3D SLAM for Powered Lower Limb Prostheses

Manan K. Shah

CMU-RI-TR-21-21

June 4, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Hartmut Geyer, PhD, *chair*
Aaron Johnson, PhD
Ashwin Khadke

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

During locomotion, humans use visual feedback to adjust their leg movement when navigating the environment. This natural behavior is lost, however, for lower-limb amputees, as current control strategies of prosthetic legs do not typically consider environment perception. With the ultimate goal of achieving environment-awareness and adaptability for prosthetic legs, we here seek as an initial step to take advantage of the rapid development in affordable hardware and advanced algorithms for simultaneous localization and mapping (SLAM) that has fueled the integration of perception in the control of mobile robots including legged machines, and propose, implement and analyze a SLAM integration for a lower limb prosthesis. To this end, we first simulate the motion of a range sensor mounted on a prosthesis and investigate the performance of state of art SLAM algorithms subjected to the rapid motions seen in lower limb movements. Our simulation results highlight the challenges of drift and registration errors stemming from the dynamic motion sensor. Based on these observations and knowledge about the walking gait, we then implement a modified SLAM pipeline in hardware that uses gait phase information to bypass these challenges by resetting the global map and odometry at the beginning of each stride. This pipeline uses an RGB-D camera to perform a dense reconstruction of the terrain directly in front of the prosthesis using a colored point cloud registration algorithm. In preliminary tests with one able-bodied subject, we find the algorithm creates dense representations of multiple obstacles with an accuracy of 11mm, while simultaneously tracking the camera pose with an accuracy of 19mm. Although we conducted the hardware experiments with the registration algorithms running offline, our results suggest that SLAM methods can be implemented on lower-limb prostheses with sufficient accuracy to enable environment perception, opening up avenues for the development of advanced control strategies of prosthetic legs that more proactively adapt to changes in the environment and, thus, unburden their amputee users.

Acknowledgments

I would first like to thank my advisor Professor Hartmut Geyer, who has been extremely helpful and patient with me while also challenging me to learn and perceive problems from a scientific approach. I would also like to thank all the members of our lab, Ashwin Khadke, Russell Xing, Saurav Kumar, Aditya Sripada, Omar El Sayed for the incredible collaborative learning environment they maintain within the group. I would like to give a shout out to Professor Aaron Johnson for his feedback and for bringing together all of us students interested in legged systems through his locomotion seminars. Lastly, I would like to thank my parents for always being pillars of supports and my guiding stars, and all other family and friends for never being more than a phone call away.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

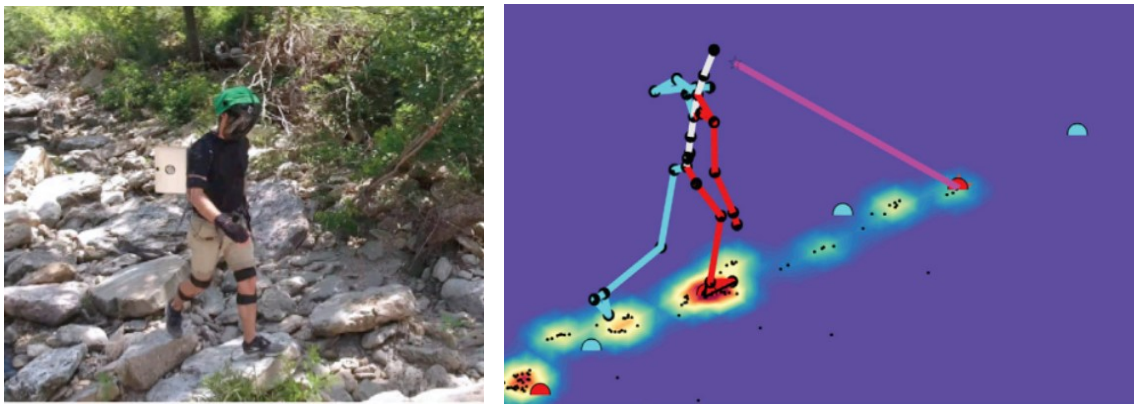
Chapter 1

Introduction

1.1 Motivation

Currently, over 1.6 million people are living with a lower-limb amputation in the United States, with this number estimated to double by 2050 [9]. The common causes for people to undergo amputations are traumatic injuries from accidents and casualties on the battlefield. However, the most common cause of lower-limb amputations is diabetes mellitus. In 2016, the Center for Disease Control and Prevention reported 130,000 hospitalizations for diabetes-related lower limb amputations among adults [10].

Passive energy restoration-based prostheses do not replicate the positive power output evident in the knee and ankle joint during different activities such as walking, running, climbing, etc. [11, 12, 13]. Hence, there has been notable progress in the research and development of control strategies for powered lower-limb prosthetic devices. These strategies are based on trajectory tracking [14], virtual impedance



(a) Subject walking on rough terrain (b) Reconstructed skeleton and gaze vector

Figure 1.1: Subject walking on rough terrain with cameras attached to track eye movement, and reconstructed skeleton and 3D gaze vector [1].

[15], phase variables [16], and neuro-muscular models [17]. The state of the art in all these control strategies predominantly attempt to reproduce normal locomotion and gait cycles [18, 19]. Due to this emphasis on achieving control over nominal gait, these strategies have little to no robustness to challenging environments.

Adding awareness about the environment is an important step to enable dynamic behavior in prostheses. [1] investigates human gaze while walking on different terrains by attaching cameras to track eye movement, shown in figure 1.1. The results demonstrate constant forward gaze during locomotion, with the look-ahead distance reducing with increasing roughness in the terrain. This awareness of the terrain in the immediate future plays an important role in the planning of the next step, hence ensuring maintenance of balance and stability in spite of varying terrain and obstacles in the path.

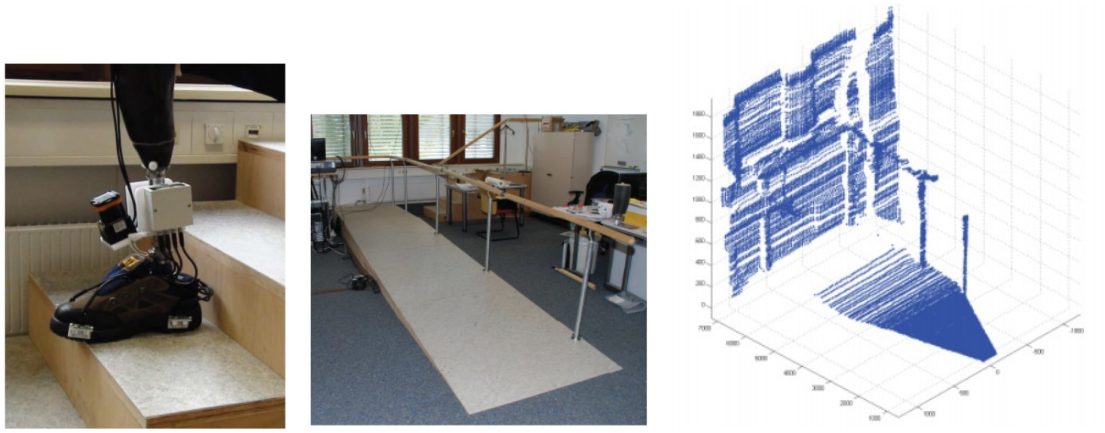
Here, we investigate the work done to date in integrating environment sensing modalities with prostheses and propose a new approach borrowing techniques from Simultaneous Localization and Mapping (SLAM). We have implemented a SLAM

pipeline specifically tailored to fit the application of powered prostheses control by generating a dense reconstruction of the immediate terrain. To the best of our knowledge, we are the first to attempt an implementation of SLAM to increase the adaptability of lower-limb prostheses. Chapter 2 provides a brief review of popular SLAM methods, specifically methods which construct a dense map of the environment. It also highlights work done on implementing SLAM for legged systems and their relevant challenges. Chapter 3 describes simulations of a simple SLAM system with a range sensor attached to a prosthesis, which highlights these challenges. This chapter also presents a custom SLAM pipeline designed after evaluating simulation results. Lastly, it describes the experimental setup to test the proposed approach in hardware on an able bodied subject, including details about software and hardware components used. The results from these experiments are presented and discussed in Chapter 4. Lastly, the impact of these results and future work in this field are discussed in Chapter 5.

1.2 Prior Work

There is a sizeable volume of work that has been done in the past decade to increase environment sensing capabilities of prostheses (and exoskeletons). [20] reviews over 30 different publications that explore integration of various sensing modalities on such devices, ranging from cameras, LIDARs, RADARs, depth cameras combined with Inertial Measurement Units (IMU) and force sensors. The methods and findings from some notable publications are outlined below to show the evolution over time in the richness of the information generated from sensing the environment.

In [2], a 2D laser range-finder was attached to the tibia of a patient wearing an

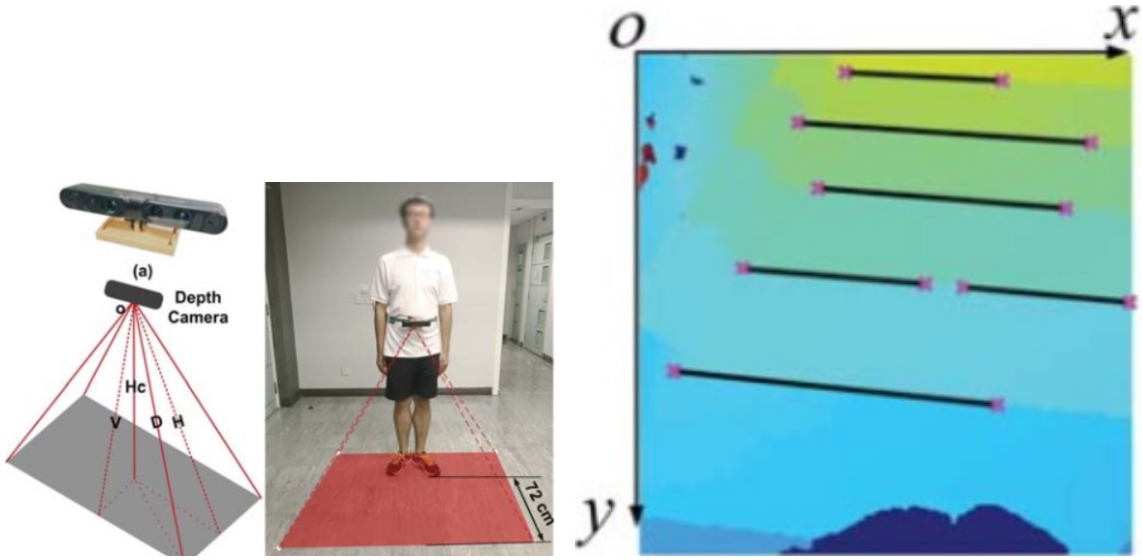


(a) Sensor attachment (b) Test environment (c) Environment map

Figure 1.2: [2] 2D laser scanner attached to tibia, the testing environment, and the map generated by combining laser scans with IMU odometry

ankle prosthesis. In standing position, the plane of the laser scan is parallel to the transverse plane of the patient. An IMU is used to estimate the odometry of the sensor, and these odometry estimates are used to transform consecutive scans and stitch them together to obtain a 3D map of the environment. The attachment point of the sensor, testing environment, and the generated map are shown in figure 1.2. One major limitation of this method is its dependence on an IMU which can generate inaccurate odometry due to noise and drift, leading to inaccuracies in the map of the environment.

In [3], a Xtion Pro Live depth camera is mounted to the waist, facing down. The depth measurements are passed through a thresholding algorithm. The resulting image is then used to classify the terrain and the locomotion mode into 5 discreet classes: flat ground, stairs (up/down), and ramps (up/down). Additionally, a Hough transform is applied on the depth image to detect straight edges of stairs. The camera attachment, camera field of view, and the processed image with stair edges detection

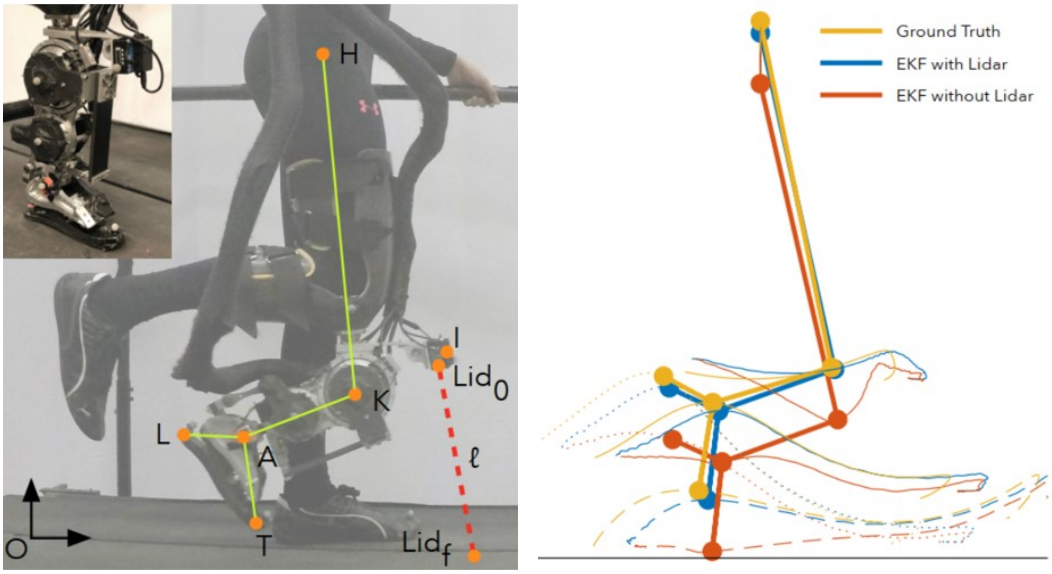


(a) Depth sensor attached to waist, pointing down (b) Processed image

Figure 1.3: [3] Depth camera mounted to waist, and thresholded depth image with stair edges detected

are shown in figure 1.3. The information gathered about the environment in this approach is lacking in richness, as it simply classifies the terrain into 5 discreet classes. Additionally, it is undesirable to have an additional sensor attached to the waist instead of integrating all sensors into the prostheses allowing it to be an isolated, self-contained system.

In [4], data from a 1D LIDAR was fused with IMU data using an Extended Kalman Filter (EKF) to accurately localize the sensor and hence the entire prosthesis (using joint encoders) in real-time. The sensor attachment is shown in figure 1.4a and the localization compared to ground truth is shown in figure 1.4b. The LIDAR data is used in the update step of the EKF, assuming the prosthesis is traversing on flat ground. This localization was then used to update the swing trajectory in real-time to avoid premature contact between the toe or the heel and the ground, reducing the number of tripping events. While achieving impressive performance

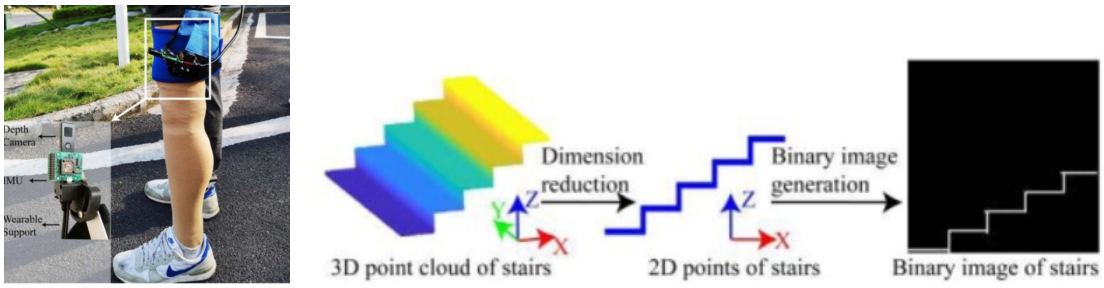


(a) 1D LIDAR + IMU attached above knee (b) Comparison of localization with ground truth

Figure 1.4: [4] 1D LIDAR + IMU attached above knee and prosthesis localization using EKF on IMU vs. EKF on IMU + LIDAR on flat ground walking.

to reduce the number of trips based on accurate localization, this system does not provide additional information about the terrain, and assumes the terrain to be flat ground.

