Lit Review FINISHED STUFF ONLY

**The Effects of SLAM on Quadrupedal Robots**  
Rey Banzaert  
Junior Research Project Seminar  
March 12, 2025

# 1. Introduction

Simultaneous Localization and Mapping (SLAM) is considered to be the holy grail of autonomous robotics. SLAM, at its core, is the question: “Given a robot placed anywhere, how does it both build a map of its environment and figure out where it is on that map at the same time?” SLAM can be seen in many areas of our world, with autonomous floor cleaning robots (commonly referred to as Roombas), VR headsets, and self-driving cars.

For as long as I can remember, robotics has been a core component of my life. Because of my parents, both of whom are engineers, I started extremely young: learning to solder when I was in kindergarten, building robot kits and messing around with circuits in early elementary school, and watching the MIT 2.007 robotics and 2.009 competitions every year, not to mention the year I spent with my dad at MIT and my mom at Wellesley. Robotics has been my escape for numerous years, all accumulating in this JRPS. I have done large-scale robotics projects before, like my fish car in grade 9 and my CNC router in grade 10, but my interest in autonomous robotics had never really been explored.

During the summer before my junior year of high school, I had an internship at MIT as a research assistant under Professor John J. Leonard and Dr. Kurran Singh, two experts in the field of SLAM. I learned under them, implementing SLAM onto a small wheeled robot, developing my love for the field. At the same time, in the building I was working in—CSAIL—quadrupedal robots were everywhere. I had been watching the development of the MIT Cheetahs my whole life, and seeing the quadrupeds at CSAIL rekindled my interest in them. I was curious about how SLAM worked and also really wanted to build a quadrupedal robot for my JRPS project.

I chose to focus on the larger aspects of SLAM and quadrupedal robot design, specifically focusing on the benefits and drawbacks of employing SLAM on a quadruped. Given time constraints, I chose to not include foothold planning, power management, gait planning, and other types of robots. Throughout my research, my central question was: How does SLAM work on quadrupedal robots and how does it impact their design?

# 2. SLAM

## Simultaneous Localization and Mapping (SLAM) is a critical technology in autonomous robotics, allowing for robots to create a map of their environment while also determining its position on said map. SLAM systems rely on various sensor inputs, such as LiDAR, cameras, and inertial measurement units, to collect data about itself and its environment. By continuously updating its map and location, SLAM allows robots to navigate unknown environments autonomously and adapt to changes in real time.

## 2.1. Sensors

Sensors are the bridge between robots and the outside world. A common challenge in employing SLAM is the usefulness of sensors, as the drive for more sensors is pushed by places other than robotics. As Nikolaus Correll, a roboticist at U.C. Boulder and author of the book *Introduction to Autonomous Robotics* acknowledges, “[The drives to develop sensors] include submarines, automatically opening doors, safety devices for industry, servos for remote-controlled toys, and more recently the cell-phone, automobiles and gaming consoles. These industries are mostly responsible for making ‘exotic’ sensors available at low cost by identifying mass-market applications, e.g., accelerometers and gyroscopes now being used in mass-market smart phones or the 3D depth sensor ‘Kinect’ as part of its XBox gaming console” (89). Sensors are the baseline for every single step—and the most defining feature—of SLAM.

## 2.2. Types of Sensors / What is needed from sensors

Engineers typically sort external collectable data into two groups: 2D and 3D. 2D SLAM often uses LiDAR sensors to generate laser scan data, where a single plane of laser measurements is used to detect obstacles and free space in an environment. This happens rapidly, noted by Correll, “A single-ray laser scanner still provides around 600 distance measurements 10 times per second” (117). This system is computationally efficient, allowing for real-time mapping without using excessive processing power. However, 2D sensors struggle in environments with obstacles that are not visible from a single plane, such as a hole in the floor or something hanging from above.

3D SLAM enhances environmental perception by using depth cameras, stereo vision, or 3D LiDAR to produce pointclouds—dense sets of spatial data points that provide depth information. These pointclouds allow for SLAM to observe complex features both above and below the single plane a 2D LiDAR module could. One thing of note is that while a Pointcloud map of an environment is inherently more detailed, the amount of computational power required to handle so much data can outweigh the benefits. As stated by Correll, “The information generated by sensors can be quite formidable. For example, a simple webcam generates 640x480 color pixels (red, green, and blue) or 921600 Bytes around 30 times per second” (117). Whereas “A single-ray laser scanner still provides around 600 distance measurements 10 times per second” (117).

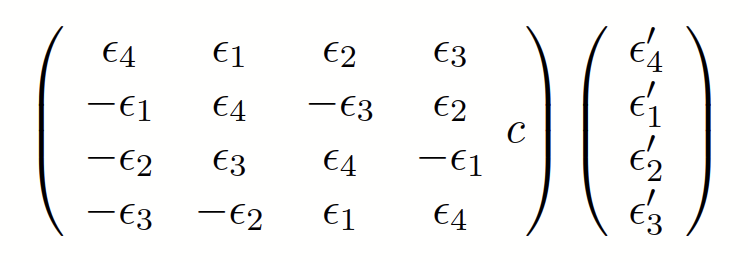
Each one of these sets of data is referred to as a frame. To use this, however, the coordinates of the data points in the frame need to be transformed to where they would be relative to our robot, or relative to a map, which will be useful later on. To achieve this, a translation and rotation are needed to shift them. While normal translation and rotational matrices can work, the most commonly used form is a quaternion matrix, as they are the most stable and computationally cheap (43-44). A quaternion is a 4-dimensional hypersphere (axes being x, y, z, ) that allows a translation of points from one quaternion to another.

“The basic idea is that each rotation can be represented as a rotation around a single axis (a vector in space) by a speciﬁc angle. Given such an axis and an angle θ, one can calculate the so-called Euler parameters or unit quaternion,” with the set of equations:

(Figs. 3.15, 3.16, 3.17, 3.18)

which are constrained by the relationship:

can be used to place a point on a quaternion. Then, two quaternions of and can be multiplied using the equation



(Fig. 3.19)

Quaternions are incredibly useful as “Unlike multiplying two rotation matrices, which requires 27 multiplications and 18 additions, multiplying two quaternions only requires 16 multiplications and 12 additions, making the operation computationally more efficient” (44). By placing all the points from the sensor on a quaternion and multiplying that by a quaternion of the entire map, we can place our sensor data on our map, giving us the ability to map our environment easily.

Placing the robot on the map is also needed in order to figure out where to place the data frames. The most common form of sensor for understanding where the robot is and how it is moving is an inertial measurement unit or IMU. IMUs typically contain two different sensors. Firstly, IMU’s contain an accelerometer, which is a device used to measure acceleration. Accelerometers are used to figure out how fast and in what direction a robot is moving, returning movement data, but missing orientation data. The function of an accelerometer “can be thought of as a mass on a dampened spring. Considering a vertical spring with a mass hanging down from it, we can measure the acting force *F = kx* (Hooke’s law) by measuring the displacement x that the mass has stretched the spring. Using the relationship *F = am*, we can now calculate the acceleration a on the mass m” (97). For orientation, a gyroscope is used, which is:

…an electro-mechanical device that can measure rotational orientation. It is complementary to the accelerometer that measures translational acceleration. Classically, a gyroscope consists of a rotating disc that could freely rotate in a system of pivots and gimbals. When moving the system, the inertial momentum keeps the original orientation of the disc, allowing it to measure the orientation of the system relative to where the system was started. (98)

The combination of sensor data and the use of quaternions to place sensor data on a map gives the input for mapping, the core of SLAM.

**2.3 Map Building**

Map building is fundamental to SLAM, allowing robots to construct a map of their environment for navigation and exploration. At its core, map building involves collecting and then processing sensor data to construct a map, piece by piece. For example, this map can be grid-based, feature-based, or some hybrid, depending on the sensors used.

**2.3.1. 2D GMapping**

Grid mapping, or GMapping, is the simplest and most frequent form of 2-dimensional mapping. GMapping is an algorithm that defines every point along the environment as a cell. By taking a 2D scan of the environment and odometry information about the robot, it sets each cell to a value of how likely it is to have something there (Shen et al. 22).

One problem with GMapping is the imprecision of the data. Rough data can mean something that is defined as a wall is actually an open space, or vice versa, making it possible to crash into something the robot thought was there or maneuver around an obstacle that doesn't actually exist. In order to mitigate this, each cell is not actually in a fixed position, allowing for analysis of moving cells. Cells that move frequently are either moving objects or errors in the data. Removing these cells also removes much of the noise (Narayanaswamy and Kanehiro 23).

**2.3.2. 3D Mapping**

The most common applications of 3D mapping are Oriented FAST (Features from Accelerated Segment Test) and Rotated BREIF (Binary Robust Independent Elementary Features), known as ORB-SLAM, and Real Time Appearance Based Mapping, referred to as RTAB-MAP. ORB-SLAM is an algorithm that takes “into account different kinds of sensors such as monocular, stereo and RGB-D cameras” (Ragot et al. 3), allowing for a wide range of robots to implement it, differing from RTAB, which does not use stereo (3).

Monocular ORB-SLAM uses a single camera as input data. When initialized, ORB creates an empty map and places the robot on the map. With a given frame from the camera, it grayscales the image and attempts to obtain features from the image. However, as depth cannot be obtained from an image, it tracks the feature across multiple frames as the robot moves, allowing it to converge on the feature's actual position. Stereo ORB-SLAM uses a stereo camera, a camera with two or more lenses. Having two cameras at a set distance on the robot allows for the same feature in two different images to be triangulated based on the offset of where it is in the image, as we know that a feature in the left image will be translated in the right by the same amount in every frame. RGB-Depth ORB-SLAM uses an RGBD camera (Red, Green, Blue, Depth) that functions the same as stereo ORB-SLAM, but adds color (4).

RTAB-MAP is an open-source library for SLAM, which typically uses 3D LiDAR modules (5). However, its biggest advantage over ORB is its ability to turn a 3D map into a 2D map, allowing for 2-dimensional navigation of a 3D space, significantly reducing computation time (Pengxiang et al. 3). These 2D maps have a large advantage over GMapping, as they can detect objects above and below where a LiDAR module can not see—such as a hole in the floor or something hanging from above RTAB and ORB maps, both 2D and 3D, are now an ever-expanding map of the environment as the robot moves around.

