking + 0.1* geometric tracking) also performs well.

Here, the algorithm benefits from being either at one of the two extremes. Either nearly all photometric tracking, or no photometric tracking at all. The weight values in between the two extremes gradually decline in accuracy as more weight is given to geometric tracking.

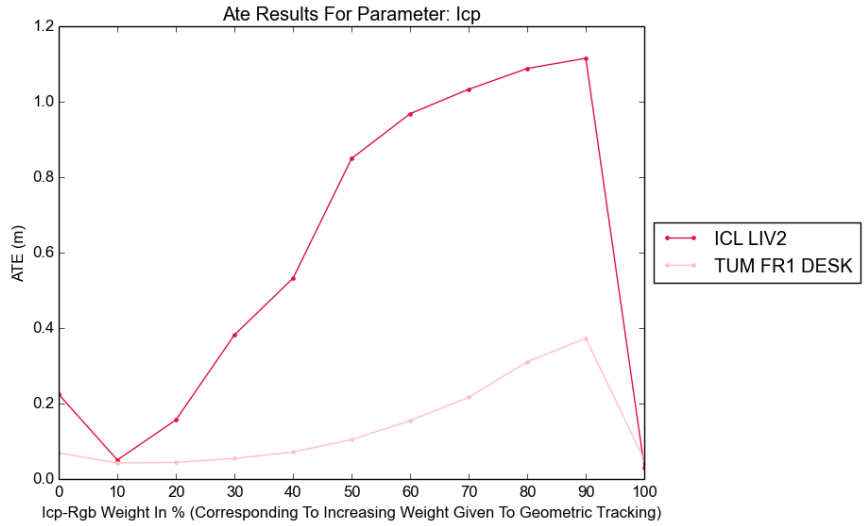


Figure 12.2: This graph shows the results on ATE for different ICP-RGB parameter values on the ICL LIV 2 dataset and the TUM FR1 dataset

The values in between are weighted as RGB + weight * ICP which would mean as weight approaches 100 the weighting between ICP and RGB is 50 50. However, once it actually gets to 100 there is a catch in the code which stops ElasticFusion performing photometric tracking making it 100% geometric tracking. This is a bit strange because it does not follow the formula in the paper at the two extremes.

TUM RGB-D Dataset:

Figure 12.3: The left is an example frame from the ICL-NUIM Living Room scene, the right is an example from the TUM real-life warehouse scene

Very different results can be seen for the real-life dataset of TUM. The living room dataset is more uniform than the TUM dataset, with much less colours - which explains the poor results of the ICL dataset with higher focus on the photometric weights.

It makes sense the performance decreases as the ICP-RGB weight approaches 100. This is because we end up with a 50:50 weighting between photometric and geometric tracking which would cause competition between the two techniques and dilution of the good decisions. Areas which "Save" the photometric tracking , e.g. such as the picture on the wall or the plant in the synthetic dataset, will not be able to be used in full effect because of the geometric influence which is also weakened.

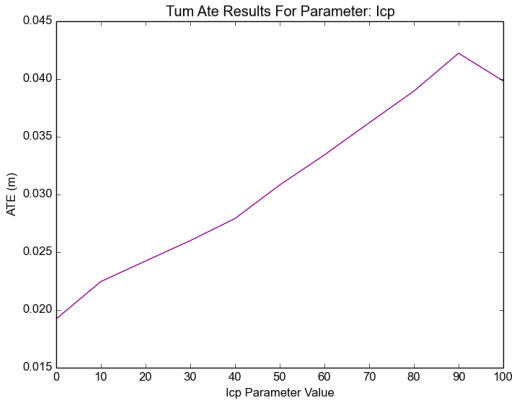


Figure 12.4: Here we see the effect of the ICP-RGB parameter on the TUM FR2 XYZ dataset. Here we note the scale on the ATE, even the worst accuracy value is under 5cm, whereas the other two datasets are both over 30cm and 1m.

We also analyse the results on the TUM FR2 XYZ dataset, and see these results in Figure 12.4. The accuracy difference is minimal here, even compared to the other

real-life dataset. This suggests that the difference in colour and texture of the scene is not the only contributing factor here.

The TUM FR2 XYZ dataset has a much smaller range of camera position than the TUM FR1 DESK or ICL-NUIM Living Room 2 dataset - the camera barely moves pose throughout the whole trajectory. This results in the alignment of the current frame with the model being much less prone to error because the pose is barely different between frames. This particular result is therefore unfair to compare directly to the synthetic dataset, however the TUM FR1 DESK and ICL-NUIM Living Room 2 have similar motions and are acceptable comparisons.

This results strengthens our conclusion that the results of design space exploration and algorithm analysis are dataset dependant. We see the dependence not only in the image of the scene, i.e. the density, the amount of colour, or the amount of texture, but also in the camera motions.

Performance

The total performance results can be seen in Figure 12.5, and the broken down performance results in Figure 12.6.

In this performance section we see that the lower times are at either of the two extremes. This is because at either two extremes the photometric or geometric tracking is completely cut out. All of the other timing data is essentially the same because the same amount of work is completed, the results are just weighted differently.

This suggests that we can chose to improve performance in tracking if we realise that purely photometric or purely geometric tracking works well enough on its own, otherwise this parameter is useless for improving performance - only good for improving execution time. Even a small weight such as 1% geometric tracking will perform the same amount of geometric tracking as a weight of 100%, which seems a tad wasteful!

In Figure 12.6 we can see that when the ICP-RGB parameter is at 100 (Here meaning that photometric tracking will *not* be performed), that there is still some execution taking place in the `rgbStep`. This is unsurprising, as the weight only affects the frame-to-model tracking. The model-to-model tracking is hard-coded to be at an ICP-RGB weight value of 10 (0.1) in ElasticFusion, so photometric tracking will still take place during the execution of the code.

We see the decrease in overall time spent in RGB tracking when we completely drop it though, so although it is not completely eradicated we do see improvement.

Summary

- The ICP-RGB weight affects ICL-NUIM more significantly that the TUM dataset due to the texture levels in the scene, and is a key parameter in ElasticFusion.
- Exclusively tracking geometrically or photometrically is the only way to improve performance for this parameter.

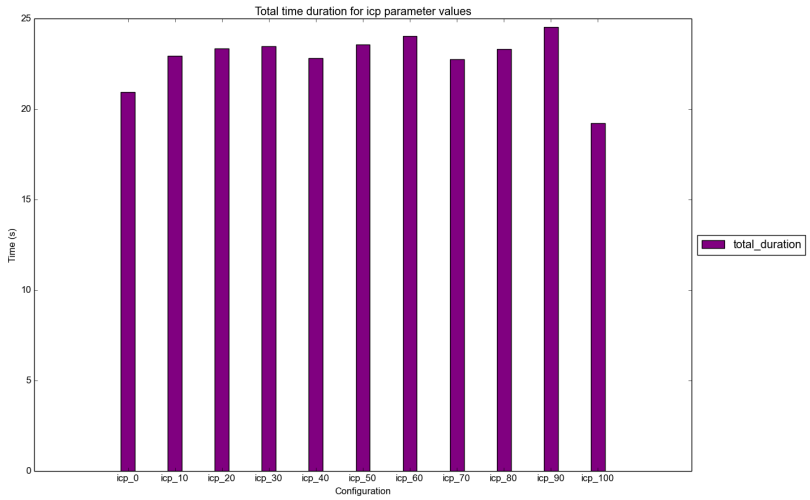


Figure 12.5: Here we see the execution time per frame results for different ICP-RGB parameter values. We see that at either two extremes we have a lower time, and all other parameter values result in the same execution time.

- Accuracy decays as the weight approaches 100, as the two techniques compete.
- Dataset choices can strongly affect the results of comparisons. It is important when comparing to pick trajectories of a similar motion, and when performing a design space exploration ensure to encapsulate as much dataset information as possible.

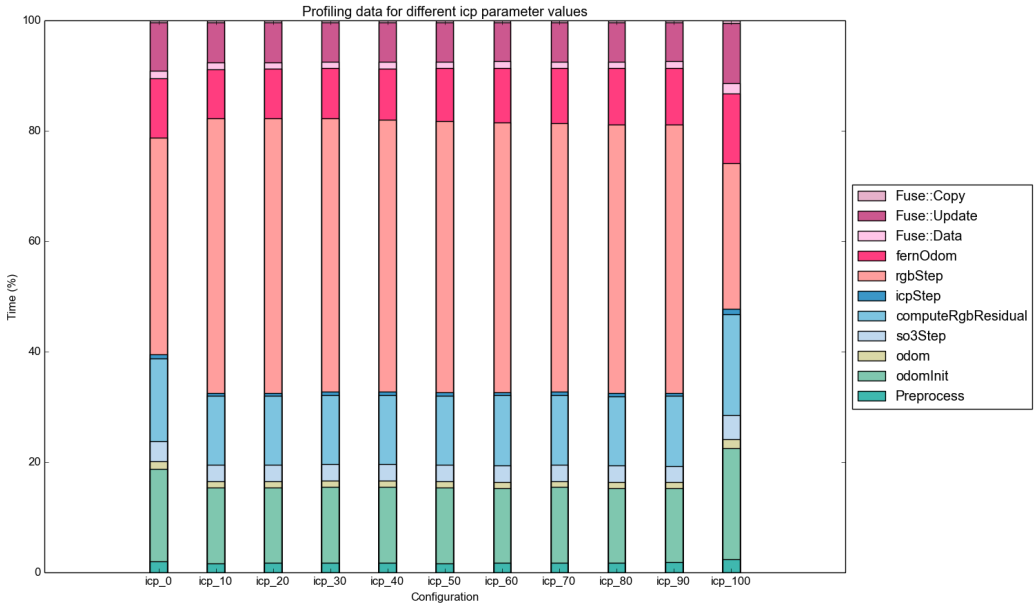


Figure 12.6: Here we see the execution time per frame results for different ICP-RGB parameter values split into kernels.

12.0.2 Depth Cut-Off

Accuracy

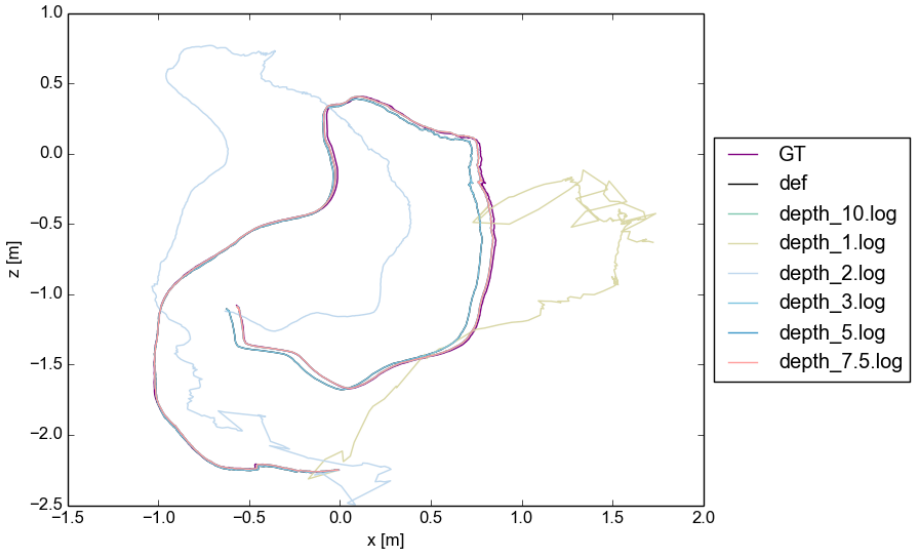


Figure 12.7: This graph shows the estimated trajectories for differing depth parameters, mapped down to 2D in the x and z planes.

The trajectories in Figure 12.7 are expected. Having only depth values less than one metre in a scene that is primarily at depths of 3-4m is going to cause extremely poor geometric tracking as we have lost most of the information in the scene. Poor pose estimates from poor tracking results in a poor accuracy. The ATE at depth 3 is pretty interesting from a performance optimisation perspective though - it results in a poorer accuracy than the larger depths, but it is still within an acceptable range and we see next that it saves on execution time.

Performance

In 12.8 we can see that the general trend for execution time per frame for depth increases as more of the depth frame is included. This is expected, because we are processing more of the frame throughout the algorithm.

We can see that after about 4/5m the time doesn't change too much, there are a few minor fluctuations, but we can see that at 3m the execution time is actually lower.

For very low depths we see a larger photometric tracking proportion of time, and a lower proportion of geometric tracking time.

For a very low depth, in Figure 11.2 we can see that a lot more time proportionally is spent in the rgbStep which is the photometric tracking step. This result is expected because the RGB images are not cut off when cutting off the depth frames. It is the

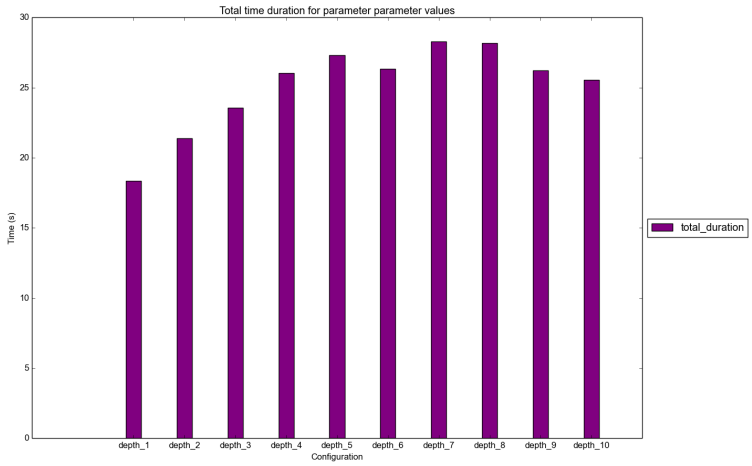


Figure 12.8: The total execution time per frame for different depth parameter values

geometric tracking part of the code which is affected by a depth cut off, and as this is lower, less time is spent here.

After the depth starts to stabilise after 4 meters we see that there isn't much change wrt. time in each section of the code because there aren't many extra values after 4m in the ICL NUIM Living Room 2 dataset, as this is the size of the living room. The actual amount of depth values cut off beyond 4m would be too minimal to see a real effect at all, let alone on a per-kernel basis.

Summary

We see for depth we can trade accuracy for execution time up to an extent, but the result isn't that huge. Depth is a very predictable result, but there's nothing wrong with this - expected results show that a SLAM implementation is doing what it should be doing.

