Tactical Edge Reprogramming for Rapid Autonomy Adaptation

A Major Qualifying Project Report submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

Author:
Megan Aloise, Robotics Engineering & Computer Science

Faculty Advisors:
Professor George Heineman, Computer Science Professor
Professor Therese Smith, Computer Science Professor
Professor Carlo Pinciroli, Robotics Engineering Professor

Mentor:
Ho Chit Siu, Engineer at MIT Lincoln Laboratory

October 13th, 2021

Contents

Abstract

Tactical Edge Reprogramming for Rapid Autonomy Adaptation (TERRAA) focuses on developing robotic agents that can work as a member of human teams in tactical settings without the involvement of a robotics engineering expert. A subset of the TERRAA project, Tactical Formulaic Language Interpretation and Prediction (TFLIP), aims to achieve this by using Tactical Language, a formulaic language utilized by many human Special Weapons and Tactics (SWAT) and Army Special Reaction teams. This language is particularly relevant to tactical room searches, where US Army officers will use this language to describe a room. This language emphasizes the geometry and features of the room that are relevant to method of entry. The TERRAA TFLIP project aims to take advantage of this tactical formulaic language already understood by humans trained in room clearing and introduce that language to a robotic operative. The robot's goal is to utilize the provided verbal description of the room and show an understanding of the description by predicting unseen features of the room. This will be achieved by combining real-time sensor readings and prior knowledge of the room's geometry. The robot entering the room will contain multiple sensors which it can use to observe the room. These sensor readings represent incomplete observations of the room.

The *particle filtering algorithm is* a common method in robotics to estimate a true state based on incomplete observations (Del Moral, 1996). Implementing such a particle filter for our application is a complex process comprised of many elements. One important aspect is determining the score of a particle provided the sensor observations. This project assesses how different methods of scoring and weighting of known information affect the accuracy in particle filter convergence toward the correct true room at various stages of receiving information. Due

to the incomplete initial state of the project, other aspects of the particle filtering algorithm, such as particle representation, particle generation, and particle elimination will also be addressed.

Unfortunately, due to the non-public nature of official tactical language dictionaries and room search protocols, existing autonomous projects created for tactical room search context are sparse to non-existent. However, the particle filtering algorithm has been used in various civilian applications which shall serve as inspirations for this project.

In investigating various room representation and generation techniques, we evaluated a coordinate approach, a geometric properties approach, and a door-centric geometric properties approach. The door-centric approach was found to be the most efficient representation and generation technique due to its simplicity and unique relevancy to tactical language room descriptors and room entry perspective. In investigating various particle elimination techniques, we evaluated a deterministic percentile approach, a probabilistic draw approach, and a probabilistic individual score assessment approach. The probabilistic individual score assessment approach was determined to be the favorable approach due to concerns of improper convergence and lack of accommodation for multimodal score distributions with other techniques. The scoring functions were investigated by creating edge-case representative test cases and collecting room estimate data from each of four iterations of the particle filter for each test case. These were collected while running the particle filter on each separate attribute's optimization score contribution. By doing this, it was determined that three of the six attempted optimization techniques were effectively isolated, where two of the three unsuccessful optimization techniques showed irregular success. The three unsuccessful optimization techniques require further investigation. Once each attribute is determined to have a successful optimization technique, the method of combining each of these score contribution values will require investigation. In addition to attribute optimization, it was discovered from gathering these results that particle generation will need to be altered to accommodate for all potential room attribute values. These findings have identified a promising room representation and generation technique, identified a promising elimination of particles technique, and evaluated individual attribute optimization scoring functions to pave way for future developments of the project.

This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

1. **Introduction**

Tactical Edge Reprogramming for Rapid Autonomy Adaptation (TERRAA) focuses on developing robotic agents that can work as a member of human teams in tactical settings without the involvement of a robotics engineering expert. A subset of the TERRAA project, Tactical Formulaic Language Interpretation and Prediction (TFLIP), aims to achieve this by using Tactical Language, a formulaic language utilized by many human Special Weapons and Tactics (SWAT) and Army Special Reaction teams. This language is particularly relevant to tactical room searches, where US Army officers will use this language to describe a room. This language emphasizes the geometry and features of the room that are relevant to method of entry.

Room searches are often regarded as the most dangerous part of tactical response because it occurs when a responder "leaves an area that he or she controls into one that he or she does not" (Blair et al., 2019). The potential for hidden threats is heightened because of the uncertainty of the room's contents. For example, a corner that cannot be seen from the point of entry could contain a hostile agent. Properties of the room geometry and location of various features drive the measure of safety for any particular room. This determines how a tactical response team would enter the room, as they prioritize clearing areas that present more danger. The US Army, along with other organizations dealing with tactical room searches, have come up with their own formulaic language used to efficiently describe these rooms. The TERRAA TFLIP project aims to take advantage of this tactical formulaic language already understood by humans trained in room clearing and introduce that language to a robotic operative. This way, the humans and the robot will share a method of communication, thereby reducing the need for a robotics engineering expert in the field to facilitate information exchange. Since the language is formulaic, it softens a common barrier found in robotics and language interpretation.

Many robotic systems struggle with understanding natural language. There are many nuances, implied social contexts, and tones involved in human communication. Tactical Language is an ideal middle ground. There is a limited scope to the implications being made in this context, as a robot only needs to be provided with room-relevant context. For example, rooms take up physical space; room walls are typically parallel to one another; a box-shaped room has four corners; etc. Though these are still implications, they are limited by the context of the problem, and therefore manageable by the system.

While TFLIP aims to extend the TERRAA project by utilizing tactical language used by room entry teams, the robot has its own role to play within the room search. The TFLIP proposal is to have a robotic agent that can understand structured tactical language and utilize it for a tangible purpose.

The robot entering the room will contain multiple sensors which it can use to observe the room. TFLIP is operating under the assumption that sensors on the robot will include Light Detection and Ranging (LiDAR) sensors, which return a value proportional to distance from an object in the room, and global positioning sensors, which locate the robot within global space. These sensor readings are incomplete observations of the room. The robot’s goal is to utilize the provided verbal description of the room and show an understanding of the description by predicting unseen features of the room. This will be achieved by combining real-time sensor readings and prior knowledge of the room’s geometry.

Since multiple readings will be taken during room entry and more knowledge of the room will become available over time, the estimate of the complete room will also change over time. Ideally, as more information becomes available, the estimate of the room will become more accurate to the true state of the room. One common method in robotics to estimate a true state based on incomplete observations is by utilizing an instantiation of the *particle filter algorithm* (Del Moral, 1996). This algorithm generates guesses for the true state of the world, assigns some value of confidence to each according to observable information, eliminates guesses driven by those confidence values, and then backfills eliminated guesses driven by attributes of the surviving guesses. It continues this process for each new observation (Thrub et al., 2005). Incomplete observations are noisy global locations of observable room features, such as corners or doors, and the true state of the room is the exact global position of all corners and doors in the room. The TFLIP project uses such limited information to estimate the true state of the world.

There are many elements to how a particle filter would be implemented for this application. One important aspect is determining the score of a particle provided the sensor observations. Since room-representing particles have many attributes, there is a lot of flexibility in how a scoring function can be implemented. The different geometric deductions made from sensor readings will need to be assessed for various stages of receiving information to make best use of the information available. Weighing and comparing these selected attributes effectively will make the score valuable. This project assesses how different scoring methods and weighting of known information affect the accuracy in particle filter convergence to the correct true room at various stages of receiving information.

Before we can assess these scoring methods, we must address the initial state of this project. This project is a small part of a larger project that has already been in development for some time. Incomplete areas of work included developing particle generation and representation, methods of particle elimination, and the various score assignment functions. As such, the further development of these areas falls within the scope of the project goals. The completion of these portions of the project will be referenced to as Stage 1, where the investigation of the scoring functions will be referenced to as Stage 2.

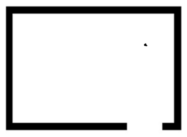
By Stage 2, multiple scoring functions will be ready for assessment. The remaining task is to analyze those functions against each other against various edge-case-representative test case rooms. To achieve this, there are a few tasks remaining: we must define what success looks like and define ways to measure success. The results will then be analyzed and addressed in the discussion section, Section 7.3.

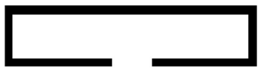
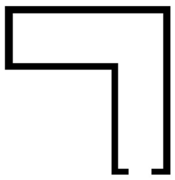
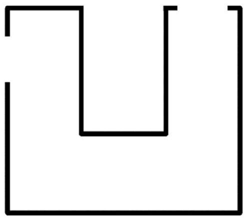
Due to time constraints and technical complications during my project period, I was only able to partially complete Stage 1 of the project, and Stage 2 was implemented for completed portions of Stage 1. All incomplete work is ongoing and left for future work.