## 2.4. Path Planning

Once we have a map, the next step is to navigate across the map. To achieve navigation across a map, we employ algorithms that perform path planning. Path planning asks, “How do I get from point A to point B the fastest, without crashing?” While simple for humans, path planning is a complex question for robots.

## 2.5. A\* and D\*



A\* is the most commonly used algorithm for pathfinding and one of the simplest. Rather than exploring everywhere until the goal is reached, A\* only looks in directions that are closer to the end goal. The A\* algorithm starts with a grid of cells, with the robot at the start, the end, and the walls. It then labels every cell with two numbers: the

Euclidean distance to get there and the Euclidean distance to the end. Then, it looks for the path that shortens that distance the fastest. Unlike A\*, D\* starts at the goal and only focuses on the distance to the goal. It allows for a live updating of the distance values of each cell, removing the need for intensive re-calculation of the path (Correll et. al. 77).

The biggest problem with A\* and D\* is the exponentially increasing computational requirements for more detailed maps. This severely limits the use cases of A\* and D\*, as a very precise map is required for consistent obstacle avoidance (77).

## 2.6. Odometry & Localization

Odometry, in its simplest sense, is knowing where a robot is by taking in different measurements from the robot: its movement, IMU data, gyroscope data, and encoder data, among others, to get the robot's location (Shen et al.). One problem in any SLAM, or any robotics challenge for that matter, is that we live in the real world, and our physics equations can never be perfect. For example, if a command is given for a robot to drive forward 1 meter, the robot may incorrectly drive 0.9 meters, 1.1 meters, or rotate a degree to the left. While small, these errors can quickly add up and cause serious problems, as the robot requires precise information (Correll et. al. 15).

In order to address these problems, localization takes place, combining two key processes: action updates and perception updates. Action updates use the robot’s proprioceptive sensors to estimate potential changes from its planned movements. While this provides a way for motion to be tracked, it adds uncertainty—by the nature of having more sensors. Perception updates, on the other hand, use external sensors to compare the robot's observations to the already-made map, reducing uncertainty by aligning sensor data with known features of the map (144-145).

The integration of action and perception updates is formally known as Markov Localization, which uses Bayes’ rule to refine the robot’s understanding of its position. Bayes’ rule connects the chance of being in spot A—given what features are currently observed through external sensors—with the chance of detecting said features at spot B. For example, as a robot approaches a wall, its positional uncertainty grows until the wall is detected. Then, once the wall is detected, it can update its estimate of where it went, reducing uncertainty.

Localization on a map has a similar process. Using odometry to get a general location, localization programs generate a series of features to look for. When a feature is detected, it estimates a position based on the orientation and distance of the feature, updating its position on the global map and its predicted path throughout the map (150). Localization in this way effectively eliminates all problems related to errors in the robot’s movement.

## 2.7. Exploration

Exploration in SLAM focuses on enabling a robot to autonomously map an unknown environment by systematically seeking out new information. A well-designed exploration strategy aims to create a complete or nearly complete map in a reasonable amount of time, using subsequent navigations. Frontier-based exploration is a widely used method that uses the concept of frontiers: regions on the edge of known and unknown parts of the map (Yamauchi 146-147).

The core principle of frontier-based exploration is to maximize the amount of information gained by moving the robot towards frontiers. By positioning itself at said frontiers, the robot can observe unexplored parts of the map, expanding the known environment and pushing back said unknown regions. This process begins with the detection of frontiers on the map. Then, the robot finds accessible, unvisited frontiers and plans a path to the nearest one. When a frontier is reached, the robot re-updates the map, which is then used to find the next batch of frontiers. This cycle allows for the robot to continue autonomously until all accessible space has been explored (146-8). The use of mapping, localizing the position of the robot on the map, planning a robot's path across said map, and exploring the unknown on the map is the current solution to the SLAM problem.

## 2.8. ROS

The Robot Operating System (ROS) is a powerful framework for integrating sensing, actuation, computation, and communication in robotic systems. However, these subsystems—aside from computation—are inherently uncertain due to environmental factors, mechanical failures, and communication instabilities. For example, sensor measurements are affected by environmental conditions and noise, actuators experience inaccuracies from backlash or wheel slipping, and wireless communication is notoriously unreliable (Jiali et al. 129). ROS helps address these uncertainties by offering a framework that distributes the computations across the computer, allowing for an easier time finding out where problems lie.

ROS also includes many different software and packages to aid in the development of SLAM robots. The navigation stack (nav stack) is the backbone to SLAM implementations in ROS. The nav stack allows for a system of packages for path planning, localization, obstacle avoidance, and much more to be easily stitched together, which allows for an entire SLAM architecture to be run through the nav stack. The nav stack consists of several components:



* Input and output: The nav stack, at its core, takes in the sensor data, the odometry, the map, and a movement goal, and returns a series of commands to be sent in order to get the robot to said goal (“Using rviz”).
* Move\_base: The move base is the heart of the nav stack, consisting of:
* The global planner, which plans a path to the current goal from the robot
* The local planner, which tries to stick to the global path, but uses obstacle detection to adjust the path in real time.
* The costmaps, which are a set of maps that represent obstacles and the risk associated with being close to them, with the biggest difference being that the local map updates in real time, and is more refined.
* The recovery behaviors, which are strategies to handle failures in any part of the nav stack (Megalingam et al. 4).

Each component in the framework is either a node or a topic. Nodes are programs that communicate over topics—a stream of data that any node can read or write to (“Using rviz”).

The ROS Visualizer, or RViz, is a 3D visualization tool built into ROS that helps to visualize sensor data, maps, and navigation paths in real time. It can use both 2D and 3D displays from ROS, allowing itself to be an incredibly helpful tool for debugging and monitoring. RViz is particularly useful as it can display the generated map, the robot's estimated pose, and the planned path, allowing for easy debugging (Megalingam et al.1-3).

RQT Graph is a graphical debugging tool built into ROS that automatically generates a visual representation of all nodes and topics and their interactions with one another. This tool is especially useful for troubleshooting missing or incorrectly corrected nodes (4).

Gazebo is a simulation tool for ROS that allows for the creation of virtual environments, which allows for the simulation of a robot and all of its sensors, as if it were in the real world (1). Gazebo uses Universal Robot Description Format (URDF) files, which define all of the physical qualities of the robot: the joints, size, mass, inertia, and location of sensors on the robot, which are then used to create a physics model of the robot. By creating an environment in Gazebo, the user can feed the output data of ROS into Gazebo, which emulates what would happen in real life and returns predicted values (2). Gazebo allows for the testing of robots without a robot or the risk of crashing a robot. Together, this set of tools provided by ROS allows for a significantly easier production of a SLAM system.

# 3. Quadrupeds

Quadrupedal robots have gone through rapid developments over the past few decades, changing from simple systems to very complicated machines capable of traversing a wide range of terrains with both agility and precision. These robots are inspired by biological quadrupeds, incorporating the main features of animal bodies into their design. The engineering behind quadrupedal robots is a delicate balance of features like strength, speed, and size to allow for the best robot possible (Narayanaswamy and Kanehiro 1330). As quadrupeds are more mobile than wheeled robots, they are able to get to and traverse through environments that wheeled robots can not. Being more versatile makes quadrupeds a very compelling platform to employ SLAM on.

## 3.1. Design

The design of quadrupeds is fundamental to their ability to work effectively in different environments. Having four legs allows for more stability than bipedal robots, allowing these robots to navigate through rough, uneven, or slippery terrains—both while keeping balance and being less heavy and costly than five or more legs.

Quadrupedal robots use various leg designs depending on their planned usage and movement requirements:

* Prismatic Legs – These legs use a singular linear extension and retraction mechanism instead of a knee, allowing the robot to adjust to terrains with significant height variation. This configuration is frequently used in robots designed for navigating hazardous terrain.
* Articulated Legs (knee joint) – These legs use multiple joints, similar to biological quadrupeds, allowing for greater flexibility and natural movement by having a knee. They provide enhanced maneuverability and speed over prismatic legs, especially in environments with obstacles that require precise stepping patterns.
* Redundant Articulated Legs (knee with ankle and other joints) – These advanced legs include additional joints after the knee, like an ankle, increasing maneuverability. They function slightly better than an articulated leg but suffer due to the increased complexity, weight, and cost of having an extra joint (Zhong et al.).



The choice of materials in quadrupedal robots significantly influences their weight (and therefore energy efficiency) and durability. Quadrupeds frequently use:

* Carbon fiber composites – Carbon fiber is the best material to use for high strength and low weight, making robots more agile without sacrificing structural integrity.
* Lightweight alloys (e.g., aluminum and titanium) – Lightweight metal alloys like aluminum and titanium are significantly more rigid and sturdy than carbon fiber while being cheaper, making them very useful for structural components. However, they are significantly heavier than carbon fiber.
* Polymer-based materials – Polymer-based materials are typically applied in non-load-bearing parts to reduce the overall weight of the robot (Katz 23).

## 3.2. Actuation Systems

Actuators are essential for allowing controlled movement in quadrupeds. Different types of actuators convert electrical, hydraulic, or pneumatic energy into mechanical motion and force, allowing quick and forceful leg movements. Legged robots require much more torque than wheeled robots, forcing the need for high-powered actuation systems. Common actuation systems include:

* Electromagnetic Actuators – Traditionally, quadrupeds utilize high gear ratio actuators, which provide precise movement with high force at high speeds (Zhong et al.). The MIT Cheetah robot series, described below, has shown the effectiveness of high-torque, low-gear-ratio quasi-direct drive (QDD) actuators, which optimize torque density, allowing for movements like jumping and running. These actuators use brushless DC motors and planetary gear reductions, maximizing the torque output (Katz 23).
* Series Elastic Actuators (SEA) – SEA actuators create elasticity between the actuator and load, improving force control and shock absorption. The shock absorption of these actuators improves the stability of the robot, particularly when operating in unpredictable environments (Zhong et al.).
* Parallel Actuators (PA) – Parallel actuators improve torque transmission efficiency, allowing robots to handle heavier loads while maintaining agility (Zhong et al.).
* Hydraulic and Pneumatic Actuators – Found in heavy-duty quadrupeds, these actuators generate high force output, making them suitable for applications that require powerful leg movements, such as industrial and military applications (Zhong et al.).