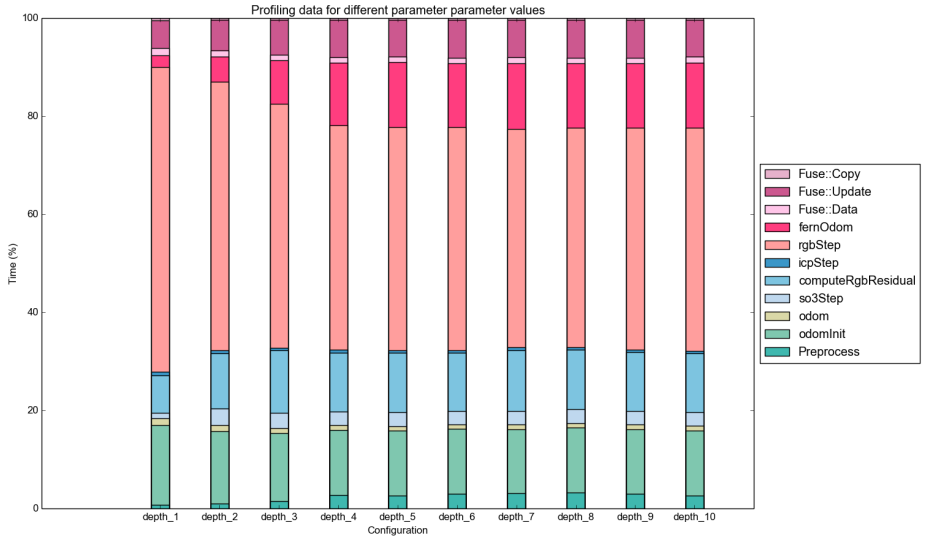


Figure 12.9: This graph shows the percentage of the execution time that is spent in each of the different parts of the code for different depth cut off points.

12.0.3 Effect of the Surfel Confidence Threshold

Accuracy

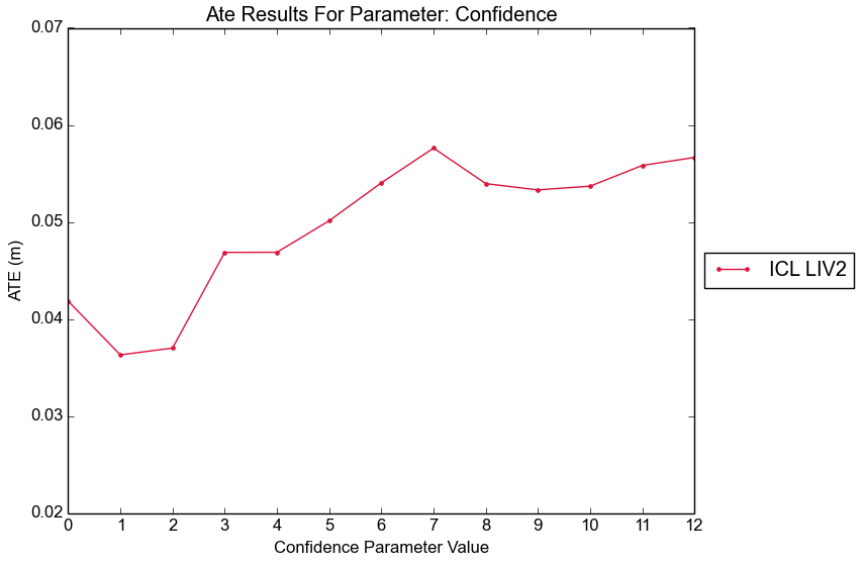


Figure 12.10: Here we can see the confidence parameter slightly worsening the trajectory accuracy as it is increased.

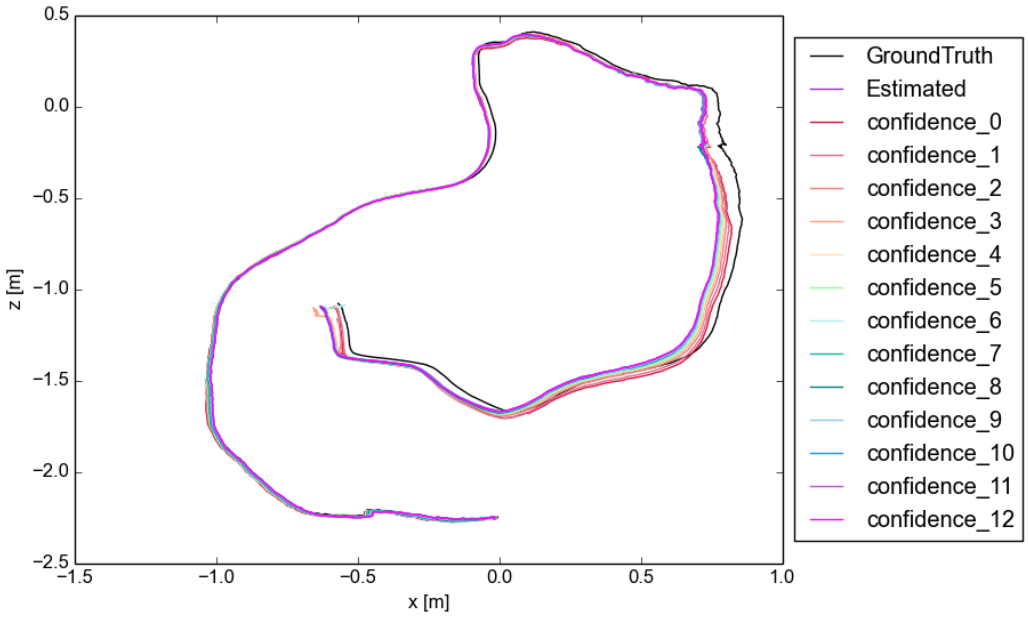


Figure 12.11: Here we can see the confidence parameter slightly worsening the trajectory accuracy as it is increased.

Performance

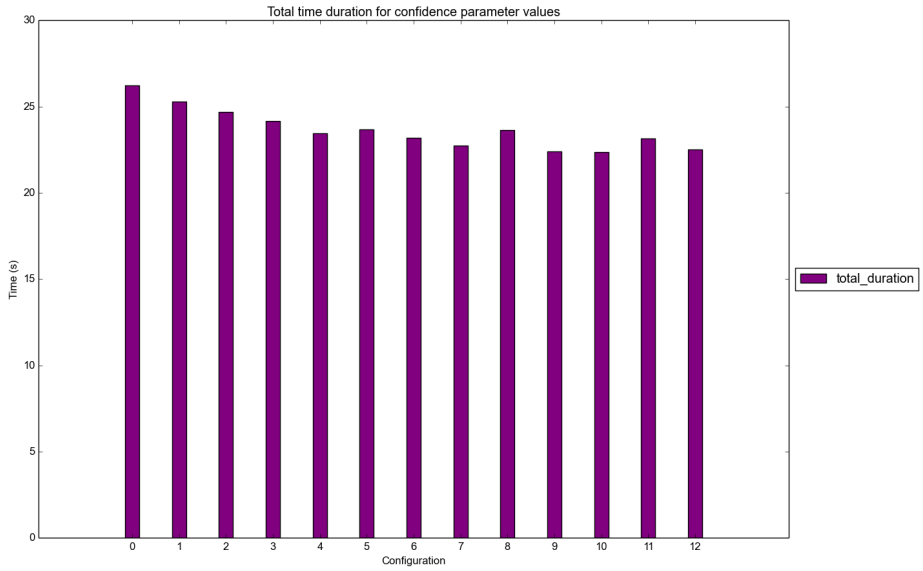


Figure 12.12: As we increase the surfel confidence threshold, we see the execution time decrease.

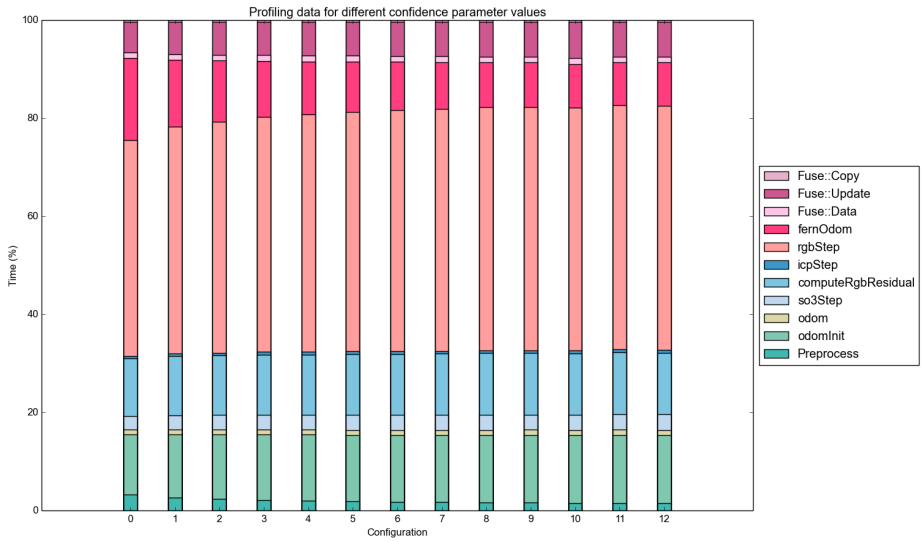


Figure 12.13: As we increase the surfel confidence threshold, we see less proportion of time being spent in the Global model fusion stages.

12.0.4 Modes

Fast Odometry accuracy results

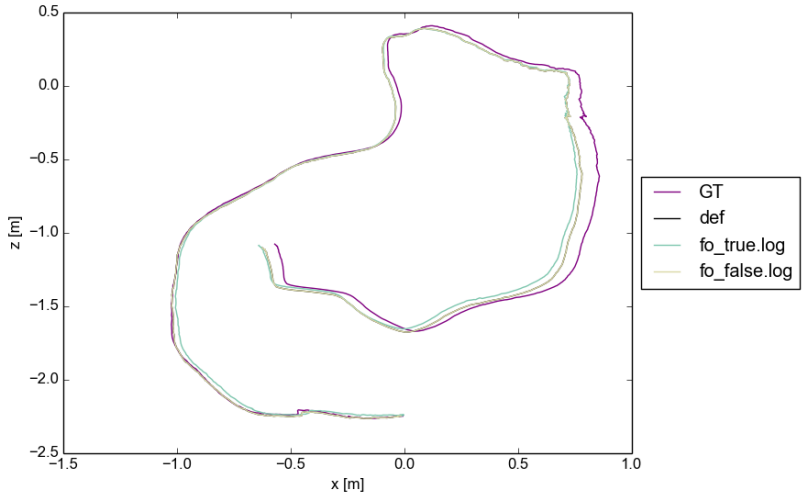


Figure 12.14: In this graph we can see the subtle difference in the accuracy with fast odometry enabled. It is slightly less accurate as we have performed less transform iterations.

Fast Odometry performance results

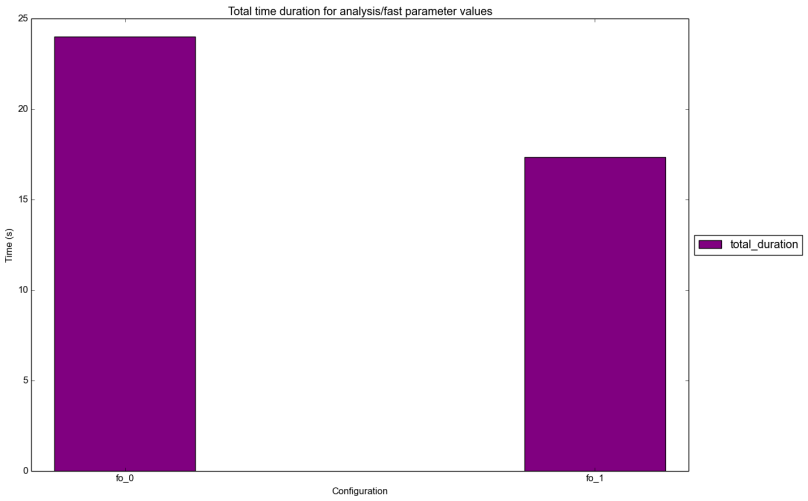


Figure 12.15: We can see that the performance is much better though. Improving from 23s to 17s.

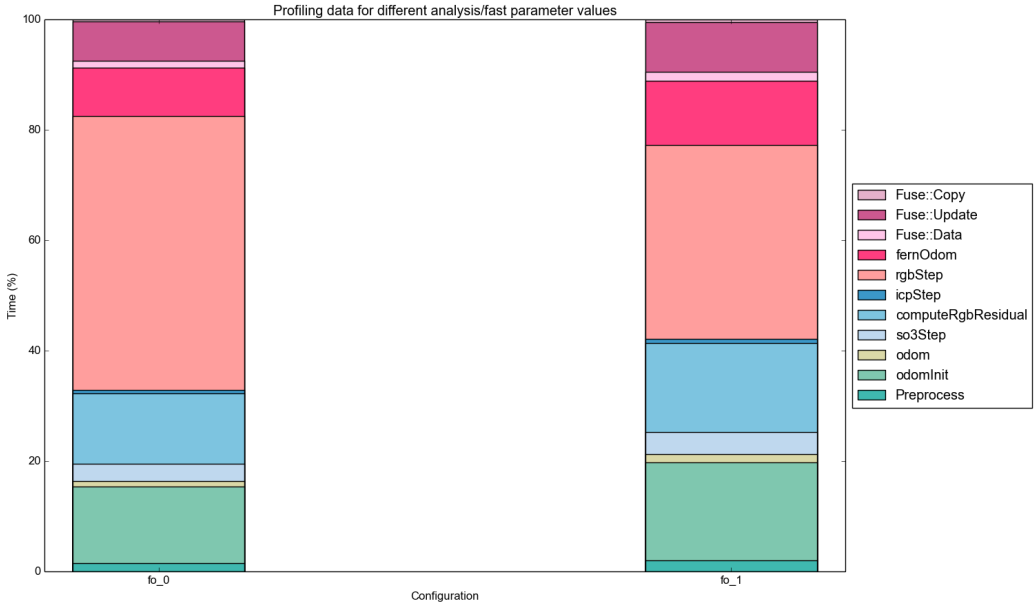
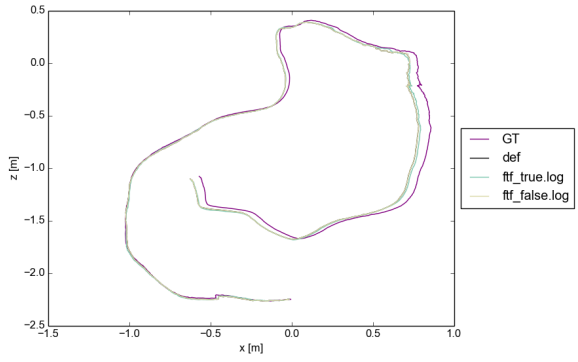


Figure 12.16: More in depth look into the fast odometry performance improvement

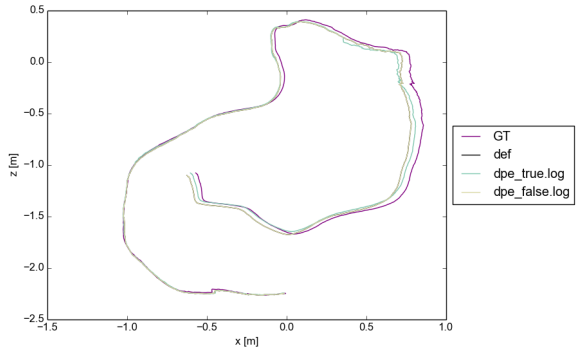
When we look deeper into the profiling results in Figure 12.16, we can see that a much lower proportion of the time is spent in tracking - this is because we perform fewer tracking iterations in the final pyramid stage. In the defualt configuration we perform $4 + 5 + 10 = 19$ tracking iterations, whereas here we will perform $4 + 5 + 3 = 12$ tracking iterations so an approximate 1.5x speed up per tracking per frame, and as the tracking dominates the performance results we see a big speedup overall.