One of the most recent publications in this direction extracts richer information about the terrain using a depth camera [5]. The depth camera is mounted on the unilateral thigh, as shown in figure 1.5a. First, the 3D point cloud recorded by the camera is rotated to align with the inertial frame using the leg angle in the sagittal plane estimated by a complimentary filter applied to an IMU. Next, this 3D point cloud is reduced to a 2D binary image of the terrain, which is classified into 5 discreet classes as above (flat, stairs, ramp) using a convolutional neural network, as shown in figure 1.5b. Additionally, traditional RANSAC-based computer vision algorithms are used to estimate feature parameters, namely, slope angle for ramps and step height



(a) Depth camera + IMU attached above knee

(b) 3D point cloud of stairs reduced to binary image

Figure 1.5: [5] Depth camera + IMU mounted to above knee, and 3D point cloud reduced to binary image which is then classified by CNN

and width for stairs. This implementation obtains slightly richer but still discreet information about the environment in the form of feature measurements for discreet terrain classes which can be used to switch the locomotion modes of the prostheses.

It is desirable to obtain a dense 3D map of the environment, which, when combined with foot-placement and trajectory planning algorithms, can allow for more robust and dynamic control of the prosthesis in variable terrain. This would also require accurate localization of the system in its environment. The field of Simultaneous Localization and Mapping (SLAM) has taken massive strides forward in solving this problem for mobile systems. We believe that research of prosthesis control stands to benefit from the recent developments in SLAM. The following section provides a brief overview of some popular SLAM techniques from seminal publications, specifically those relevant to this work.

Chapter 2

SLAM Background

2.1 Current Applications of SLAM

Simultaneous Localization and Mapping (SLAM) has become one of the primary areas of focus in robot perception in recent years. When a mobile robot is placed in an unknown environment, it is important to incrementally build a map of the environment while also estimating the robot's location and pose in this environment. Solving this problem is imperative before planning actions to accomplish any tasks. This is the problem solved by SLAM. Recently, there has been a rapid growth in the mobile robots industry, from drones to autonomous vehicles. A major contributing factor to this has been improvements in perception hardware and algorithms. iRobot's Roomba Robot Vacuums use visual SLAM to maintain a map of an indoor environment to plan an efficient cleaning route around the room [1]. Self-driving vehicles with varying

[1] www.irobot.com

levels of autonomy from companies like Waymo [2] and Tesla [3], industrial and outdoor robots like Boston Dynamics' Spot [4], as well as surveillance drones from Skydio [5] and American Robotics [6], are some examples of commercial products enabled by developments in SLAM. With the development of methods that generate dense representations of the environment (such as the ones described later in this chapter), SLAM has also found applications in augmented reality. These methods are able to generate dense maps and surfaces to represent the environment. Access to such dense representations of the environment with accurate localization of the system can enable more robust foot placement and swing trajectory planning for a lower-limb prosthesis. Hence, adaptation and implementation of SLAM is the next logical step in the field of powered prosthesis control research.

2.2 Popular Dense SLAM Algorithms

To help familiarize the readers, some popular SLAM algorithms that yield a dense representation of the environment and are relevant to our implementation for a prosthesis are summarized below.

KinectFusion [6] was one of the first attempts at dense volumetric scene reconstruction and sensor tracking. It generates a dense vertex and normal map from the raw depth measurements obtained from a Kinect, an infrared-based time-of-flight depth sensor. It then uses a point-to-plane iterative closest point (ICP) algorithm to map live depth frames to a globally fused map. This constructs 3D map while also

[2] www.waymo.com
[3] www.tesla.com
[4] www.bostondynamics.com
[5] www.skydio.com
[6] www.american-robotics.com

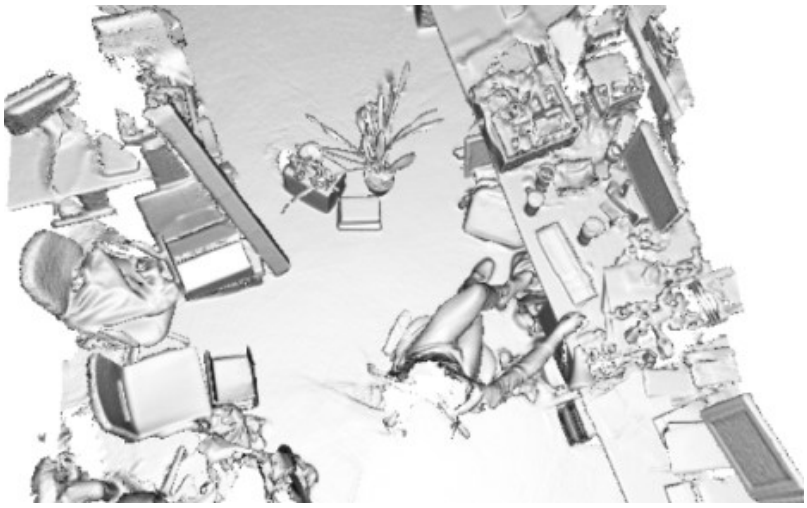


Figure 2.1: Large scale indoor reconstruction obtained in real time by KinectFusion [6]

tracking sensor odometry. The rigid body transform for each new source depth frame is computed by minimizing the energy equation shown in equation 2.1 [6] below.

$$\boldsymbol{E}(T_{g,k}) = \sum ||(T_{g,k}\dot{\boldsymbol{V}}_k(\boldsymbol{u}) - \boldsymbol{V}_{k-1}(\hat{\boldsymbol{u}}))^T \hat{\boldsymbol{N}}^g_{k-1}(\hat{\boldsymbol{u}})||_2 \tag{2.1}$$

Here, $\dot{\boldsymbol{V}}_k$ represents the vertices at time k, while $\dot{\boldsymbol{V}}_{k-1}$ represents the corresponding vertices from the previous frame, with $\hat{\boldsymbol{N}}^g_{k-1}$ being the normals. $T_{g,k}$ represents the transform for the global pose estimate for frame k. The term inside the summation is simply the distance of each new vertex from its corresponding plane in the global map. Since this system tracks live depth frame against the globally fused map, unlike frame-to-frame pose estimation, it is also able to perform some loop closure at small scales. Loop closure is a system's ability to assert that it has previously visited the current location, and correct for any drift in localization and mapping based on this assertion. Figure 2.1 [6] shows a dense reconstruction of an indoor scene obtained in real-time using KinectFusion.

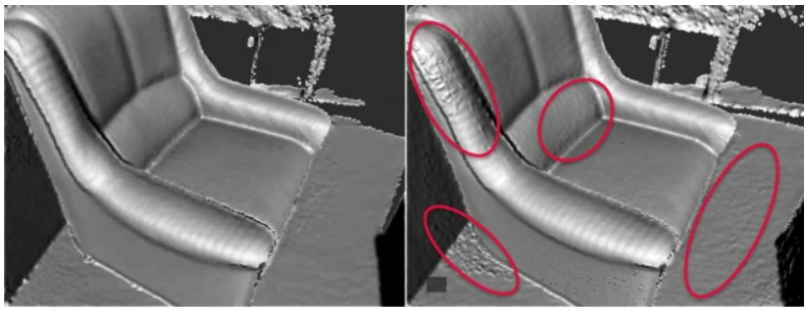


Figure 2.2: Weighted averaging for point-fusion from [7] yielding less noisy output map (left) compared to KinecFusion [6] (right)

Point-based fusion [7] builds on KinectFusion with a couple of key improvements. In this system, the camera pose estimation uses a hierarchical ICP to align the input depth frame with the global map. It uses three hierarchy levels, starting from course downsampling and ending with the finest level at the camera's resolution. The biggest improvement of this system is the technique of fusing new depth maps with the global map, once the camera pose has been estimated. Every point in the map evolves from unstable to stable status based on its confidence score. Each new point is merged or fused with its closest correspondence from the global map, using a weighted average. The confidence score of this fused point is updated. If no correspondences are found, then the point is added to the global map as an unstable point. This method of modeling sensor uncertainty yields better denoising, as shown in figure 2.2. Building on this simple approach, the system is also able to perform accurate reconstruction in dynamic scenes, by marking outliers from the ICP algorithm as unstable.

Another popular visual-SLAM algorithm from recent years is Real-Time Appearance-Based Mapping (RTAB-Map) [21, 22, 23], which is also available as an open source ROS package [24]. The most important contribution of RTAB-Map is an efficient loop-closure technique that is independent of time and map size, which allows large-scale

and long-term operation of a mobile system. RTAB-Map uses SURF [25] descriptors to derive visual words for each image and uses a bag-of-words [26] approach to create a signature of each image, which defines a unique location. It then uses a discrete Bayesian filter to estimate the probability of the current location matching any of the previously visited locations, and hence make loop closure hypotheses. Long-term and large-scale performance independent of time and map size is achieved by smart memory management for past locations. RTAB-Map uses a probabilistic approach to store more frequently visited locations in its working memory, while storing the remaining locations in long-term memory. Maintaining an efficient working memory allows it to perform long-term real-time loop closure without increasing computation time. This method can be adapted to a variety of sensors including stereo cameras, lidar sensors, RGB-D cameras.

An in depth comparison of various SLAM systems based on RGB-D sensors with respect to their underlying algorithms is provided in [27]. While these systems have their limitations in terms of robustness and required computation for real-time performance, they show great promise for solving the problem of real-time 3D reconstruction. The data sets these systems are evaluated on have trajectories of hand-held sensors [28], [29] or similar sensors attached to a wheeled mobile robot. While these trajectories have some vibrations, they do not exhibit the dynamic oscillations (especially pitch) which would be expected on a legged system, exacerbating the constraints of field of view and motion blur of recorded images.

2.3 SLAM for Legged Systems

SLAM and environment sensing for legged robots and systems has been studied to some degree in the last few years. In [30], a 2D LIDAR scan is segmented to detect objects with a rectangular cross section that a four-legged robot has to jump over. Performance of a total of four state-of-the-art RGB-D and visual-only SLAM systems on a six-legged platform are evaluated in [31]. As expected, dynamic, unpredictable sensor motion such as increased vibrations and oscillations prove to be the major challenge in accurate localization and trajectory tracking of the sensor. When the SLAM pipelines were modified to either capture frames when at least five of the six legs were in contact with the ground, or when the gait was adapted to be slower, the error in trajectory estimation were significantly reduced. This indicates the necessity to couple the SLAM pipeline with gait control. More recently, [32] implemented a LIDAR-SLAM system that performs ICP-based scan registration for dynamic motion of a sensor mounted on a four-legged mobile robot, using IMU and encoder measurements recorded at a high sampling rate as an initial guess for the ICP algorithm. These methods evaluated in both, [31] and [32] are aimed at obtaining a sparse map of the environment, but more importantly, get accurate trajectory estimation and loop closures. The reconstructed map is used in planning the path of the system, not for adjusting the gait of legs. Lastly, a notable recent example of a SLAM system used in a bipedal system is [33], which also attempts to address the challenges of SLAM systems due to dynamic motion of an RGB-D sensor resulting from bipedal locomotion. The system discards image pairs with an estimated blur score above a certain threshold, using methods from [34].

Despite such widespread use of SLAM systems in a variety of use cases, they have

not found their way into assistive legged systems like prosthetics and exoskeletons, as seen in the broad review done by [20] on environment sensing capabilities of such systems. For a prosthesis, it is desirable to integrate all sensors into the device, as opposed to requiring users to attach additional sensors elsewhere on the body, for example on the torso. This also allows the prosthesis to be an isolated mobile system free from requiring repeated sensor calibrations. However, a sensor such as a LIDAR or a depth camera mounted to a lower-limb prosthesis would go through higher accelerations and oscillations, compounding the challenges faced by SLAM implementations summarized above where the sensors were mounted to the trunk of a legged system. In the next chapter, we describe the reasoning behind selecting an attachment point for the sensor on the prosthesis. We then simulate a simple 2D range finder going through motion that would be expected at this attachment point to observe the performance of a scan matching SLAM algorithm. Based on these results, we propose a custom SLAM pipeline for a sensor mounted on a prosthetic device and share our experimental setup to test the performance of this pipeline.

Chapter 3

Proposed Approach

As mentioned in the previous chapter, it is desirable to mount any additional sensors on the prosthesis itself, instead of requiring users to attach sensors elsewhere on the body. The attachment point for the sensor on the prosthesis should be carefully selected. To alleviate the challenges stemming from dynamic sensor motion, the attachment point should have the lowest possible velocity and acceleration during normal gait. This is because typical environmental scan registration algorithms are less robust to large motion between consecutive frames and may additionally require fusion with an Inertial Measurement Unit (IMU) [35]. Additionally, it is important to attach the sensor high enough on the prosthesis such that its field of view over the terrain in the direction of locomotion is maximized. Keeping these factors in mind, we mount a depth sensor above the knee of the prosthesis, oriented approximately 45 degrees below the horizontal to get direct vision of the immediate terrain. Figure 3.1 shows a simple representation of the sensor attachment point.

Before implementing any algorithms on hardware, we tested our approach in

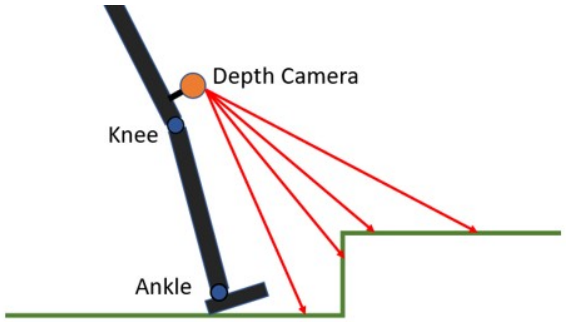


Figure 3.1: Schematic representation showing the depth camera mounted above the knee joint of the prosthesis, facing down from the horizontal axis

simulation. The main goal of the simulations was to observe the expected challenges caused by dynamic sensor motion at the selected attachment point. The sections below describe the basic theory for matching depth scans to map the environment and localize the sensor in simulation using a point-to-plane iterative closest point (ICP) algorithm, followed by the simulation environment and results. Based on these results and knowledge of the basic walking gait, we then propose a custom SLAM pipeline for the specific application of enhancing environment awareness of prostheses.

3.1 2D Depth Scan Matching

Figure 3.2 shows the overall pipeline implemented in simulation. The following subsections describe the theory behind each block in the pipeline. For this formulation, the terrain being scanned is assumed to be comprised of steps, ramps, and obstacles with a rectangular cross section. Once a point cloud is generated by depth sensor, like a LIDAR, a transform needs to be computed to align this new point cloud with the global point cloud representing the terrain. The algorithm used for computing this transform is iterative closest point (ICP). Since points may not have direct

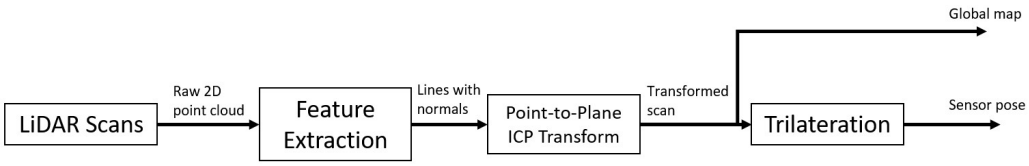


Figure 3.2: 2D depth scan registration and sensor localization pipeline used in simulation

correspondences across scans, a point-to-plane variant of ICP is used, which matches points with the plane on which they are located [36]. However, computing a transform using point-to-plane ICP first requires feature extraction from the raw point cloud. Once the point cloud is transformed and aligned with the global map, a trilateration step is used to localize the sensor.