2. Background

2.1. Tactical Language and Room Search Entry

Tactical language is a formulaic language used by US Army Special Reaction team officers to describe the geometric properties of a room relevant to search tactics (B. Koo, personal communication, August 29, 2022). A formulaic language is defined as some form of language with consistent structure (Piirainen et al, 2020). These geometric properties determine how the Special Reactions team will enter to clear that room of potential threats. These descriptions are driven largely by clearance priority - areas with the highest likelihood of potential threats. The properties of tactical language include shape, feed, and weight. The most important property in tactical language used to describe a room is the shape of the room. There are many different shapes that can be described using tactical language. Tactical language terms used by the US Army for shape are detailed in the table below.

Shape	Description	Example
Box-Shaped Room	The most common room is the *box-shaped room*. This is a room that has four corners that are all right angles with a rectangular shape.	

Linear Room	Linear rooms have four corners with some defined ratio between the wall lengths that determine its classification as a linear room over a box-shaped room.	
L-Shaped Room	L-Shaped Rooms have six corners, consisting of five traditional 90-degree corners and one inverted 90-degree corner. This makes the L-shaped room take on an L-like shape with two distinct areas of the room.	
U-Shaped Room	U-Shaped rooms have 8 corners, consisting of 6 traditional 90-degree corners and 2 inverted 90-degree corners. This particular configuration makes the U-Shaped room take on a U-like shape with three distinct areas	

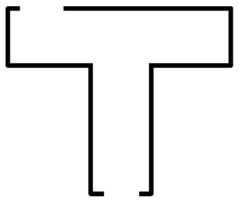
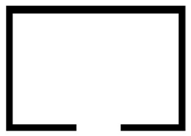
	of the room.	
T-Shaped Room	T-shaped rooms have 8 corners, consisting of 6 traditional 90-degree corners and 2 inverted 90-degree corners. In this particular configuration, this makes the T-Shaped room take on a T-like shape with 3 distinct areas of the room.	
Irregular Room	The irregular room is used to describe any room that does not meet any of the prior defined room types.	

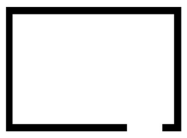
Table 1: Tactical Language Shapes

Various room shape types are listed in Table 1. This includes the tactical language shape, a verbal descriptor of how that defines the room geometry, and an example of how an instance of the respective room could be visually represented.

The tactical language descriptors for room shapes are described above in Table 1. Each of these room shapes carry different tactical implications when it comes to tactical room search. As an example, linear rooms and box-shaped rooms are distinguished from one another because linear rooms have the potential to be hallways. Hallways tend to have more doors and exits leading off into adjacent rooms. This makes linear rooms the most dangerous room to enter in a tactical response, since all adjacent rooms could conceal threats or hidden enemies, meaning that there are more areas to clear. This is a risk that US Army officers communicate to their teams. The four other rooms that can be classified by tactical language are *L-shaped rooms*, *U-shaped rooms*, and *T-Shaped rooms*; all other configurations are referred to as *"irregular rooms"* and will be described relative to the unique shape of that room. The shape of a room will drive how the Special Reactions team will enter the room to clear the area quickly and safely.

In addition to the shape of the room, there are other geometric properties that are important to note in tactical language. It is not enough to just know the shape of the room; the feed is also important. The feed defines from what position relative to the room the door is located.

Feed	Description	Example
Center-Fed	Center-Fed denotes when a door feeds into a room from the center of the wall of entry within some tolerance.	

Corner-Fed	Corner-Fed denotes when a door feeds into a room from non-center section of the wall of entry within some tolerance.	

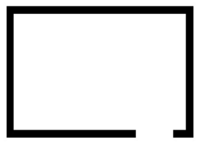
Commented [HG1]: The image associated with "Corner Fed" rooms appears to be corrupt or missing. Please fix.

Table 2: Tactical Language Feeds

Various room feed types are listed in Table 2. This includes the tactical language feed, a verbal descriptor of how that defines the room geometry, and an example of how an instance of the respective room could be visually represented.

Doors can be denoted as *center-fed* or *corner-fed*. Center-fed is when, within some tolerance, the door enters the room from the center of the wall of entry. Corner-fed is when, within some tolerance, the door enters the room with some significant difference to either side. This means that there is a prioritized side to be cleared when entering from a corner fed room.

In order to specify which side the door is entering from relative to the wall of entry, tactical language uses the property of weight. This is an optional tactical language property that may not be relevant to all rooms.

Weight	Description	Example
Left-Heavy	There is more room area to the left of the point of entry.	

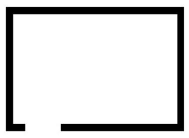
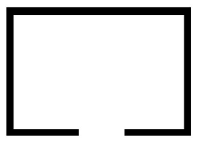
Right-Heavy	There is more room area to the right of the point of entry.	
N/A	The area to the right and the left of the point of entry are approximately the same.	

Table 3: Tactical Language Weights

Various room weight types are listed in Table 3. This includes the tactical language weight, a verbal descriptor of how that defines the room geometry, and an example of how an instance of the respective room could be visually represented.

The weight consists of the two terms left-heavy or right-heavy. Where left-heavy meaning more space is to be clear to the left relative to entry, and right-heavy meaning there is more space to be cleared to the right relative to entry. Additionally, there are ways to describe each corner in a room.

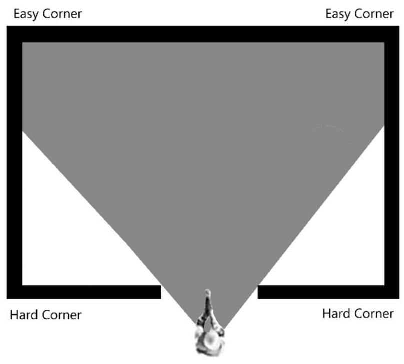


Figure 1: Tactical Language Lines of Vision

Figure 1 displays a visual representation of an expected line of vision from the entrance of a room. Corners that rest within this line of vision are denoted as easy corners, whereas corners that rest outside of this line of vision are denoted as hard corners.

Since corners are areas with high potential for hidden threats, it is important to know as much information as possible about these areas. To classify different types of corners, they are classified as either hard corners or easy corners. Easy corners are the corners that are cleared immediately upon entry from the responder’s primary line of vision, therefore, they are low priority. Hard corners are the corners that are obscured from the view at the point of entry, such as those along the wall of entry. For this reason, it is important to clear hard corners as soon as possible, as they could expose responders to hidden threats.

Tactical language has been developed with the intention of clarifying relevant information to tactical room searches and the high-risk areas within each room. Being a formulaic language, tactical language reduces difficulty in transferring learning over to the robotic agent as compared to traditional, unstructured language.

Since developing a tactical language interpreter is not within the scope of this particular project, from now on, we assume that the tactical language input to the system was a *box-shaped* room for the purpose of simplicity.

2.2. Accounting for Implied Information

When we consider introducing a robot agent into the field in a tactical room search situation, we must acknowledge the robot's ignorance of room properties. Humans have many expectations for what defines and constrains a room, where robots inherently do not. This information must be communicated to the robot for it to gain a human-comparable understanding of the space. If the robots are working as team members with a human team, this understanding is important to be a valuable team member. This type of understanding is reflected by one of the primary technical objectives - to accurately predict locations of features of the entire room based off a few observable measurements. This will help the robot act similarly to how a human agent would, as human agents can take the prior information and room assumptions and use that to deduce features about the room during entry. To do this, we must establish to the robot determined constraints that rooms adhere to via the tactical language descriptions and room-relevant context.

This introduces a common Machine Learning (ML) problem with implied information and subjectiveness of language. Robots do not have any prior information about the world, so we must provide this information to them, in this case, via tactical language. The act of understanding traditional language is called Natural Language Processing (NLP). Typically, ML models are trained with a large dataset with the expectation the robot would be exposed to real data similar to this training data. Humans use a variety of sayings, tones, phrases, and implied information that can drastically change the meaning of any sentence. This is a common issue with NLP that ML implementations attempt to solve (Xiang and Foo, 2021). In this application, there is limited variation in the language being interpreted by the robot, as it is a formulaic language with a limited scope of information. With this in mind, it is possible to manually code each of these specifications for the robot. The benefit of this is a decreased complexity of the code.

2.3. Sensors and Representation of Noise

As the robot enters the space, it needs more than just a general understanding of qualities of a room and some verbal description to create a complete image of the world state. The robot also needs to collect readings as observations about the world. To collect observations, the robot needs a variety of sensors. The assumed sensors to be utilized on this robot would be LiDAR sensor and some global positioning sensor. Each of these sensors has some method of observing the world.