Do frame-to-frame RGB tracking



In this graph we can see the subtle difference in the ATE, with Frame-To-Frame enabled the ATE is slightly more accurate.

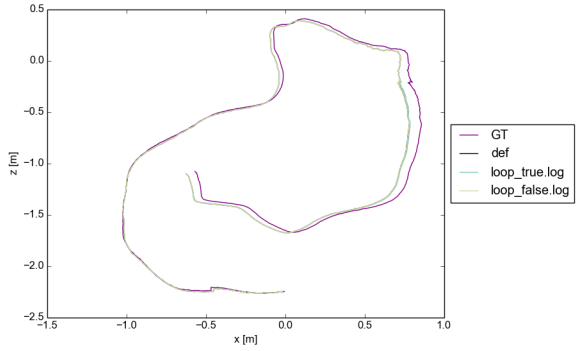
Disable SO(3) pre-alignment in tracking



Similarly, with so3 alignment disabled, the accuracy slightly decreases.

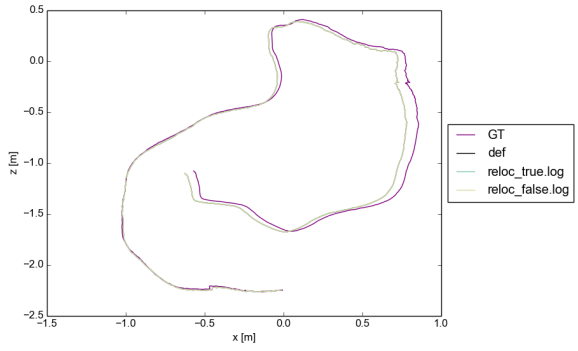
The next two metrics show almost no effect on the accuracy of the trajectory. This is because the following metrics are very dataset and configuration dependant. When changing these parameter with the default coniguration, nothing actually happens as a result of them.

Open loop mode



This is the parameter which controls whether or not loop closures will be performed. The reason that we see no difference here is that due to default configuration being pretty good - no local loop closures are actually needed so it doesn't affect the accuracy of the trajectory.

Enable re-localisation



Similarly here, even when re-localisation is on, it doesn't have a chance to improve the accuracy of the trajectory by getting the camera pose on track again, because it never loses track at all in the first place.

12.0.5 Summary

This is the summary of the algorithmic parameter effects on ElasticFusion for the ICL NUIM dataset representaed graphically in Figure 12.17. The results for the TUM datasets are almost identical, but with a less high impact on accuracy the ICP-RGB weight.

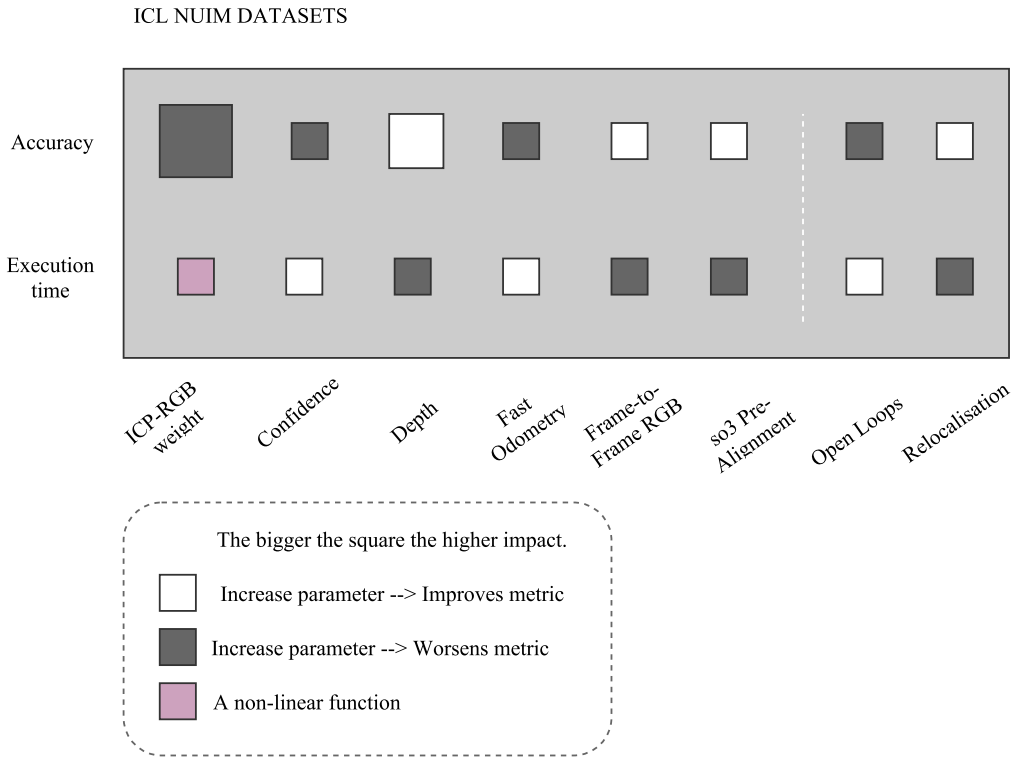


Figure 12.17: Summary of the effect of the algorithmic parameters on the ICL-NUIM datasets for accuracy and execution time. The Open Loops and Re-localisation parameters are sectioned to the side here - as we don't see a huge difference in our empirical data on the default configuration as no loop closures or frame re-localisations are needed regardless.

12.0.6 Conclusion

- In this chapter we have identified the key algorithmic parameters for ElasticFusion, explaining their effects and the significance of them compared to other parameters.
- We have seen how the ICP-RGB parameter performs very differently for different datasets, and how it wastes a lot of time executing both tracking techniques. There are diagnostic insights, one of the potential uses of SLAMBench.

  Diagnostic suggestions following on from the insights gained from this analysis:

  - Frame-to-Frame dynamic behaviour. The weighting between photometric and geometric tracking can change on a frame-to-frame basis depending on how much texture or angular information is in the frame. E.g. in ICL-NUIM we can increase photometric tracking around the photographs or plants, and reduce it across plain wall dominant frames.
  - If using one technique exclusively would be sufficient for accuracy, then the other could be dropped to improve performance. This could be performed as a pre-processing step - evaluating the information in each frame, for example dropping photometric tracking if we see a dataset such as ICL-NUIM which has little texure.
  - Alternatively, less iterations could be performed on the lower weighted technique so as to still contribute, and give the higher weighted, higher influencing technique a higher proportion of the processing time.
- We have shown how some parameters such as the depth cut-off and fast odometry (the pyramid iterations) behave exactly as expected, which are ideal for performance and accuracy trade-offs, as well as providing a SLAM developer confirmation that their algorithmic parameters are working correctly.

Chapter 13

Random sampling

The following chapter is concerned with the results of random sampling on ElasticFusion, on both synthetic and real-life datasets. These experiments choose random configuration values to try within the bounds of each parameter. We evaluate the effectiveness of these results, and suggest some improvements to basic random sampling that can be performed if there exist algorithmic insights, such stepping back and redefining parameter ranges at the design space analysis stage and re-exploring with a more limited range - filtering out pathological cases.

Trajectory Accuracy vs Performance

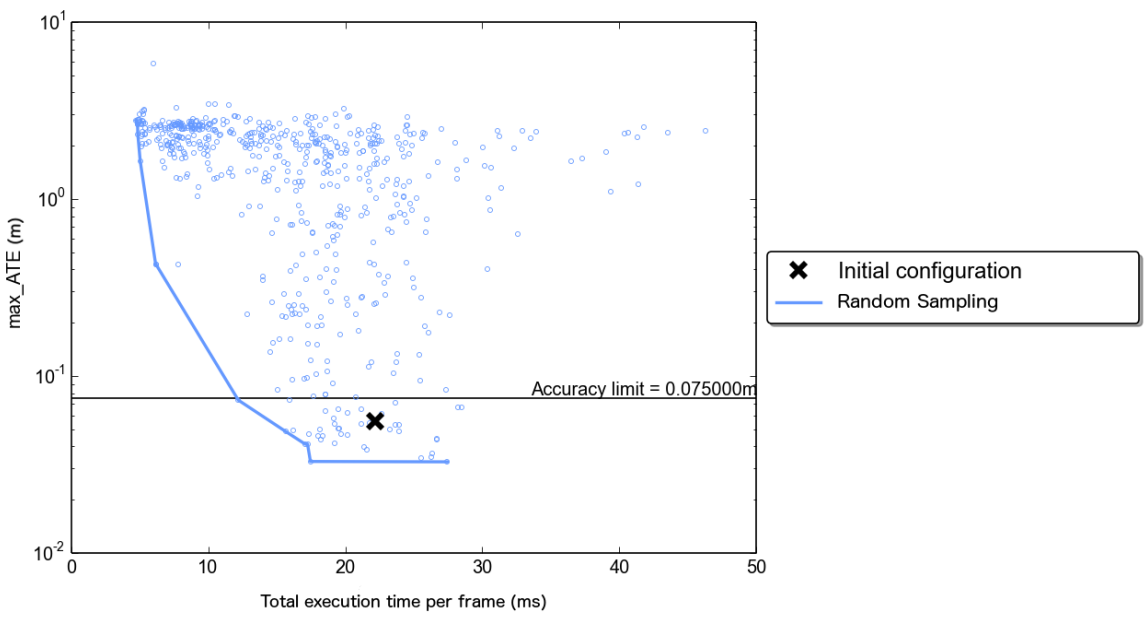
The following experiments evaluate the accuracy of the trajectory using the ATE calculated with a range of different average metrics, as well as measuring the total execution time per frame along with the execution time per kernel.

The following pareto graph plots the absolute trajectory error (m) against the total execution time per frame (ms) for all of the configurations run in each experiment. It then plots the pareto front of these results and marks the default configuration specified by ElasticFusion:

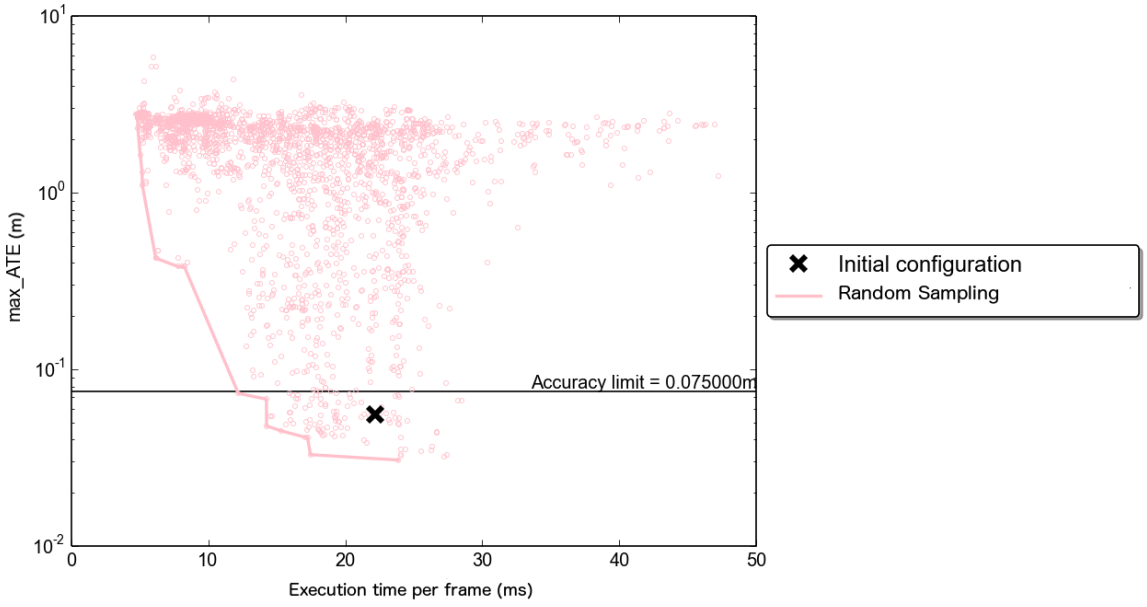
13.1 Basic random sampling

Dataset: ICL-NUIM Living Room 2
Parameter space: All parameters excluding thresholds. Confidence threshold, ICP-RGB, Depth cut-off, all 5 boolean parameters

Results generated: 500



Results generated: 2500



Analysis

- There are a lot of really bad results

  This is expected. The second graph shows that regardless of the number of random points, the results are still not very good for random sampling. Although there are a few more good configurations found, there is an even higher proportion of pathological case results. This shows that not much is to be gained by increasing the number of random experiments, and design space exploration would sophisticated method.

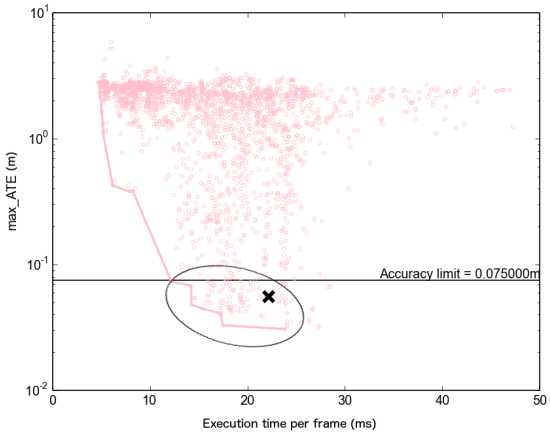
- Better configurations exist than the default configuration

  This isn't easy to find though as a huge proportion of the results are abysmal. Only 2% percent are within suitable range of accuracy, and 90% percent are over a metre inaccuracy (a lot of these are untracked results)

  It shows how it is difficult for a computer vision expert to manually fine tune parameters to find optimal configurations. Although a good configuration has been found, there do exist better ones.