3.1.1 Feature Extraction

First step upon receiving a 2D range scan is to extract line features from it. This is done by computing a roughness factor c [35] for each point. To compute this roughness factor for every point i in a scan $\boldsymbol{P}_t$, let S be a subset of $\boldsymbol{P}_t$ comprised of 3 consecutive points on either side of i. Then the roughness factor for each point i with $[x, y]$ co-ordinates in the laser frame expressed by $\boldsymbol{X}_i^L$ is computed by equation (3.1).

$$c_i = \frac{1}{|S|.||\boldsymbol{X}_i^L||} || \sum_{j \in S, j \neq i} (\boldsymbol{X}_i^L - \boldsymbol{X}_j^L) || \tag{3.1}$$

Now, points with a roughness value below a threshold (tuned at 0.5 here) are determined with a high confidence to lie on a line. Consecutive groups of points with such low roughness values are grouped into lines, and the normal for each line is computed. These points located on lines and their normals will be used for the point-to-plane iterative closest point (ICP) block described below.

3.1.2 Point-to-Plane ICP

This block is used to iteratively determine the relative transform to align the current scan with a reference scan from the previous time step. This algorithm is adapted from [36] for a simpler 2D implementation. At the beginning of each iteration, a set of data associations are determined. Let $\boldsymbol{P}_{t-1}$ be the reference scan from the previous time step, and $\boldsymbol{P}_t$ be the current scan. Then, for each point $\boldsymbol{p}_i$ in the current scan $\boldsymbol{P}_t$, the nearest point $\boldsymbol{r}_i$ is found from $\boldsymbol{P}_{t-1}$. Not all points in $\boldsymbol{P}_{t-1}$ are the closest point to at least one point in $\boldsymbol{P}_t$. Also, some points in $\boldsymbol{P}_t$ can detect the same point in $\boldsymbol{P}_{t-1}$ as their nearest point, in which case, the point with the smaller distance is chosen. This results in subsets of $\boldsymbol{P}_t$ and $\boldsymbol{P}_{t-1}$, $\hat{\boldsymbol{P}}_t$ and $\hat{\boldsymbol{P}}_{t-1}$ respectively, where each point in $\hat{\boldsymbol{P}}_t$ has a direct correspondence to its nearest neighbor in $\hat{\boldsymbol{P}}_{t-1}$.

Now that the data association has been established, the rigid body transform to make $\boldsymbol{P}_t$ match with $\boldsymbol{P}_{t-1}$ can be computed using point-to-plane ICP on $\hat{\boldsymbol{P}}_t$ and $\hat{\boldsymbol{P}}_{t-1}$ using reference normals to all points in $\hat{\boldsymbol{P}}_{t-1}$ determined in the feature extraction step. Let $\boldsymbol{n}_i$ be the normal to reference point $\boldsymbol{r}_i$. Then the optimal transform $\boldsymbol{T}_{opt}$ can be found by minimizing the sum of squared distance of each point $\boldsymbol{p}_i$ from its corresponding reference normal, as defined by equation (3.2)

$$\boldsymbol{T}_{opt} = \arg\min_{\boldsymbol{T}} \sum_i ((\boldsymbol{T}\boldsymbol{p}_i - \boldsymbol{r}_i).\boldsymbol{n}_i)^2 \tag{3.2}$$

Here, the rigid body transform is comprised of the 2D rotation and translation components $\boldsymbol{R}$ and $\boldsymbol{t}$, respectively which are shown in equation (3.3). The rotation matrix is also simplified using the small angle assumption, since the angle between consecutive scans is assumed to be small.

$$\boldsymbol{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} = \begin{bmatrix} 1 & -\theta \\ \theta & 1 \end{bmatrix}; \boldsymbol{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{3.3}$$

Next, the argument inside the argmin function is expressed in homogeneous form in equation 3.4

$$(\boldsymbol{T}\boldsymbol{p}_i - \boldsymbol{r}_i).\boldsymbol{n}_i = \left(\begin{bmatrix} 1 & -\theta & t_x \\ \theta & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{ix} \\ p_{iy} \\ 1 \end{bmatrix} - \begin{bmatrix} r_{ix} \\ r_{iy} \\ 1 \end{bmatrix} \right). \begin{bmatrix} n_{ix} \\ n_{iy} \\ 0 \end{bmatrix} \tag{3.4}$$

The above equations for all m points in $\hat{\boldsymbol{P}}_t$ can be expanded and stacked in a linear from as $\boldsymbol{Ax} - \boldsymbol{b}$ where, $\boldsymbol{A}$, $\boldsymbol{x}$, and $\boldsymbol{b}$ are shown in equations (3.5) - (3.7)

$$\boldsymbol{A} = \begin{bmatrix} n_{1x} & n_{1y} & p_{1x}n_{1y} - p_{1y}n_{1x} \\ ... & ... & ... \\ n_{mx} & n_{my} & p_{mx}n_{my} - p_{my}n_{mx} \end{bmatrix} \tag{3.5}$$

$$\boldsymbol{x} = \begin{bmatrix} t_x \\ t_y \\ \theta \end{bmatrix} \tag{3.6}$$

$$\boldsymbol{b} = \begin{bmatrix} r_{1x}n_{1x} + r1yn_{1y} - p_{mx}n_{mx} - p_{my}n_{my} \\ ... \\ r_{1x}n_{1x} + r1yn_{1y} - p_{mx}n_{mx} - p_{my}n_{my} \end{bmatrix} \tag{3.7}$$

Here, the $\boldsymbol{x}$ to minimize $\boldsymbol{Ax} - \boldsymbol{b}$ can be found by setting $\boldsymbol{Ax} = \boldsymbol{b}$ and solving using a linear least squares solver. Since this $\boldsymbol{A}$ matrix is not sparse, in this implementation

the linear system was solved simply using the pseudoinverse of $\boldsymbol{A}$ instead of factorizing $\boldsymbol{A}$ using Cholesky or QR factorization. Solving this leads to the transform variables $\boldsymbol{x} = [t_x; t_y; \theta]$ that should align the scan $\boldsymbol{P}_t$ with the reference scan $\boldsymbol{P}_{t-1}$.

3.1.3 Trilateration

In this block of the pipeline, scanned points transformed to the world frame are used to determine the location of the sensor $\boldsymbol{x_s} = [x_s; y_s]$ in the world frame. Once the optimal transform for the current time step has been found, it is added to the accumulated transform from all previous steps as shown in equation (3.8)

$$\boldsymbol{x}_{current} \leftarrow \boldsymbol{x}_{current} + \boldsymbol{x} \tag{3.8}$$

This $\boldsymbol{x}_{current}$ is then used to transform the current scan $\boldsymbol{P}_t$ not to the reference scan, but to the world co-ordinates, expressed as $\boldsymbol{P}_t^W$. The distance d_i from each point $[x_i, y_i]$ in $\boldsymbol{P}_t^W$ to the lidar sensor is known from the range measurements. For n scanned points, you get n equations using the distance formula as follows:

$$(x_s - x_i)^2 + (y_s - y_i)^2 = d_i^2 \tag{3.9}$$

Two such equations can be subtracted from each other to obtain the following system of linear equations

$$\begin{aligned} &(2x_i - 2x_{i-1})x_s + (2y_i - 2y_{i-1})y_s \\ &= d_{i-1}^2 - d_i^2 - x_{i-1}^2 + x_i^2 - y_{i-1}^2 + y_i^2 \end{aligned} \tag{3.10}$$

At least two distinct equations similar to 3.10 are required to solve for the 2D

position of the sensor. However, since the distance readings include noisy measurements, an over-constrained system of more than two equations can be used to find the linear least squares solution for $[x_s; y_s]$. After finding the position of the sensor using this trilateration and the angle of the sensor from $\boldsymbol{x}_{current}$, the full 2D pose of the sensor has been estimated.

3.2 Simulation

Simulations were implemented in MATLAB programming language. First a simulation environment in the form of a 2D occupancy grid was developed to generate data which was used to test the SLAM pipeline. This environment simulates a variety of terrain such as flat ground, stairs, obstacles with rectangular cross sections and ramps. Next, trajectories for the sensor in random regions through this occupancy grid are generated. These random trajectories have oscillations in θ and y to mimic the motion of a sensor that would be mounted on the thigh portion of a human leg in locomotion over a period of two full strides at different walking speeds. Lastly, 150 LIDAR range measurements within a span of 75 degrees are generated at a sampling rate of 45Hz using ray casting and injected with Gaussian white noise to mimic a real LIDAR. One such example trajectory is shown in figure 3.3, where the green line shows the simulated trajectory of the sensor shown as a blue triangle and its depth measurements shown as red rays.

Simulation results give some key insights about expected performance of a SLAM system when a depth sensor is mounted to the leg. This performance can be analyzed by looking at results from two example trajectories, with simulated average walking speeds of 0.83 m/s and 1.33 m/s. Figure 3.4a shows the frame at which the scan

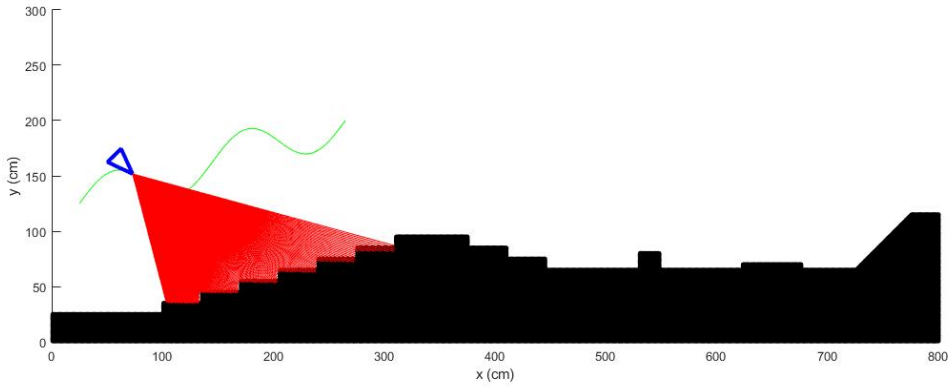
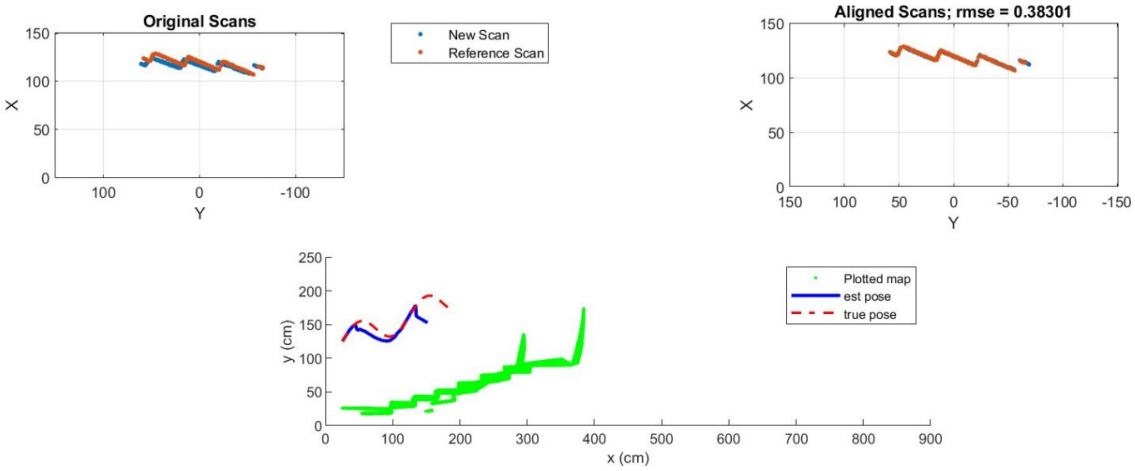


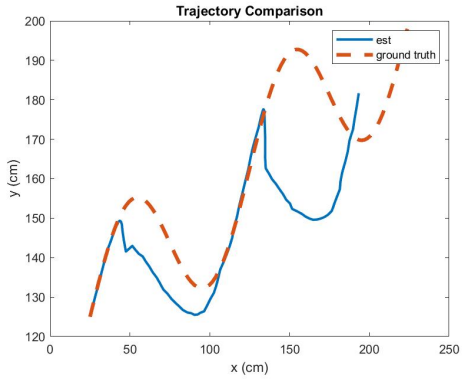
Figure 3.3: Simulation Environment: Occupancy grid showing variety of terrain, green line showing and example trajectory of sensor, red rays showing noise-injected ray casting to generate depth scans.

matching algorithm fails for the lower speed of 0.83 m/s. This happens right after the sensor collects a scan at one of the highest points during swing. In this scan, a set of points which are not a part of the terrain are recorded. These points represent the maximum range of the sensor, and are recorded because the terrain is not within the sensor's point of view. In this figure, the top left block shows the raw 2D depth scans in the sensor reference frame. The top right block of the figure shows the after the new scan has been aligned with the reference scan and the bottom center block shows the generated map (green) and the estimated and ground truth trajectories (blue and red, respectively). Figure 3.4b shows the entire estimated trajectory compared to the ground truth and figure 3.4c shows the absolute trajectory error. The trajectory estimation seems to be inaccurate when the sensor approaches the highest point in swing due to a part of the terrain being outside its field of view, as explained above.

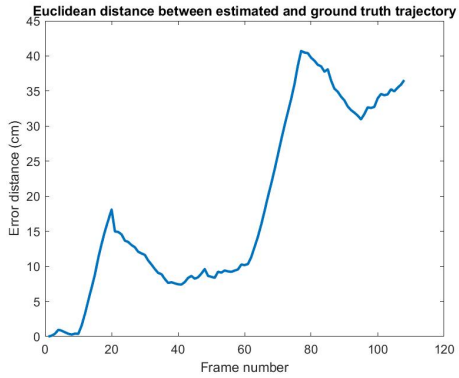
Figure 3.5 contains the equivalent results from a simulated average walking speed of 1.33 m/s. At this higher speed, a different kind of error is observed. Figure 3.5a



(a) Top left block shows two consecutive scans in the sensor frame, top right block shows the current scan transformed to align with the previous scan, and bottom center block shows the final map (green) generated with a comparison of the estimated (blue, from SLAM) and ground truth (red, from simulation) trajectory of the sensor.