2.3.1. LiDAR Sensor

LiDAR sensors (also known as optical radars) are a form of imaging sensor (Ready, 1997). These sensors are used to collect information about the space by producing pulses of electromagnetic waves, that typically fall in the infrared range of light (McManamon, 2019). This information is primarily representative of the distance between the sensor and some surface. Using this information, it is also possible to estimate velocity of the sensor, velocities of objects in the sensor's field of vision, or the texture of material being observed. This sensor consists of three major components: a light source, a receiver, and an optical positioning system. The light source is typically a laser beam which is used to pulsate light onto the observed area. The photodetector receiver will then measure the reflection of this backscattering of light from the observed area in units of seconds, which can be mapped to a distance measurement. This ability is essential in identifying feature locations within a room. Since LiDAR sensors often measure an array of values in space, angles created by walls can additionally be deduced.

Since the current state of this project does not take input from an existing LiDAR sensor, we must produce some way of simulating this received data. We assume the received data have been processed from a LiDAR sensor, rather than being raw LiDAR point clouds. We also assume that the data has been interpreted to identify wall angles about corners and about the door entrance. When combined with the global positioning sensor, simulated LiDAR measurements can be used to deduce the global coordinates of each feature, along with the deduced unit vector identifying wall directions off each corner and the global angle created by the wall of entry.

In addition to simulating data, we must also simulate predicted noise on this data. This ensures an accurate representation of receiving data and forces the team to accommodate for noise prior to introducing physical sensors to the system, promoting robustness. If there were a decided LiDAR sensor for the physical robot at this stage of the project, we could check the noise specifications for that particular model. However, since that information is not currently available, we must estimate this noise using other means.

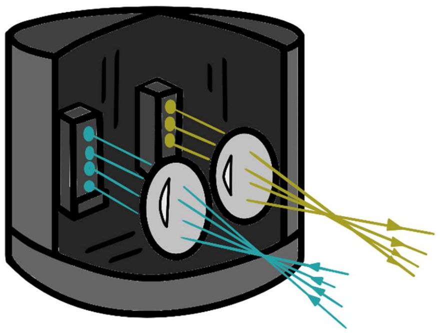


Figure 2: Traditional LiDAR Operation Graphic

Figure 2 displays a representation of the inner workings of a traditional LiDAR sensor. Where outgoing light is represented by the yellow lines and incoming light is represented by the blue lines.

2.3.2. Simulating Reasonable Noise

As robotic systems observe their environment using sensor data, there will always be noise introduced into the system. Though some noise will exist only in small magnitudes, it is still important to account for this noise in our observations. This noise has the potential to greatly affect our observations of the room if not accounted for correctly. An analysis was conducted on pre-crash prediction system in standard automobiles which illustrates this point.

In a high-risk scenario, such as predicting a head-on collision, the false negative prediction must be avoided at all costs - where a positive case is a head-on collision (Dirndorfer et al., 2011). This prediction is put in jeopardy when noise is introduced to the system. Without properly accounting for inaccuracy in the system, the system could inaccurately fail to detect an oncoming collision.

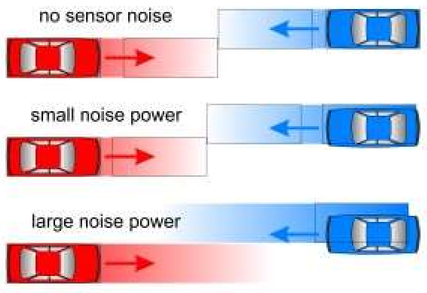


Figure 3: Effect of Noise Incorporation Example

Figure 3 displays a graphic from the study on impact of noise incorporation into a pre-crash detection system. The true paths of the cars are denoted by arrow headings. The system's belief of the paths of the cars are denoted by gradient rectangles of the respective colors of the cars.

This concept is illustrated in Figure 3. When noise was introduced to the pre-crash detection system, there was some inaccuracy in the observed path of the car and the true path of the car. Where the true paths of the cars are denoted as the car figures paired with arrow headings, and the system-observed state of the cars are denoted by the rectangular trajectories. With no simulated sensor noise introduced to the system, the cars are correctly predicted to collide. With small, simulated noise power introduced to the system, the cars are still correctly predicted to collide. This changes with a large noise power where the prediction changes, now predicting no collision, when, in fact, the cars are on course to collide. This could be a potentially fatal false prediction.

In the physical world, there will always be some level of noise or inaccuracy of data present in our system. In the pre-crash detection example, these factors may include noisy distance measurements, time delays, delays in communication between various control units, or otherwise inaccurate information (Dirndorfer et al., 2011). Each of these potential sources of inaccuracy must be taken into consideration within our system and assigned an appropriate noise handling method. With the TERRAA TFLIP project, we must similarly consider the whole signal processing chain. This could include LiDAR sensor noise, global positioning noise, inaccurate tactical language input, and other factors.

There are multiple ways in which systems can account for noise. For this project, the current technique of simulating noise is assuming Gaussian noise. This is done for simplicity's sake for the current state of the project.

2.4. Particle Filter Algorithm

The Particle Filtering Algorithm is an algorithm commonly used in robotics applications. The filter becomes particularly helpful when attempting Simultaneous Localization and Mapping

(SLAM), the most important component for an autonomous robot (Song et al., 2018). The particle filtering algorithm operates under the condition that there exists some observable variable which is related to unobservable information. Using this observable variable, we aim to estimate the unobservable information. Regarding TERRAA-TFLIP's application, the observable variables are our sensor readings, and our unobservable information is the true state of the room we are entering, including features of the room we have yet to see. It is important to understand a traditional application of a particle filter to be able to implement it for the TERRAA-TFLIP project.

SLAM is a necessity in many robotics applications. When responders enter a room, they use their senses to create an image of that room, and, assuming decent depth perception, deduce/recognize where they are located within that space. A robot needs to do just the same. Unfortunately, the adaptability of human's neural pathways is not easy to replicate. There needs to be a procedure for the robot to map the space while accurately updating its position within that space. The particle filtering algorithm is used to execute SLAM in a structured and procedural way, constantly updating predictions about the world with incoming sources of information. Though this project is not following a traditional implementation of SLAM particle filtering, the generalized particle filtering algorithm can still be applied for the purposes of this project.

```
$\boldsymbol{Algorithm\ Particle\ filter(X_{t-1} = None, z_t)}$:
1:    $X_t = [\,]$
2:    $if\ X_{t-1}\ is\ None$:
3:        $for\ m = 1\ to\ M\ do$:
4:            $n = generate\ random\ particle$
5:            $add\ n\ to\ X_{t-1}$
6:    $score_m \alpha\ P(z_t | x_{t-1_m})$
```

```
7:      $add\ < x_{t-1_m}, score_m >\ to\ X_{bel}$
8:      $for\ m = 1\ to\ M\ do$:
9:          $x_{t_m} = accept\ or\ reject(X_{bel})$
10:         $if\ x_{t_m}\ is\ not\ None$:
11:             $add\ x_{t_m}\ to\ X_t$
12:     $for\ m = size(X_t)\ to\ M\ do$:
13:         $n_m = generate\ particles\ (X_t)$
14:         $add\ n_m\ to\ X_t$
15:     $return\ X_t$
```

Figure 4: Particle Filter Algorithm Pseudocode

Figure 4 displays a pseudocode example of how a particle filter could be implemented.

2.4.1. Particle Generation

The primary step in any implementation of a particle filtering algorithm is creating an initial set of *particles*. This is represented in lines 2-5 of Figure 4, where X_{t-1} is used to represent the initial set of particles when running the algorithm. For the run, X_{t-1} will be generated. After the first reading, X_{t-1} will be provided by the previous run of the particle filter. Particles represent different estimates of what we are trying to predict. In our application, particles represent possibilities for all the features of the room the robot is entering. The information contained in these particles will vary based on how we decide to represent rooms within our code. This is introduced in Section 5.2 and further discussed in Section 7.1. This initial set of particles is varied in nature and large in size (potentially over 1000) to increase the possibility of having a truth-representing particle within the initial generation (Thrub et al., 2005).

Initial particle generation can be executed in multiple ways. Particles can be generated randomly to have an unbiased initial sample. This is a method we have chosen to execute for this project. However, there are alternative methods to consider. These particles could be strategically selected. In certain applications, it may be preferable to use the same distribution of initial particles with each particle filtering attempt.

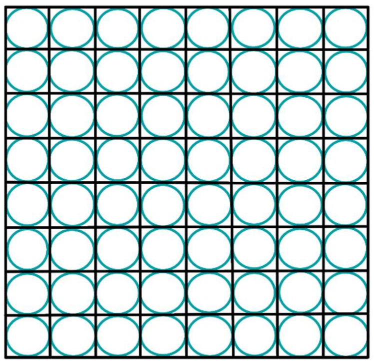


Figure 5: Methods of Particle Generation

Figure 5 displays a visual representation of systematic initial particle generation.

This is demonstrated in Figure 5 where the grid space represents all potential possibilities of state and dots represent some determined, varied set of initial particles. This systematic generation has particles uniformly spaced throughout the possibilities of states (Wang et al., 2018).