Some preliminary algorithmic reasoning behind the poor results follow. This will be explored in more detail later.

- A huge amount of results linger around a very fast execution time with an extremely poor ATE. Whilst a few of these points will be on the pareto front, they aren't actually good configurations. There are not any interesting results here - as the algorithm simply loses track at this point (rendering the map and trajectory useless).



Interesting Configurations:

In this graph the the interesting configuration area has been circled in black. This section has points with both a low ATE and reasonable execution time.

- The ICP-RGB weight parameter can also cause really bad ATE results (explained in more detail later in section x). A low ICP-RGB weight is great, but as this parameter increases giving more weight to the geometric tracking, the pose estimates becoming increasingly poor.

- The re-localisation parameter, where tracking is attempted, causes a lot of low time results. When this parameter is enabled, if the camera is lost, the

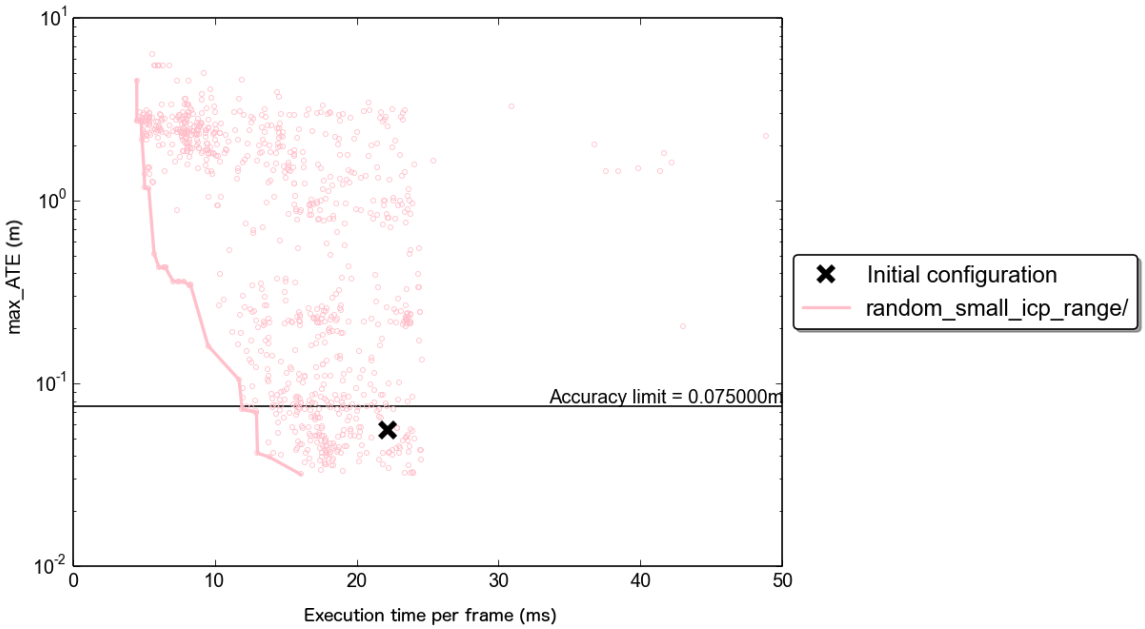
13.2 Redefine parameter range

Dataset: ICL-NUIM Living Room 2
Results generated: 1000
Parameter space: All parameters excluding thresholds. Confidence threshold, ICP-RGB, Depth cutoff, all 5 boolean parameters

Changed the range of the ICP-RGB to be 0-12 instead of 0-100 for this experiment. This is because this parameter performs increasingly badly as the ICP-RGB weight increases. The intention is that this redefinition of the range will act as a filter to some of the pathological results which are obtained during random sampling.

No value higher than 12 is ever seen in the list of good parameter configurations achieved by either the random sampling or active learning results, hence the choice of this parameter as the upper bound. The lower bound of the range remains the same.

The following graph shows the results of this experiment:

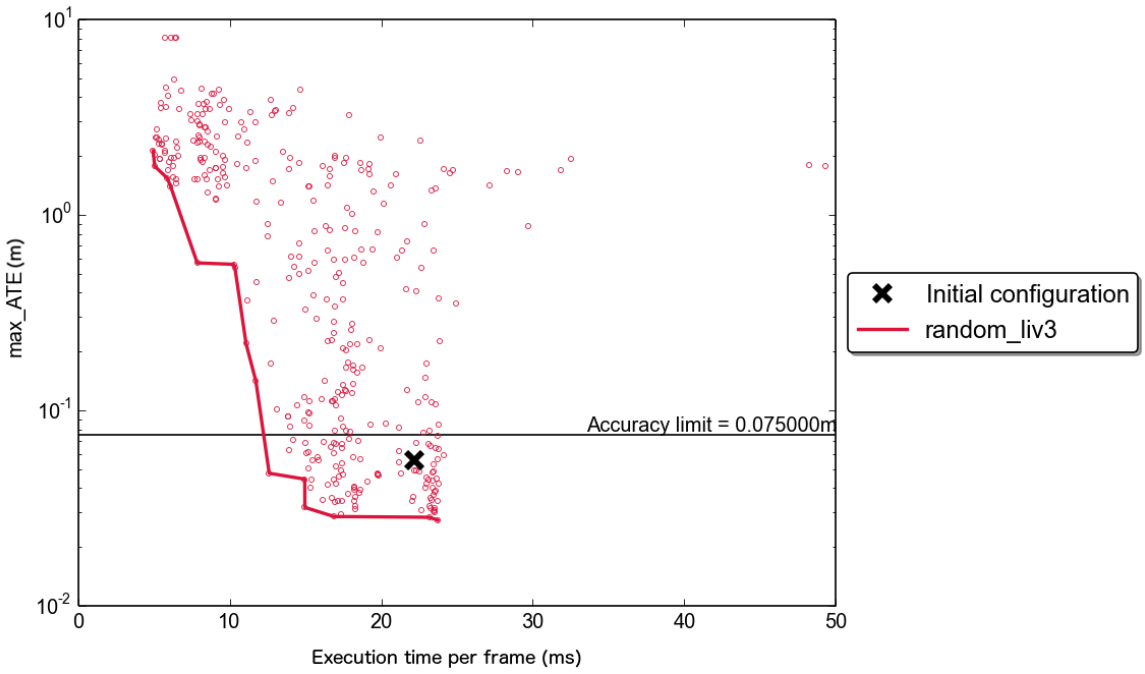


In this new experiment, 10% of the results are good, and only 50% bad results. This better than the old range, where there was only 2% good results and 90% bad results.

The problem with this method is that whilst it does get rid of a huge amount of bad results, there might be some really strange configurations that a high ICP-RGB range works well with. For this reason, it may be best to use a mixture of the random samples, weighting more heavily to the filtered results, but still allowing some of the other unfiltered configurations to be included.

13.3 A different synthetic dataset

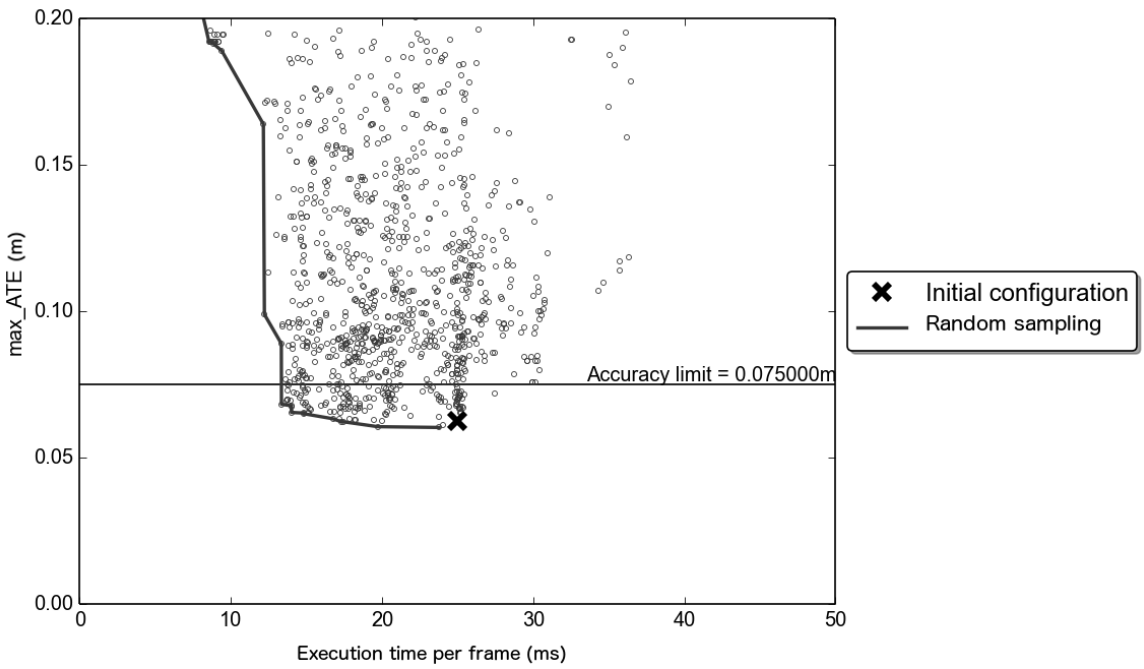
Dataset: ICL-NUIM Living Room 3
Results generated: 500
Parameter space: All parameters excluding thresholds. Confidence threshold, ICP-RGB, Depth cutoff, all 5 boolean parameters



The point of this experiment is to see how random sampling performs on a different synthetic dataset. The ICP-RGB weight range is set small here as it was in the previous experiment, the results are very similar. This strengthens our arguement throughout the paper that of of the key algorithmic parameters for elastic fusion is the geometric and photometric tracking weight.

13.4 A real-world dataset

Dataset: TUM RGB-D Freiburg1 Desk
Results generated: 100
Parameter space: All parameters exluding thresholds. Confidence threshold, ICP-RGB, Depth cutoff, all 5 boolean parameters



The point of this experiment is to see how random sampling performs on a real-world dataset, rather than just the synthetic ICL datasets. We can see that it is still able to find better configurations than the default configuration.

13.5 Conclusion - Random sampling insights

- Random sampling is conceptually very simple, allowing insights to be gained quickly
- Random sampling actually produces some good results - there are a lot of configurations which are much better performance and much better accuracy than the default configuration. However, it's down to pure luck for finding better configurations. More sophisticated methods are needed in order to get reliable and efficient optimisations from experiments.
- It's not as efficient as running active learning, but it can still achieve really high results. This is a good option if efficiency is actually not a problem and it will be very representative of the design space without variance or bias (variance and bias will still exist though even though because of the choice of parameter range).
- Random sampling easily falls into poor performance traps. It does not do anything to avoid pathological cases which we have seen can be improved by something as simple as filtering parameter ranges. Active learning will be able to avoid problems like this automatically. It will predict that those configurations will cause pathological results and avoid re running these experiments.

Chapter 14

Pareto Based Active Learning

14.1 Synthetic design space exploration

14.1.1 Configuration

Dataset: ICL NUIM Living Room 2
Original data file: 2500 random points
Active learning configuration: 100 samples per iteration, 10 iterations

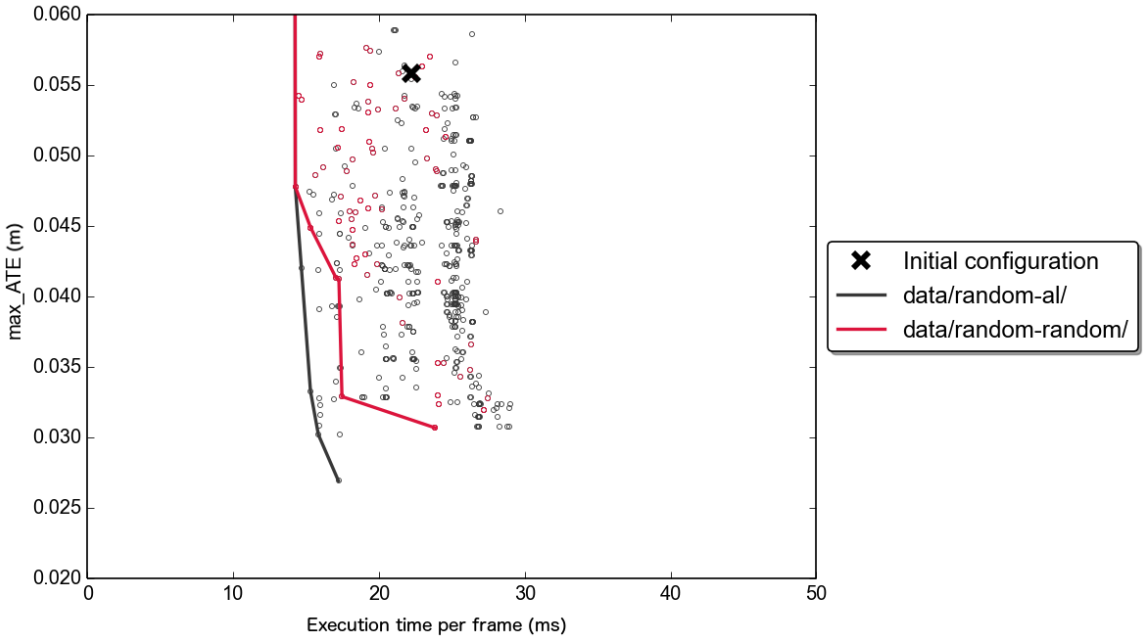


Figure 14.1: The pareto front for the design space exploration using active learning on the ICL-NUIM Living Room 2 dataset, along with the pareto front for the random sampling.

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.04202	14.673	5.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03328	15.297	4.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.0302	15.838	2.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
0.02692	17.214	1.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
		<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

Figure 14.2: The Pareto efficiency points as a result of the design space exploration on the ICL NUIM Living Room 2 Dataset

14.1.2 Results

Points to note:

- An improvement for trajectory accuracy

  ```
  Default configuration:          Best ATE configuration:
  max_ATE:          0.05585m      max_ATE:          0.02692
  total_duration:   22.2ms        total_duration:   17.214ms
  ```

  This is a **2x** accuracy improvement

  (this configuration is also a **1.25x** speed up)

  As we can see from this result, the best ATE score does not necessarily have the long execution time, i.e. allowing more time will not necessarily improve results.