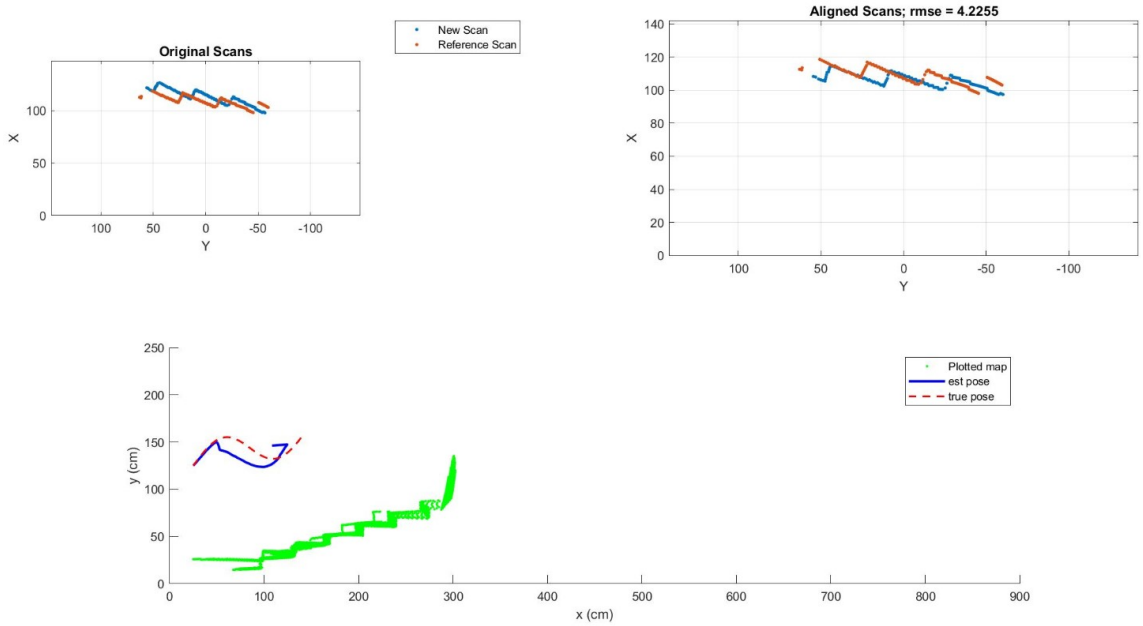


(b) Absolute trajectory, estimation and ground truth

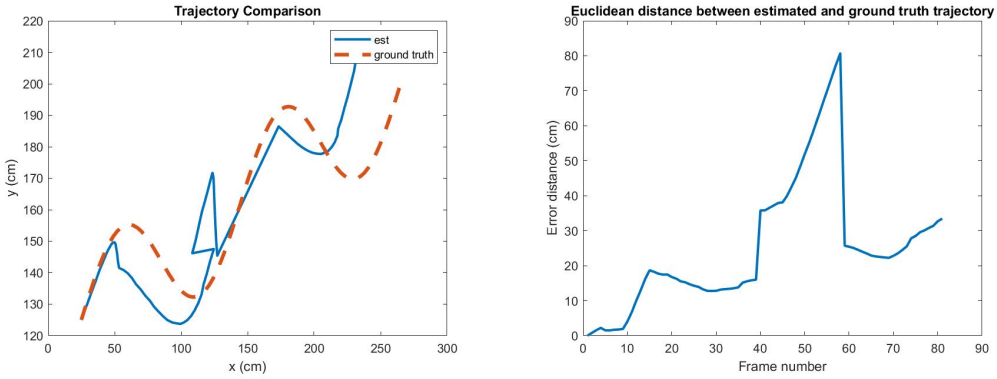


(c) Absolute Euclidean distance between estimated and ground truth trajectories from simulation

Figure 3.4: Simulated 2D SLAM pipeline performance from average walking speed of 0.83 m/s. Figure (a): Intermediate frame from showing mapping and trajectory estimation right after a failed scan matching. (b) and (c): Figures showing the complete trajectory estimation performance, comparison between estimated and ground truth on the left, absolute distance between estimated and ground truth on the right



(a) Top left block shows two consecutive scans in the sensor frame, top right block shows the current scan transformed to align with the previous scan, and bottom center block shows the final map (green) generated with a comparison of the estimated (blue, from SLAM) and ground truth (red, from simulation) trajectory of the sensor.



(b) Absolute trajectory, estimation and ground truth

(c) Absolute Euclidean distance between estimated and ground truth trajectories from simulation

Figure 3.5: Simulated 2D SLAM pipeline performance from average walking speed of 1.33 m/s. Figure (a): Intermediate frame from showing mapping and trajectory estimation right after a failed scan matching. (b) and (c): Figures showing the complete trajectory estimation performance, comparison between estimated and ground truth on the left, absolute distance between estimated and ground truth on the right

shows this failure instance. The ICP algorithm fails to correctly match two consecutive scans. This is likely because at a higher speed, the difference between consecutive scans is too large for the scans to be aligned by the ICP algorithm. ICP is a local registration algorithm, whereas matching scans that are further apart from each other would require a global registration algorithm, like in [37], which is algorithmically more complex, and computationally slower. The trajectory comparison and absolute error shown in figures 3.5b and 3.5c also show the failures related the sensor field of view, similar to the previous case with a slower walking speed.

In both example cases shown above, the failure occurs at some point during the simulated swing phase of leg (when the foot is not in contact with the ground). The trajectory comparison and error plots in both cases show that the scan matching algorithm seems to correct for these errors after a few frames. However, after each cycle, the accumulated error grows as the estimated trajectory drifts away from the ground truth. These simulation results validate and highlight the challenges of implementing SLAM on legged systems, as described in [31, 32, 33], especially with the sensor mounted to the limb and not the trunk/torso. There are two main expected sources of error: the terrain going beyond the sensor field of view leading to points recorded from sensor's maximum range and large differences between consecutive scans causing a local registration algorithm like ICP to fail. We now take a brief look at the typical human walking gait and propose a SLAM pipeline which integrates information from gait phase to circumnavigate some of these challenges. When implemented in hardware, we expect to be able to obtain maps of the environment that can be effectively used for planning safe swing trajectories of the prosthesis in spite of these challenges.

3.3 SLAM Pipeline For Prosthesis

To design a SLAM pipeline to be used to control a lower-limb prosthesis through a gait cycle, it is important to first observe a typical walking gait cycle, as shown in figure 3.6 [8]. An entire gait cycle, which is the combined period between two consecutive foot strikes for the same leg, can be broadly divided into two phases, stance and swing, with stance being the duration of the stride when a part or all of the foot is in contact with the ground and swing being the duration when the foot is moving forward with respect to the ground just before the next heel strike. Once the foot makes contact with the ground at foot strike, there is negligible relative motion between the foot and the ground through the stance phase. This stance phase lasts for roughly 0.6s for average walking speed. Swing phase is where most tripping events and falls occur, especially if there is an obstacle in the normal swing trajectory. Hence, adapting to obstacles or variations in the terrain requires the selection of a safe foothold for the placement of the foot to begin the next stance phase, and also a planned trajectory that avoids any contact with the environment before safe foot strike. This can be accomplished by obtaining a dense map of the terrain directly in front of the prosthesis during stance phase. Additionally, map of the terrain behind the foot at the beginning of stance is of no use for planning a safe swing trajectory. This could be used for resetting the SLAM system, providing a few different benefits. Firstly, it would eliminate issues of long term drifts commonly observed in visual SLAM systems. Secondly, implementation of a graph optimization-based loop closure can be eliminated as well. These loop closure methods can require expensive computation and storage which is not ideal for packaging a computer of a small form factor on a mobile, ergonomic device like a prosthesis.

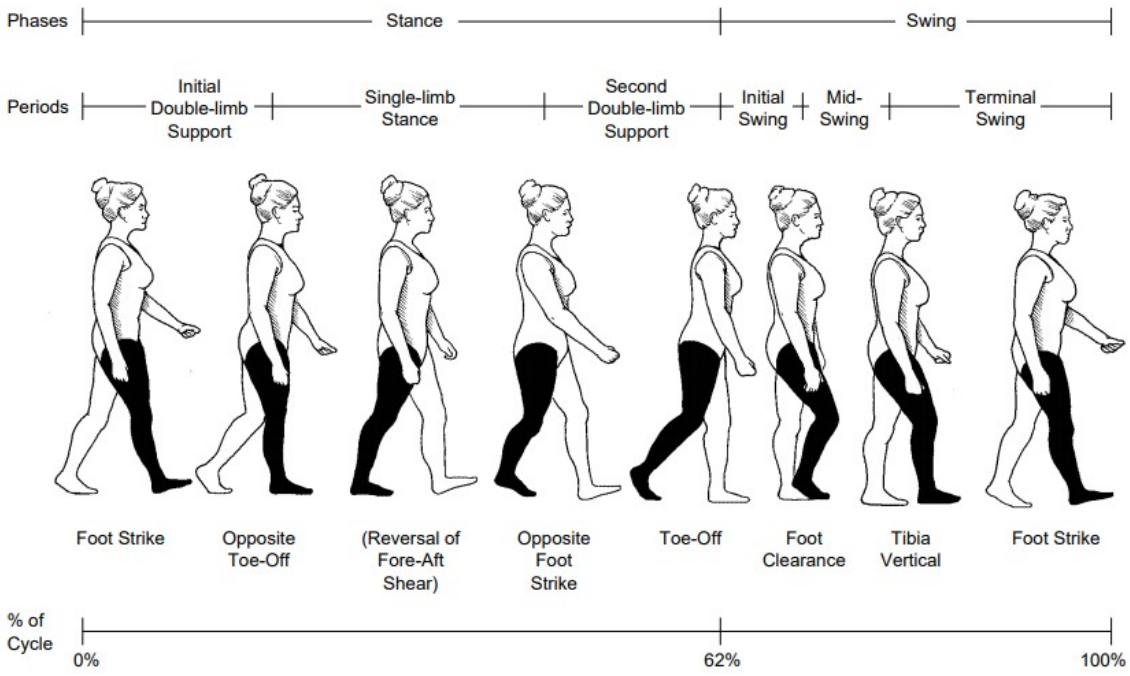


Figure 3.6: A typical normal walking gait cycle [8]

Keeping this in mind, the SLAM system shown as a flowchart in figure 3.7 was designed for the application of prosthesis control and planning. This pipeline extends the 2D simulations to 3D by using a depth sensor that generates 3D point clouds. The goal of the system is to obtain a dense global map of the terrain and a pose estimation (or odometry) of the sensor, from which the pose of the prosthesis can be estimated. At the beginning of every stride phase at foot strike, which is detected by a load cell integrated with the prosthesis, the sensor odometry can be reset using encoder measurements giving joint angles, and IMU measurements giving orientation. Since there is assumed to be no relative motion between the foot and the ground during stance phase, the terrain map from the previous stride provides no useful information for prosthesis control. Hence, the global map can be erased and reset as the most recent point cloud obtained at the time of foot strike, registered based on the

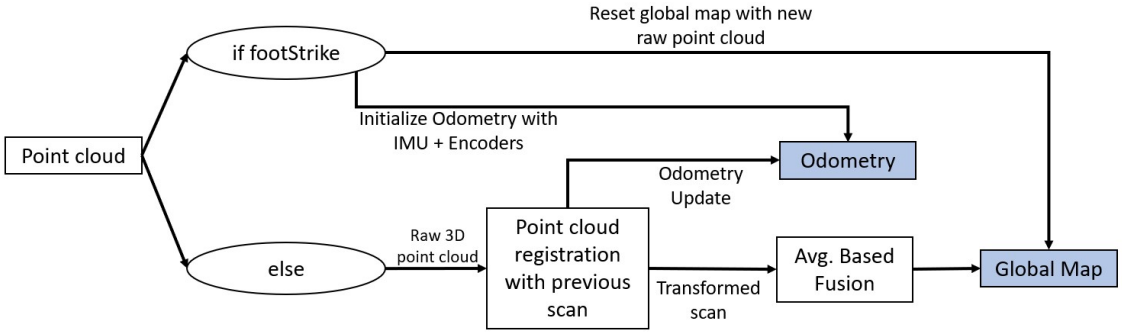


Figure 3.7: A flow chart showing the high level pipeline of SLAM system designed to aid with prosthesis control

odometry initialization. For each subsequent point cloud, a transform is computed using a point-to-plane ICP algorithm (similar to [36]) and the most recent odometry as an initial guess. This transform is then used to update the sensor odometry. It is also used to register the new point cloud in the global map and fused with the global map based on averaging with point correspondence used in the ICP algorithm. This process of updating sensor odometry and registration of new point clouds into the global point cloud is repeated till the next foot strike, when both, the odometry and the global map are reset again. This provides a dense 3D map of the immediate terrain which can be used for adaptive swing control of the prosthesis.

3.4 Experimental Setup

We implemented and tested the above SLAM pipeline which integrates gait phase information in hardware. The sections below describe the software and hardware components used in these experiments.

3.4.1 Software

Experiments were run using two different depth sensing cameras, a CamBoard Pico Flexx[1] and an Intel RealSense D455[2]. Both sensors generate 3D point clouds of the surfaces in their field of view. The Pico Flexx only provides a depth image which can be converted into a point cloud, whereas the RealSense is a full RGB-D camera which can provide colored point clouds. For tests using the Pico Flexx, 3D point cloud registration is performed using a point-to-plane ICP algorithm as described in [36]. However, the added color information from the RealSense can be used to lock the point cloud alignment in the tangent plane and hence, improve the registration performance. The color point cloud registration algorithm implemented here is from [38] and summarized below.

The algorithm runs ICP iterations which minimize the energy given by equation 3.11

$$E(\boldsymbol{T}) = (1-\delta)E_C(\boldsymbol{T}) + \delta E_G(\boldsymbol{T}) \tag{3.11}$$

where $\boldsymbol{T}$ is the transform to be estimated. E_G and E_C are the geometric and photometric or color energy terms, weighted with a weight parameter $\delta \, \epsilon \, [0, 1]$. E_G is the geometric energy term for a generic point-to-plane ICP algorithm, which is equal to the term under summation in equation 3.2, shown again below in equation 3.12.

$$E_G(\boldsymbol{T}) = \sum_i ((\boldsymbol{T}\boldsymbol{q}_i - \boldsymbol{p}_i).\boldsymbol{n}_{i,p})^2 \tag{3.12}$$

where $\boldsymbol{q}$ and $\boldsymbol{p}$ are sets of corresponding points from the source (current) and

[1] www.pmdtec.com
[2] www.intelrealsense.com/

target (previous) or global point clouds respectively, and $\boldsymbol{n}$ are the normals to the points $\boldsymbol{p}$. E_C is the photometric or color energy term which is a measure of the difference in color between points $\boldsymbol{q}$ and the color of their projection on the tangent plane of $\boldsymbol{p}$ shown in equation 3.13

$$E_C(\boldsymbol{T}) = \sum_i \left(C_{\boldsymbol{p},i}(\boldsymbol{f}(\boldsymbol{T}\boldsymbol{q_i})) - C(\boldsymbol{q_i})\right)^2 \tag{3.13}$$

where $C_{\boldsymbol{p},i}$ is a pre-computed color function continuously defined on the tangent plane of $\boldsymbol{p}$ and the function $\boldsymbol{f}()$ projects a 3D point to the tangent plane. More details can be found in [38].

Additionally, we also implemented a multi-scale registration scheme which was used in both [38] and [7] to improve the overall accuracy and performance efficiency of the registration algorithm. In multi-scale registration, the source and target point clouds are downsampled with decreasing voxel radii. Applying the ICP registration after downsampling with the largest voxel radius gives a transform to coarsely align the source point cloud with the target point cloud. This coarse transform is then used as an initial guess for ICP registration with a slightly smaller voxel radius. We run this overall process for three iterations, with the final iteration providing fine adjustment to register the source point cloud.