## 3.3. Control and Motion Planning

## A quadruped's ability to move efficiently and adapt to different environments relies on its control algorithms and motion planning strategies based on its map of the environment. Good motion planning gives optimal balance, energy efficiency, and real-time responses to obstacles. To achieve an optimal motion plan, quadrupeds use various leg movement plans known as gaits:

## Walking Gait – Ensures stability by keeping at least three legs in contact with the ground at all times, making it ideal for navigating rough terrain.

## Trotting Gait – Balances speed and stability by moving diagonal leg pairs simultaneously.

## Galloping Gait – Allows for high-speed movement by moving like a horse.

## Bounding Gait – Used for fast acceleration by synchronized movement from both the front and rear legs.

## In addition to gaits, modern quadrupeds integrate reinforcement learning into their control algorithms, allowing them to change their gaits based on real-time sensor feedback. This gives them the ability to catch themselves when falling or counteract slipping, reducing the frequency of robot crashes (Narayanaswamy and Kanehiro 1330).

By choosing different configurations of materials, actuators, legs, and gaits, a quadruped can be built to perform many different tasks, including monitoring industries, search and rescue, military aid, and a platform to employ SLAM.

## 3.4. Evolution of Quadrupedal Robots

The beginnings of quadrupedal robots can be traced back decades. The first designs focused on mimicking the basic walking patterns of animals. These early quadrupeds relied on simple mechanical linkages and predefined motion sequences, causing them to lack real-time adaptability and making their inner workings very complicated (Zhang et al. 1165). The 1980s and 1990s saw significant improvements in quadruped design, especially in the areas of stability and energy efficiency. Newer designs introduced active control mechanisms that allowed robots to adjust their leg positions in response to environmental changes (Narayanaswamy and Kanehiro 1330), which is exiting as it provides a way for the robot to adjust to a shift in the ground.

The biggest advancement in quadrupedal robotics comes from Sangbae Kim and the Biomimetic Robotics Laboratory at MIT. Each of the four robots they have created has redefined how quadrupeds are made and has laid the groundwork for most quadrupeds since the early 2010s. The MIT Cheetah series of robots represents a significant breakthrough in quadrupedal robotics, focusing on high-speed movement, agility, and energy efficiency. These robots use high-torque, low-gear-ratio actuators, allowing them to achieve rapid acceleration, perform mid-movement corrections, and do efficient running gaits with as little energy consumption as possible (Katz 23). 

The first iteration of the MIT Cheetah series, MIT Cheetah 1, laid the foundation for all future quadrupedal robots by having an emphasis on efficiency and biomimicry. It used lightweight materials, QDD actuators, and very simple control algorithms to mimic the natural motion of cheetahs in the wild. Though relatively simple compared to later versions, MIT Cheetah 1 showed the feasibility of energy-efficient legged movement and was a stepping stone for more advanced versions (23).

Building upon its predecessor, MIT Cheetah 2 incorporated more powerful actuators, better balance control, and refined trajectory optimization algorithms. One of its biggest innovations was its ability to run and jump autonomously without external support. Additionally, the MIT Cheetah 2 significantly advanced the field of quadrupeds by showing fast, unsupported running at speeds of over 6 m/s (13 mph) (23).

The MIT Cheetah 3 is the first in the MIT Cheetah series to have movement capabilities without external sensors, by having advanced control algorithms that allow it to traverse obstacles and maintain stability. By integrating force control algorithms and real-time predictive modeling, Cheetah 3 could anticipate the ground reaction forces and adjust its gait accordingly, making it function very well on manipulatable, unstable, and rough surfaces. Unlike earlier versions, MIT Cheetah 3 could climb stairs, recover from falling down, and do various other tasks its predecessors were incapable of (23).

The MIT Mini Cheetah is the most recent quadruped in the line of MIT Cheetahs. The Mini Cheetah is a more compact and agile version of the larger MIT Cheetah models. Despite its smaller size, it also uses high-performance actuators and an advanced control system, allowing it to do backflips, rapid direction changes, and recovery maneuvers (23). The MIT Cheetah series has been the foundation for quadrupedal robots, allowing future robots to build off of them and be used professionally, industrially, in research, aid, and as a platform for SLAM.

Based on the MIT Cheetahs, Boston Dynamics’ Spot robot is one of the most advanced quadrupedal robots to date, featuring modular attachments and being the most recognizable SLAM quadruped. It is mainly used in industrial inspections and security, but has also been used for research purposes and other activities. Equipped with depth cameras, IMUs, and LiDAR, Spot is purpose-built for SLAM. Its robotic arm allows it to manipulate things in the real world and do tasks such as opening doors, carrying payloads, and interacting with objects in areas that would be extremely hard for humans to reach (Narayanaswamy and Kanehiro 1330). Spot's versatility has made it a valuable tool in disaster response scenarios, where remote operation and mobility in hazardous conditions are essential.

Another notable quadruped that employs SLAM is the ANYmal robot. Designed for extreme environments, the ANYmal quadruped by ETH Zurich has high-mobility legs, allowing it to adapt to the most complex of terrains. It has been used in offshore platforms, underground tunnels, and disaster sites, showing off its use in challenging conditions where humans could get hurt. Unlike Spot, which is designed for a variety of applications, ANYmal specializes in autonomous inspection tasks using SLAM, particularly in industrial settings. Its onboard sensors allow it to assess equipment, detect gas leaks, and perform safety inspections (1330). ANYmal's ability to operate in remote locations without human supervision has made it an incredibly valuable tool for menial or dangerous tasks, for example, oil and gas industry inspections, nuclear facility monitoring, and environmental hazard assessments.

The Boston Dynamics Spot and the ETH Zurich ANYmal have fundamentally changed how SLAM is used, both within research and industry. They have paved the way for future companies to create cheaper platforms to employ SLAM on.

# 4. Implementation of SLAM in Quadrupeds

Beyond the challenges of building a quadrupedal robot, additional difficulties arise when integrating them with SLAM due to the complexity of their movement. Unlike wheeled robots, which are relatively stable during operation, quadrupeds experience significant movement and instability because they are walking, requiring more advanced localization, mapping, and data collection. Additionally, having more moving parts creates more uncertainty about where each of the robot's parts are, adding to the need for localization and control systems.

## 4.1. Sensor Fusion and Stabilization on Quadrupeds

The effectiveness of SLAM on quadrupeds relies heavily on sensor fusion and stabilization methods that can compensate for the movements of walking. Traditional 2D LiDAR based SLAM methods, while effective on wheeled robots, struggle significantly on quadrupeds, as quadrupeds have more degrees of freedom than wheeled robots (Zhang et al. 1165). This limitation has pushed the development of alternative SLAM systems and strategies that incorporate both visual and internal sensors. Typically, quadrupeds tend to use 3D LiDAR and RGBD camera SLAM systems, like RTAB-MAP or ORB-SLAM (Ragot et al. 5). However, LiDAR sensors have tradeoffs, notably that they weigh more, consume more power, and cost, making them less suitable for smaller quadrupeds (Zhang et al. 1165). However, RGBD SLAM covers all of these fields, being smaller, lighter, cheaper, and having less energy needs than LiDAR. Additionally, quadrupeds can use internal sensors, like IMU’s and gyroscopes, to help estimate their position (Correll et. al. 97-98). This combination of visual and internal measurements greatly improves the SLAM algorithm being used by being able to aid when one of the sensors is harder to read than the others.

However, while IMU’s and gyroscopes are useful, they can sometimes struggle with walking due to the rapid acceleration and deceleration associated with each step, which is made worse when the robot trips or slips. Every time a foot hits the ground or a leg knocks into something, the shake and vibrations can travel throughout the whole robot, manipulating the readings coming out of the sensors. This can make it difficult to get a clean and reliable estimate of the robot’s position and can make the data from its external sensors essentially worthless (Fink and Semini 101919). To counteract these issues, quadrupeds can use a mix of both hardware and software solutions. Shock-absorbing legs and shock-absorbing sensor mounts can help aid in stability. In software, one solution is algorithms that can account for the predicted shakes. By taking the robot’s gait and only taking readings during stable points in the gait, the noise from impacts can be minimized (101915).

# 5. Conclusion

SLAM is one of the most complex problems in the field of autonomous robotics, and in return takes robot’s autonomy to a whole new level. The ability to overcome the inconsistencies of movement and create maps applies to a wide range of fields, such as self driving cars, autonomous house cleaners, search and rescue, research, industrial manufacturing and inspection, and exploration.

Quadrupedal robots are an excellent platform for SLAM because their ability to reach almost anywhere on land surpasses that of wheeled robots. However, designing a quadruped for SLAM is a quite difficult task, requiring a robot with a high payload to hold the computers and sensors required for SLAM, high stability to get non-noisy sensor readings, and high maneuverability to access many places. Ignoring the high costs of actuators and materials to build a robot — and the even higher costs of sensors and computers for said robot — the balance of size, weight, payload, and maneuverability makes the design and creation of a quadruped very difficult.

During this process, I deeply struggled with the style of writing required in the literature review. Writing in a style to summarize a collection of sources into a cohesive argument was challenging for me, as I am used to writing directly from my thoughts. Having to express my thoughts through a collection of quotes — especially quotes that did not have the complete thought I was trying to convey — forced me to do a deeper level of analysis to pull meaning from my sources. In addition, writing this literature review while simultaneously designing and manufacturing a quadrupedal robot allowed me to see the true complexities of my topic, such as reckoning with different design constraints to maximize the efficiency of a leg.

As both SLAM and technologies evolve over time, quadrupedal robots will incorporate increasingly complicated localization and mapping techniques, especially those that use real-time machine learning. One promising direction is the use of hybrid SLAM, which can switch between different mapping techniques based on its environmental conditions. For example, a quadruped could use visual SLAM in well-lit areas and transition to LiDAR-based SLAM in low visibility areas (Zhang et. al. 1165). By continuously refining the approaches used in SLAM, quadrupedal robots will become increasingly capable of navigating and mapping complex environments, making them extremely valuable tools.