- We can achieve an improvement on execution time for the same ATE as the default configuration

  ```
  Default configuration:          Other configuration:
  max_ATE:          0.05585m      max_ATE:          0.05584m
  total_duration:   22.2s         total_duration:   12.1s
  ```

  This is a **1.85x** speedup.

- There are **more good results** when using active learning instead of random sampling

  Here, a good result is defined as a configuration which has an ATE lower than 6cm and an execution time lower than 22.5s (which are soft boundaries around the default configuration)

Active learning:	**Random sampling:**
- Number of points = 999	- Number of points = 2401
- Number of good points = 83	- Number of good points = 48
8.3% good results	2% good results

There are about **4x** as many good results

- There are **less poor results** when using active learning instead of random sampling

  Here, a bad result is defined as a configuration which has an ATE higher than 1m or an execution time higher than 25s

Active learning:	**Random sampling:**
- Number of points = 999	- Number of points = 2401
- Number of bad points = 597	- Number of bad points = 2121
60% bad results	88% bad results

  There are about **50%** fewer bad results

- The best results found for active learning are better than the best results for random sampling

  We can see in 14.1 that the active learning is able to push forward the pareto front beyond doing extra random sampling for the same number of configurations. This shows us that a pareto based active learning method of design space exploration will lead us to more accurate and better performance results.

- We don't get many high performance points at a good ATE

  To view the unzoomed pareto front graph see the Appendix A at Figure A.4. The problem here is that the predictor is acting really aggressively. It is predicting points that are high performance and should be achieving good accuracy, however because it is so aggressive, tracking ends up failing - hence the large number of results which have really poor accuracy and minimal execution time. All of these points have the same execution time because over 400 frames early tracking failure results in the median time being the same (the time per frame to execute the re-tracking code).

Improving Accuracy

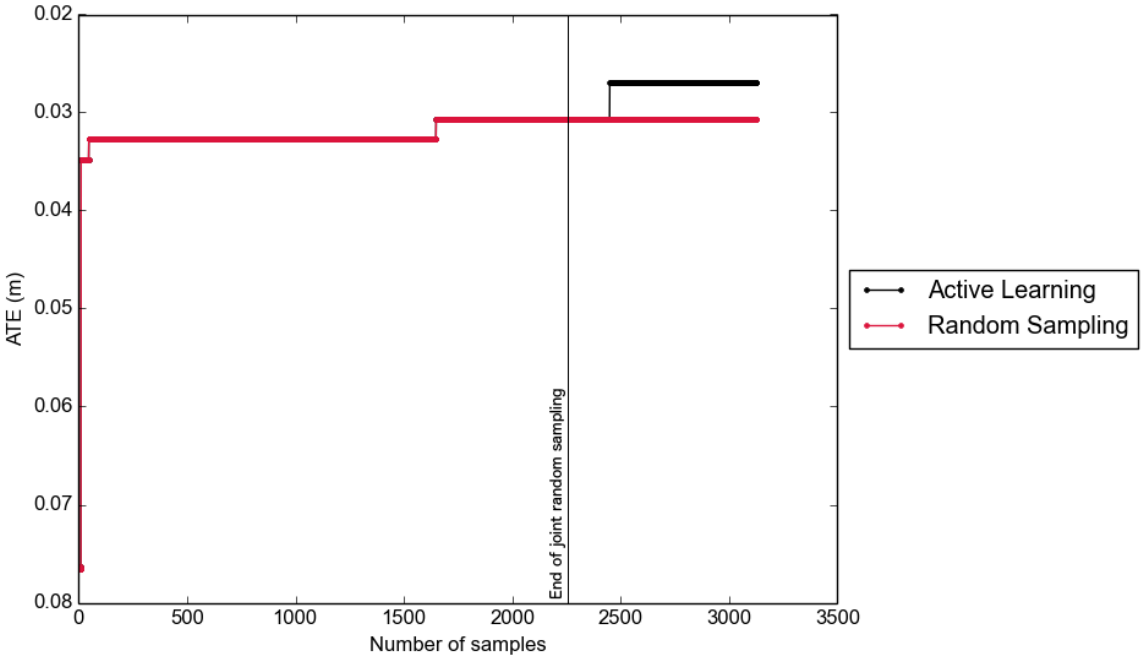


Figure 14.3: A plot showing the best ATE achieved after n frames, for both random sampling and active learning

Here we can see that the active learning method can achieve better accuracy results than the random sampling. If we were to continue for more iterations, we may see random sampling finding higher points - however we achieve it faster with active learning and may find an even better point with active learning.

Improving Performance

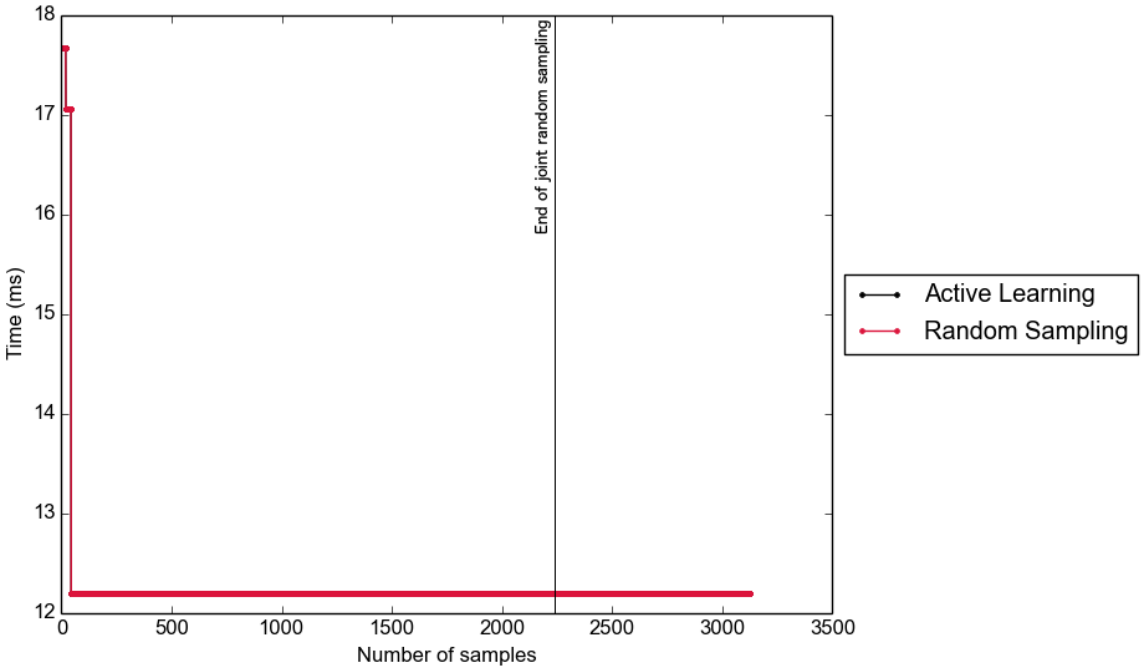


Figure 14.4: A plot showing the best performance achieved after n frames (Under an ATE limit of 10cm), for both random sampling and active learning

Here we don't see an improvement on the performance metric when comparing active learning with random sampling. This is due to the point we have explored where the active learning predictor is too aggressive on improving performance, and the configurations it predicts fail on tracking and end up having too poor of a performance to show up on this graph.

14.1.3 An Optimal Configuration?

icp	depth	confidence	so3	Close-Loops	Reloc	Fast-Odometry	FTF RGB
<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

This optimal configuration has been constructed from analysing the correlation between the pareto-points Table 14.2, and other really good points for accuracy, and for performance (This raw data can be viewed in Appendix A Tables A.2 and A.1.

The results here correlate really well. This results in another configuration that could be used instead of the default configuration. For the ICL dataset, we actually see that with a few exceptions we have a very similar configuration when focusing optimisation on either accuracy or performance.

The main difference is for the fastest results, we want to enable fast odometry, and for the most accurate results, we want to disable fast odometry. This result is generated by looking at the pareto points, the best results for ATE, and the best results for execution time. The full tables of these configurations can be seen in the appendices.

Of course, this result is dataset dependant. Whilst it proves to work well on other synthetic datasets, it actually performs more poorly on the dataset of TUM. (Default configuration ATE for TUM: 0.04m, New configuration ATE for TUM: 0.046m)

Enable Fast Odometry

Fast odometry enabled is achieving more accurate results on the Living Room 2 dataset. This is a strange result - the more transform iterations done during tracking the better the ATE will be surely.

In the best pareto front configurations, fast odometry is enabled and still achieving really high accuracy results. This is because it is optimising for execution time and accuracy.

With fast odometry disabled we can expect even better accuracy results here. Lots of iterations are better i.e. slow odometry. Times much faster than the default configuration can still be achieved with 10 iterations over the pyramid. (good times can still be achieved)

ICP-RGB weight

Results: 2/3/4 (Lower is better)

The low ICP values really perform well for the ICP dataset. This follows from the simple single parameter exploration that was performed earlier. Photometric tracking performs extremely badly for the ICL synthetic datasets. This is because of the lack of colour and texture in the scene compared with the TUM real world datasets. This reiterates the point that a simple weighting between photometric and geometric tracking is not enough - a relatively texture-less dataset will not run well.

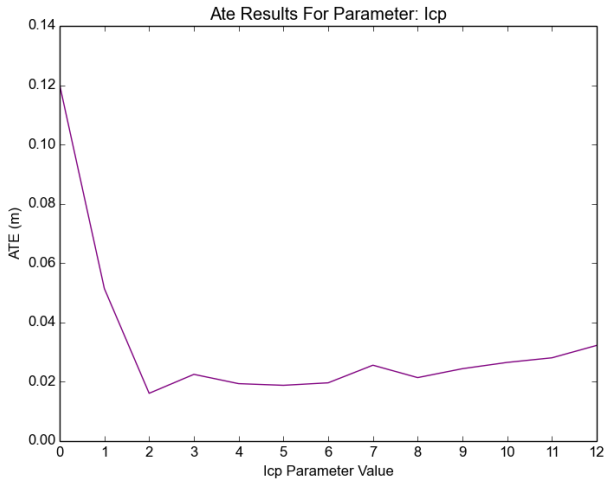


Figure 14.5: Different parameter values for the ICP-RGB weight parameter, ranging from 0-12 (0.0 - 0.12), on the otherwise optimal configuration

Depth

Results: Between 5 and 10 give the best results

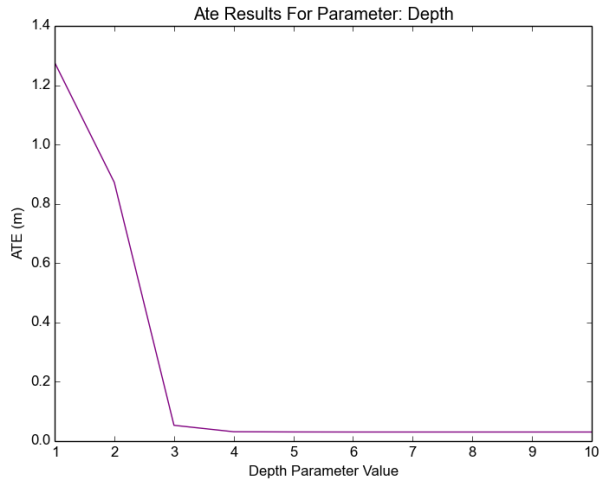


Figure 14.6: Different parameter values for the depth cut off parameter, ranging from 0-12 (0.0 - 0.12), on the otherwise optimal configuration. This graph backs up the results from the good parameters found. The depth stops affecting the ATE very much after about 5m, and 3 and 4m depths are still pretty accurate.

Profiling data

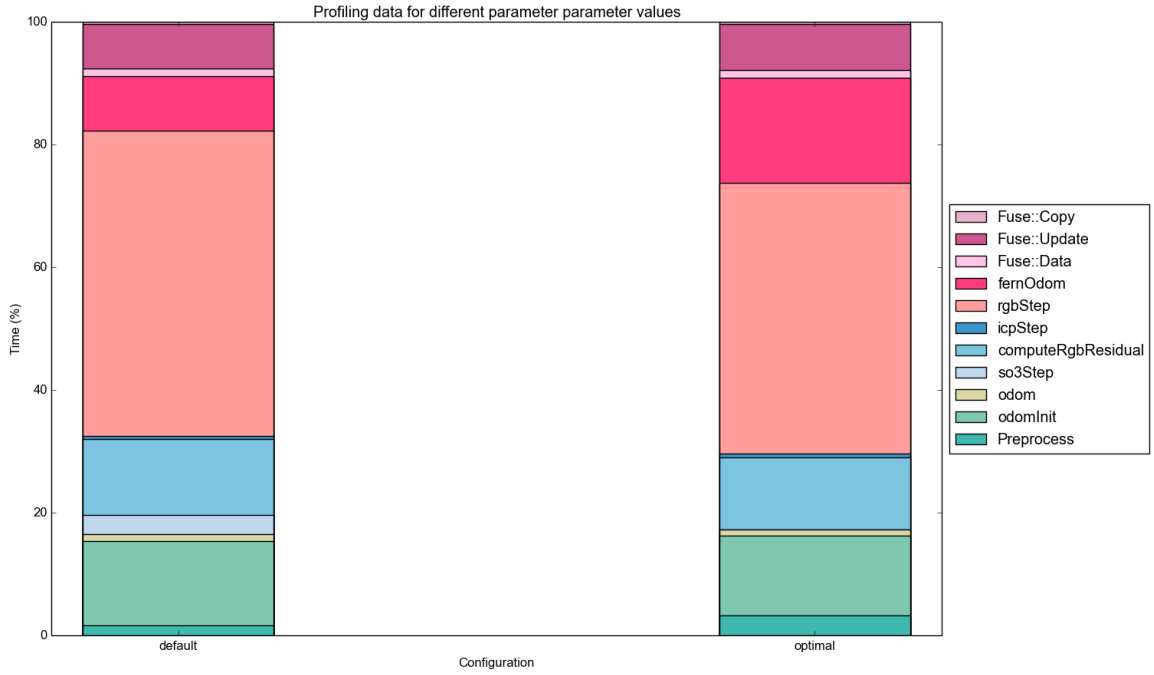


Figure 14.7: Profiling data for the default configuration of ElasticFusion, and the new optimal configuration found

Here we see that a lower proportion of time is spent in tracking in the optimal configuration - this is due to the fast odometry parameter causing us to perform less tracking iterations. We see a higher proportion of time spent in the fern odometry stage, which is because we see a comparatively lower amount of time spent in the other odometry step. We also note the loss of the so3step, which saves us execution time also. Preprocessing takes up a larger proportion of time - this due to the skewing from other kernels though and actually takes the same amount of raw time.