The software for reading and registering point clouds is implemented as nodes in a ROS environment [24]. ROS allows for multiple nodes to run independently while also communicating and sharing data with other nodes running in the same core environment. For this application, it is desired for the prosthesis controller to run stance and swing phase controllers while the SLAM system runs independently as a separate node. There is, however, some communication required between the

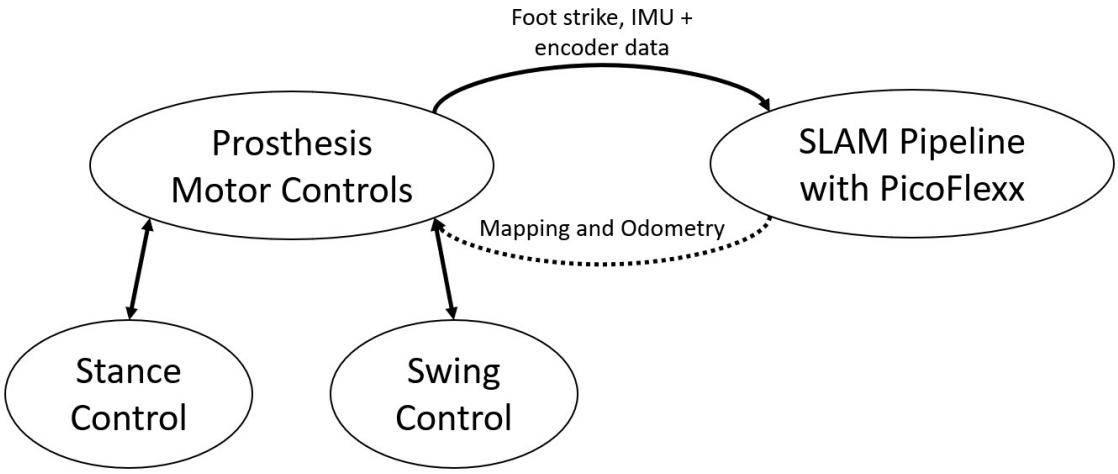


Figure 3.8: Higher level ROS nodes with communication between the prosthesis controller and SLAM system

prosthesis controller and the SLAM system. Every time the prosthesis controller detects a foot strike marking the beginning of a new stance phase, this needs to be communicated to reset the SLAM system. The SLAM node is also subscribed to the IMU and encoder data from the prosthesis which are used to reset the odometry at foot strike. In the future it will also be necessary for the SLAM node to share the mapping and localization information back to the prosthesis controller. Running the software in a ROS environment makes implementation of these communication lines seamless. The higher level architecture of this ROS environment is shown in figure 3.8. To process point clouds, including storage, downsampling, registration, we also used Open3D [39], which is an open source library that supports rapid development of software that deals with 3D data. Open3D provides a python API for a variety of popular algorithms such as the point-to-plane ICP for colored and simple point clouds, described above.

Object	Depth	Width	Height
Box1	203.20	266.70	190.50
Box2	266.70	355.60	76.20
Box3	336.55	438.15	63.50

Table 3.1: Dimensions of obstacles in mm shown in figure 3.9

3.4.2 Testing Environment and Hardware

The goal of these experiments is to validate the following hypotheses about implementing a SLAM system on a prosthetic device:

1. The point clouds recorded at available sampling rate while the camera undergoes dynamic motion due to its attachment point can be registered accurately, forming a dense map of the terrain.
2. The transforms computed from this registration can be used to accurately localize the camera and track its trajectory.
3. The generated map of the environment accurately represents the locations and sizes of the obstacles, making it useful for planning the next foot placement and swing trajectory.

For all experiments testing and validating the initial approach shared here, a terrain with different sizes of boxes to simulate cuboid obstacles was setup for the subject to navigate, shown in figure 3.9. All the boxes were marked with infrared reflective markers which can be recorded with a Vicon motion capture system, to be used as ground truth. Table 3.1 includes the dimensions of the three obstacles labeled in figure 3.9 as box 1, 2, and 3. These dimensions will be used as ground truth to evaluate the mapping performance of the SLAM system.

The cameras were mounted to the thigh bone of an able-bodied adult, pointing 30

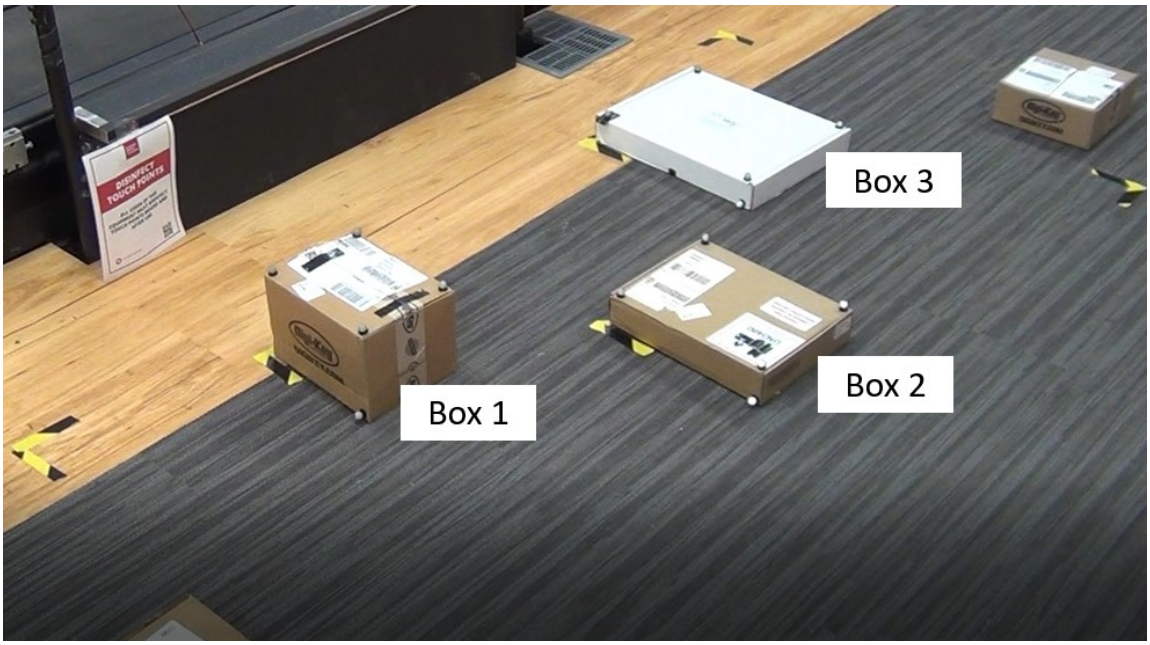
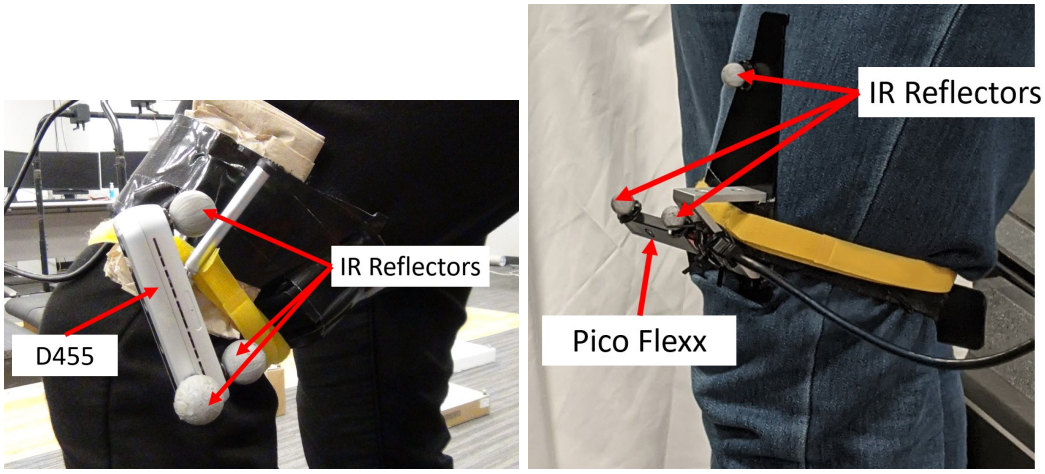


Figure 3.9: Testing terrain with various sizes of boxes used as obstacles.

to 45 degrees below the horizontal, when the subject is standing straight. The cameras also have reflective markers attached to them to record the ground truth trajectory of the camera using the Vicon motion capture system. The two camera mounts with reflective markers are shown in figure 3.10. The resetting of the SLAM system with each foot strike, which would be detected by the load cell on a prosthesis, is simulated by the press of a button on a computer carried by the subject. For these experiments, the point clouds were simply recorded on a consumer grade laptop running on an Intel i7-8550U CPU, at a sampling rate of 90Hz (even though this can vary, as limited by available computational resources). Since real-time registration would require further optimization of the code, with parallelization and implementation in a compiled language like C++ on a GPU, the registration is performed offline for the purposes of these experiments.



(a) Intel RealSense (b) Pico Flexx

Figure 3.10: Intel RealSense D455 RGB-D Camera and Pico Flexx depth camera mounted to the thigh bone. Both camera mounts fitted with infrared reflective markers.

Chapter 4

Results and Discussion

Non-colored point cloud data collected using the Pico Flexx camera generated poor results. There are a few possible reasons contributing to these results. The Pico Flexx camera has a smaller field of view and lower resolution than the RealSense, hence when there is a large change between two consecutive point clouds, the registration algorithm can fail. Additionally, the Pico Flexx also has lower depth resolution at larger distances, leading to noisy point clouds. Lastly, due to the lack of a color image, in point clouds with minimal geometric features, accurate registration can be harder to achieve. Experimental evaluation of this camera can be re-attempted with different camera orientation and sampling rates.

One major change in the experimental set up using the Intel RealSense D455 was mounting the camera such that the camera's larger field of view (horizontal dimension) was oriented vertically, parallel to the sagittal plane. Since the camera undergoes significant rotation in this plane, orienting the larger field of view increases the probability of capturing common features in consecutive frames, increasing the

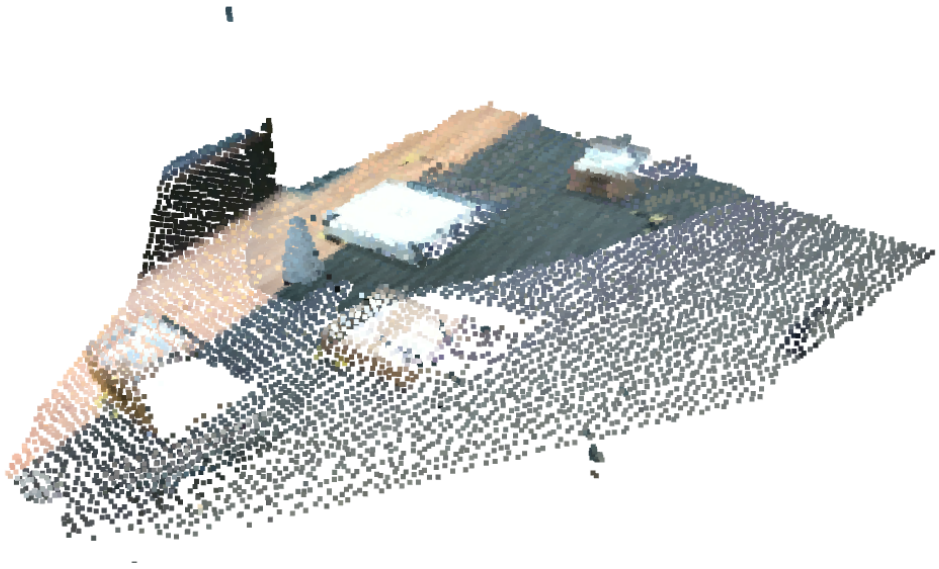
probability of successful registration.

Registration performance for point clouds recorded during 25 different strides, distributed over 3 different walking speeds was evaluated. The fitness score of registration for any point cloud is the ratio of the number of points in the new source point cloud that have a correspondence with a point in the target point cloud to the total number of points in the target point cloud. Hence, the fitness score lies in the range $[0, 1]$, with a higher score indicating better registration. Instances where registration fails leading to multiple copies of the same stationary object in different locations in the global point cloud, and a lack of continuity in the resulting camera trajectory typically have low fitness scores, < 0.15. Therefore, the registration and localization was halted if the fitness score dropped below a threshold of 0.3.

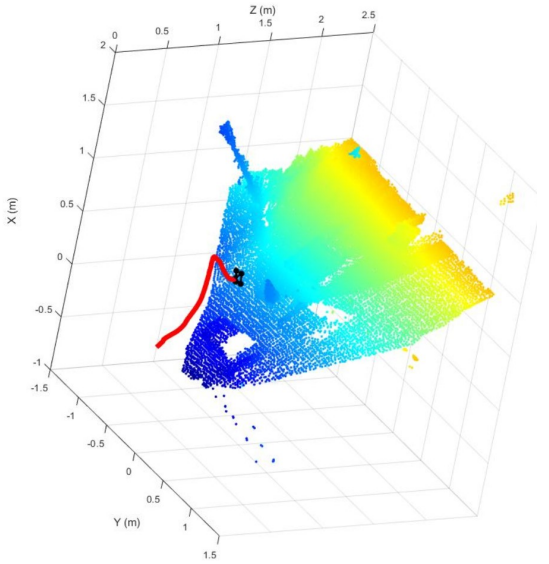
4.1 Strides With Successful Registration

From the 25 recorded strides, only 2 strides had successful registration up to the last frame. Figures 4.1 and 4.2 show the results from running the SLAM pipeline over these two full strides, in which figures 4.1a and 4.2a show the final global colored point cloud formed by registering individual frames during that stride. Figures 4.1b and 4.2b on the other hand show the same point cloud without the color information, but with the camera pose at the last frame and the tracked camera trajectory. Lastly, figures 4.1c and 4.2c show the ground truth of the trajectory and obstacle locations for comparison, obtained using the Vicon motion capture system.

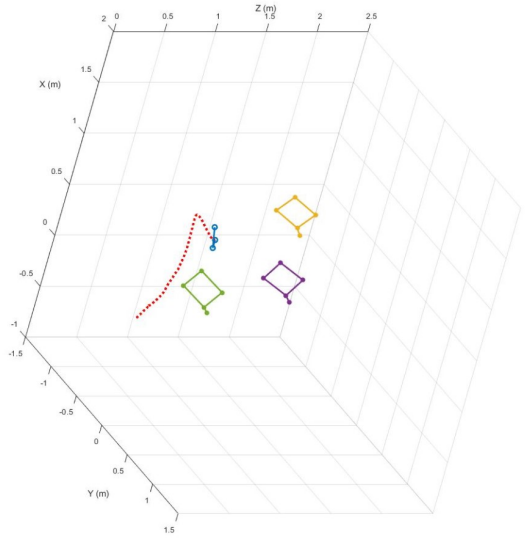
Quantitative comparison between estimated and ground truth trajectories is a common challenge, and common benchmarking methods are described in [40] and [28]. The main challenges stem from the difference in coordinate frames and sampling



