

# Thesis

Tom Zurales

July 2025

## Abstract

This research details the implementation and analysis of a novel, viewpoint aware method for map point culling for use in keypoint based visual SLAM systems. This method makes use of perspective dependent shells around each map point, allowing for the storage of overall observability metadata using constant additional space. This metadata allows for the overall probability of a point’s existence to be continuously calculated using a simple Bayesian update step. This existence probability can be used in a myriad of ways. This research explores its use as a method of culling outdated map points, such as those originally seen on an object which has since moved, and as an extension to the RANSAC algorithm, providing the ability to select more robust map points. We provide the implementation of the perspective aware metadata shell as an open-source library, as well as an ORB\_SLAM3 implementation utilizing the library for point culling and RANSAC improvement. Additionally, a set of co-registered visual-inertial-lidar datasets are released, containing scenarios specifically intended to exercise and characterize the performance of the system. Through our analysis, (discuss effects on well known, non-dynamic SLAM datasets, along with my datasets)

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Objectives and Scope . . . . .	5
1.2	Contribution . . . . .	5
1.3	Road Map . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	SLAM Overview . . . . .	7
2.1.1	The SLAM Pipeline . . . . .	7
2.1.2	Keypoint-Based Visual SLAM . . . . .	8
2.1.3	Extensions to Core SLAM . . . . .	8
2.2	Additional Fields of Research . . . . .	8
2.2.1	Directional Probability . . . . .	8
<b>3</b>	<b>Related Work</b>	<b>9</b>
3.1	Point Removal Optimizations . . . . .	9
3.1.1	Semantics Based Implementations . . . . .	9
3.1.2	Probability Based Optimizations . . . . .	9
<b>4</b>	<b>Implementation</b>	<b>10</b>
4.0.1	REWORDING . . . . .	10
<b>5</b>	<b>Experimental Analysis</b>	<b>16</b>
5.1	Dataset Creation . . . . .	16
5.1.1	Hardware . . . . .	16
5.1.2	Structure . . . . .	16
5.1.3	Dataset Overview . . . . .	16
5.2	Evaluation Metrics . . . . .	16
5.3	SLAM System Configurations . . . . .	16

5.3.1	Parameter Tuning . . . . .	16
5.4	Results . . . . .	16
5.4.1	Quantitative Evaluation . . . . .	16
5.4.2	Qualitative Evaluation . . . . .	16
5.4.3	Ablation Study . . . . .	16
<b>6</b>	<b>Discussion</b>	<b>17</b>
<b>7</b>	<b>Conclusion and Future Work</b>	<b>18</b>
	<b>Appendices</b>	<b>19</b>
	Algorithm Implementations . . . . .	19
	Full Results . . . . .	19

# 1 Introduction

This section provides an introduction to the limitations of SLAM systems, and discusses how the research presented in this thesis intends to solve those shortcomings.

## 1.1 Objectives and Scope

This research intends to build upon the previously developed probability models, in order to distill the update step of each map point’s probability of existence into a simple Bayesian update step. The goals for this model are as follows:

1. To utilize constant additional space for each map point
2. To complete the update step in constant time
3. To resist updating confidence levels with redundant data

## 1.2 Contribution

Through this research, we introduce an incrementally updated directional confidence model for the existence of map points. This model differs from other point removal optimizations in several ways. First, this implementation avoids the use of neural networks, facilitating use on resource constrained hardware without facilities optimized to run them. Second, while other probability based point removal optimizations have been developed, this model introduces the idea of utilizing a continuously updated perspective dependent shell of metadata for each keypoint, which can be used to reduce the problem of point existence to a simple Bayesian update step. This implementation allows perspective of observation to play a role in the point’s existence probability update step, and avoids some of the common a priori work such as prior estimation common to other point removal optimization techniques. This shell is implemented in both finite and continuous modalities, utilizing regular convex polyhedral shells in the finite case, and von-meiser fisher distributions on the sphere in the continuous case.