Some systems may already have some known information that could contribute to the generation of particles. This is particularly apparent in swarm applications, where there are multiple robots in the area of interest communicating with one another. In this case, the positions of the robots relative to one another may already be known (Saeedi et al, 2015). In an application where some information about the system is apparent, this may impact how particle generation is seeded. Where, if relative robot positions are known with some amount of confidence, each particle would reflect those relative distances.

2.4.2. Score Assignment

Once readings have been obtained, some score assignment function will compare each particle to the known information to describe the likelihood of that particle being correct. This will be reflected by the score. The goal is for the score to be a single value that defines the goodness of fit for any particle given current observations about the world. This is represented by lines 6 and 7 in Figure 4, where z_t is the collection of readings from sensors and X_{t-1m} is the particle being evaluated within the set of particles. The score, score_m will be generated in some way by comparing z_t and X_{t-1m}. For each implementation of the particle filtering algorithm, there will be different attributes relevant to that project. This will determine how z_t and X_{t-1m} are compared to one another. In this case, we are using attributes relevant to defining a room. As analyzing this portion of the particle filter is a primary focus of this project, these representations and scoring implementation will be further detailed in Sections 5.2 and 5.3 and discussed in Sections 7.1 and 7.1.

2.4.3. Elimination of Particles

Based on the score assigned to each particle, there will be an elimination of particles from the set. This is represented by lines 8-11 in Figure 4, where X_{bel} (X belief) contains all particles and their respective scores. This elimination of particles is done in order to remove poor-fit rooms from consideration.

As the range of score values can change per run and per implementation based on the calculation of scores, it is important to have a method of elimination that is valid across various score distributions. This can be done in many ways. As implementing this portion of the particle filter is a primary focus of this project, three notable approaches are detailed in Section 5.4 and discussed in Section 7.2. Particles that are not eliminated from the set are then used to drive the backfilling of eliminated particles in the following resampling step.

2.4.4. Resampling

Now that poor-scoring particles have been eliminated from the set, it is anticipated that these higher-scoring remaining particles have attributes that are more accurate to the true room. Under this assumption, the algorithm will backfill eliminated particles using the attributes of the remaining particles. This is represented by lines 12-14 in Figure 4, where X_t is the set of surviving particles and n_m is a newly generated particle based on X_t. n_m is then added to X_t to create the final set of particles. In our implementation, we have decided to take the mean and standard deviation of all attribute values and generate new particles by selecting attributes from a gaussian distribution using the stated mean and standard deviation. Though this is what was implemented for this project's resampling step, there are other methods. For example, the mean and standard deviation fed into particle generation do not have to be the same as X_t. These can be varied to suit the needs of the project.

After resampling, the final set, X_t, will be returned by the algorithm and reused in the next call with new received readings. For this reason, it is important to know *when* we would like to resample (call the particle filtering algorithm again). Resampling on the same observed information is something that is warned against. This is because, by doing this, we risk converging too sharply on incomplete information and eliminating diversity in our particle set early on. If this becomes a problem, there are two known ways to solve it. The first approach, which is the caution we are currently taking, is to resample infrequently. Particularly, we are only resampling when we receive an additional measurement. In a practical application, this may not be necessary, though it is a noted precaution. The second approach is to intentionally add some completely random particles to the particle set with each iteration. This allows for added variance in case the particle filter has converged too confidently to the incorrect value. Introducing these random particles provides an opportunity to escape this poor conclusion (Thrub et al., 2005). These various methods of resampling and precautions to take regarding resampling are important to keep in mind through the implementation of the project.

2.5. Related Work

For this project, we utilize tactical language descriptions as prior information and fuse that with real-time sensor readings to create an updating estimate of the current room via particle filtering. To our knowledge, tactical language has not yet been utilized in automation for tactical environments. This is largely because the US Army intentionally doesn't publicly release their room entry protocols, including the tactical language descriptions (B. Koo, personal communication, August 29, 2022). In our unique position, we have verification of some language used in these room entries, allowing us to use this in our autonomous robot.

Though the use of tactical language is novel to our use, estimating states via particle filtering has been around for decades, and there are many existing implementations of this. Various optimizations have been made to the particle filtering algorithm across projects. In the paper, *Box Particle Filtering for SLAM with Bounded Errors*, the SLAM particle filtering problem is addressed with the intention of optimizing the size of the set of particles. Traditional particle filters run on hundreds or thousands of particles, leading to expensive computation time. Decreasing the particles needed to run an effective particle filter leads to faster computation time (Wang, 2018). This could be of future use to our project.

Critical Rays Self-Adaptive Particle Filtering SLAM is a project using a LiDAR sensor on a moving robot to perform SLAM on a grid of streets. Like our project, they use a LiDAR sensor to receive measurements, and will therefore experience similar noise (Song, 2018). Though, since our project combines this observed information from the LiDAR and our global positioning reading, our noise will be combined with the global positioning noise. We will not be receiving similar observed information as from the raw LiDAR distance measurement. This project can be useful in assessing what a reasonable LiDAR noise contribution could be as well as referencing for chosen implementation for various portions of the particle filtering algorithm.

Both particle filtering implementations are useful in referencing the various potential implementation of steps in the particle filtering algorithm. However, due to our unique representation of particles, inspiration for methods of scoring these particles will not be represented in their implementations. This original scoring approach is the primary focus of this project.

3. **Methods**

3.1. Simulating Readings

Test case readings were simulated by determining global positional coordinates for each of the eighteen test cases. Corner readings are currently represented in the form (string feature_type, float x_position, float y_position). Door readings are currently represented in the form (string feature_type, float x_position, float y_position, float global_angle). feature_type represents the feature type of the reading, either ‘door’ or ‘corner’. Where x_position is the numerical value for the position of the feature along the global x-axis, and y_position is the numerical value for the position of the feature along the global y-axis. global_angle is parameter that only is associated with a door reading.

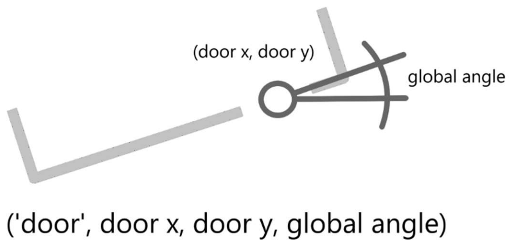


Figure 6: Door Reading Information

Figure 6 displays a visual representation of the data contained in a door feature type reading. This includes the door x position in global space, the door y position in global space, and the global angle. As displayed in the figure, the global angle represents the angle created by the wall of entry relative to a global coordinate system.

3.1.1. Noise Incorporation

A function was developed to incorporate Gaussian noise to the simulated position coordinates. This function takes in some reading and a standard deviation for the noise. It then selects out all features other than the feature_type and selects the noisy value from a Gaussian distribution using the values provided to the function. The function returns a reading in the same format received, but now the values are considered noisy.

3.1.2. Introduction of Readings to Particle Filter

In practical applications, there are common lines of sight when entering a room. The first observation will consistently be the door, as you will always enter a room through the door. After this, the next most likely observations within common lines of sight from the door are the easy corners. We can apply this knowledge in ordering sensor readings for accuracy purposes. A function to achieve this was not completed by the end of the project period and remains a task for future work. Due to this, to gather results, the door measurement was introduced first for all test cases, with purely random introduction of corner readings in succession.

The particle filter runs through each test case in the provided order. Seeing the first reading, then the first through second, first through third, and, finally, the first through fourth. For the sake of analyzing the effectiveness of scoring functions, the estimated state of the world is calculated for each stage of receiving measurements.

3.2. Room Representation & Generation

To implement the particle filter algorithm, we must establish some intuitive way to represent our particles. This is a multidimensional problem with many possible solutions.
We receive our sensor data in global 2-D positional coordinates, referred to as the x-axis and the y-axis. When door measurements are provided, we also have a measurement for the global angle of the wall of entry, about the door.

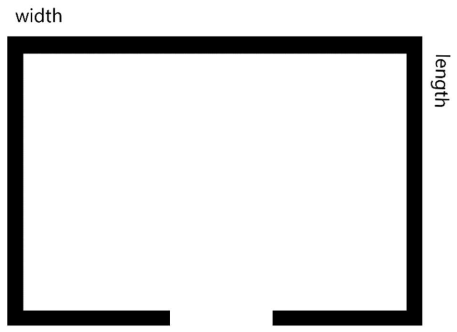


Figure 7: Box Shaped Room with Geometry

Figure 7 displays a visual representation of common geometric properties to describe rectangular shapes: length and width.

We need to determine some convenient seed for particle generation and particle scoring. This project assumes a tactical input of box-shaped rooms. Due to this, there are some box-shaped specific parameters we can work with. Box-shaped rooms are rectangular shapes with doors located throughout. All rectangular shapes have the common properties of length and width, as shown in Figure 7. These geometric properties can be used to our advantage to describe the proportions and size of the room with only two parameters. This is the type of optimization that will be important in generating particles.