Exceptions to the rule which have high accuracy:

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.03347	20.462	**7.0**	8.0	**3.0**	0.0	0.0	**0.0**	**0.0**	0.0
0.03426	19.998	4.0	7.0	8.0	0.0	**1.0**	1.0	1.0	**1.0**
0.03426	21.603	4.0	6.0	8.0	0.0	**1.0**	1.0	1.0	**1.0**
	Avg	<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

Figure 14.8: These good accuracy points have worse performance than the optimal configurations, but still have better performance than the default configuration.

First exception in Table 14.8

Fast Odometry is disabled in this example. It shows how for a sacrifice in time by doing extra tracking we can achieve a good accuracy still. ElasticFusion can essentially "recover" from the poorer choices of a higher weighting to photometric tracking and a lower surfel confidence threshold for the ICL dataset.

The effect of re-localisation is not important here - if the other parameters are good, the camera wont loose track. We believe that on the synthetic dataset with a strong weighting to geometric tracking, i.e. a good configuration for synthetic, ElasticFusion does not lose track.

Second exception in Table 14.8

This configuration performs well on accuracy, however it sacrifices performance as loop closures are enabled.

Summary

However, analysis aside, the ultimate point to this story on exceptions is that there exist different ways of reaching optimal configurations, this is to say there is very likely a non-convex optimal design space. The machine learning methods are excellent at predicting new points on the pareto front based on previous configurations, however finding completely new points will be more challenging for it because predictors often work with methods like gradient descent which can fall prey to local minima - implying a convex result which is not actually there.

14.1.4 Decision tree for ICL-NUIM

Here we see a decision tree that helps us pick good configurations for ElasticFusion. We have constructed this with the strongest parameters we have found in the algorithmic analysis of ElasticFusion and insights from the optimal configurations from the design space exploration. This decision tree is specific to the ICL datasets, whilst depth and fast odometry are consistent over the TUM datasets - we would want to make slightly different choices regarding the ICP-RGB weight parameter.

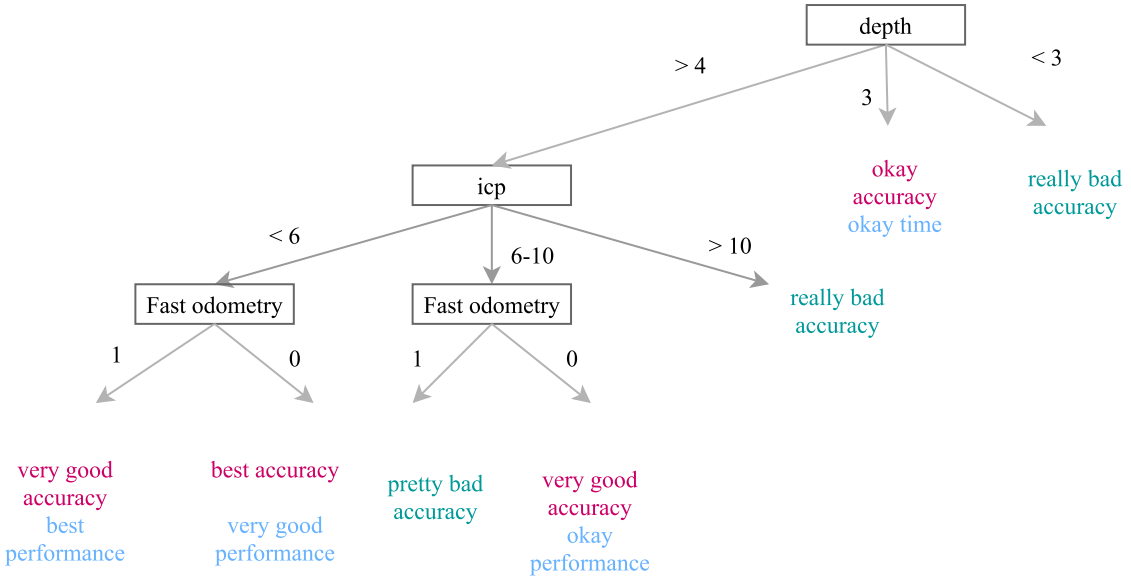


Figure 14.9: This diagram shows a decision tree for reaching optimal configurations. {Accuracy results - Red. Performance results - Blue. Poor results - Green}. We see the best results following a high depth cut off, a low ICP-RGB weight, and Slow odometry.

14.1.5 Trade-Offs

As suggested towards in previous chapters, the following are the results for good trade-offs that can be performed to choose whether to improve performance or trajectory accuracy.

Depth Cut-Off

The following results are the depth parameter value for the best 5 on ATE, and the best 5 configurations on time duration with an ATE under 5cm. See Appendix A Table A.1 and Table A.2.

The rest of the parameters are pretty similar to the optimal configuration.

Best 5 ATE results depth parameter: 7.0, 7.0, 8.0, 7.0, 9.0
Best 5 Time results depth parameter: 3.0, 3.0, 3.0, 3.0, 3.0

	BEST ATE	BEST TIME
ATE (m)	0.0323	0.0480
Execution time (s)	17.5	13.4

The strict depth cut off corresponds here to a speed up of 1.31x (30% performance increase), but a 1.49x reduction in accuracy (50% accuracy decrease)
What this tells us is that cutting off further depth data can give an easy speedup without affecting the ATE too badly for small values.

Fast odometry

This parameter indicates that changing the number of iterations over tracking is an easy way to find trade-offs. This correlates well with the results of other SLAM implementations. Enabling fast odometry reduces the number of iterations in the final pyramid, and we have seen that it decreases the accuracy but increases the performance quite significantly.

14.2 Real world Design Space Exploration

To contrast the results of the ICL Living Room Datasets, we would like to perform a design space exploration using a real life dataset.

14.2.1 Configuration

Dataset: TUM RGB-D FR1 Desk
Original data file: 2000 random points
Active learning configuration: 100 samples per iteration, 10 iterations

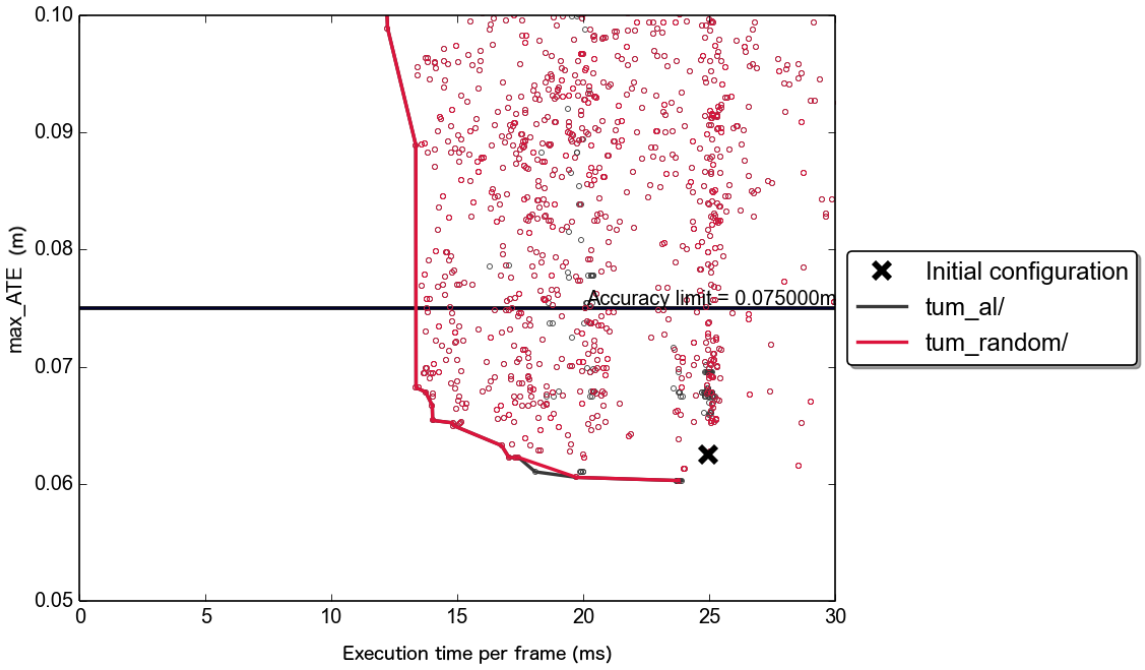


Figure 14.10: The pareto front for random sampling design space exploration in red, and the active learning results in black.

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.06784	13.736	11.0	8.0	11.0	1.0	0.0	0.0	1.0	0.0
0.06669	13.996	11.0	7.0	9.0	1.0	0.0	0.0	1.0	0.0
0.06541	14.011	9.0	2.0	7.0	1.0	0.0	1.0	1.0	0.0
0.06521	14.807	10.0	4.0	9.0	1.0	0.0	1.0	1.0	0.0
0.06496	14.846	7.0	4.0	10.0	0.0	0.0	1.0	1.0	0.0
0.06326	16.749	9.0	6.0	10.0	1.0	0.0	0.0	0.0	0.0
0.06224	17.426	10.0	9.0	9.0	1.0	0.0	1.0	0.0	0.0
0.06224	17.044	10.0	8.0	9.0	1.0	0.0	1.0	0.0	0.0
0.06055	19.725	7.0	8.0	2.0	0.0	0.0	0.0	1.0	0.0
0.06028	23.784	9.0	4.0	2.0	0.0	1.0	0.0	1.0	0.0
	Avg	>= 7	>= 4	2, 10	Mixed	0.0	Mixed	Mixed	0.0

Figure 14.11: Pareto Front Configurations for the TUM dataset

From the table in Figure 14.11 which shows the pareto parameters for the TUM dataset, and the additional tables in the appendix showing other good results, we can conclude the following:

- We can achieve an improvement on performance

  ```
  Default configuration:          Better performance configuration:
  max_ATE:          0.0624m       max_ATE:          0.06224m
  total_duration:   24.95ms       total_duration:   17.044ms
  ```

  This is a **1.5x** speedup for the same accuracy, and can reach a **1.8x** performance increase with a small decay in accuracy.

- An improvement for trajectory accuracy

  ```
  Default configuration:          Best ATE configuration:
  max_ATE:          0.0624m       max_ATE:          0.06028
  total_duration:   24.95ms       total_duration:   23.784ms
  ```

  This is a only a **5%** accuracy improvement, as the TUM default configuration is already really good for accuracy.

- Good vs. bad points

Results	Random sampling	Active Learning
Good	9%	9%
Bad	20%	7%

On this dataset, we see a similar proportion of results are good, but there are significantly fewer bad results. We can see a plot in the appendix of the whole graph (not cropped for only interesting accuracy results) which will show the proportion of bad results is much less in active learning.

14.2.2 An optimal configuration for the TUM dataset

icp	depth	confidence	so3	Close-Loops	Reloc	Fast-Odometry	FTF RGB
>= 7	>= 4	2, 10	Mixed	0.0	Mixed	Mixed	0.0

The majority of these parameters are the same as for the ICL-NUIM Living Room 2 dataset. This is unsurprising as the majority of parameters have predictable and expected effects on performance and accuracy.

The points that are different/interesting:

- Higher ICP-RGB weight

  The TUM dataset performs much more accurately using a higher ICP-RGB weight. This simply extends what we have seen throughout this project - the TUM dataset has texture and colour, whereas the synthetic dataset does not.

- Fast Odometry

  The Fast Odometry results are shown more clearly with these Pareto parameters, we see the general trend that the fastest Pareto points have Fast Odometry enabled, and the most accurate Pareto points disable it, performing all of the tracking iterations.

Accuracy and Performance Plots

On the TUM FR1 DESK dataset , we don't see an improvement in the best accuracy results from Figure 14.12. This is also shown in the Pareto front plot Figure14.10. Likewise we see similar results for exection time.

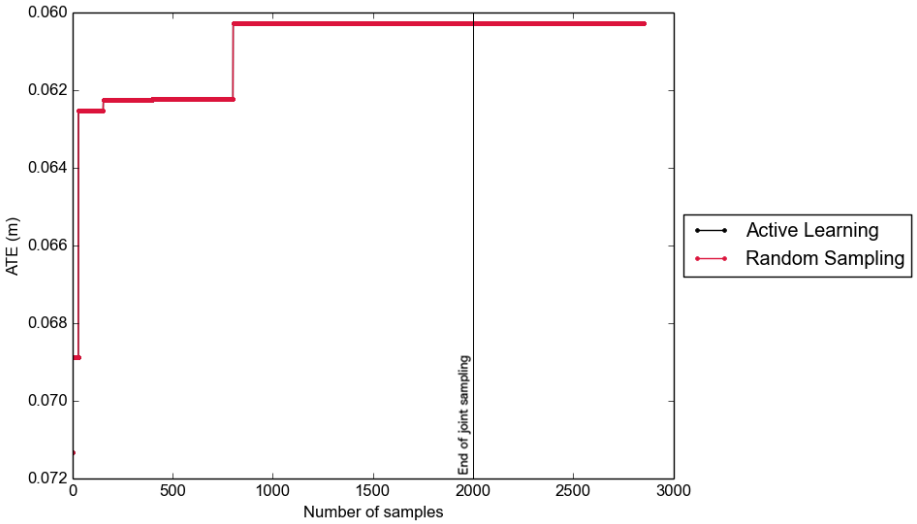


Figure 14.12: A plot showing the best ATE achieved after n frames, for both random sampling and active learning

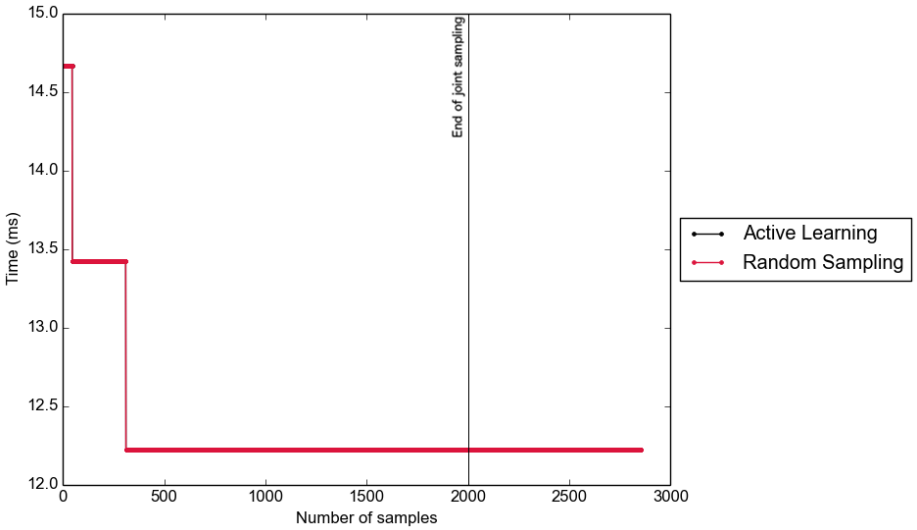


Figure 14.13: A plot showing the best performance achieved after n frames (Under an ATE limit of 10cm), for both random sampling and active learning

14.2.3 Conclusion of TUM findings

- We don't get better results on ATE on this dataset, but we do get better performance

  We don't see better results for ATE because the default configuration benchmark is higher - the default configuration for the TUM dataset has been picked out very well so that it already performs well on accuracy. We can still get much better performance by up to 1.8x on this dataset for the same ATE.