To facilitate future research, this model is released as an open-source library, which is

compatible with any keypoint based visual SLAM implementation. Additionally, a collection of co-registered visual-inertial and LIDAR datasets is provided, containing instances of multiple traversals through the same environments with changes to scene contents. Information regarding the locations of these environmental changes is included in the dataset, facilitating the benchmarking of point removal optimization implementations.

### **1.3 Road Map**

Chapter X of this thesis discusses the background of the general SLAM problem, covering the history, use cases, and general pipeline utilized by SLAM systems. This is followed by a deeper dive into keypoint-based visual SLAM, the sensor modality targeted by this research. We provide a brief survey of widely utilized extensions to the core SLAM algorithm which target deficiencies in the core pipeline. Finally, we discuss fields outside the scope of SLAM which provide insight and methodology into this research.

Chapter X discusses works related to this research, specifically focusing on extensions targeting improved performance in dynamic situations, with additional focus given to those methods which utilize point removal optimizations.

## 2 Background

In this chapter, we provide a high level overview of the definitions, objectives, and history of SLAM, in addition to an overview of the common sensor modalities found in SLAM systems. Following this, we discuss the stages of the SLAM pipeline in the generic case, followed by a more in-depth exploration of keypoint-based visual SLAM, the sensor modality targeted by this research. Next, we look into several of the widely adopted extensions to the base SLAM pipeline which address core issues and enhance performance. A discussion of extensions which have similar goals to this research is held for the chapter on related works.

### 2.1 SLAM Overview

This research is acts as an extension to keypoint-based visual SLAM; a term which warrants some explanation. But before exploring the specifics of keypoint-based visual SLAM, some background on the general SLAM problem is required. The idea behind SLAM is to simultaneously produce a map of an environment, and determine the position of the observer within the map based on a set of sensor data. The process differs heavily based on the sensor types being utilized. For example, LIDAR provides a direct measurement of 3D distances from the sensor, while an RGB camera must calculate them from correspondences between multiple frames. While implementations differ heavily, a common SLAM pipeline could be described as follows:

#### 2.1.1 The SLAM Pipeline

This stage is responsible for the creation of the initial map.

There have been hundreds of SLAM implementations for a wide variety of sensors, commonly targeting combinations of monocular, stereo or RGBD cameras, IMUs, LIDARs, etc.

Due to it providing the motivation for this project, the Astrobbee robots will be mentioned several times throughout this work. The Astrobbee project was motivated by the desire

to research human/robot interaction, robotic automation and inspection, and to provide a research platform on which companies and researchers could deploy software and hardware for testing in a micro-gravity environment. The Astrobees platform has been used to develop satellite rendezvous control algorithms, grippers to capture tumbling orbital debris, inspection methods to autonomously detect anomalous operation, and many other space habitation focused endeavors.

### **2.1.2 Keypoint-Based Visual SLAM**

The term Keypoint-Based Visual SLAM refers to the SLAM modality which primarily utilizes key points extracted from images as the primary means of mapping and navigating. This is distinct from systems like LIDAR-based SLAM, which utilize direct distance measurements from a LIDAR sensor, or Dense Visual SLAM, which

### **2.1.3 Extensions to Core SLAM**

## **2.2 Additional Fields of Research**

### **2.2.1 Directional Probability**



## 3 Related Work

Numerous methods have been developed to improve long-term SLAM performance, generalizing the problem to remove the constraint that the environment remains static. <https://arxiv.org/pdf/2209.1>

- ChangingSlam - Uses a Bayes filter to remove changing map points. Utilizes semantic detection to determine dynamic objects before filter is used, which prevents it from working on deformable objects.