# 6. Dedications

Firstly, I want to thank everybody at MIT, specifically the shop staff of Papalardo Lab, Bill, Scott, all of the Steves, Danny, and previously Tasker, for inspiring me and helping my ridiculous projects over the years, and treating me as their own. You all have allowed me to accomplish what I have, and I can not thank you enough. I would also like to thank Ben Katz and Sangbae Kim from the Biomimetics Lab, who have paved the way for quadrupeds and let me ask a million questions. I would also like to thank all of MRG@CSAIL, especially Hayden, who interned with me and let me play with Russ Tedrake’s Spot robot, Alan, who was super cool my whole time there, Jungseok, who let me hide in and use the MoCap room, and especially Prof. John Leonard and Kurran Singh, who let me intern there and study under them, while asking a million questions.

I'd also like to thank my closest friends, notably the new email chain friend group, along with all of the Tigers (go tigers!!!), and all of Div 4, for being the greatest group of friends I could have ever asked for. I'd like to give a special mention to:

to fishboy, my favorite friend. to fishboy, the greatest fish who ever lived. to fishboy, the greatest boy who ever lived. to fishboy, more fish than boy. - Jason Glick Macalalad

My closest friend Anna Olivera, who was an unwavering source of advice and guidance in my times of need. In addition to her steadfast support of the efforts towards this paper, she was of great assistance during the creation of this declaration.

Thank you to my ever longing nemesis, Evelyn “Mr. Beast” St. Clair, for our long standing rivalry, and for rage baiting me for a full hour. Great times had by all, all except for me.

To my wonderful family, Steve Banzaert, Amy Banzaert, and Taff Banzaert, who have listened to my countless rants about the topic, have dealt with my project being strewn throughout our house, along with other annoying things, and especially my parents, who have dedicated much of their time and money to my project. Seriously, this could not happen without you, and I can’t thank you all enough.

And of course, the one person who I could not have done this without, my wonderful JRPS advisor, Emily Naito: who gave me the most wonderful comments on this lit review, who encouraged me to work through the most challenging problems I've ever dealt with, who endured my stupidity, and who tolerated me demanding that I build an absurdly big robot, and who bullied me relentlessly, etc. Emily, thank you for your support of me and more so tolerance of me, and more ;).

# 

# 7. Bibliography

Balasuriya, B.L.E.A. et al. "Outdoor robot navigation using Gmapping based SLAM algorithm." *Moratuwa Engineering Research Conference*, 2016, pp. 403-408. doi: 10.1109/MERCon.2016.7480175.

Bledt, Gerardo et al. "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot." *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 2245-2252. doi: 10.1109/IROS.2018.8593885.

Correll, Nikolaus et al. Introduction to Autonomous Robots: Mechanisms, Sensors, Actuators, and Algorithms. MIT P, 2022.

Da Silva, Vagner et al. "Autonomous navigation strategy in quadruped robots for uneven terrain using 2D laser sensor." *Latin American Robotics Symposium*, 2023, pp. 290-295. doi: 10.1109/LARS/SBR/WRE59448.2023.10333035.

Fink, Geoff and Claudio Semini. "Proprioceptive Sensor Fusion for Quadruped Robot State Estimation." *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 10914-10920. doi: 10.1109/IROS45743.2020.9341521.

Katz, Benjamin. “A Low Cost Modular Actuator for Dynamic Robots.” *A Low Cost Modular Actuator for Dynamic Robots*, MIT, 2018. dspace.mit.edu/handle/1721.1/118671.

Katz, Benjamin et al. "Mini Cheetah: A Platform for Pushing the Limits of Dynamic Quadruped Control." *International Conference on Robotics and Automation*, 2019, pp. 6295-6301. doi: 10.1109/ICRA.2019.8793865.

Maurović, Ivan et al. "Path Planning for Active SLAM Based on the D\* Algorithm With Negative Edge Weights." *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 8, pp. 1321-1331, Aug. 2018. doi: 10.1109/TSMC.2017.2668603.

Megalingam, Rajesh et al. “Comparison of Planned Path and Travelled Path Using ROS Navigation Stack.” *International Conference for Emerging Technology*, 2020, pp. 1-6. doi: 10.1109/INCET49848.2020.9154132.

Narayanaswamy, Nara and Fumio Kanehiro. "Vision-Based Software System for Indoor Quadrupedal Locomotion: Integrated with SLAM, Foothold Planning, and Multimodal Gait." *IEEE/SICE International Symposium on System Integration*, 2024, pp. 1330-1335, doi: 10.1109/SII58957.2024.10417432.

Ragot, Nicholas et al. “Benchmark of Visual SLAM Algorithms: ORB-SLAM2 vs RTAB-Map”. *Eighth International Conference on Emerging Security Technologies*, July 2019. doi: 10.1109/EST.2019.8806213.

Shen, Dong et al. "Research and Implementation of SLAM Based on LIDAR for Four-Wheeled Mobile Robot." *IEEE International Conference of Intelligent Robotic and Control Engineering*, 2018, pp. 19-23. doi: 10.1109/IRCE.2018.8492968.

“Using rviz with the Navigation Stack” *ROS Wiki*, 8 Jan. 2014.wiki.ros.org/navigation/Tutorials/Using%20rviz%20with%20the%20Navigation%20Stack.

Xia, Pengxiang et al. “Comparison of 3D SLAM for Quadrupedal Robot-Based Scanning.” *Computing in Civil Engineering 2021*, pp. 1059–1066. doi: 10.1061/9780784483893.130.

Xu1, Jiali et al. “IOPscience.” *Journal of Physics: Conference Series*, IOP Publishing, 1 Nov. 2020. iopscience.iop.org/article/10.1088/1742-6596/1651/1/012120/meta.

Yamauchi, B. "A frontier-based approach for autonomous exploration." *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97*. 1997, pp. 146-151. ieeexplore.ieee.org/document/7480175.

Zhang, Chi et al. "FRL-SLAM: A Fast, Robust and Lightweight SLAM System for Quadruped Robot Navigation." *IEEE International Conference on Robotics and Biomimetics*, 2021, pp. 1165-1170. doi: 10.1109/ROBIO54168.2021.9739499.

Zhong, Yuhai. “Analysis and research of quadruped robot’s legs: A comprehensive review.” *International Journal of Advanced Robotic Systems*, 2019. doi: 10.1177/1729881419844148.

Lit Review (working on)

# INCOMPLETE:

# 1.0.0 Introduction

# 

# 2.0.0 SLAM

## 2.1.0 General Overview

The Field of SLAM stems from a singular question: Given a robot dropped somewhere in the world, without forms of global positioning, how does the robot build a map of its environment, and simultaneously figure out where it is on said map. While trivial for a human being, in robotics, it is a mix of integration and genius algorithms that allow this to be solved.

## 2.2.0 Sensors

One of the biggest hurdles in SLAM is receiving information about our environment. For humans, this is relatively simple with our eyes. However, for robots, it is a much trickier problem, with vastly different types of sensors available commercially, and a lack of innovation for sensors specifically for the field of autonomous robotics, as AUTHOR acknowledges, “The development of sensors is classically driven by industries other than robotics. These include submarines, automatically opening doors, safety devices for industry, servos for remote- controlled toys, and more recently the cell-phone, automobiles and gaming consoles. These industries are mostly responsible for making “exotic” sensors available at low cost by identifying mass-market applications, e.g., accelerometers and gyroscopes now being used in mass-market smart phones or the 3D depth sensor “Kinect” as part of its XBox gaming console. (p 89 b5)”

### 2.2.1 Types of Sensors / What is needed from sensors

Generally speaking, all forms of data collection for the environment can be categorized into two groups: 2D and 3D. 3D slam typically consists of data sets known as Pointclouds, a collection of XYZ coordinates that create a three-dimensional depth map of the environment. Typically Pointcloud data comes from sensors with 3D capture systems such as death cameras or 3D lidar.If a robot knows where it took different pointclouds from it can merge said Point clouds together to create a cohesive map of an environment. Additionally robots can use cameras and object finding algorithms to locate where certain objects are in an environment. 2D slam typically consists of late laser scan data which is normally obtained through 2D lidar sensors. These sensors emit laser beams to measure the time taken for the beams to reflect back to determine the distance to objects. Additional information on where the robot is is also needed. The most common form of sensor for understanding where the robot is and how it is moving is an inertial measurement unit or IMU. IMU’s typically contain two different sensors, an accelerometer which ”can be thought of as a mass on a dampened spring. Considering a vertical spring with a mass hanging down from it, we can measure the acting force F = kx (Hooke’s law) by measuring the displacement x that the mass has stretched the spring. Using the relationship F = am, we can now calculate the acceleration a on the mass m.( p 97 b5 )” And a gyroscope, which is “an electro-mechanical device that can measure rotational orientation. It is complementary to the accelerometer that measures translational acceleration. Classically, a gyroscope consists of a rotating disc that could freely rotate in a system of pivots and gimbals. When moving the system, the inertial momentum keeps the original orientation of the disc, allowing it to measure the orientation of the system relative to where the system was started.” p 98 b5. One thing of note is that while a pointcloud map of an environment is inherently better, the amount of computational power required to handle so much data can outweigh the benefits. As stated by BOOK, “The information generated by sensors can be quite formidable. For example, a simple webcam generates 640x480 color pixels (red, green and blue) or 921600 Bytes around 30 times per second.” Whereas “A single-ray laser scanner still provides around 600 distance measurements 10 times per second.(p 117 b7)”

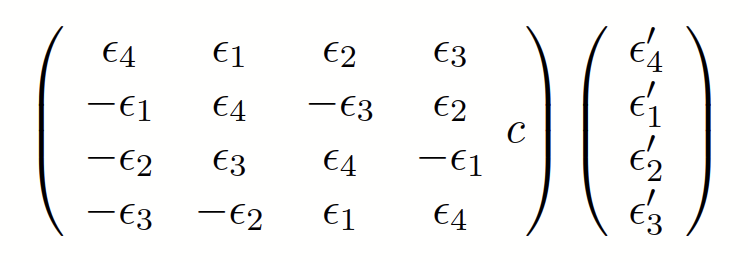
Each one of these sets of data (excluding IMU’s and other forms of odometry) are referred to as a frame. In order to use this, however, the coordinates of the data points in the frame need to be transformed to where they would be relative to our robot, or relative to a map which will be useful later on. In order to achieve this, a translation and rotation is needed to shift them. While normal translation and rotational matrices can work, the most commonly used form is a quaternion matrix, as they are the most stable and computationally cheap (p43-44b). A quaternion is