- Optimal configurations for TUM are different than on ICL

  This makes sense - as the datasets are very different, and we have seen in this chapter and others that dataset dependence is really important.

- The active learning doesn't push the pareto front forward very much

  A speculative point is that this is algorithmically dependant. On the TUM dataset the parameter ranges don't cause as much of an affect as on the ICL datasets. There is much more room for an improvement on the synthetic dataset because it performs so poorly on the photometric tracking, which skews all of the results much more significantly - allowing interesting parameter configurations which may be more easily found by machine learning. This scope for improvement may not exist for the real-life dataset. The parameters are simply not as sensitive here. We have seen a similar point to this earlier on in the project, where the TUM FR2 XYZ dataset does not have as much scope for a change of accuracy when alterting the ICP-RGB weight parameter as the TUM FR1 DESK dataset as the camera barely moves pose.

14.3 Summary of design space exploration findings

- Generalisation of the design space exploration techniques

  We have performed design space exploration methods on Elastic fusion, with both a real world dataset and a synthetic dataset, and have shown the techniques perform well on these too. We can achieve performance improvements on both the synthetic and real-world datasets, and achieve performance results using ElasticFusion showing that these techniques generalise to other implementations aside from KinectFusion.

- Machine learning techniques perform better than random sampling techniques

  We have shown that the machine learning design space exploration techniques more efficient find good solutions than random sampling. On the ICL-NUIM dataset we see that we can achieve better accuracy results than random sampling with active learning. We have also shown that a higher proportion of results are good configurations, and we get much fewer pathological configurations with active learning as well.

- We have shown dataset knowledge is important for picking good configurations for a SLAM algorithm

  Ideally design space exploration would be run over 100s of different types of dataset, encapsulating all possible rotations and translations of the camera, different type of surface textures and scene scale and density. This would allow more insight into why the things happen when running with specific parameters, and allow the vision experts a more useful debugging aid, and additionally more assurance that their specifically designed SLAM algorithm will perform well where it says it will.

Part VI

Conclusion

Chapter 15

Conclusion

15.1 Summary of Achievements

- We integrated ElasticFusion into SLAMBench, ensuring that it executes in a process-every-frame mode. We extracted per-kernel performance timings to aid our analysis of ElasticFusion.
- We added support for ElasticFusion to run with the ICL-NUIM and TUM RGB-D datasets
- We integrated ElasticFusion into the design space exploration code, and added support to run it with the TUM RGB-D dataset

Using this integration:

- Successful Design Space Exploration of ElasticFusion

  We performed a design space exploration showing that the machine learning methods outperform the randomised methods - proving that active learning techniques generalise to other SLAM implementations.

- Performance improvements and analysis

  We have achieved significant performance improvements by performing design space exploration on ElasticFusion. We are able to improve the execution time of ElasticFusion on the ICL datasets by almost 2x with no loss in accuracy, and improving the trajectory accuracy by about 2x whilst also improving execution time. For the TUM RGB-D dataset we can achieve an 80% performance increase without losing accuracy, pushing the performance of ElasticFusion from about real time, to definitely real time. We analyse and justify these results to draw conclusions in how to best run the ElasticFusion algorithm for certain dataset types.

- Identifying Key Parameters and Trade-Offs

  We performed an analysis of the design space exploration results, identifying pareto points for ElasticFusion, showing that we can find trade-offs that can be made between trajectory accuracy and performance, such as pyramid iteration levels or depth cut off limit. We identify the key parameters we can change that result in performance affects on the algorithm.

- Dataset dependence

  By performing a design space exploration on the TUM dataset we showed that performance is dependant on the dataset, and an optimal solution for one dataset is not necessarily optimal for another. The optimal configuration for the ICL NUIM Living Room 2 datasets improves the accuracy by 2x, but in fact decreases the accuracy of the TUM RGB-D dataset by approximately 12.5% compared to the ElasticFusion default configuration.

  We have also seen the dataset dependence issue seeing the parameter ICP-RGB weight affect the ICL NUIM synthetic datasets far more than the TUM datasets, and the difference in trajectory motion affect the rate of accuracy decline for the same parameter ranges.

  Analysis using SLAMBench must be run using a large range of datasets, encapsulating all different camera motions, scene textures and scales for comparisons and evaluations to be accurate and fair.

  This ties into the holistic approach - the whole system must be considered as optimum configurations are dependant on everything. This can be extended further than just looking at the whole system as the architecture and the algorithm - it also concerns the number of cameras used by a robot, or the speed at which the robot runs. Another interesting layer is that of dynamic execution, i.e. changing the approach dependant on the results being found at the time. For example, slowing the robot down if it is lost, or as we have seen in this paper perhaps focusing on geometric tracking when the frames are more texture-less, and refocusing on photometric tracking when this isn't a problem.

- Diagnostic SLAM

  We further show SLAMBench can be used in a diagnostic way by computer vision developers, having found some potential issues in the ElasticFusion algorithm, highlighted by poorer performance on the synthetic datasets. We can then suggest improvements to ElasticFusion, such as dynamically selecting photometric and geometric tracking weight depending on the information in the frame.

  We have identified potential issues within the ElasticFusion algorithm showing that SLAMBench can be used in a diagnostic way. The issues are related to the photometric and geometric tracking weighting being dependant on dataset for performance, and we suggest various optimisations as a result of our analysis e.g. dynamic frame-to-frame tracking weighting decisions, or reducing tracking pyramid iterations for the lower weighted technique.

15.2 Future work

- Mapping accuracy measurement

  As discussed earlier in the report, statistics on map accuracy are important for the evaluation of a SLAM algorithm. Without this we only have half the story, the trajectory accuracy. Although the trajectory accuracy definitely implies a solid map representation, the small errors can cause problems for highly specific use-cases e.g. a robot bumping into misplaced objects in the scene.

  The first problem is how to asses the accuracy between the estimated maps and ground truth maps. This involves finding registration methods, ensuring that different formats of map can be registered together, and choosing how to evaluate the difference between them after alignment e.g. a metric such as ATE. The other problem is finding ground truth models of scenes. ICL-NUIM provides a set of ground truth maps but there are few on offer.

  For this to run using SLAMBench, the results must be generated automatically which adds another level to the complexity of this problem as there are a wide variety of surface models for SLAM algorithms e.g. signed truncated distance function, surfel based model.

- Port ElasticFusion to C++ and OpenCL to allow evaluation on different hardware

  ElasticFusion analysis is currently limited to nVidia GPUs due to it's tracking code being written in CUDA. Whelan et al. have suggested that it would be possible to rewrite the tracking code in OpenCL to allow it to be run on other GPUs or integrated graphics units. We could also run on different nVidia hardware, it would be interesting to run on the embedded platform and analyse power readings here.

- Dataset choices

  Information gain per frame is about how much changes per frame on a dataset. A good dataset would have a lot of new information, this could be new camera rotations or a dense scene. It will tell us about how good a dataset is at evaluating all the different metrics. This is currently being looked into by Sajad Saeedi of the software performance optimisation group. This work is all about finding exhaustive dataset configurations, because as we have seen this this project the dataset dependence is a big issue for accurately implementing SLAM algorithms.

- Further analysis of ElasticFusion

  Implementing some of the suggested optimisations could be an interesting step into improving the ElasticFusion code for robustness on different datasets, and to improve performance. It would also be useful to see how ElasticFusion compares to another algorithm such as KinectFusion or LSD-SLAM.

Appendices

Appendix A

Additional Design Space Exploration: Synthetic dataset

Tables showing the interesting parameters of the ICL Living room dataset:

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.03226	15.95	3.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03162	15.94	4.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03083	15.931	5.0	8.0	4.0	0.0	0.0	1.0	1.0	0.0
0.03907	15.925	2.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03279	15.922	5.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
0.04445	15.904	4.0	6.0	4.0	0.0	0.0	1.0	1.0	1.0
0.04591	15.87	2.0	9.0	4.0	0.0	0.0	1.0	1.0	1.0
0.0302	15.838	2.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
0.04185	15.81	5.0	10.0	4.0	0.0	0.0	1.0	1.0	1.0
0.04724	15.498	1.0	10.0	5.0	0.0	0.0	1.0	1.0	0.0
0.03328	15.297	4.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.04746	15.219	2.0	10.0	6.0	0.0	0.0	1.0	1.0	0.0
0.04202	14.673	5.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
	Avg	<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

Figure A.1: ICL LIV 2: Points close to the pareto front in time

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.03494	17.366	3.0	9.0	4.0	0.0	0.0	1.0	1.0	0.0
0.03494	17.285	3.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
0.03396	17.018	3.0	10.0	5.0	0.0	0.0	1.0	1.0	0.0
0.03328	15.297	4.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03279	15.922	5.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
0.03273	16.902	2.0	10.0	5.0	0.0	0.0	1.0	1.0	0.0
0.03226	15.95	3.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03162	15.94	4.0	6.0	9.0	0.0	0.0	1.0	1.0	0.0
0.03083	15.931	5.0	8.0	4.0	0.0	0.0	1.0	1.0	0.0
0.0302	17.309	2.0	9.0	4.0	0.0	0.0	1.0	1.0	0.0
0.0302	15.838	2.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
0.02692	17.214	1.0	10.0	4.0	0.0	0.0	1.0	1.0	0.0
	Avg	<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

Figure A.2: ICL LIV 2: Points close to the pareto from in accuracy

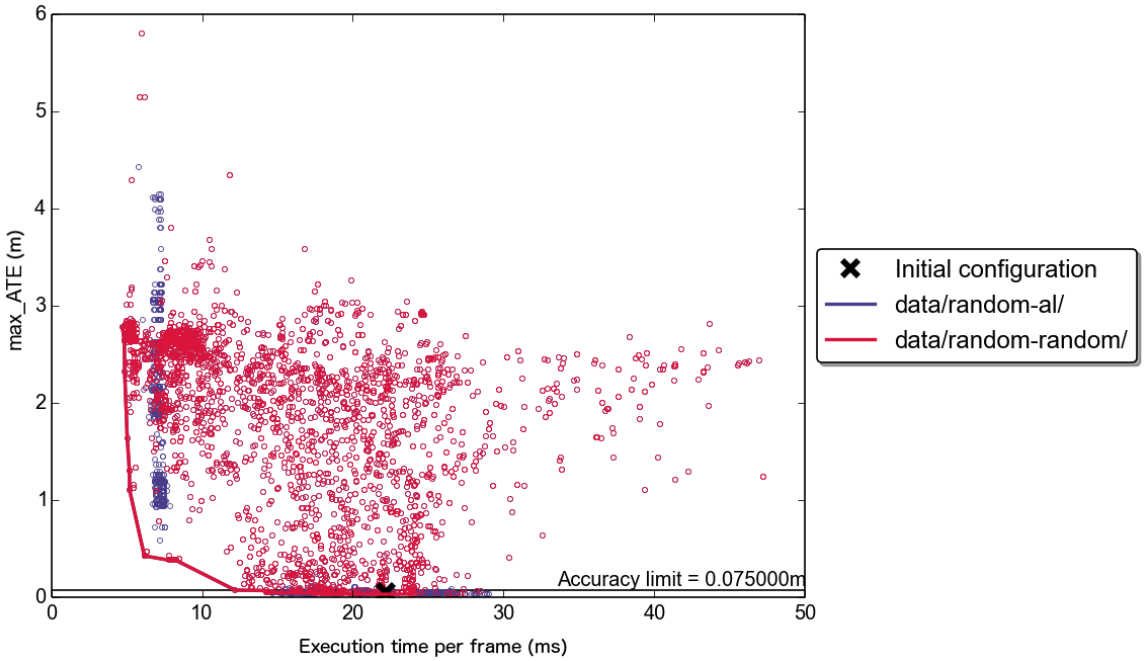


Figure A.3: The pareto curve for active learning on the ICL Living Room 2 dataset, un-cropped to include all results, not just those in the interesting accuracy and performance range.

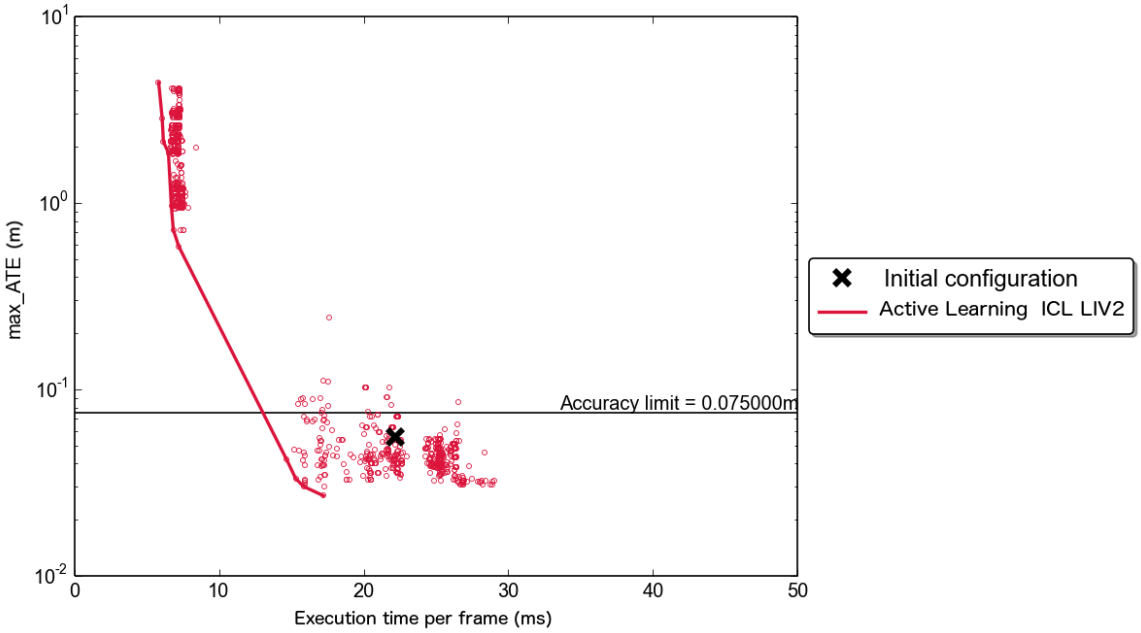


Figure A.4: The pareto curve for active learning on the ICL Living Room 2 dataset, without including the random sampling results that drive it. This graph is for clarity on a per-iteration basis of active learning.