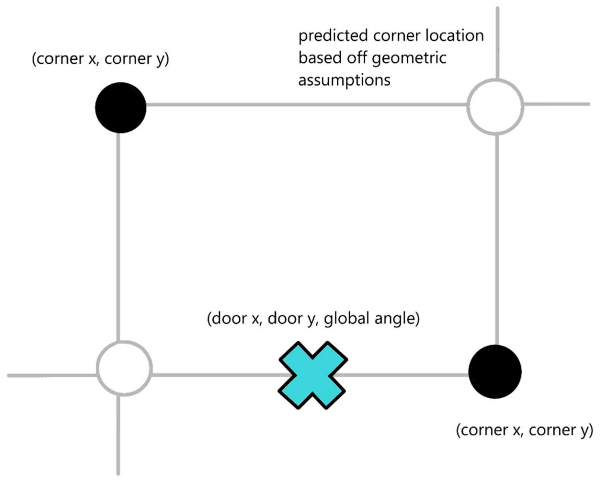

(corner x, corner y)
predicted corner location based off geometric assumptions
(door x, door y, global angle)
(corner x, corner y)

Figure 8: Feature Prediction Using Geometric Assumptions

Figure 8 displays how geometric assumptions can be used in feature prediction. Where bolded sections are received readings and faded sections are predicted features based on geometric assumptions.

Using geometric properties of these shapes will also provide great benefits regarding feature prediction. By using these assumptions of the standard geometric properties of rooms (90-degree corners, parallel walls, etc.), we are intuitively predicting unseen features of the room. Three different implementations of room representation and generation are described below.

In the graph package currently being used by the project, a copy of particles is being stored in coordinate representation. The generation techniques below will cover the conversion of the respective representations to coordinate representations. It is important to keep this information across each particle for later scoring implementations and helpful graph package visualization tools.

3.2.1. Coordinate Method

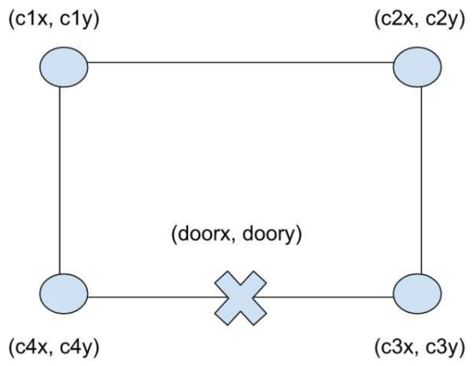
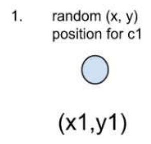
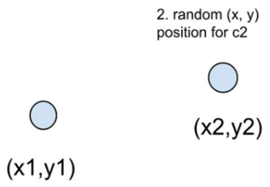
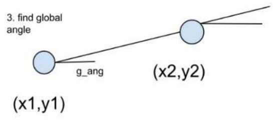
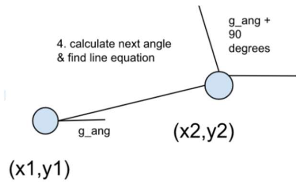
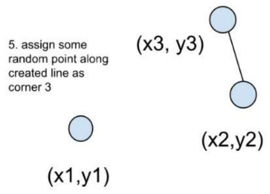


Figure 9: Coordinate Method Representation

Figure 9 displays a visual representation of the coordinate method of room representation. The lines represent walls, the blue circles represent corners, and the blue X represents the door.

The coordinate method represents a room solely based on its features' global coordinate positions. In a four-corner room with single-point entry, the door would be represented in global coordinates, (door x position, door y position). Each corner would be respectively represented in their global coordinates: (corner1 x position, corner1 y position), (corner2 x position, corner2 y position), (corner3 x position, corner3 y position), and (corner4 x position, corner4 y position). This represents all the feature information in the room using 10 parameters. We can shorthand refer to corner 1 as c1, corner 2 as c2, etc.

Step #	Description	Visual Representation

1	Random values chosen for c1 x1 and y1 positions.	
2	Random values chosen for c2 x2 and y2 positions.	
3	From this, global angle calculated between c1 and c2.	
4	90 degrees incremented onto the global angle to create the angle23. The line created from angle23 and c2 drives the location for c3.	 4. calculate next angle & find line equation g_ang + 90 degrees g_ang (x2,y2) (x1,y1)
5	Some random position selected along the line becoming c3.	 5. assign some random point along created line as corner 3 (x3, y3) (x2,y2) (x1,y1)

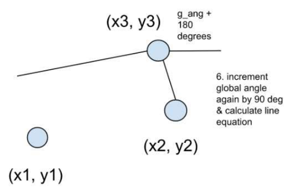
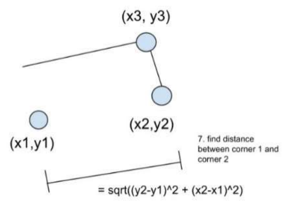
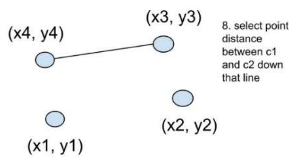
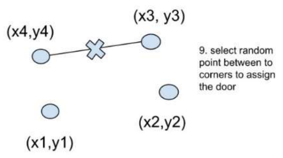
6	90 degrees incremented onto the angle23, creating angle34. Line equation calculated created by angle34 and c3.	 (x3, y3) g_ang + 180 degrees 6. increment global angle again by 90 deg & calculate line equation (x2, y2) (x1, y1)
7	Distance between c1 and c2 calculated.	 (x3, y3) (x1,y1) (x2,y2) 7. find distance between corner 1 and corner 2 = sqrt((y2-y1)^2 + (x2-x1)^2)
8	c4 placed along the line using distance calculated in step 7.	 (x4, y4) (x3, y3) 8. select point distance between c1 and c2 down that line (x2, y2) (x1, y1)
9	Two random adjacent corners selected to include the door. Line equation calculated between these two corners, and door location selected randomly from somewhere along this line between the two corners.	 (x4,y4) (x3, y3) 9. select random point between to corners to assign the door (x2,y2) (x1,y1)

Table 4: Coordinate Method Generation Process

Table 4 displays a step-by-step process for generating rooms with the coordinate method of room representation. This details the step number, a description of the actions taken, and a figure representation of that step.

This representation was generated randomly by selecting random values for (c1 x position, c1 y position). The rest of the room was constructed about these points. The rest of the corners are assigned by iteratively adding 90 degrees to the angle created from the two preceding corners, calculating the line equation at that angle and previous corner location, and then selecting a point along the line to assign the corner location. For the final corner, this will be driven by the distance between c1 and c2.

Once the corners have been placed within global space, two random adjacent corners were selected to be the wall with the door. The line equation was calculated between these two selected corners, and the door location was selected randomly from somewhere along this line in between the two corners. This was now reflected in the door x position and the door y position.

This full generation is for the assumption of random room generation. Generating a particle set based on given attribute values, we would just assign these values to the input seeds and graph directly for the coordinate representation method, as no conversion is necessary for the graph.

3.2.2. Geometric Method

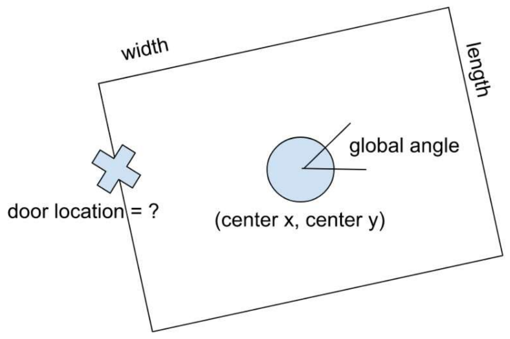


Figure 10: Geometric Representation

Figure 10 displays a visual representation of the geometric method of room representation. The lines represent walls, the blue circle represents the center point of the room, the blue X represents the door, the V at the center point represents the global angle, and the length and width are labelled.

The geometric method represents a room based on traditional geometric properties. These include length, width, center point x, center point y, rotation about center, and some positional door representation. This represents all the feature information in the room using 6 (or 7) parameters. Different door representations are listed below.

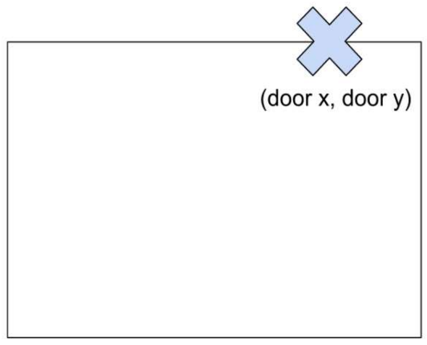


Figure 11: Door Coordinate Representation

Figure 11 displays the door coordinate representation for the geometric room representation approach. In this figure, the blue X represents the door location in global space.