(a) Colored global point cloud

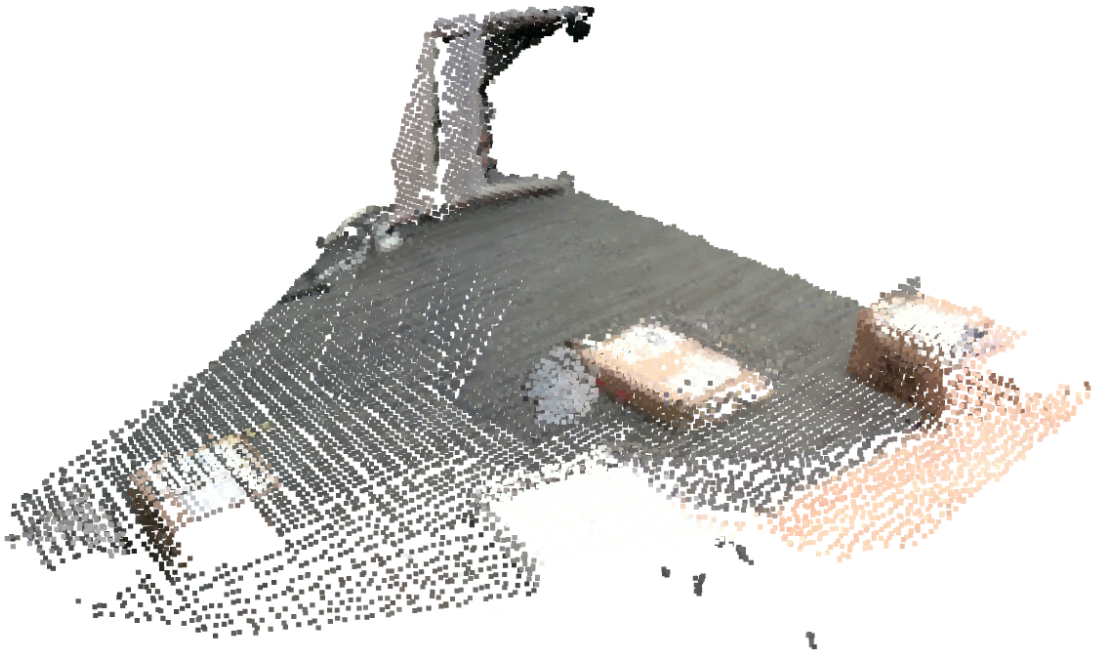


(b) Global point cloud with camera trajectory

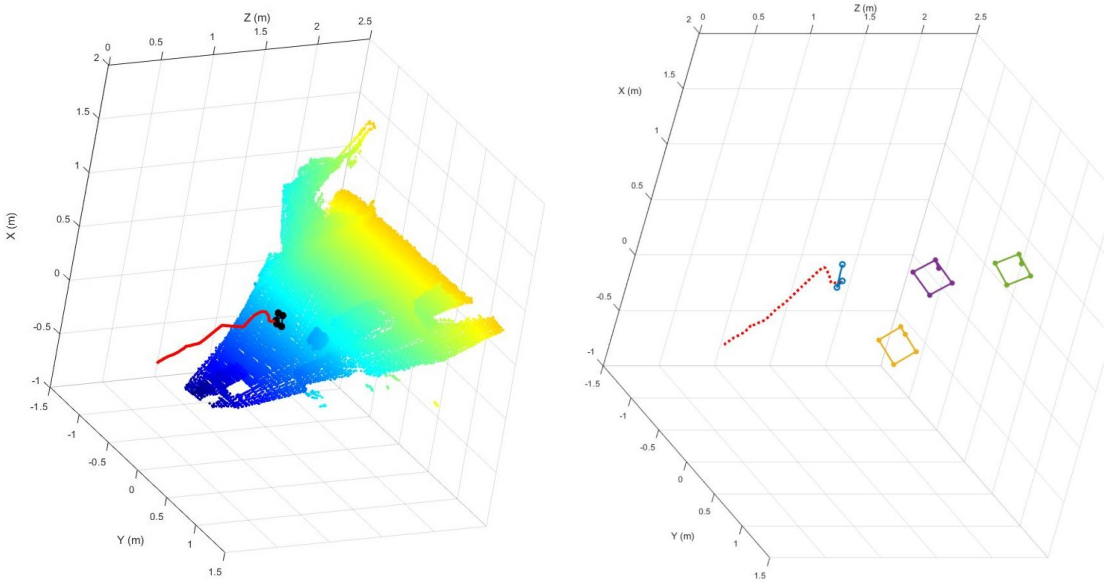


(c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

Figure 4.1: Global registered point cloud and camera trajectory resulting from registration of point clouds recorded during one stride and ground truth from Vicon motion capture



(a) Colored global point cloud



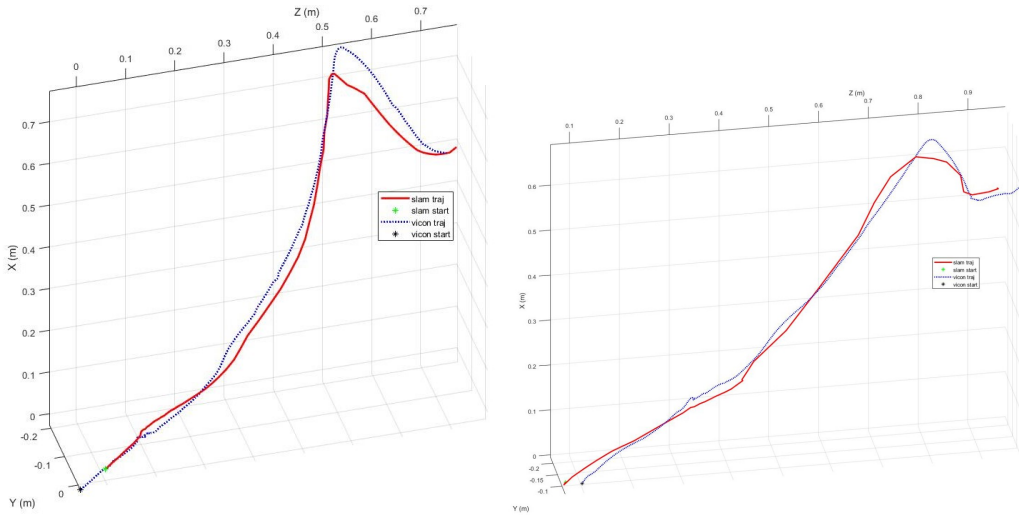
(b) Global point cloud with camera trajectory (c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

Figure 4.2: Another example of global registered point cloud and camera trajectory resulting from registration of point clouds recorded during one stride and ground truth from Vicon motion capture

rates and the lack of timestamp synchronization between the ground truth and SLAM outputs. For our experiments, the Vicon measurements and point cloud measurements ran with different co-ordinate frames and asynchronously on independent computers. This makes it challenging to achieve comprehensive qualitative analysis of trajectory tracking. However, it is possible to estimate a transform to best match these trajectories and compare them. To do this, first we manually segment the data stream from Vicon into individual strides. We then compute a coarse rigid transform to transform the Vicon co-ordinate frame into an estimated camera frame. The point-clouds and trajectories obtained from the SLAM pipeline are in the camera frame. Lastly, we align the trajectories using ICP, and observe the differences and the root mean squared error between the trajectories. Following this process provides a consistent method for evaluating the sensor pose estimation performance of our SLAM pipeline. Figures 4.3a and 4.3b show the manually aligned estimated and ground truth trajectories for the two strides shown above. Our pipeline is able to localize the sensor and estimate its trajectories producing root mean squared errors of 0.02114m and 0.03407m respectively. This is comparable to localization obtained in [4] with root mean squared error of 0.019m.

4.2 Mapping Performance for All Strides

To evaluate the mapping performance of the pipeline, we use the obstacles as landmarks. In the absence of accurate transforms to compare the exact locations of these obstacle, we instead compare manually measured dimensions instead. Table 4.1 shows the mean and standard deviation of the dimensions of the obstacles fro figure 3.9 measured from the final global point cloud generated by the mapping algorithm for



(a) Trajectories from stride in figure 4.1 (b) Trajectories from stride in figure 4.2

Figure 4.3: Comparison between estimated and ground truth trajectories for two strides from figures 4.1 and 4.2, after manual alignment and synchronization of the camera and Vicon co-ordinate frames.

all 25 strides. It also lists the root mean squared errors of these dimensions compared to the ground truth values of these dimensions shown earlier in table 3.1. The colored point cloud registration algorithm is able to map the heights of these obstacles within 10mm of the ground truth. The Intel RealSense has a depth uncertainty specification of 80mm for distances up to 4m. Since the sensor is mounted to right above the knee, the obstacles are never more than 1m away from the camera, which explains the lower uncertainty. The registration algorithm estimates the height with more accuracy than the other dimensions, since the ground surface and top surface of the obstacles are significantly larger in the camera's field of view, leading to more accurate point-to-plane correspondences.

Object	Width			Depth			Height		
	Mean	Stdev	RMSE	Mean	Stdev	RMSE	Mean	Stdev	RMSE
Box1	267	9	6.5	216	20	16.5	180	8	10.1
Box2	357	10	7.1	283	22	19.8	75	3	2.7
Box3	443	10	9.9	351	11	16.4	66	2	2.5

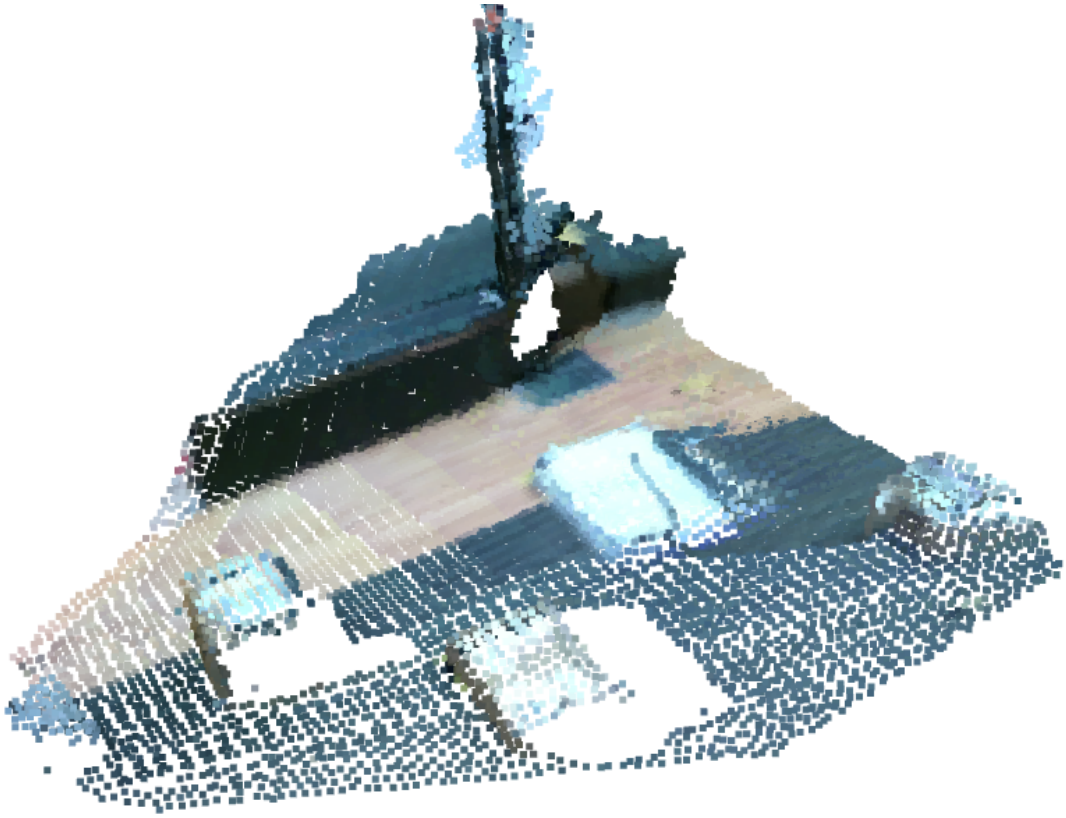
Table 4.1: Mean and standard deviation of measured dimensions of obstacles shown in figure 3.9 and their root mean squared errors relative to ground truth measurements listed in table 3.1, in mm, obtained from the global map from generated by the registration algorithm across all 25 strides.

4.3 Strides With Partially Successful Registration

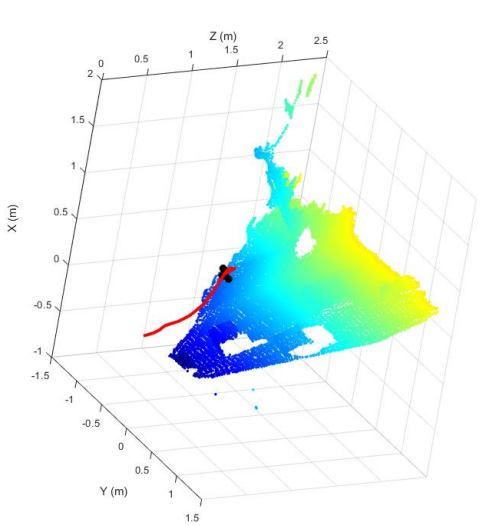
For the remaining strides, the registration fitness score dropped below the threshold, leading to a failed registration after a few frames recorded with the leg in swing mode. This is an expected challenge of implementing a SLAM system using a sensor that is undergoing such dynamic motion. The reason for these failed registrations through the swing phase can be attributed to two factors. Firstly, the recorded images and point clouds have detrimental motion artifacts such as blur and distortion, as was also observed in the implementation of [31], during which the sensor motion is less dynamic as it is mounted to the trunk of a six-legged robot as opposed to being mounted on the leg of a bipedal system shown above. Secondly, even though the camera was set to record point clouds at a sampling rate of 90Hz, as noted in chapter 3.4, limited computational resources while writing these point clouds to disk led to multiple dropped frames. The effective sampling rate was observed to be only in the range of 35-40 Hz. At such low sampling rates and high sensor velocities, consecutive point clouds can have larger differences between them, causing registration algorithms

like ICP to fail due to poor point correspondences. The 2 strides that resulted in successful registration for the entire stride were all the first strides in individual runs when the queue of point clouds to be written is smaller and there are fewer dropped frames. More than 50 point clouds were recorded for each of these strides, compared to under 35 for the remaining strides. This could be addressed by optimization and parallelization of code using more computational resources and implementation in a compiled language like C++ which can be up to a 100 times faster than an interpreted language like Python. Another possible solution would be to replace the registration algorithm used. ICP and colored point cloud registration are both local registration methods which rely on some initial alignment of the point clouds. This could be replaced by a global registration method, similar to the one implemented by [37], which would also require increased computational resources.

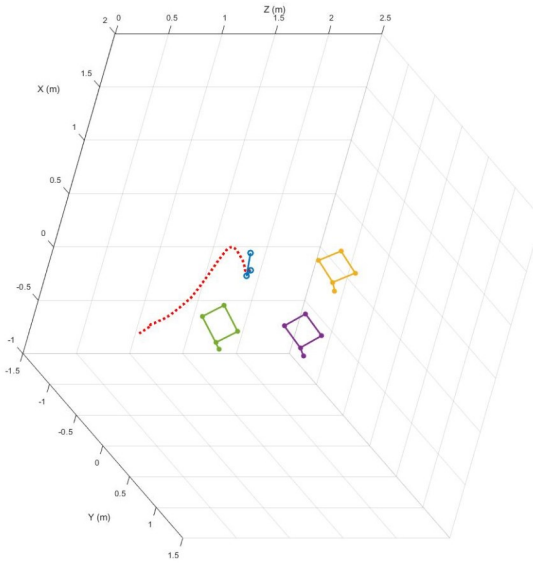
In spite of these instances of failed registration and localization mid-way through the stride, the results show promise. Figures 4.4 and 4.5 also show example results for a few such strides, similar to the results shown in figure 4.1. The trajectory comparisons for these two strides are shown in figures A.5c and A.5d respectively. These two strides recorded trajectory tracking root mean squared errors of 0.026m and 0.014m respectively. Table 4.2 below summarizes results for all strides, showing the number of point clouds captured, number of point clouds successfully registered, and the root mean squared error in trajectory tracking against manually transformed ground truth. For every stride, the system was able to generate a map of the environment and localize the camera through the entire stance phase and the initial portion of the swing phase. This is the period during which a prosthesis controller, in real-time, plans the location of foot placement for the next stance phase and plans the swing trajectory to drive the foot to this location without tripping against the terrain