Appendix B

Additional Design Space Exploration: Real-life dataset

B.1 TUM Dataset Results

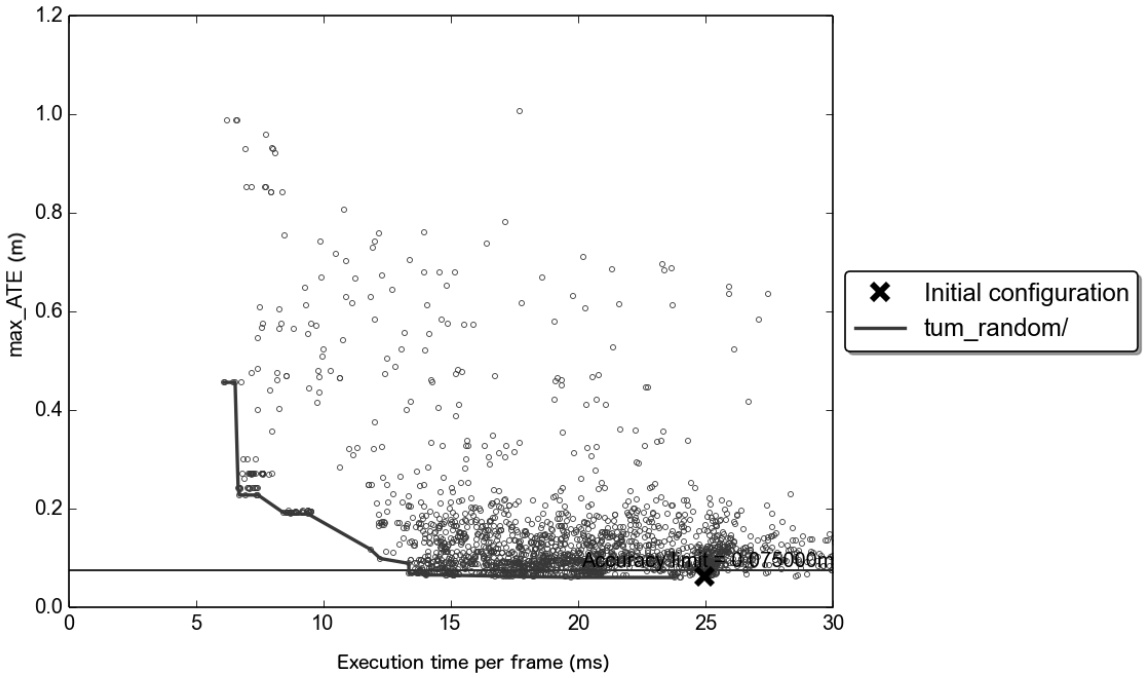


Figure B.1: The results of random sampling on the TUM FR1 Desk - Unzoomed version showing un-interesting results too

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.06496	14.846	7.0	4.0	10.0	0.0	0.0	1.0	1.0	0.0
0.06521	14.807	10.0	4.0	9.0	1.0	0.0	1.0	1.0	0.0
0.0746	14.786	6.0	7.0	11.0	1.0	0.0	0.0	1.0	0.0
0.07391	14.766	5.0	2.0	7.0	0.0	0.0	1.0	1.0	0.0
0.06859	14.731	7.0	2.0	9.0	1.0	0.0	1.0	1.0	0.0
0.07133	14.663	11.0	2.0	9.0	1.0	0.0	1.0	1.0	0.0
0.07056	14.556	7.0	5.0	11.0	0.0	0.0	0.0	1.0	0.0
0.07179	14.365	6.0	3.0	8.0	1.0	0.0	0.0	1.0	0.0
0.06952	14.275	10.0	7.0	11.0	0.0	0.0	0.0	1.0	0.0
0.07343	14.243	8.0	8.0	7.0	0.0	0.0	1.0	1.0	0.0
0.07225	14.152	9.0	6.0	7.0	0.0	0.0	1.0	1.0	0.0
0.07025	14.043	6.0	3.0	9.0	1.0	0.0	1.0	1.0	0.0
0.06944	14.038	6.0	9.0	9.0	1.0	0.0	1.0	1.0	0.0
0.06748	14.016	9.0	5.0	9.0	1.0	0.0	0.0	1.0	0.0
0.06541	14.011	9.0	2.0	7.0	1.0	0.0	1.0	1.0	0.0
0.06769	14.01	7.0	5.0	9.0	1.0	0.0	0.0	1.0	0.0
0.06669	13.996	11.0	7.0	9.0	1.0	0.0	0.0	1.0	0.0
0.06943	13.937	10.0	2.0	11.0	1.0	0.0	0.0	1.0	0.0
0.06897	13.899	9.0	2.0	9.0	1.0	0.0	1.0	1.0	0.0
0.07462	13.805	10.0	7.0	9.0	0.0	0.0	0.0	1.0	0.0
0.06992	13.787	8.0	6.0	9.0	0.0	0.0	0.0	1.0	0.0
0.07297	13.756	9.0	9.0	9.0	0.0	0.0	1.0	1.0	0.0
0.06992	13.742	8.0	9.0	9.0	0.0	0.0	0.0	1.0	0.0
0.06784	13.736	11.0	8.0	11.0	1.0	0.0	0.0	1.0	0.0
0.06944	13.694	10.0	4.0	11.0	1.0	0.0	1.0	1.0	0.0
0.06947	13.653	6.0	2.0	10.0	0.0	0.0	0.0	1.0	0.0
0.06821	13.451	6.0	2.0	9.0	0.0	0.0	0.0	1.0	0.0
0.06821	13.353	4.0	2.0	10.0	0.0	0.0	0.0	1.0	0.0
	Avg	<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

Figure B.2: TUM FR1 Desk results: Points close to the pareto front in time

ATE	duration	icp	depth	confidence	so3	CloseLoops	Reloc	FastOdom	FTF RGB
0.06287	18.276	5.0	2.0	11.0	0.0	1.0	0.0	1.0	0.0
0.06287	18.207	5.0	2.0	11.0	0.0	1.0	0.0	1.0	0.0
0.06272	19.472	6.0	9.0	2.0	1.0	0.0	1.0	1.0	0.0
0.06253	19.999	8.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06227	20.012	7.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06226	17.288	11.0	5.0	10.0	1.0	0.0	0.0	0.0	0.0
0.06224	17.426	10.0	9.0	9.0	1.0	0.0	1.0	0.0	0.0
0.06224	17.044	10.0	8.0	9.0	1.0	0.0	1.0	0.0	0.0
0.06103	19.883	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	18.118	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.956	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.975	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.903	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.924	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.953	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.982	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	18.09	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	18.144	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	18.136	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.939	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	20.056	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	19.91	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06103	20.061	9.0	4.0	2.0	1.0	0.0	0.0	1.0	0.0
0.06055	19.725	7.0	8.0	2.0	0.0	0.0	0.0	1.0	0.0
	Avg	<= 5	>= 6	4, 9	0.0	0.0	1.0	1.0	0.0

Figure B.3: TUM FR1 Desk results: Points close to the pareto front in accuracy

References

Bibliography

[1] Durrant-Whyte, Hugh, and Tim Bailey, *Simultaneous Localization and Mapping: Part I* , Robotics & Automation Magazine, IEEE 13.2: 99-110, 2006.

[2] Emanuele Vespa, *A fast volumetric fusion based on octrees with automatic surface evaluation*, Early Stage Assessment Report, Department of Computing, Imperial, 2016.

[3] Whelan, Thomas and Leutenegger, Stefan and Salas-Moreno, Renato F and Glocker, Ben and Davison, Andrew J, *ElasticFusion: Dense SLAM Without A Pose Graph*, Proceedings of Robotics: Science and Systems (RSS), 2015.

[4] Glocker, Ben, et al, *Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding*, Visualization and Computer Graphics, IEEE Transactions on 21.5: 571-583, 2015.

[5] Martin Sayers, Colin Smith, Researchers closer to developing vision for household robots, December 2015, `http://www3.imperial.ac.uk/newsandeventspggrp/imperialcollege/newssummary/news_26-11-2015-10-31-15`, [Retrieved 12-06-2016].

[6] Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., Csorba, M., *A Solution to the Simultaneous Localisation and Map Building (SLAM) Problem*, Robotics and Automation, IEEE Transactions on, 17(3), 229-241, 2001.

[7] Jakob Engel, Thomas Schops, Daniel Cremers, *LSD-SLAM: Large-Scale Direct Monocular SLAM*, In Computer VisionECCV 2014 (pp. 834-849). Springer International Publishing, 2014.

[8] Izadi, Shahram and Kim, David and Hilliges, Otmar and Molyneaux, David and Newcombe, Richard and Kohli, Pushmeet and Shotton, Jamie and Hodges, Steve and Freeman, Dustin and Davison, Andrew and others, *KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera*, Proceedings of the 24th annual ACM symposium on User interface software and technology. Pages 559–568, 2011.

[9] Whelan, Thomas, et al, *Real-time large scale dense RGB-D SLAM with volumetric fusion*, The International Journal of Robotics Research 34.4-5: 598-626, 2015.

[10] Keller, Matthias, et al., *Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion*, 3D Vision-3DV 2013, 2013 International Conference on. IEEE, 2013.

[11] Nardi, Luigi and Bodin, Bruno and Zia, M Zeeshan and Mawer, John and Nisbet, Andy and Kelly, Paul HJ and Davison, Andrew J and Luján, Mikel and O'Boyle, Michael FP and Riley, Graham and others, *Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM*, arXiv preprint arXiv:1410.2167, 2014 .

[12] *PAMELA Project*,

`http://apt.cs.manchester.ac.uk/projects/PAMELA/about/index.html`, [Retrieved 12-06-2016].

[13] Baker, Chuck A and Grossman, Bernard and Haftka, Raphael T and Mason, William H and Watson, Layne T, *HSCT configuration design space exploration using aerodynamic response surface approximations*, Proc. of 7th AIAA/USAF/-NASA/ISSMO Symp. on Multidisciplinary Anal. and Optim, 769–777, 1998

[14] Saltelli, Andrea. *Sensitivity analysis for importance assessment.* Risk Analysis 22.3, Pages 579-590, 2002.

[15] Fornaciari, William, Donatella Sciuto, Cristina Silvano, and Vittorio Zaccaria. *A sensitivity-based design space exploration methodology for embedded systems.* Design Automation for Embedded Systems 7, no. 1-2: 7-33. 2002

[16] R. C. Baker, *Design Of Experiments*,

`http://wweb.uta.edu/insyopma/baker/QUALITY%20MASTER` ...

`/DOEpp%20Reduced%20PRESENTATION5.ppt`, [Retrieved 12-06-2016].

[17] Zia, M. Zeeshan, Luigi Nardi, Andrew Jack, Emanuele Vespa, Bruno Bodin, Paul HJ Kelly, and Andrew J. Davison., *Comparative Design Space Exploration of Dense and Semi-Dense SLAM.* arXiv preprint arXiv:1509.04648, 2015.

[18] Luigi Nardi, M. Zeeshan Zia, Paul H. J. Kelly, Bruno Bodin. *Design exploration across domain and implementation in dense 3D scene understanding*, Review Paper, 2015.

[19] , Dr Ben Glocker, Machine Learning, Medical Image Computing, 407H, `http://www.imperial.ac.uk/computing/current-students/courses/407H/`, 2015

[20] *CUDA*,

`http://www.nvidia.com/object/cuda_home_new.html`, [Retrieved 12-06-2016].

[21] *Creating an OpenGL Context*,

`https://www.opengl.org/wiki/Creating_an_OpenGL_Context_(WGL)`, [Retrieved 12-06-2016].

[22] John MacCormick, *How does the Kinect work?*, http://users.dickinson.edu/ jmac/selected-talks/kinect.pdf

[23] Tracy Counts. *RAPL*,

https://01.org/blogs/tlcounts/2014/running-average-power-limit---rapl 2014, [Retrieved 12-06-2016].

[24] Sturm, Jrgen, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. *TUM Computer Vision Group - Useful tools for the RGB-D benchmark*, http://vision.in.tum.de/data/datasets/rgbd-dataset/tools, 2012, [Retrieved 12-06-2016].

[25] Sturm, Jrgen, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. *TUM Computer Vision Group - Useful tools for the RGB-D benchmark*, http: //vision.in.tum.de/data/datasets/rgbd-dataset/file_formats, 2012, [Retrieved 12-06-2016].

[26] Sturm, Jrgen, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. *A Benchmark for the Evaluation of RGB-D SLAM Systems*, In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 573-580. IEEE, 2012.

[27] A. Handa and T. Whelan and J.B. McDonald and A.J. Davison, *A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM*, IEEE Intl. Conf. on Robotics and Automation, ICRA, May 2014.

[28] A. Handa and T. Whelan and J.B. McDonald and A.J. Davison, *SurfReg: Automatic surface registration tool for ICL-NUIM datasets*,

https://github.com/mp3guy/SurfReg, April 2015, [Retrieved 12-06-2016].

http://www.dept.aoe.vt.edu/ mason/Mason_f/AIAA98-4803.pdf

[29] Whelan, Thomas, et al. *ElasticFusion*, https://github.com/mp3guy/ElasticFusion 2015, [Retrieved 12-06-2016].

[30] Whelan, Thomas, et al. *Logger1*, https://github.com/mp3guy/Logger1, 2014, [Retrieved 12-06-2016].

[31] Whelan, Thomas, et al. *Logger2*, https://github.com/mp3guy/Logger2 2015, [Retrieved 12-06-2016].