The (door x, door y) method of door representation consists of representing the door using its global positional coordinates. This representation does not actively communicate the door's position relative to the room geometry. This representation also brings the representation to 7 parameters, unlike the 6 in methods detailed below.

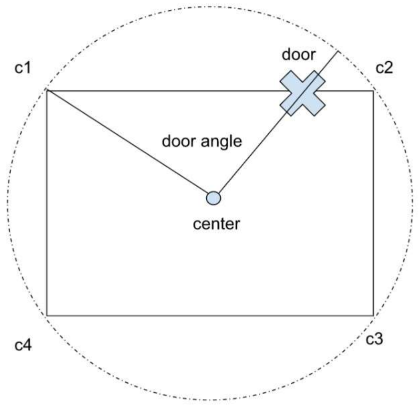


Figure 12: Door Angle Representation

Figure 12 displays the door angle representation for the geometric room representation approach. In this figure, the blue X represents the door location in global space, the blue circle represents the center point of the room, and the V at the center point of the room represents the angle between c1 position, the center point, and the door position. This door angle is the value used to represent the door within the room.

The door angle method represents the door by using the angle created from the first corner to the door location about the center point of the room. This method uses 6 parameters to describe the room, and it actively communicates the door's position relative to the room geometry.

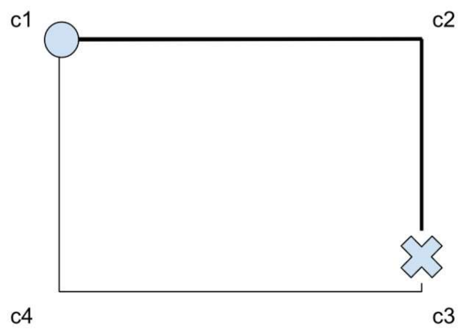
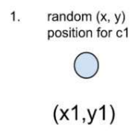


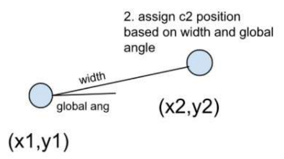
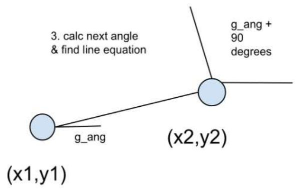
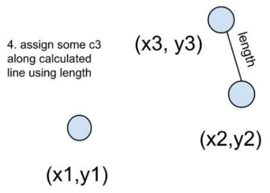
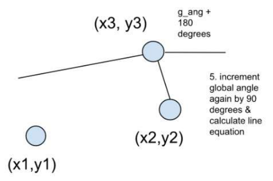
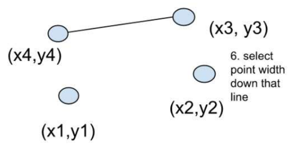
Figure 13: Door Distance Representation

Figure 13 displays the door distance representation for the geometric room representation approach. In this figure, the blue X represents the door, the blue circle represents the first corner, and the bolded line represents what segments are included in the door distance value. The length value of the bolded sections is used to represent the door within the room.

The door distance method represents the door by using the distance around the perimeter of the room from the first corner to the door location. This method uses 6 parameters to describe the room, and it does not actively communicate the door's position relative to the room geometry.

This representation was generated similarly to Section 5.2.1.

Step #	Description	Visual Representation
1	Random values selected for c1's x1 and y1 positions	

2	Width and global angle used to calculate the desired c2.	 2. assign c2 position based on width and global angle width global ang (x2,y2) (x1,y1)
3	90 degrees incremented onto the global angle to create the angle23. The line created from angle23 and c2 drives the location for c3.	 3. calc next angle & find line equation g_ang + 90 degrees g_ang (x2,y2) (x1,y1)
4	Length used to place c3 along generated line.	 4. assign some c3 along calculated line using length (x3, y3) length (x2,y2) (x1,y1)
5	90 degrees incremented onto the angle23, creating angle34. The line equation is calculated created by angle34 and c3.	 (x3, y3) g_ang + 180 degrees 5. increment global angle again by 90 degrees & calculate line equation (x2,y2) (x1,y1)
6	Width used to place c4 along the calculated line.	 (x3, y3) (x4,y4) 6. select point width down that line (x2,y2) (x1,y1)

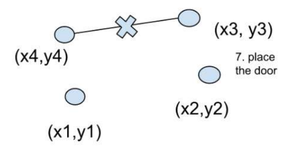
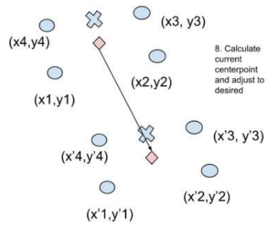
7	Door location parameter used to drive placement of door.	 (x4,y4) (x3, y3) 7. place the door (x2,y2) (x1,y1)
8	The difference between current center point and desired center point calculated. Each node adjusted to this center point.	 (x4,y4) (x3, y3) 8. Calculate current centerpoint and adjust to desired (x2,y2) (x1,y1) (x'4,y'4) (x'3, y'3) (x'2,y'2) (x'1,y'1)

Table 5: Geometric Method Generation Process

Table 5 displays a step-by-step process for generating rooms with the geometric method of room representation. This details the step number, a description of the actions taken, and a figure representation of that step.

This representation was generated randomly by selecting random values for each parameter: length, width, center point x, center point y, rotation about center (global angle), and some positional door representation. First, a random value was generated for (c1 x position, c1 y position). The rest of the room was constructed about this point. The rest of the corners are assigned by iteratively adding 90 degrees to the angle created from the two preceding corners, calculating the line equation at that angle and previous corner location, and then selecting a point along the line provided length or width seeds to assign the corner location, respectively.

Once all the corner positions and the door positions were calculated, the center point of the current room representation was calculated. The difference between this center point and the desired center point for both x and y coordinates was calculated. This difference was then added to each feature coordinate to readjust the desired center point.

When a door is represented in (door x location, door y location) global coordinate system, no conversion is necessary. Though this representation does not explicitly ensure the door lies along a wall. To generate this door location from a seed or randomly generated (door x location, door y location), the values would be directly passed to the graph.

When the door is represented in some angle created between the first corner coordinate, the center point, and the door location, there is some additional computation needing to be done to acquire the door x position and door y position to be passed to the graph.

To acquire these global position coordinates while ensuring the door rests directly between two corners, we first had to figure out which two corners the door was between. Since we know the global coordinates of each corner location, we were able to deduce the distances from each corner to the center point, as well as each corner from one another. This was all the information needed to calculate the internal angles between each successive pair of nodes and the center point using Side/Side/Side Law of Cosines. Once these center angles were acquired, we could deduce at which angle the door location would switch walls in any given room. This, coupled with the information of the door angle, revealed which wall the door was located on. For the respective wall, the angular percent along that wall was determined by taking the door angle minus start angle of that wall over the range of degrees on that quadrant. The line equation created by the wall was then calculated, and length of the line was determined. The door was then placed that percentage of length down the wall. This (door x position, door y position) was deduced and set in the graph.

When the door location is represented in some distance around the perimeter from the first corner, there is some additional computation that needs to be done to acquire the (door x position, door y position) to be passed into the graph.

Though this specific door representation did not have an opportunity to be implemented, it likely would have been executed as follows. The distance from the first corner would be provided to represent the door location. The distance between each successive corner would be calculated from the global coordinate system. The distance from the first corner to the door would be calculated from these values. Wherever the door location surpasses these cutoffs, we would have determined which wall the door lies along. We would take this cutoff corner and calculate the equation for a line between this corner and its successive corner. Then we would take the difference between the door location value and the cutoff corner and calculate the (x position, y position) that distance down the line. The (door x position, door y position) would be then deduced and included in the graph.

3.2.3. Door-Centric Representation

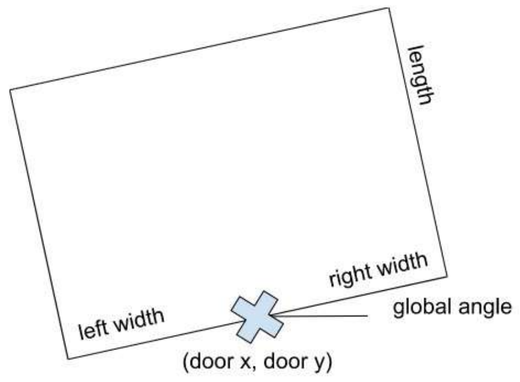
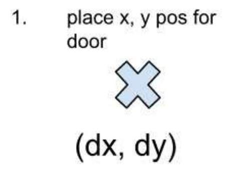
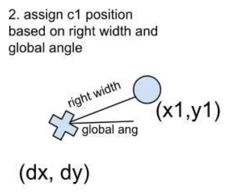
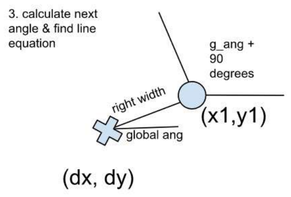
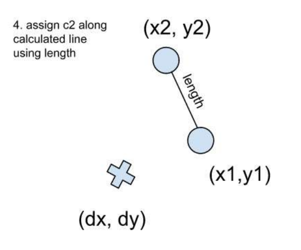
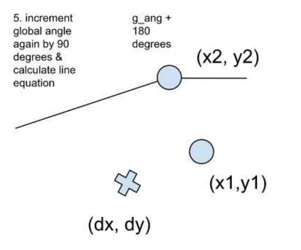
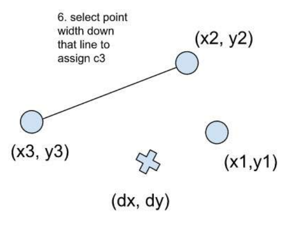
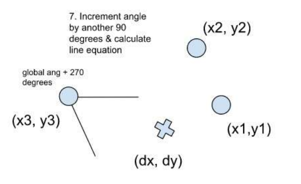


Figure 14: Door-Centric Room Representation

Figure 14 displays a visual representation of the door-centric method of room representation. The lines represent walls, the blue arrow loop represents the global angle, and the gap in the wall represents the door. The length, right width, and left width are labelled.