(a) Colored global point cloud

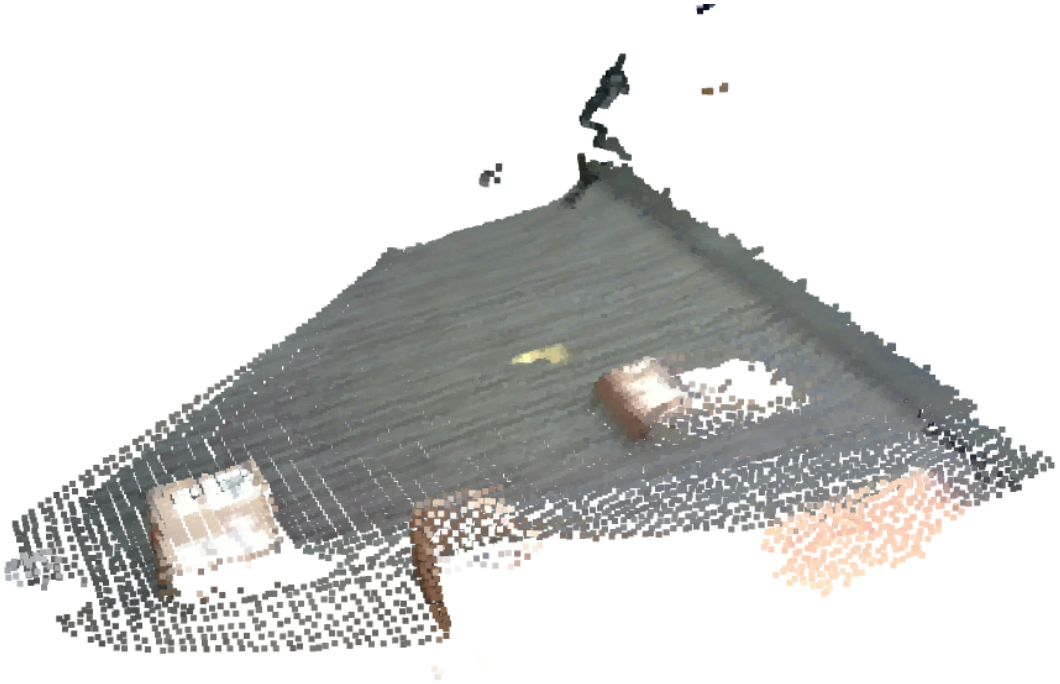


(b) Global point cloud with camera trajectory

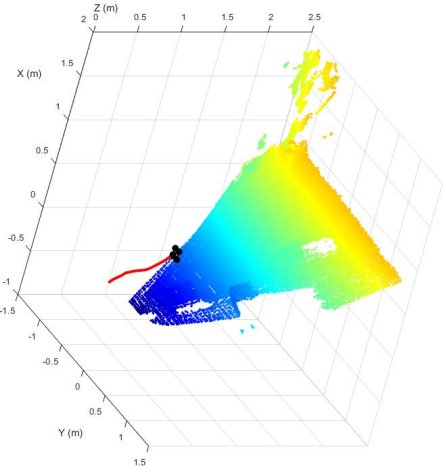


(c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

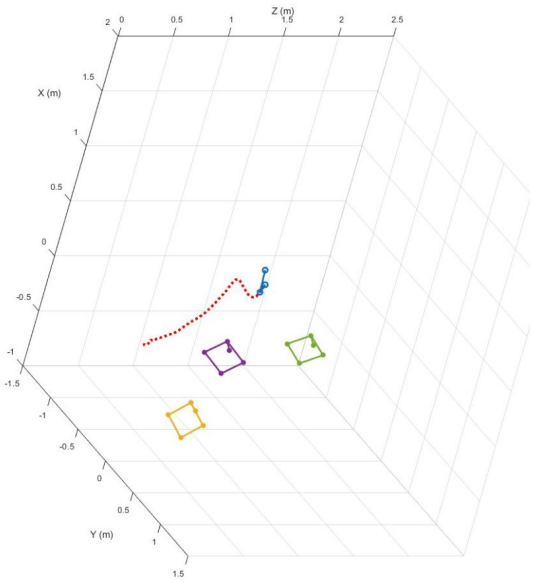
Figure 4.4: Example of global registered point cloud and camera trajectory resulting from incomplete registration of point clouds recorded during one stride and ground truth from Vicon motion capture



(a) Colored global point cloud

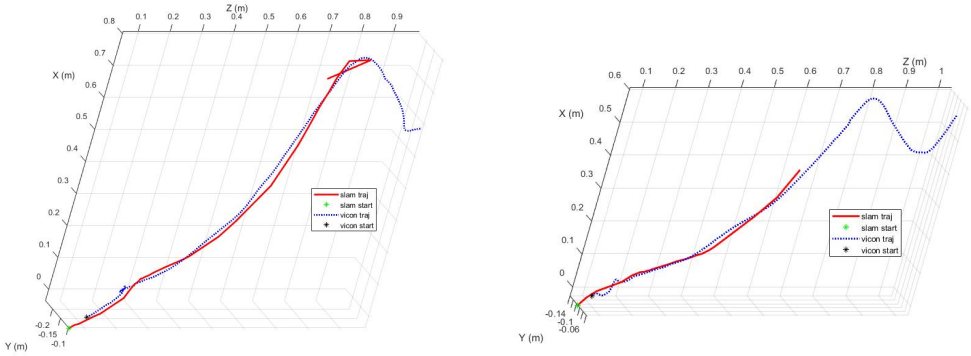


(b) Global point cloud with camera trajectory



(c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

Figure 4.5: Another example of global registered point cloud and camera trajectory resulting from incomplete registration of point clouds recorded during one stride and ground truth from Vicon motion capture



(a) Trajectories from stride in figure 4.4 (b) Trajectories from stride in figure 4.5

Figure 4.6: Comparison between estimated and ground truth trajectories for two strides from figures 4.4 and 4.5, after manual alignment and synchronization of the camera and Vicon co-ordinate frames.

[4]. Hence, these results shows promise for successfully increasing the environment adaptability of a lower limb prosthesis even with a functioning SLAM system only during the stance phase.

Table 4.2: Summarized results for all recorded strides

Walking speed level	Stride #	# Pcd recorded	# Pcd registered	Traj RMSE (m)
Slow	1	70	70	0.021
	2	62	37	0.014
	3	43	32	0.030
	4	35	20	0.011
	5	42	29	0.019
	6	31	19	0.010
	7	35	27	0.030
	8	32	20	0.019
	9	30	18	0.016

Table 4.2: Summarized results for all recorded strides

Walking speed level	Stride #	# Pcd recorded	# Pcd registered	Traj RMSE (m)
Normal	1	35	35	0.034
	2	33	21	0.024
	3	26	13	0.015
	4	34	20	0.015
	5	33	22	0.010
	6	40	28	0.025
	7	33	24	0.012
	8	33	23	0.009
	9	32	23	0.014
Fast	1	36	33	0.026
	2	21	14	0.015
	3	25	5	0.009
	4	19	12	0.011
	5	24	13	0.010
	6	33	27	0.021
	7	26	14	0.017

Chapter 5

Conclusions and Future Work

In this thesis, we show that the field of prosthetics control stands to benefit from the large strides taken in the research and development of SLAM algorithms, especially methods to reconstruct dense maps of the environment using RGB-D cameras. Reviewing the literature about the state of the art in integration of environment-sensing for prosthesis control, there is a clear opportunity for obtaining richer environment maps and accurate localization to guide swing trajectory planning. A brief review of literature for dense RGB-D SLAM indicates many advanced techniques to obtain such rich maps and accurate localization. However, dynamic motion of a sensor on a legged system adds additional challenges for SLAM algorithms, which is evident upon reviewing the prior efforts of implementing SLAM on hexa-, quadra-, and bi-pedal systems. This literature underscores the importance of integrating gait information with the SLAM algorithms.

Simulations of a 2D SLAM system using registrations of LIDAR scans taken from a sensor mounted on the thigh segment highlighted these challenges for implementing

such a system on a prosthesis. Based on observation of these simulation results, current knowledge about the human walking gait, and common prosthesis control strategies we designed a 3D SLAM system which uses a single RGB-D camera. The pipeline is designed to circumnavigate the challenges stemming from dynamic sensor motion by resetting the global map and sensor odometry with every stride. This also helps minimize the requirement for expensive computation and memory resources by only maintaining a short term map of the terrain directly in front of the prosthesis, which is required to control safe swing trajectories. To the best of our knowledge, this is the first ever attempt at deploying a SLAM pipeline to enhance the environment sensing capabilities of a lower-limb prosthesis. This pipeline was then implemented on an experimental setup using an able-bodied subject walking on a terrain with variable sized obstacles.

Results from these experiments show great promise in the implementation of such a SLAM system on a prosthetic device. Even though the colored point cloud registration often fails mid-way through the swing phase for most strides, the map generated during stance and early swing phase can be used for planning an adaptable swing trajectory, enabling the prosthesis-user to navigate obstacles and variations in terrain. This demonstrates the usefulness of modern sensors and SLAM algorithms to increase the adaptability and robustness of prostheses, and hopefully opens future pathways of research in the field of prosthetics control.

To deploy a fully functioning SLAM system on a lower-limb prosthesis there is still a significant amount of work that needs to be done in the future. This future work ranges from improvements in implementation of both, hardware and software to improvements in fundamental SLAM algorithms. A few key points of future work and challenges are listed here:

1. To achieve real-time online performance of the SLAM system, the computational resources need to be scoped and the registration algorithms need to be optimized and parallelized. Failures due to dropped frames would also be alleviated by these computational improvements.
2. More robust registration performance throughout the stride can be partially addressed by minimizing dropped frames. However, there is also room for improvement in implementing existing methods of reducing artifacts from fast sensor motion such as motion blur [34].
3. This initial pipeline can also be optimized to minimize storage capacity yet have access to information about previously observed terrain. For example, at each foot strike, instead of resetting the entire global map, only the points behind the current prosthesis location can be erased.
4. Currently, the map of the terrain is generated as a colored point cloud. To use this map for planning the swing trajectory of a prosthesis, the point cloud would need to be further processed. Promising areas of investigation would be semantic segmentation of the point cloud to help identify possible foot placement, or even simply converting to a 3D occupancy grid. Existing approaches of real-time online swing trajectory control [4] would also need to be extended and validated on non-uniform terrain.
5. The current registration algorithms depend on geometric and/or photometric features for matching the source point cloud with the target point cloud. Hence, a uniform flat terrain, which is the most frequently navigated condition, would lead to failed registration. While this is an open challenge in the field of SLAM research, simple segmentation of individual point clouds and identification of

flat terrain could provide necessary information for effective prosthesis control.

6. The SLAM pipeline implemented here also assumes a static environment with no dynamic objects, which can fail when deployed outside a controlled lab environment. This stands to benefit from several existing SLAM methods such as [7] or [33] that detect and segment moving objects.
7. These approaches should also be extended and explored for controlling exoskeletons.

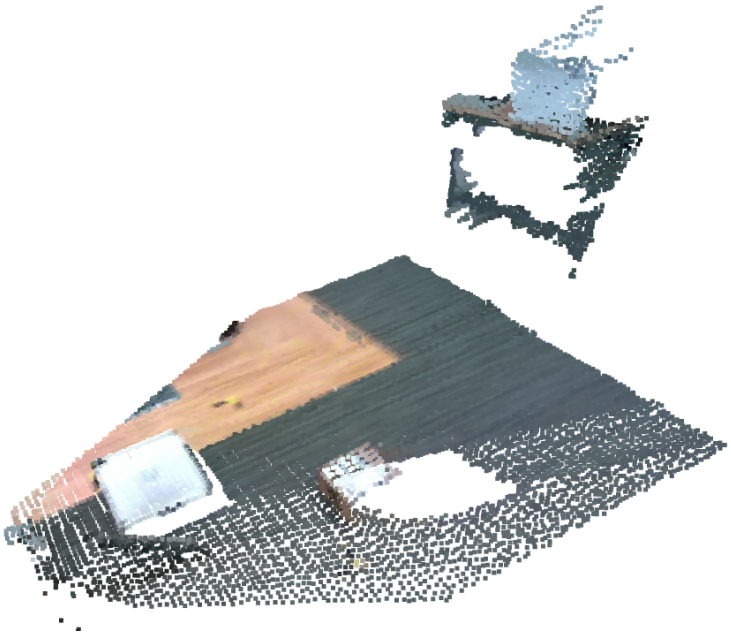
While there are a significant number of challenges that need to be addressed, we are hopeful that this is a good stepping stone in the march toward robust robotic devices to restore or enhance human locomotion.

Appendix A

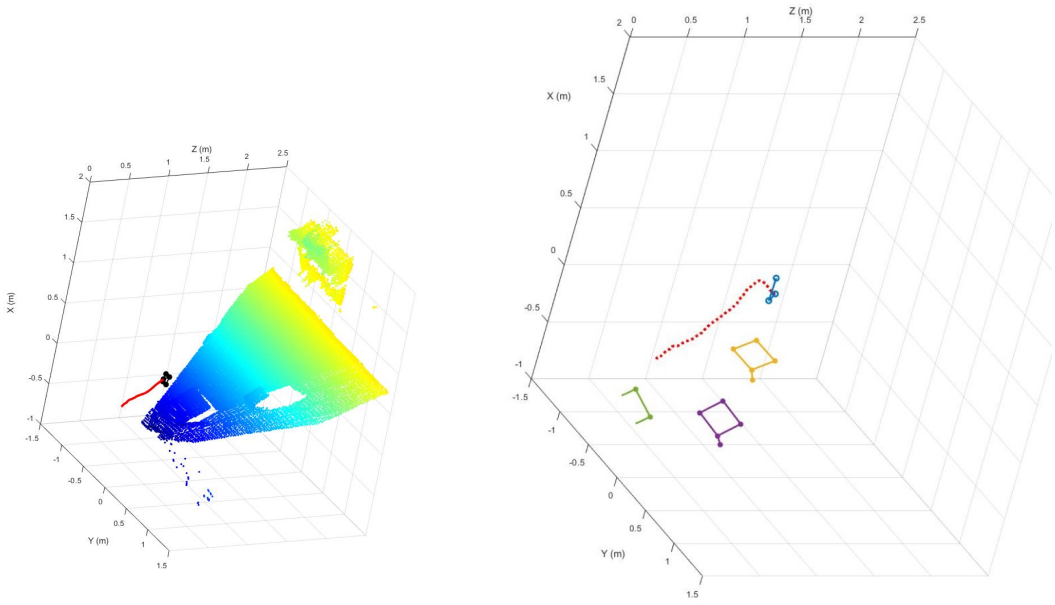
Appendix

A.1 Additional Example Strides Using Intel RealSense

Mapping and trajectory results from four more example strides

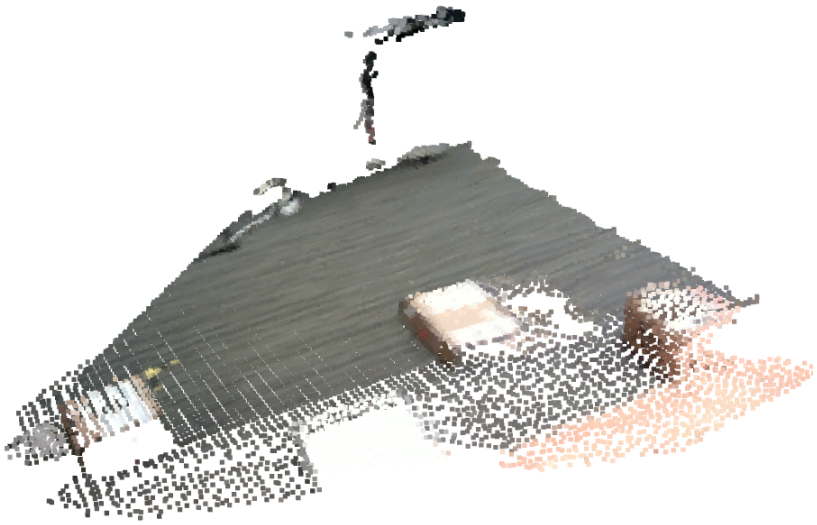


(a) Colored global point cloud

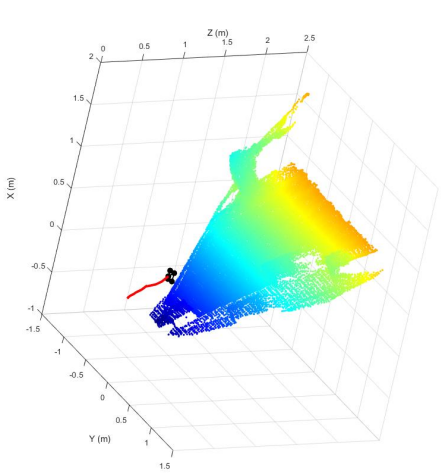


(b) Global point cloud with camera trajectory (c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