“a 4-tuple that extends the complex numbers with very general applications in mathematics and representing orientation and rotation in particular. The basic idea is that each rotation can be represented as a rotation around a single axis (a vector in space) by a speciﬁc angle. Given such an axis and an angle θ, one can calculate the so-called Euler parameters or unit quaternion:”

(Figs. 3.15, 3.16, 3.17, 3.18, b)

Which are constrained by the relationship

Can be used to place a point on a quaternion. Then, two quaternions of and can be multiplied using the equation



(Fig. 3.19, b)

“Unlike multiplying two rotation matrices, which requires 27 multiplications and 18 additions, multiplying two quaternions only requires 16 multiplications and 12 additions, making the operation computationally more eﬃcient. In addition, the quaternion representation does not suﬀer from singularities for speciﬁc joint angles, making the approach computationally more robust. (p44b)”

**2.3.0 Map Building**

**2.3.1 Overview**

As stated previously, the aim of SLAM is to build a map. I am going to ignore this right now because for gods sake i cant write an overview

**2.3.2 2D GMapping**

GMapping is the simplest and most frequent form of 2-dimensional mapping. GMapping, or grid mapping, is an algorithm that defines every point along the environment as a cell. Taking a 2D scan of the environment and odometry information about the robot, it changes each cell to a certain value of how likely it is to have an object there (n1p22).

One of the biggest problems with GMapping is how rough the data typically is. This can mean something that is defined as a wall is actually an open space, or vice versa. In order to mitigate this, each cell is not actually in a fixed position, allowing for analysis of moving cells, and the removal of a lot of noise. (p23n5)

While commonly used, GMapping also has its problems, notably because it is a 2D map, that takes in 2D data. This means that if there is an object in any place that the sensor can not see (a hole in the floor, something dangling from the ceiling, etc.), it is not added to the map. However, one of the biggest advantages for a 2D map is the simplicity of the map, allowing for quicker computation. This map is the framework for the rest of SLAM.

**2.3.3 3D Mapping**

The most common employments of 3D mapping are ORB-SLAM and RTAB-MAP. ORB-SLAM is an algorithm that takes “into account different kinds of sensors such as monocular, stereo and RGB-D cameras” (p3n9) allowing for a wide range of robots to implement it, while RTAB focuses solely on “monocular and RGB-D images” (p3n9).

Monocular ORB-SLAM uses a single camera as input data. When initialized, ORB creates an empty map, and places the robot on the map. With a given frame from the camera, it grayscales the image, and attempts to obtain features from the image. However, as depth cannot be obtained from an image, it tracks the feature across multiple frames as the robot moves, allowing for it to converge on the features actual position. Stereo ORB-SLAM uses a stereo camera (a camera with two or more lenses). Having two cameras at a set distance allows for the same feature in two different images to be triangulated based on the offset of where it is in the image, as it is known that a feature in the left image will be translated in the right by the exact same amount in every frame. RGB-Depth ORB-SLAM uses an RGBD camera (Red, Green, Blue, Depth), but applies an error propagation to the data (p4n9).

RTAB-MAP, or Real Time Appearance Based Mapping, is an open source library for SLAM. Similarly to ORB, RTAB can use both feature and depth mapping, but also allows for LIDAR modules to be used, letting a wider variety of robots use them (p5n9). However, its biggest advantage over ORB is its ability to turn said 3D map into a 2D map, allowing for 2 dimensional navigation (p3n7).

**2.4.0 Path Planning**

**2.4.1 A\* and D\***

A\* is the most commonly used algorithm for pathfinding, and one of the simplest. Rather than exploring everywhere until the goal is reached, A\* only looks in directions that are closer to the end goal. The A\* algorithm starts with a grid of cells, with the robot at the start, the end, and walls. It then labels every cell with two numbers; the euclidean distance to get there, and the euclidean distance to the end. Then, it looks for the path that shortens that distance the fastest. Unlike A\*, D\* starts at the goal and only focuses on distance to the goal. It allows for a live updating of the distance values of each cell, removing the need for intensive re-calculation of the path. (b4p77)

The biggest problem with A\* and D\* is the exponentially increasing computational requirements for more detailed maps. This severely limits the use cases of A\* and D\*, as a very precise map is required for consistent obstacle avoidance. (b4p77)

**2.5.0 Odometry**

**2.6.0 Localization**

In order to build frames of reference on our map, we must identify certain sets of data that let us get our own XYZ and PRY from the given data. The simplest version of this is to take the current input data, and try to fit it to every part of the map. However, this is very inefficient, as iterating through every part of the map is not only computationally expensive, but also increasingly time consuming as the map grows. To overcome this, we create sets of data on our map known as features; the most discernible data set in a certain area. Then, taking the current input data, we can try to fit it to only the features. This not only cuts down on computational power, but cuts down on time remarkably. Given that the feature has been seen again, the robot can update its current position based on where it sees the feature, and where it saw the feature previously, given their relative angles and differences in positions. The simplest version of this is “a map that has only a single feature. We assume that the robot is able to obtain the relative range and angle of this feature, each with a certain variance … The position of this measurement mi = [αi, ri] in global coordinates is unknown, but can now easily be calculated if an estimate of the robot’s position xˆk is known. The variance of mi’s components is now the variance of the robot’s position plus the variance of the observation. Now consider the robot moving closer to the obstacle and obtaining additional observations. Although its uncertainty in position is growing, it can now rely on the feature mi to reduce the variance of its old position (as long as it is known that the feature is not moving). Also, repeated observations of the same feature from diﬀerent angles might improve the quality of its observation. The robot has therefore a chance to keep its variance very close to that with which it initially observed the feature and stored it into its map.” p 172 b11

**2.7.0 Exploration**

**2.8.0 ROS**

**2.8.1 What is ROS?**

**2.8.2 Why is ROS used?**

**2.8.3 NAV Stack (lets go baby love the nav stack) (okay its childish gambino homegirl drop it like the nav stack)**

**3.0.0 UGV’s**

**3.1.0 Overview**

**3.2.0 Wheeled Robots**

**3.2.1 Introduction**

**3.2.2 How they work & Versions**

**3.2.3 Design**

**3.3.0 Quadrupeds**

**3.3.1 Introduction**

**3.3.2 How they work & Versions**

**3.3.3 Design**

**4.0.0 Comparison**

**4.1.0 Usage of SLAM in the real world**

**4.2.0 Pros and Cons of UGV’s and sensor implementation**

**5.0.0 Dedications**

**6.0.0 Bibliography**

Quotes and stuff

# 1. Introduction

# 

# 2. SLAM

## 2.1 General Overview

## 2.2 Sensors

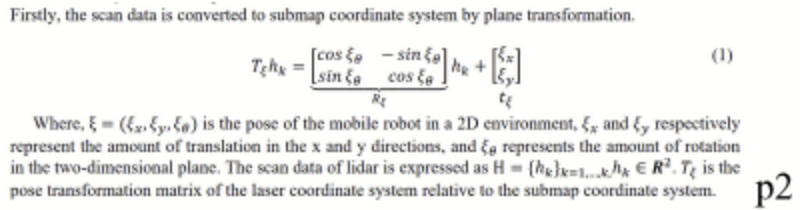
“The development of sensors is classically driven by industries other than robotics. These include submarines, automatically opening doors, safety devices for industry, servos for remote- controlled toys, and more recently the cell-phone, automobiles and gaming consoles. These industries are mostly responsible for making “exotic” sensors available at low cost by identifying mass-market applications, e.g., accelerometers and gyroscopes now being used in mass-market smart phones or the 3D depth sensor “Kinect” as part of its XBox gaming console.” p 89 b5

### 2.2.1 Types of Sensors / What is needed from sensors

One of the biggest hurdles in SLAM is receiving information about our environment. Generally speaking, all forms of data collection for the environment can be categorized into two groups: 2D and 3D (P1 n7). 3D slam typically consists of data sets known as Pointclouds, a collection of XYZ coordinates that create a three-dimensional depth map of the environment. Typically Pointcloud data comes from sensors with 3D capture systems such as death cameras or 3D lidar.If a robot knows where it took different pointclouds from it can merge said Point clouds together to create a cohesive map of an environment. Additionally robots can use cameras and object finding algorithms to locate where certain objects are in an environment. 2D slam typically consists of late laser scan data which is normally obtained through 2D lidar sensors. These sensors emit laser beams to measure the time taken for the beams to reflect back to determine the distance to objects. Additional information on where the robot is is also needed. The most common form of sensor for understanding where the robot is and how it is moving is an inertial measurement unit or IMU. IMU’s typically contain two different sensors, an accelerometer which ”can be thought of as a mass on a dampened spring. Considering a vertical spring with a mass hanging down from it, we can measure the acting force F = kx (Hooke’s law) by measuring the displacement x that the mass has stretched the spring. Using the relationship F = am, we can now calculate the acceleration a on the mass m.( p 97 b5 )” And a gyroscope, which is “an electro-mechanical device that can measure rotational orientation. It is complementary to the accelerometer that measures translational acceleration. Classically, a gyroscope consists of a rotating disc that could freely rotate in a system of pivots and gimbals. When moving the system, the inertial momentum keeps the original orientation of the disc, allowing it to measure the orientation of the system relative to where the system was started.” p 98 b5. One thing of note is that

“The information generated by sensors can be quite formidable. For example, a simple webcam generates 640x480 color pixels (red, green and blue) or 921600 Bytes around 30 times per second. A single-ray laser scanner still provides around 600 distance measurements 10 times per second.” p 117 b7

### 2.2.2 Transformation of sensor data to 0, 0

****n8

“Among these, the preferred representation for computational and stability reasons are Quaternions. A quaternion is a 4-tuple that extends the complex numbers with very general applications in mathematics and representing orientation and rotation in particular. The basic idea is that each rotation can be rep- resented as a rotation around a single axis (a vector in space) by a specific angle. Given such an axis K [kakykz]T and an angle 0, one can calculate the so-called Euler parameters or unit quaternion” p43 b3