### 3.1 Point Removal Optimizations

#### 3.1.1 Semantics Based Implementations

#### 3.1.2 Probability Based Optimizations

## 4 Implementation

A probability model for the elimination of inaccurate map points For each new frame which comes in, the tracking thread completes its estimation of the new pose, based on the motion model, the local map, or the reference keyframe. This pose is then optimized to improve local consistency. Post localization, the estimation of the current pose will be as good as possible based on the currently available information. This estimated pose, along with the set of map points in the map are then processed to update each map point’s probability of existence. Step one involves projecting each point to the camera’s image plane, determining the set of map points which should be visible given the currently estimated pose. We can then split these “expected” map points into two groups: those that are seen, and those that are not seen. For the map points which are seen, we set their probability of existence to 1, since we are sure they still exist. Next, their `icosFaceProbs` is increased using Bayes theorem to indicate that the point is visible from that perspective. Finally, the nearest vertex in `icosVertexDists` is set to the current distance between the camera and the map point, if it’s larger than the current setting. If all map points associated with the face are 0, we do something else I guess. If the point is not seen, things are a bit more interesting. First of all, the `icosFaceProbs` is updated using Bayes theorem, this time decreasing it. If the point’s overall probability drops enough for a specific face, the vertex distances begin to drop, in the hopes of moving to the inside of any obstructions. This prevents observations from one perspective from having too much power over the overall probability of existence. The distance is dropped by  $\frac{1}{2}$ , and the face probability is set to the average of the adjacent face’s probabilities.

### 4.0.1 REWORDING

For each observed map point, we require the ability to determine both the perspectives from which it can be observed, and the maximum distance from that perspective at which the point can be observed. This effectively eliminates instances where a point is observable from a

perspective while in front of a wall, but not while behind the wall.

### Icosahedron Observability Shell

An icosahedron was selected as the geometry for the observability model because it has the highest number of faces for a convex, regular polyhedra. Effectively, by using 12 parameters for vertex distances MAYBE CHANGE TO 20 FACE DISTANCES LATER and 20 parameters for face weights, we are able to estimate the probability of observing a map point from every possible perspective and distance. This probability is utilized in a Bayesian update step to modify the map points overall probability of existence.

### Observability Shell Construction

A standard construction of an icosahedron utilizes three perpendicularly intersecting rectangles of ratio  $1:(1+\sqrt{5})/2$ . The vertices of these rectangles become the vertices of the icosahedron, and the edges are formed from the short edges of the rectangles, and regular triangles drawn between them. A consistent naming scheme is needed for the vertices and faces, and can be arbitrarily decided. For simplicity, rectangles lying on the coordinate planes and centered on the origin are utilized for the construction. ORB-SLAM3 utilizes a right handed coordinate system based on the first frame of a new map with x pointing right in the image plane, y pointing down, and z pointing forward. Figure XX shows the relationships between the coordinate frames, construction rectangles, vertex names, and face names utilized for this construction. Placing this icosahedron shell around a map point is accomplished by adding the map point's global coordinates to the vertex coordinates.

### Continuous Observability Shell Evaluation

Simulation methods for the quantification of the effect of inaccurate maps on tracking and relocalization

### Before implementing the probability model,

significant work can be done through simulation to determine the degree to which an outdated map affects SLAM performance. I generated several 3D environments, and several visual-inertial SLAM datasets through these environments for the Dataset generation

### Dataset generation

takes place in three steps. The first step is Path Recording, followed by Dynamics Recording, and finally, sensor generation. These steps are described in detail below.

### Path Recording

In this step, a world (SDF) is selected, and the simulated robot is driven through it. At each frame update, the controls provided by the user are recorded

for future playback. The output of this step is a directory structure containing a copy of the world, and the file containing the recorded path. Dynamics Recording At this stage, the world is reloaded, and the paths are played one-by-one. The user can move simulated objects which are part of the world around, along with loading new objects to place around the world. Care is taken to avoid these objects getting in the way of the paths. The output of this stage is a patch containing the ‘diff’ between the original world and the modified world. A manual step is required to add the “dynamic” tag to the moved/modified objects.