This door-centric method consists of utilizing traditional geometric properties about the door location. This includes: (door x location, door y location), angle about the door, width to the left of the door, width to the right of the door, and length. This represents all the feature information about the room within 6 parameters.

Step #	Description	Visual Representation
1	Door x and door y positions placed.	
2	Right width and global angle used to calculate the desired c1.	 2. assign c1 position based on right width and global angle right width global ang (x1,y1) (dx, dy)
3	90 degrees incremented onto the global angle to create the angle12. The line created from angle12 and c1 drives the location for c2.	 3. calculate next angle & find line equation g_ang + 90 degrees right width global ang (x1,y1) (dx, dy)

4	Length used along generated line to place c2.	 4. assign c2 along calculated line using length (x2, y2) length (x1,y1) (dx, dy)
5	90 degrees incremented onto the angle12, creating angle23. Line equation calculated created by angle23 and c2.	 5. increment global angle again by 90 degrees & calculate line equation g_ang + 180 degrees (x2, y2) (x1,y1) (dx, dy)
6	c3 placed along the calculated line using width. Where width = right width + left width.	 6. select point width down that line to assign c3 (x2, y2) (x3, y3) (x1,y1) (dx, dy)
7	90 degrees incremented onto the angle23, creating angle34. Line equation calculated created by angle34 and c3.	 7. Increment angle by another 90 degrees & calculate line equation (x2, y2) global ang + 270 degrees (x3, y3) (x1,y1) (dx, dy)

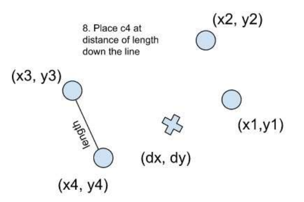
8	Length used along generated line to place c4.	 8. Place c4 at distance of length down the line (x2, y2) (x3, y3) (x1,y1) length (dx, dy) (x4, y4)

Table 6: Door-Centric Method Generation Process

Table 6 displays a step-by-step process for generating rooms with the door-centric method of room representation. This details the step number, a description of the actions taken, and a figure representation of that step.

This representation generated random values or used seed values for each attribute: door x location, door y location, angle about the door (global angle), width to the left of the door, width to the right of the door, and length. First, (door x position, door y position) was set. The rest of the room was constructed about this point. The corners are assigned by iteratively adding 90 degrees to the angle created from the two preceding corners, calculating the line equation at that angle and previous corner location, and then selecting a point along the line provided length, right width, or left width seeds to assign the corner location, respectively. This representation was chosen to represent the rooms within this project.

3.3. Scoring Functions

In determining the scores for the particles, there are two parts to this problem. Primarily, we must determine the various contributions to our score. In this case, the contributions could be comparing the lengths of the measurements vs the corner locations, global angles, etc. of the particles. We must determine which of these room attributes could be related to a closer match room. After these have been isolated, we must determine the best way to take these values and combine them into a final score. This will be the final scoring function.

3.3.1. Determining Attributes

To determine the contributions to a score, we must consider what we are trying to optimize. We are trying to optimize the attributes of whichever room representation technique we use. We must then determine some method to isolate the success of each of these attributed based off readings received. In this collection of results, the door-centric method of room representation was used. The methods used for scoring will reflect this from now on.

Since the first reading received is the door feature reading. We have noisy values of door x location, door y location, and global angle. The door-centric room representation represents a room by using door x location, door y location, global angle, left width, right width, and length. This means that just from the first reading, we have information about three of our six room representing attributes. We can now score particles based off these three attributes. To create some value that where a higher value is associated with a better fit room, we can take the difference between the reading value and the particle value and invert it. This way, the closer the particle value is to the true value, the higher score we receive for that attribute. These would then somehow be combined to create an overall score for that particle.

An additional assessment can be made with one door reading that can score particles. Since we know a point location along the wall of entry and the angle that wall makes, we can calculate the line equation for the wall of entry. This helps deduce potential locations for the hard corners of the room, which we can score based on distance from this line. We will further refer to this as the corner distance scoring method. A corner distance function was implemented. This function calculated the line equation of the wall of entry using the door location and global angle. It then found the orthogonal line equations to this line for each particle corner location. It calculated the intersection between each orthogonal line and the line of the wall of entry. It then determined the distance between each particle corner location and the wall of entry by calculating the distance between the particle corner and the point of intersection. The closest two particle corners to the wall of entry were isolated. These two particle corners were used to drive additional scoring by inverting the distance from the wall of entry. This is intended to further optimize the global angle and improve accuracy of particle hard corner locations.

When two readings have been received, we will have a door reading and a corner reading. Depending on the location of the corner about the room, it can give us a variety of information. If it is the leftmost hard corner, we can now calculate the estimated left width. If it is the leftmost easy corner, we can now calculate both the estimated left width and estimated height. If it is the rightmost hard corner, we can now calculate the estimated right width. And if it is the rightmost easy corner, we can now calculate both the estimated right width and estimated height.

To apply this scoring correctly, we will have to determine if the reading is to the left or right of the door. To do this, a right or left function was written. This function took the orthogonal line to the wall of entry using the door location and used that as the cutoff of left or right of door. It then takes the orthogonal line angle. If the angle is between (270, 0] or [0, 90), then if the point is above that line, it is to the right of the door. If it is below that line, it is to the left of the door. If the angle is between (90, 270), then if the point is above that line, it is to the left of the door. If it is below that line, it is to the right of the door. If the angle is 90 degrees, then if the x value of the point is greater than that line, it is to the right. If it is smaller, then it is to the left. If the angle is 270 degrees, then if the x value of the point is greater than that line, it is to the left. If it is smaller, then it is to the right.

Using the information from the right or left function, we determine which side of the door the corner reading is located. If it is to right, it will affect the right width. If it is to the left, it will affect the left width. We ignore length for now, as we do not yet have comparative information to distinguish a hard corner from an easy corner. To find the estimated value for either right width or left width, we take the wall of entry line equation again and find the orthogonal line from the corner reading location. We then calculate the intersection between the orthogonal line and the wall of entry line. We then take the distance between the point of intersection and the door location. If this corner reading was to the left of the door, this distance is our estimated left width. If this corner reading was to the right of the door, this distance is our estimated right width. We can then take the inverse of the absolute value of the difference between the estimated value and the particle value to drive scoring that attribute.

When three readings have been received, we will now have some scoring for the door x position, door y position, global angle, and either left width or right width. If the second corner reading was on the opposite side of the door as the previous corner, we applied the same implementation again but contributing to the opposing width score. If the second corner reading is located on the same side of the door as the previous reading, we can compare these two to distinguish hard corners from easy corners and, from that, determine an estimate for the length of the room. Unfortunately, the length approximation was not completed due to time constraints. But if it were to have been implemented, it would likely have been as detailed here.

If the corner location is on the same side of the door as the prior, we can check how far from the wall of entry each of the corner positions are. Whichever corner is the farthest from the wall of entry is the hard corner. The distance between this corner and the wall of entry is the estimated length. We would then take the inverse of absolute value of the difference between the estimated length and the particle length attribute and use that to drive scoring that attribute.

Each of these would account for each of the attributes used to represent a room within this system.

3.3.2. Combining Attributes

Once ways to score each of these attributes have been isolated, we must combine them in some useful way to benefit the simultaneous optimization of each of these attributes using a singular score value. Unfortunately, due to time constraints, various methods of combining these values were not fully implemented or evaluated. This will be left for future work.

3.4. Elimination of Particles

Once a scoring function has been applied to each particle in the generated set, there must exist some method of eliminating particles based on their scores. Three different approaches implemented and evaluated in this project are described below.