### 2.2.3 Features

## Map Building

### Overview

### Loop Closure

“Robots are able to keep track of their position using a model of the noise arising in their drive train and their forward kinematics to propagate this error into a spatial probability density function (Section 8.2). The variance of this distribution can shrink as soon as the robot sees uniquely identiﬁable features with known locations. This can be done for discrete locations using Bayes’ rule (Section 9.2) and for continuous distributions using the Extended Kalman Filter (Section 11.3). The key in- sight here was that every observation will reduce the variance of the robot’s position estimate. Here, the Kalman ﬁlter per- forms an optimal fusion of two observations by weighting them with their variance, i.e., unreliable information counts less than reliable one. In the robot localization problem, one of the observations is typically the robot’s position estimate whereas the other observation comes from a feature with known location on a map. So far, we have assumed that these locations are known.” p 169 b11

P170-172 book

“Consider a map that has only a single feature. We assume that the robot is able to obtain the relative range and angle of this feature, each with a certain variance. An example of this and how to calculate the variance of an observation based on sen- sor uncertainty is described in the line ﬁtting example (Section 8.2.1). This feature could be a wall, but also a graphical tag that the robot can uniquely identify. The position of this mea- surement mi = [αi, ri] in global coordinates is unknown, but can now easily be calculated if an estimate of the robot’s position xˆk is known. The variance of mi’s components is now the variance of the robot’s position plus the variance of the observation. Now consider the robot moving closer to the obstacle and obtaining additional observations. Although its uncertainty in position is growing, it can now rely on the feature mi to re- duce the variance of its old position (as long as its known that the feature is not moving). Also, repeated observations of the same feature from diﬀerent angles might improve the quality of its observation. The robot has therefore a chance to keep its variance very close to that with which it initially observed the feature and stored it into its map. We can actually do this using the EKF framework from Section 9.5. There, we assumed that features have a known location (no variance), but that the robot’s sensing introduces a variance. This variance was propagated into the covariance matrix of the innovation (S). We can now simply add the variance of the estimate of the feature’s position to that of the robot’s sensing process.” p 172 b11

“Usually, a robot obtains an initial estimate of where it is us- ing some onboard sensors (odometry, optical ﬂow, etc.) and uses this estimate to localize features (walls, corners, graphical patterns) in the environment. As soon as a robot revisits the same feature twice, it can update the estimate on its location. This is because the variance of an estimate based on two inde- pendent measurements will always be smaller than any of the variances of the individual measurements. As consecutive observations are not independent, but rather closely correlated, the reﬁned estimate can then be propagated along the robot’s path. This is formalized in EKF-based SLAM. A more intu- itive understanding is provided by a spring-mass analogy: each possible pose (mass) is constrained to its neighboring pose by a spring. The higher the uncertainty of the relative transformation between two poses (e.g., obtained using odometry), the weaker the spring. Every time a robot gains conﬁdence on a relative pose, the spring is stiﬀened instead. Eventually, all poses will be pulled in place. This approach is known as Graph-based SLAM.” p173 b11

### Gmapping

“Gmapping is an open source real-time SLAM solution from ROS. It was written by Giorgio Grisetti et al., University of Freiburg, Germany [16]. The sensor used by the mobile robot is a lidar sensor. Gmapping uses an adaptive resampling technique to reduce the effect of particle degradation. At the same time, the current observation value is introduced when the particle is distributed, and the uncertainty of the particle estimation is reduced.

The Gmapping algorithm is based on the RBPF method and is used to solve the grid-based SLAM problem. Odometry information is required as input, and Lidar data is used as observations. The SLAM of the R-B particle filter can be understood as the decomposition of the posterior probability into two parts of the robot path and the feature map [17] (EQUATION IS HERE) where is the robot trajectory; is the characteristic map observed by ; is the odometer measurement.” (22) Notes 1

“Gmapping algorithm processing includes fixed and non-fixed landmarks [19]. During the composition process, the detected non-static landmarks will be removed from the created map. Only the static landmarks will be added to the state vector.” (23) notes 5

### ORB

"We therefore present in this paper a benchmark comparative study between two visual SLAM approaches: ORB SLAM and RTAB SLAM. The ORB SLAM has been implemented taking into account different kinds of sensors such as monocular, stereo and RGB-D camera. The RTAB SLAM, meanwhile, it has been experimented taking into account both monocular and RGB-D images. The wheelchair has been modified to fit the developed platform. A new motor controller has been developed, the wheelchair is connected within a wireless communication system, and the instrumentation is carried out within a Realsense camera including a monocular, stereo and a RGB-D sensors. A ground truth data based VICON system is used to validate the results obtained.” p3 n9

“1) Monocular ORB-SLAM 2: Monocular SLAM requires just a single camera. The ORB SLAM converts the image to grayscale for its application. The first step is to detect features and initialize the map and its position. Once it gets initialized, it starts creating a map. Monocular SLAM requires a procedure to create an initial map because depth cannot be recovered from a single image. One way to solve the problem is to initially track a known structure. In the context of ltering approaches, points can be initialized with high uncertainty in depth using an inverse depth parametrization, which hopefully will later converge to their real positions. The figure below shows the flow of process in ORB-SLAM monocular.” p4n9

“

2) Stereo ORB-SLAM 2: Stereo ORB-SLAM requires a stereo camera. The feature detection in stereo SLAM is better than monocular SLAM. The initialization is fast and has many advantages over monocular SLAM. For stereo cameras, we extract ORB in both images and for every left ORB we search for a match in the right image. This can be done very efficiently assuming stereo rectified images, so that epipolar lines are horizontal. We then generate the stereo keypoint with the coordinates of the left ORB and the horizontal coordinate of the right match, which is subpixel refined by patch correlation.“ p4 n9

“3) RGB-Depth ORB SLAM 2: RGB-D SLAM requires an RGB image with its depth image. For RGB-D cameras, we extract ORB features on the RGB image, for each feature with coordinates (uL; vL) we transform its depth value d into a virtual right coordinate:

where fx is the horizontal focal length and b is the baseline between the structured light projector and the infrared camera. The uncertainty of the depth sensor is represented by the uncertainty of the virtual right coordinate. In this way, features from stereo and RGB-D input are handled equally by the rest of the system.” p4 n9

### RTAB

“We chose RTAB-Map (Real-Time Appearance-Based Mapping) for the depth-camera-based SLAM method. Figure 2 shows the SLAM and RRT integrated system and workflow. Point cloud data and relevant transform data is sent to rtab-map node and the 3D model is created and visualized in rviz or rtabmapviz (Labbé and Michaud 2019). To enable an automated navigation with RRT algorithm, a 2D slam, gmapping, is running at the same time to provide a 2D map for navigation plan (Umari 2017). Then the RRT algorithm would calculate and plan the route to cover all the unexplored space and provide orders of next navigation target point to the robot. P3 n7

“RTAB-MAP stands for Real Time Appearance Based Mapping. It is distributed as an open source library since 2013. RTAB-Map started as an appearance-based loop closure de- tection approach with memory management (shown in below figure) to deal with large-scale and long-term online operation. It then grew to implement Simultaneous Localization and Mapping (SLAM) on various robots and mobile platforms. RTAB-Map supports both visual and lidar SLAM, providing in one package a tool allowing users to implement and compare a variety of 3D and 2D solutions for a wide range of applications with dierent robots and sensors. It uses depth image with RGB images to construct maps. The graph is created here, where each node contains RBG and depth images with corresponding odometry pose. The links are transformation between each node. When the graph is updated, RTAB-Map compares the new image with all previous ones in the graph to find a loop closure. When a loop closure is found, graph optimization is done to correct the poses in the graph. For each node in the graph, we generate a point cloud from the RGB and depth images. This point cloud is transformed using the pose in the node. The 3D map is then created [11].” p5 n9

### Cartographer

**“**For 3D LiDAR-based method, Google Cartographer (Xu et al. 2017) was selected as the integration platform. Unlike RTAB-MAP, Cartographer cannot display the modeling process in real time, indicating an extra data conversation time cost. P3 n7

### etc.

## Path planning / obstacle avoidance

### A\* & D\*

“There has been a large amount of approaches on plan- ning the robot trajectories, path planning, and path follow- ing [23], [24]. In this paper, we use a graph representation of the environment as a collection of nodes and edges, G(N, E, W) with N representing nodes and Ɛ representing edges and W referring to edge weights. The environment is represented by a discrete grid where each grid cell is occu- pied or empty, representing nodes √ in the graph. Such a grid is called the occupancy grid map created by approximate cell decomposition of the environment [25], [26]. The current robot position is denoted with x = [xy0].” p1324 n10

“Search for a shortest path in a graph uses the criterion as a sum of all edge weights from the robot position to the goal position. The edge weight of the empty, i.e., the cost of enter- ing to an empty cell is set to the Euclidean distance between two adjacent cells depending on the cell size while the edge weight of an occupied cell is set to oo. The localization cells should attract the robot to go through the localization cell thus the edge weight or the cost to enter the localization cells has to be lower than the other cells in the grid. Setting the edge weight to 0 is not giving anything to attract the robot. Actually the edge weight needs to have a negative value. The edge weight of the localization cell compensates a detour the robot takes from the shortest path on the way to the goal position what can be done only using negative cell edge weight.”

P1324 n10

**“**All mentioned methods suffer from the problem of replan- ning a trajectory/path in the scenario with moving obstacles where the robot needs to quit executing the planned trajectory and replan a new one in real time. In approaches like in [1], [2], and [7] the robot's path is only locally optimal where the path is planned only few steps ahead. The advan- tage of here proposed algorithm is a complete path from the robot position to the goal position. Active SLAM approach presented in [10] do not consider obstacles while the robot is moving, i.e., it is necessary to calculate the path from the beginning if obstacles appear, unlike in our approach where the path is efficiently recalculated in the presence of obstacles.” p1322 n10

“1) Path Planning: When the robot has all information about the static and dynamic obstacles in its surrounding it obtains a shortest, obstacle free path to reach the goal. In the pro- posed method the additional input to the path planning module are loop closing points from the active SLAM module. If the cost of the detour the robot takes to reach localization points is acceptable according to localization accuracy increase, the path will be planned through the localization point. The path planning module is based on the D\* path planning algorithm and it is the main contribution of this paper.” p1324 n10

### Odometry

**“**“In this way, a robot using frontier-based exploration will eventually explore all of the accessible space in the world-assuming perfect sensors and perfect motor control.” p147 Notes 3.