Sensor Generation The final step is to generate the simulated sensor readings which will be fed to the set of SLAM systems under test. For each combination of path and dynamics file, a stereo image stream, segmentation image stream, IMU stream and ground truth position stream are recorded to the world’s directory. Combination of pre-generated environments and custom generated environments using free assets Objects which are moved in future runs are marked “dynamic”, allowing them to be identified by the simulated segmentation camera

Multiple runs though the same environments with different levels of change from the baseline for different difficulties. Expect to see higher levels of inaccuracy when environment changes significantly between a run and its map generation Camera and IMU are attached to a simulated robot capable of 5dof motion in 3d environments. Evaluation of base performance

Evaluation of an “ideal” probability model The best possible performance of the model can be simulated by eliminating all keypoints which are moved between an initial environment and a modified environment. This is achieved using a segmentation camera as part of the dataset. Determining which keypoints fall on dynamic objects during the initial map generation allows those keypoints to be culled prior to loading the map for subsequent runs. This provides a simulacrum of an idealized “perfect” result of the probability model by using foreknowledge of which keypoints would ideally be marked as out-of-date. Limitations This model for “perfect” performance is capable of identifying objects in the original dataset which are moved or deleted, but is not capable of identifying objects which become occluded by new or existing objects. The decision could be made that objects which become occluded

should be marked “dynamic”, but this would raise issues if the occluded object remains visible from some perspectives contained in one or more test paths. It could be decided that occluded objects are marked “dynamic” if they are not seen due to the occlusion in a subsequent test path, but this would require additional decisions to be made (e.g. What should be done about objects which would not be seen by a path even if no occlusion exists? To what extent does an object need to be seen in order to avoid the “dynamic” mark?) For these reasons, objects may become occluded by other objects in a scene, but will not be culled by the ideal model. Test Plan

Implementation of a probability model for the elimination of outdated map points Definitions The following functions and parameters are being defined here for use in the several probability model implementations discussed below.

Implementation 1 The goal of this probability model is to estimate the probability that a keypoint

$$kp_n \in map_{initial}$$

exists at time

$$t$$

, given the current sensor input

$$Z_t$$

and state estimate

$$x_t$$

. Bayes’ theorem is used to determine a new probability for the existence of

$$kp_n$$

, written as:

This equation makes several assumptions: If

$$kp_n$$

is observed by

$$Z_t$$

, then we can be sure it exists. The probability of a keypoint's existence will not change without some observation of the area where

$$kp_n$$

is expected to exist. This implies that if we would not expect

$$Z_t$$

to observe

$$kp_n$$

(based on

$$kp_n$$

's last observed location and the current

$$x_t$$

),

$$p(\textbf{exists}(kp_n, t))$$

will not be changed. This means we can fix

$$p(\text{exists}(kp_n, t)) = p(\text{exists}(kp_n, t - 1))$$

. Each of these assumptions will be interrogated and analyzed to determine their validity in simulated and real world examples. Implementation 2 The assumption that the probability of a keypoint's existence does not change as a function of time may be false. This implementation makes use of the time between the loaded map's generation and the current time to increase the probability that a keypoint no longer exists based on the time delta to when it was last observed. Evaluation of probability model on simulated and real-world datasets

!– This is going to contain a section on comparisons with other methods of dynamics removal. This is difficult to do in practice due to the complications of using other people's numbers in direct comparison to my own. Add a section on potential research into context independent slam benchmarking. –j

## 5 Experimental Analysis

### 5.1 Dataset Creation

#### 5.1.1 Hardware

#### 5.1.2 Structure

#### 5.1.3 Dataset Overview

### 5.2 Evaluation Metrics

### 5.3 SLAM System Configurations

#### 5.3.1 Parameter Tuning

### 5.4 Results

#### 5.4.1 Quantitative Evaluation

#### 5.4.2 Qualitative Evaluation

#### 5.4.3 Ablation Study



## 6 Discussion

## 7 Conclusion and Future Work

# Appendices

## Algorithm Implementations

## Full Results

test