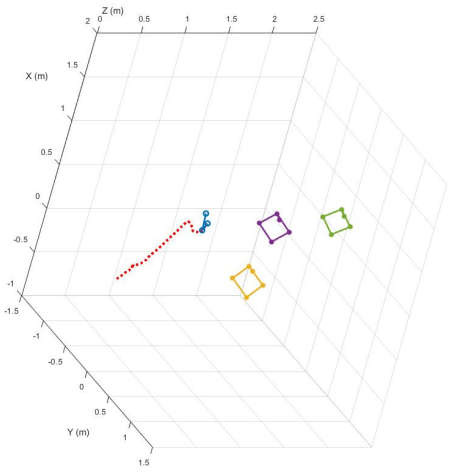
Figure A.1: Additional example 1 of global registered point cloud and camera trajectory resulting from incomplete registration of point clouds recorded during one stride and ground truth from Vicon motion capture



(a) Colored global point cloud



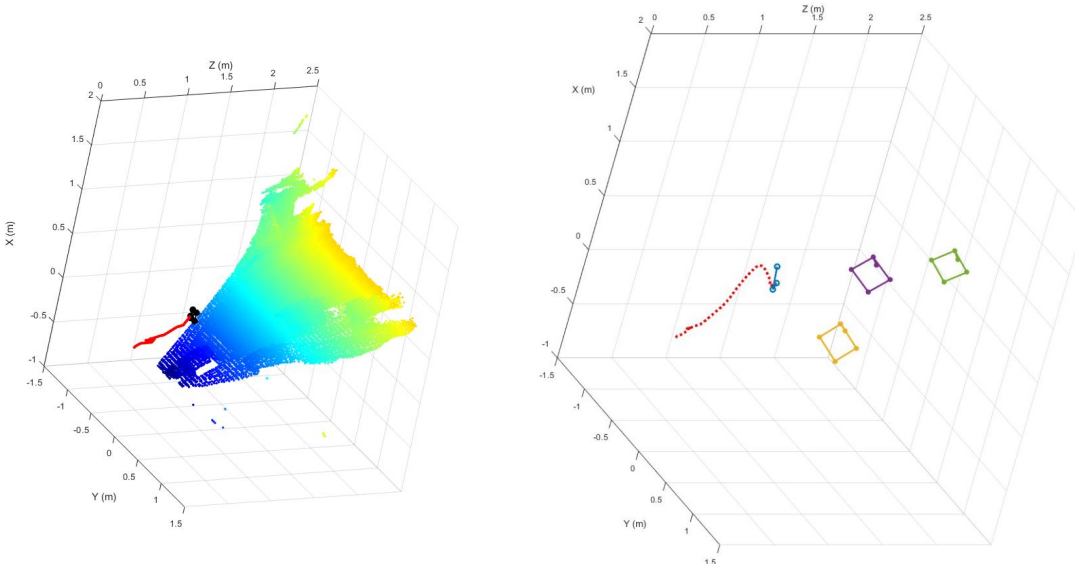
(b) Global point cloud with camera trajectory



(c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

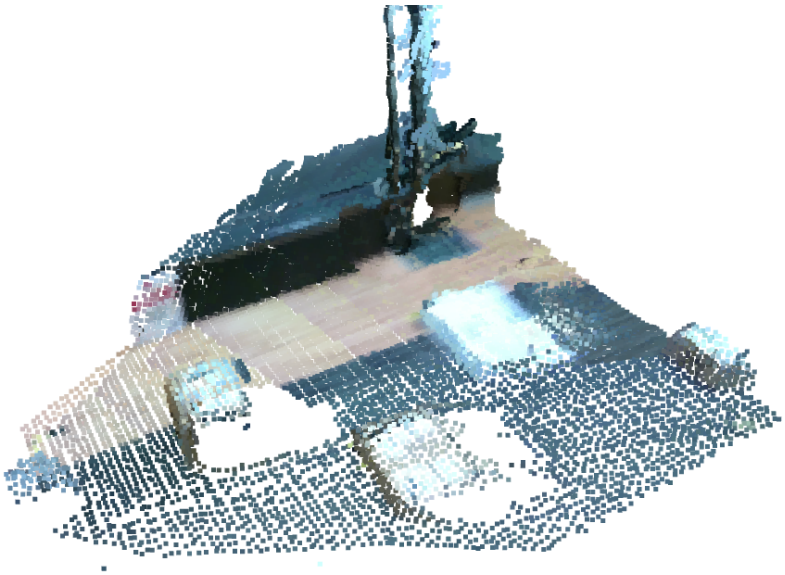
Figure A.2: Additional example 2 of global registered point cloud and camera trajectory resulting from incomplete registration of point clouds recorded during one stride and ground truth from Vicon motion capture

(a) Colored global point cloud

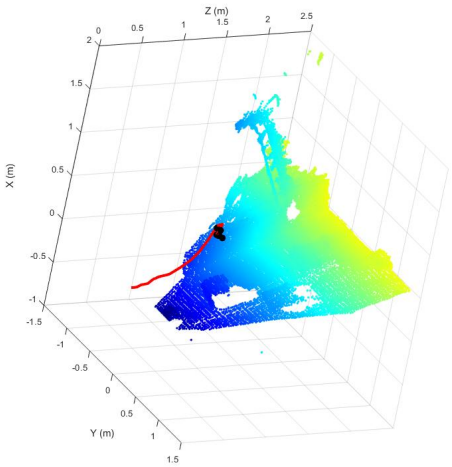


(b) Global point cloud with camera trajectory (c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

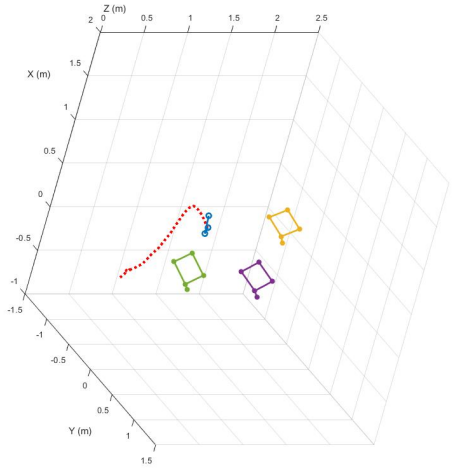
Figure A.3: Additional example 3 of global registered point cloud and camera trajectory resulting from incomplete registration of point clouds recorded during one stride and ground truth from Vicon motion capture



(a) Colored global point cloud

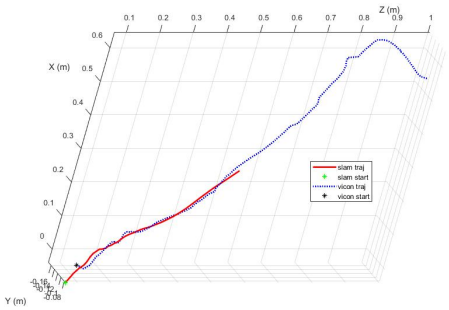


(b) Global point cloud with camera trajectory

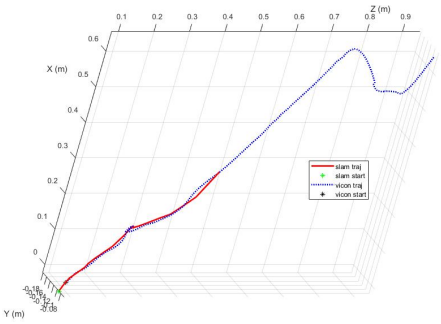


(c) Ground truth trajectory and obstacle locations as recorded by Vicon motion capture

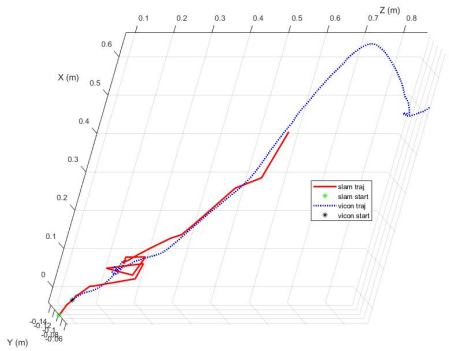
Figure A.4: Additional example 4 of global registered point cloud and camera trajectory resulting from incomplete registration of point clouds recorded during one stride and ground truth from Vicon motion capture



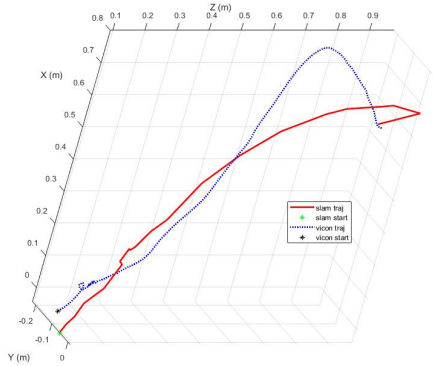
(a) Trajectories from stride in figure A.1



(b) Trajectories from stride in figure A.2



(c) Trajectories from stride in figure A.3



(d) Trajectories from stride in figure A.4

Figure A.5: Comparison between estimated and ground truth trajectories for two strides from figures A.1, A.2, A.3, and A.4 after manual alignment and synchronization of the camera and Vicon co-ordinate frames.

Bibliography

[1] J. S. Matthis, J. L. Yates, and M. M. Hayhoe, "Gaze and the control of foot placement when walking in natural terrain," *Current Biology*, vol. 28, no. 8, pp. 1224–1233, 2018. (document), 1.1, 1.1

[2] B. Kleiner and D. Cesmeci, "Foresighted control of active foot prostheses," 2011. (document), 1.2, 1.2

[3] T. Yan, Y. Sun, T. Liu, C.-H. Cheung, and M. Q.-H. Meng, "A locomotion recognition system using depth images," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6766–6772, IEEE, 2018. (document), 1.2, 1.3

[4] N. Thatte, N. Srinivasan, and H. Geyer, "Real-time reactive trip avoidance for powered transfemoral prostheses.," in *Robotics: Science and Systems*, 2019. (document), 1.2, 1.4, 4.1, 4.3, 4

[5] K. Zhang, C. Xiong, W. Zhang, H. Liu, D. Lai, Y. Rong, and C. Fu, "Environmental features recognition for lower limb prostheses toward predictive walking," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 27, no. 3, pp. 465–476, 2019. (document), 1.2, 1.5

[6] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*, pp. 127–136, IEEE, 2011. (document), 2.2, 2.1, 2.2, 2.2

[7] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *2013 International Conference on 3D Vision-3DV 2013*, pp. 1–8, IEEE, 2013. (document), 2.2, 2.2, 3.4.1, 6

[8] H. G. Chambers and D. H. Sutherland, "A practical guide to gait analysis," *JAAOS-Journal of the American Academy of Orthopaedic Surgeons*, vol. 10, no. 3, pp. 222–231, 2002. (document), 3.3, 3.6

[9] K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and

R. Brookmeyer, "Estimating the prevalence of limb loss in the united states: 2005 to 2050.," *Archives of physical medicine and rehabilitation*, vol. 89, pp. 422–9, Mar 2008. 1.1

[10] "National diabetes statistics report 2020," *Medical Benefits*, vol. 37, no. 4, pp. 3–4, 2020. 1.1

[11] A. D. Grant, "Gait analysis: Normal and pathological function," *JAMA : the journal of the American Medical Association*, vol. 304, no. 8, pp. 907–910, 2010. 1.1

[12] S. Nadeau, B. J. McFadyen, and F. Malouin, "Frontal and sagittal plane analyses of the stair climbing task in healthy adults aged over 40 years: what are the challenges compared to level walking?," *Clinical biomechanics (Bristol, Avon)*, vol. 18, pp. 950–9, Dec 2003. 1.1

[13] F. L. Buczek and P. R. Cavanagh, "Stance phase knee and ankle kinematics and kinetics during level and downhill running.," *Medicine and science in sports and exercise*, vol. 22, pp. 669–77, Oct 1990. 1.1

[14] T. Lenzi, L. Hargrove, and J. Sensinger, "Speed-adaptation mechanism: Robotic prostheses can actively regulate joint torque," *IEEE robotics automation magazine*, vol. 21, no. 4, pp. 94–107, 2014. 1.1

[15] F. Sup, H. Varol, J. Mitchell, T. Withrow, and M. Goldfarb, "Preliminary evaluations of a self-contained anthropomorphic transfemoral prosthesis," *IEEE/ASME transactions on mechatronics*, vol. 14, no. 6, pp. 667–676, 2009. 1.1

[16] D. Quintero, D. J. Villarreal, and R. D. Gregg, "Preliminary experiments with a unified controller for a powered knee-ankle prosthetic leg across walking speeds," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2016-, pp. 5427–5433, 2016. 1.1

[17] A. Wu, F. Dzeladini, T. Brugh, F. Tamburella, N. Tagliamonte, E. Asseldonk, H. van der Kooji, and A. Ijspeert, "An adaptive neuromuscular controller for assistive lower-limb exoskeletons: A preliminary study on subjects with spinal cord injury," *Frontiers in Neurorobotics*, 2017. 1.1

[18] M. R. Tucker, J. Olivier, A. Pagel, H. Bleuler, M. Bouri, O. Lambercy, J. R. Del Millán, R. Riener, H. Vallery, and R. Gassert, "Control strategies for active lower extremity prosthetics and orthotics: A review," *Journal of neuroengineering and rehabilitation*, vol. 12, no. 1, pp. 1–1, 2015. 1.1

[19] T. Yan, M. Cempini, C. M. Oddo, and N. Vitiello, "Review of assistive strategies in powered lower-limb orthoses and exoskeletons," *Robotics and autonomous systems*, vol. 64, pp. 120–136, 2015. 1.1

[20] M. Tschiedel, M. F. Russold, and E. Kaniusas, "Relying on more sense for

enhancing lower limb prostheses control: a review," *Journal of neuroengineering and rehabilitation*, vol. 17, no. 1, pp. 99–99, 2020. 1.2, 2.3

[21] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, pp. 1271–1276, IEEE, 2011. 2.2

[22] M. Labbe and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013. 2.2

[23] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. 2.2

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009. 2.2, 3.4.1

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008. 2.2

[26] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, IEEE International Conference on*, vol. 3, pp. 1470–1470, IEEE Computer Society, 2003. 2.2

[27] R. Jamiruddin, A. O. Sari, J. Shabbir, and T. Anwer, "Rgb-depth slam review," *arXiv preprint arXiv:1805.07696*, 2018. 2.2

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, IEEE, 2012. 2.2, 4.1

[29] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *2014 IEEE international conference on Robotics and automation (ICRA)*, pp. 1524–1531, IEEE, 2014. 2.2

[30] H.-W. Park, P. M. Wensing, S. Kim, *et al.*, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," 2015. 2.3

[31] M. R. Nowicki, D. Belter, A. Kostusiak, P. Cížek, J. Faigl, and P. Skrzypczyński, "An experimental study on feature-based slam for multi-legged robots with rgb-d sensors," *Industrial Robot: An International Journal*, 2017. 2.3, 3.2, 4.3

[32] M. Ramezani, G. Tinchev, E. Iuganov, and M. Fallon, "Online lidar-slam for legged robots with robust registration and deep-learned loop closure," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4158–

4164, IEEE, 2020. 2.3, 3.2

[33] T. Zhang, E. Uchiyama, and Y. Nakamura, "Dense rgb-d slam for humanoid robots in the dynamic humans environment," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 270–276, IEEE, 2018. 2.3, 3.2, 6

[34] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martinez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 3, pp. 314–317, IEEE, 2000. 2.3, 2

[35] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, 2014. 3, 3.1.1

[36] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*, pp. 145–152, IEEE, 2001. 3.1, 3.1.2, 3.3, 3.4.1

[37] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *European Conference on Computer Vision*, pp. 766–782, Springer, 2016. 3.2, 4.3

[38] J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proceedings of the IEEE international conference on computer vision*, pp. 143–152, 2017. 3.4.1, 3.4.1

[39] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018. 3.4.1

[40] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7244–7251, IEEE, 2018. 4.1