“As a ROS node, the position control module first analyzes the parameters such as line speed and angular velocity sent by the base controller node to obtain parameters such as line speed, angular velocity and displacement that the robot needs to move in the left and right wheels. IMU-9 gyroscope measured the robot’s actual moving linear velocity, angular velocity and displacement attitude parameters to calculate, get the robot movement speed and direction error, and through the PID algorithm to the left and right wheel motor speed control, The robot makes the robot move according to the trajectory established by the movement command issued by the host and avoids the obstacle through the laser distance meter. Finally, the robot sends the attitude parameters such as the linear velocity, angular velocity and displacement to the mileage node.” Notes 1

### Localization

“We have now learned two methods to update the belief distribution of where the robot could be in the environment. First, a robot can use external landmarks to update its position. This is known as perception update and relies on exteroception. Second, a robot can observe its internal sensors. This is known as action update and relies on proprioception. The combination of action and perception updates is known as Markov Localization. You can think about the action update to increase the uncertainty of the robot’s position and the perception update to shrink it.” p 144-145 b9

“The location of a robot is subject to uncertainty due to wheel- slip and encoder noise. We learned in the past how the variance in position can be derived from the variance of the robot’s drive train using the error propagation law and the forward kinematics of the robot. One can see that this error is continuously increasing unless the robot has additional observations, e.g., of a static object with known location. This update can be formally done using Bayes’ rule, which relates the likelihood to be at a certain position given that the robot sees a certain feature to the likelihood to see this feature at the hypothetical location. For example, a robot that drives towards a wall will become less and less certain of its position (action update) until it encoun- ters the wall (perception update). It can then use its sensor model that relates its observation with possible positions. Its real location must be therefore somewhere between its original belief and where the sensor tells it to be. Bayes’ rule allows us to perform this location for discrete locations and discrete sensor error distributions. This is inconvenient as we are used to represent our robot’s position with a 2D Gaussian distribution. Also, it seems much easier to just change the mean and variances of this Gaussian instead of updating hundreds of variables” p 150 b9

“In order to localize a robot using a map, we need to perform the following steps 1. Calculate an estimate of our new position using the for- ward kinematics and knowledge of the wheel-speeds that we sent to the robot until the robot encounters some uniquely identiﬁable feature. 2. Calculate the relative position of the feature (a wall, a landmark or beacon) to the robot. 3. Use knowledge of where the feature is located in global coordinates to predict what the robot should see. 4. Calculate the diﬀerence between what the robot actually sees and what it believes it should see. 5. Use the result from (4) to update its belief by weighing each observation with its variance.” p 152 b9

Pages 152 to 161

### Exploration

“We introduce a new approach for exploration based on the concept of frontiers, regions on the boundary between open space and unexplored space. By moving to new frontiers, a mobile robot can extend its map into new territory until the entire environment has been explored. We describe a method for detecting frontiers in evidence grids and navigating to these frontiers.” p 146 notes 3

“Exploration has the potential to free robots from this limitation. We define exploration to be the act of moving through an unknown environment while building a map that can be used for subsequent navigation. A good exploration strategy is one that generates a complete or nearly complete map in a reasonable amount of time.” p146 notes 3

“The central question in exploration is: Given what you know about the world, where should you move to gain as much new information as possible? Initially, you know nothing except what you can see from where you’re standing. You want to build a map that describes as much of the world as possible, and you want to build this map as quickly as possible. The central idea behind frontier-based exploration is: To gain the most new information about the world, move to the boundary between open space and uncharted territory. Frontiers are regions on the boundary between open space and unexplored space. When a robot moves to a frontier, it can see into unexplored space and add the new information to its map. As a result, the mapped territory expands, pushing back the boundary between the known and the unknown. By moving to successive frontiers, the robot can constantly increase its knowledge of the world. We call this strategy frontier-based exploration” p146-147 notes 3

“Once frontiers have been detected within a particular evidence grid, the robot attempts to navigate to the nearest accessible, unvisited frontier. The path planner uses a depth-first search on the grid, starting at the robot's current cell and attempting to take the shortest obstacle-free path to the cell containing the goal location.” p 148 notes 3

“When the robot reaches its destination, that location is added to the list of previously visited frontiers. The robot performs a 360 degree sensor sweep using laser-limited sonar and adds the new information to the evidence grid. Then the robot detects frontiers present in the updated grid and attempts to navigate to the nearest accessible, unvisited frontier.”p148 notes 3

“If the robot is unable to make progress toward its destination, then after a certain amount of time, the robot will determine that the destination is inaccessible, and its location will be added to the list of inaccessible frontiers. The robot will then conduct a sensor sweep, update the evidence grid, and attempt to navigate to the closest remaining accessible, unvisited frontier”p148 notes 3

“In this way, a robot using frontier-based exploration will eventually explore all of the accessible space in the world-assuming perfect sensors and perfect motor control.” p147 notes 3

## ROS

### What ROS is

### Why it is useful

“Robots are systems that combine sensing, actuation, computation, and communication. Except for computation, all of its subsystems are subject to a high degree of uncertainty. This can be observed in daily life: phone calls often are of poor quality, making it hard to understand the other party, characters are diﬃcult to read from far away, the front wheels of your car slip when accelerating on a rainy road from a red light, or your wireless device has a hard time getting a connection. In robotics, measurements taken by on-board sensors are sensitive to changing environmental conditions and subject to electrical and mechanical limitations. Similarly, actuators are not accurate as joints and gears have backlash and wheels do slip. Finally, communication, in particular, wireless either via radio or infrared, is notoriously unreliable.” p 129 n8

“Since map creation and robot positioning both require a large amount of data processing and it is difficult to use a single card type computer, this study makes full use of the ROS distributed processing framework. First, the server and the robot master controller are connected to the same LAN and are on the same network. Segments, then create Node Managers on the server and create keyboard debugs, IMU corrections, mobile robot linear velocity and angular velocity corrections and SLAM nodes, and create Lidars, Mileage, and Base Controller nodes on the mobile robot host controller Then, after all the nodes are registered in the node manager and managed by the node manager, they can perform TCP/IP communication through the peer-to-peer topology in the same network to achieve effective communication between nodes on different hosts. Finally, use the 3D visualization tool RVIZ to realize the SLAM work in the unknown environment on the server.” (19) Notes 1

“As a ROS node, the position control module first analyzes the parameters such as line speed and angular velocity sent by the base controller node to obtain parameters such as line speed, angular velocity and displacement that the robot needs to move in the left and right wheels. IMU-9 gyroscope measured the robot’s actual moving linear velocity, angular velocity and displacement attitude parameters to calculate, get the robot movement speed and direction error, and through the PID algorithm to the left and right wheel motor speed control, The robot makes the robot move according to the trajectory established by the movement command issued by the host and avoids the obstacle through the laser distance meter. Finally, the robot sends the attitude parameters such as the linear velocity, angular velocity and displacement to the mileage node.”

### NAV Stack (lets go baby love the nav stack) (okay its childish gambino homegirl drop it like the nav stack)

# UGV’s

## Overview

“Specifically, we differentiate between locomotion as the ability of the robot to move and manipulation as the ability to move objects in the environment of the robot. Both activities are”p23 b2

“The way in which the individual parts of a robot can move with respect to each other and the environment is called the kinematics of the robot. Kinematics are only concerned with the position and speed (first derivative of position) of those parts, but not its dynamics, which include acceleration (second derivative of position) and jerk (third derivative of position).” p24 b2

“A fundamental difference between locomotion mechanisms is whether they are statically or dynamically stable. A statically stable mechanism will not fall even when all of its joints freeze (Figure 2.2, left). A dynamically stable robot instead requires constant motion to prevent it from falling. Technically, stabil- ity requires the robot to keep it's center of mass to fall within the polygon spanned by its ground-contact points. For example a quadruped robot's feet span a rectangle. Once such a robot lifts one of its feet, this rectangle becomes a triangle. If the pro- jection of the center of mass of the robot along the direction of gravity is outside of this triangle, the robot will fall.” p26 b2

“The concept of degrees-of-freedom, often abbreviated as DOF, is important for defining the possible positions and orientations a robot can reach. An object in the physical world can have up to six degrees of freedom, namely forward/backward, sideways, and up/down as well as rotations around those axes. These rotations are known as pitch, yaw and roll and are illustrated in Figure 2.3.” p27 b2

“Every robot assumes a position in the real world that can be described by its position (x, y and z) and orientation (pitch, yaw and roll) along the three major axes of a Cartesian Co- ordinate system (See also Section 2.3, "Degrees of freedom").” p35 b3

“More complicated systems, such as mobile manipulators or multi-legged robots, make life much easier by defining multiple coordinate systems, e.g. one for each leg and one that describes the posi- tion of the robot in the world frame. These local coordinate systems are known as Frames of Reference. “ p37 b3

“Proprioception refers to the perception of internal states of a robot. This is diﬀerent from exteroception, which describes sensing of anything outside of the robot. Proprioception includes awareness of the robot’s joint angles, its speeds, as well torques and forces.” p 91 b5

## Quadrupedal Robots

### Introducing

### How they work

**“**Legged robots possess the advantages of robust and agile locomotion in challenging terrains, rendering them a viable substitute for human labor in various fields. However, to fully harness these advantages, several crucial considerations must be taken into account.

P1330” n5

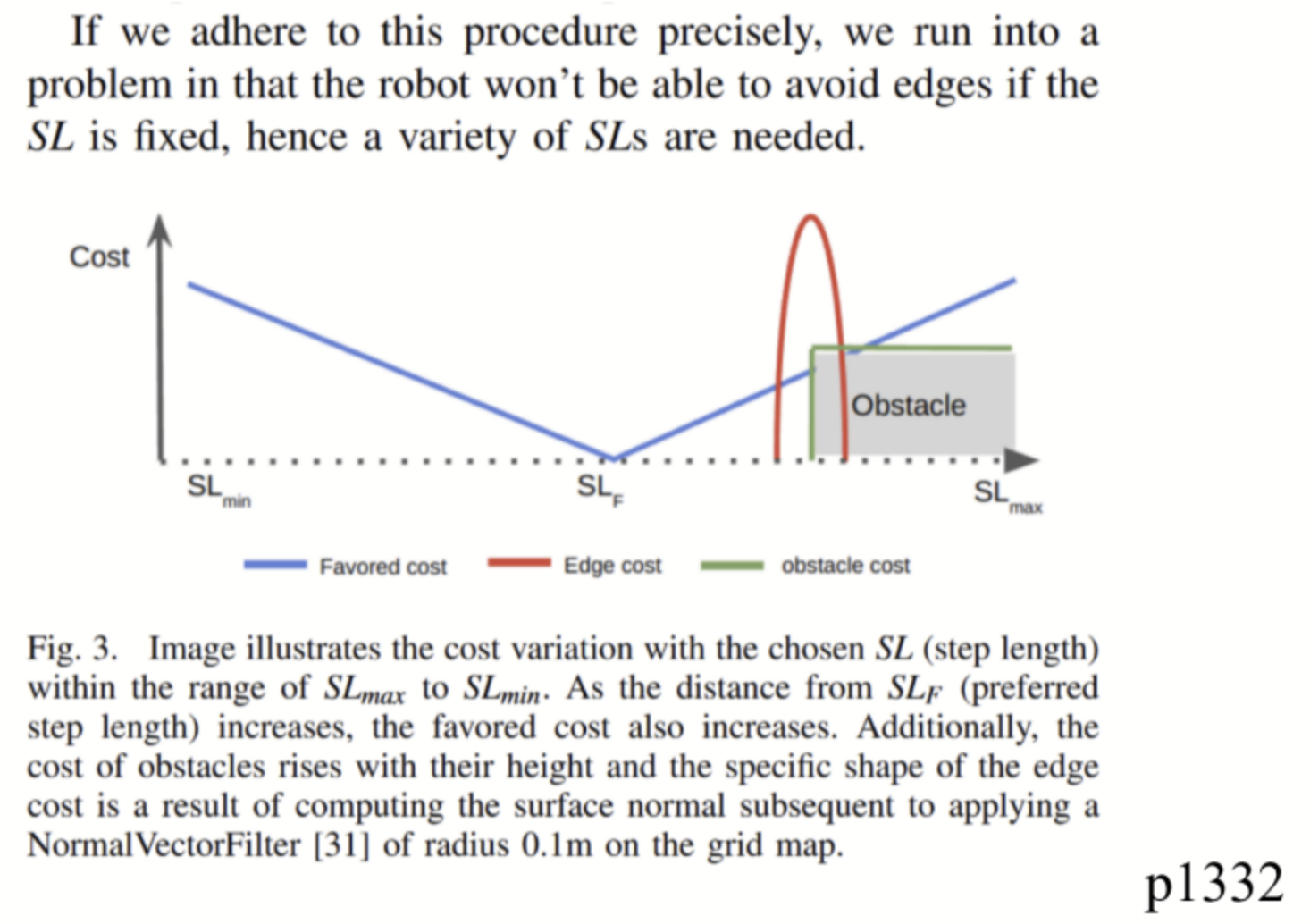
“Unfortunately, due to the quadruped robots will shake violently during walking, the robot navigation system requires that the pose estimator must have extraordinary robustness. In recent years, except for a few research institutions (e.g. Boston Dynamics and ETH Robotic Systems Lab), there are no active studies on quadruped robot environmental perception and comprehending. Furthermore, quadruped robots with fully autonomous navigation capabilities are particularly rare.” Notes 4

P1165

“Unfortunately, the performance of present SLAM technologies are unsatisfactory for quadruped robot navigation. Because the trunk of a legged robot will swing strongly during locomotion, which will easily lead to SLAM system localizing error and makes quadruped robot unable to reach correct destination. 2D LIDAR-based SLAM [10] has only 3 degrees of freedom, which can not meet the needs of legged robots (6 degrees of freedom). So it is usually used to wheeled robot navigation. 3D LIDAR SLAM [11]-[13] can estimates the robot pose accurately, but it has the disadvantages of large volume (nearly one kilogram), high power consumption (approximately 10W) and price (more than two thousand dollars). Compared with LIDAR, visual sensors not only can simply capture abundant semantic environment information and realize drift-free state estimation with loop detection, but also are usable and inexpensive.”p1165 notes 4

“Foothold Planning: The foothold planner receives the optimal velocities from the collision validation module and generates footholds by taking into account the provided step length. This is done while also avoiding the edges of obstacles, which might lead to the robot's leg slipping. P1331 n5

“Another notable framework is CHAMP, an open-source development platform for constructing quadrupedal robots and developing new control algorithms [27], which employs the MIT cheetah controller [19] as its default controller. While it provides considerable flexibility, it is implemented within the ROS navigation stack, limiting its use to two-dimensional scenarios. P1330 n5

n5

“

While quadruped robots offer numerous advantages, they still encounter several challenges. One such challenge is managing changes in the center of mass resulting from the integration of new sensors or robotic manipulators. This can affect the stability and overall performance of the robot. When a legged robot equipped with a manipulator experiences an external force, it can lead to undesired effects such as slipping or the inability to accurately track a desired path with its end- effector [4]. Hybrid robots, which combine components from diverse robot types, offer a versatile and practical solution for a wide range of modern applications [5]. By combining the strengths of manipulators and mobile platforms, these robots excel in tasks requiring both dexterity and mobility. However, the integration of such diverse subsystems introduces new challenges in terms of high-level control and integration [6]. P290 n6

### Versions

### Design

“As an important basic component of quadruped robots, mechanical legs provide the robots with excellent maneuverability and versatility, which determine the core application performance such as job adaptability, walking speed, and load capacity. A large number of robotics institutes for the last few decades have studied mechanical legs used by quadruped robots and published many research results. In this article, we collect these research results and classify them into three categories (prismatic legs, articulated legs, and redundant articulated legs) according to the degrees of freedom and then introduce and analyze them. On this basis, we summarize and study the design methods of the actuators and mechanical leg structures. Finally, we make some suggestions for the development of quadruped robot’s legs in the future. The motivation of this review is to summarize and analyze previous research efforts and provide useful guidance for future robotic designers to develop more efficient mechanical legs of quadruped robots.” n11

“As a kind of legged robot, the quadruped robot has better load and high stability than the biped robot and has larger leg movement space, less mechanism redundancy, and less complexity than the multi-legged robot.4 Just as the famous Japanese foot robot research scholar Shigeo Hirose believes that the quadruped robot is the best form of legged robots from the aspects of stability and control difficulty and manufacturing cost. Therefore, the development of quadruped robots has always been highly valued, and it is of great theoretical and practical value to carry out relevant theoretical and technical research in depth.” n11

### Actuator

“Mechanical leg motion involves repeated dynamic events such as impact, rapid leg swing, and high force interaction with uncertain terrain. Designing actuator systems for high dynamic legged robots has always been one of the major challenges in robotics research. The ideal actuator is to maximize torque, bandwidth, and power while minimizing friction, inertia, and mass loss. However, due to the multifactor coupling relationship between them, it is not clear how to design an actuator to meet these contradictory requirements.” n11

“Conventional electromagnetic actuators are often combined with gearbox to utilize high gear ratio to ensure maximum positional accuracy and stiffness. However, increasing the gear ratio will amplify the impact loads and increase the overall mechanical impedance of the system, while reducing the high gear ratio will increase friction losses and reduce the overall mechanical strength of the system. When the mechanical leg interacts quickly with the environment or there is an impact, a very high impact load is generated at the end actuator and the force control capability is limited. Conventional actuators with high gear ratio do not reduce the effects of the high mechanical impedance of the actuator system resulting from the high impact load. In order for a quadruped robot to dynamically interact with the ground and walk steadily in an unstructured environment, minimizing mechanical impedance is critical. To solve this problem, two new paradigms of actuator design have been proposed, namely the SEA and the PA.” n11

“This actuator builds off the actuation paradigm used in the MIT Cheetah series of robots, using a high torque density electric motor, coupled to a low gear ratio transmission to achieve high torque density, high backdriveablility, and high band- width force control through proprioception [1]. While the MIT Cheetah uses custom- designed motors optimized for torque density, the design presented here leverages the proliferation of high performance brushless motors for RC drones and airplanes, which are manufactured overseas in huge quantities, at very low cost.” p 23 n14

“Thest motors have been tightly integrated into an actuator which also includes a 6:1 single-stage planetary gear reduction, motor controller with built-in position sensor and joint-level control capabilities, output which can handle substantial moment loads for directly attaching limbs to the actuators, and daisy-chainable power and communication to simplify wiring.” p 23-24 n14

## Wheeled robots

### Introducing

### How they work & versions

### Design

“On flat terrain, the wheeled system has the advantages of high speed, outstanding performance in terms of payload-to-weight ratio, stable operation, and easy control. While legged system can perform tasks in rugged terrain. The wheel-leg hybrid drive system combines the characteristics of the wheel and leg system to design a quadruped robot’s leg, which combines the advantages of both forms of motion, that is, walking is used in rough-terrain and rolling is used in flat terrain. This design method improves the running efficiency of the quadruped robot. The application of the wheel-leg hybrid drive system is rarely found in quadruped robots. The robot handle developed by Boston Dynamics uses the wheel-leg hybrid drive system. By combining the rough-terrain capability of legs with the efficiency of wheels, Handle has the best of both worlds.” n11

# Comparison

## Usage of SLAM in the real world

## Pros and Cons of each ugv (ex. accessibility, stability, etc.)

“Commercially, the most dominant form of locomotion is rolling. This is due to the fact that rolling provides by far the most ef- ficient energy-speed ratio (Figure 2.1), making the invention of the wheel one of the greatest technological breakthroughs in history. Consequently, humans have modified their environ- ment to have smooth surfaces of large extent such as the road network, but also warehouse and residential floors. In contrast, evolution has not evolved a single animal with wheel-like actu- ators.” p24 b2

## How each get used

# Dedications

# Bibliography