3.4.1. Deterministic Approach

The deterministic approach for particle elimination eliminates particles based on the percentile their score aligns with among the distribution of particle scores. This approach assures the survival of the best-scoring particles of that iteration. This was done by taking in some value representative of the cutoff percentile. This percentile was mapped to the distribution of scores. All particles with a score underneath this percentile were then eliminated from the set. The mean and standard deviation of the surviving particles' attributes were used to drive the backfilling for the next set of particles. These seed parameters were imputed into the particle generator for resampling.

3.4.2. Probabilistic Draw Approach

The probabilistic draw approach for particle resampling drew some determined number of particles with replacement with a probability relative to their score across the distribution of scores. To do this, it was programmed to iterate through the distribution of scored particles for some determined number of times. Each iteration, it would calculate the sum of all scores, and uniformly randomly select some number within the range of 0 and that sum of scores. It would then iterate through summing the scores, once that sum exceeded the selected number, that particle was decided to survive the process. These surviving particles would then drive the resampling of an entirely new distribution based on the mean and standard deviation of the surviving attributes.

3.4.3. Probabilistic Individual Assessment Approach

The probabilistic individual assessment approach assessed each particle separately and determined whether to accept or reject each particle based on their respective scores. This was accomplished by iterating through the particles. Each individual particle was determined to be accepted or rejected based on the cumulative distribution function of its score across the distribution of scores. The surviving particles then drove the backfilling of particles by using their mean and standard deviation as a seed for generating new particles for the next set.

Commented [HG2]: Define this term

3.5. Test Cases

Test cases were generated manually with the following edge-case features in mind: room scale, global angle, room proportions, and door location. Twelve test cases were developed across these properties. These test cases are represented using the door-centric method of room representation.

#	Representation	door x	door y	global angle	left width	right width	length
1	Square	0	1	270	4	1	5
2	l < w	0	1	270	9	1	4

3	l > w	0	1	270	1	3	6
4	l << w	0	1	270	1	9	0.2
5	l >> w	0	0.1	270	0.1	0.1	10
6	Small	0	0.05	270	0.195	0.005	10
7	Large	0	50	270	1450	50	650
8	Corner	5	0	0	0	5	15
9	Ang = 90	5	4	90	4	11	5
10	Ang = 180	3	15	180	12	3	5
11	Ang = 0	3	0	0	3	2	15
12	Ang ≠ 0, 90, 180, 270	1	7.5	262	7.566	7.566	5.3852

Table 7: Test Cases Represented

Table 7 displays each test case used to collect results during the project. Scale options included: small, medium, and large. Small contained positional locations on the order of magnitude 10^-2. Medium contained positional locations on the order of 10^1, and large contained positional locations on the order of 10^3. Global angle options included: 0 degrees, 90 degrees, 180 degrees, 270 degrees, and ≠ 0, 90, 180, 270 degrees. Room dimension options were categorized as the ratio between length and width. This included length to width ratios of 1:1 (square room), 5:2 (length > width), 50:1 (length >> width), 2:5 (length < width), 1:50 (length << width). Door location options included on a corner or on a wall.

3.6. Measures for Success

To measure the accuracy of the room state estimate, we needed to find some way to compare the estimated rooms to the true state of the room. We wish to do this in some normalized way so that the room accuracy is measured representatively across various scoring methods used and test case rooms.

To represent accuracy, we decided to use the absolute difference between the values of each attribute of the estimated room vs the true state of the room. This included the absolute differences between the door x values, door y values, global angles, left widths, right widths, and lengths. This would represent the accuracy of the rooms over a number of measurements received, where a smaller value indicates better accuracy.

4. **Results**

The following results were obtained by running the particle filter through each of the test cases. For each test case, the particle filter was run for multiple phases of receiving information. First, it was run on the first reading received. The output set of particles from that run was used to run the particle filter on the first and second reading. The output set of particles from that run would then be used to run the particle filter on the first, second, and third reading. Finally, that output set of particles was used to run the particle filter on the first, second, third, and fourth reading. This may be implemented differently in future work to better represent reality, though this simplified method was used to test these scoring functions.

These results were obtained by applying a noise standard deviation of 0.3 units. If units are interpreted as meters, then this noise would be equivalent to an average of 30cm inaccuracy for all readings. Noise is reapplied to reading values for each run of the particle filter.

These results were obtained running the deterministic elimination of particles approach due to time-limitation related technical issues with the probabilistic independent assessment approach and concerns with the probabilistic draw approach. The top 20% of scores were kept using this approach for each iteration.

These results were obtained using the door-centric representation of rooms. Reasons for this are detailed in Section 7.1.

In all cases below, returned values represent the absolute value of the difference between the average of the returned set from the particle filter and the anticipated return values for the true state of the test case room. These differences are represented independently for each room attribute.

To assess whether attributes were being isolated correctly, the particle filter was run on all the test cases for each attribute only using the portion of the score meant to drive that attribute. For example, the particle filter was run using only the portion of the score meant to optimize the door x location, then run again only using the portion of the score meant to optimize door y location, etc. for each attribute of the door-centric room representation. This was done to ensure each attribute was being correctly isolated. Once each of these attributes are correctly isolated, the remaining scoring tasks will be to map these values to a score that represents each attribute effectively, driving the estimated state of the world to success.

For all figures and tables below: At 1, the first reading is available, a random particle generation occurs, and the particle filter runs. At 2, the particle filter is running its second iteration on two readings received. At 3, the particle filter is running its third iteration on three readings received. At 4, the particle filter is running its fourth iteration on four readings received.

#	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
1	2.015996	0.915861	0.473144	2.203446	1.303911	3.34067	2.15052	2.864146	0.137627	0.146768	3.843306	0.839612
2	0.35592	0.452078	0.215984	0.196601	0.090854	0.210256	1.58178	0.059641	0.142111	0.39566	1.455644	0.168679
3	0.149028	0.273529	0.22907	0.225364	0.065869	0.509488	1.012489	0.227918	0.182838	0.019298	0.021988	0.053737
4	0.099311	0.260984	0.170523	0.189574	0.140909	1.004583	0.218164	0.076664	0.086788	0.221342	0.036912	0.047535

Table 8: Door X Difference Measurement for Each Test Case Over Readings Received

Table 8 displays the difference between the expected true value of door x position and the average door x position value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of door x location and door y location. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

#	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
1	2.083701	2.550944	1.502129	1.669182	2.952002	1.765715	30.44636	1.80636	1.903219	0.148828	0.42362	0.894927
2	0.051146	0.53546	0.576092	0.809995	0.280593	1.073744	27.02711	0.587505	0.043092	0.231845	0.043882	0.522697
3	0.427289	0.078201	0.221682	0.028109	0.063788	0.151479	24.05212	0.282209	0.279999	0.000918	0.050868	0.154835
4	0.729258	0.142355	0.046723	0.259012	0.23476	0.008404	22.55341	0.008713	0.017614	0.169307	0.090349	0.053988

Table 9: Door Y Difference Measurement for Each Test Case Over Readings Received

Table 9 displays the difference between the expected true value of door y position and the average door y position value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of door x location and door y location. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

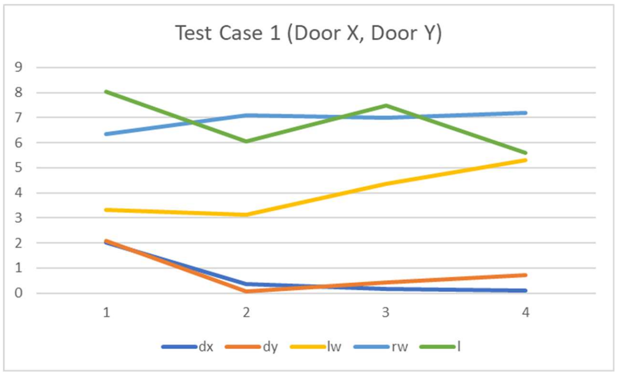


Figure 15: Test Case 1 Example of Attributes Over Readings Received Optimizing (Door X, Door Y)

Figure 15 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for Test Case 1. This run was scored on solely optimization of door x location and door y location. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

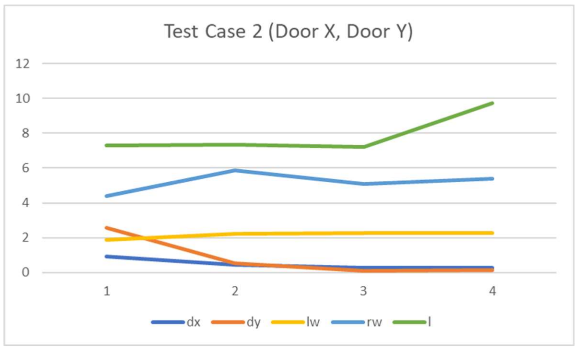


Figure 16: Test Case 2 Example of Attributes Over Readings Received Optimizing (Door X, Door Y)

Figure 16 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for

Test Case 2. This run was scored on solely optimization of door x location and door y location. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

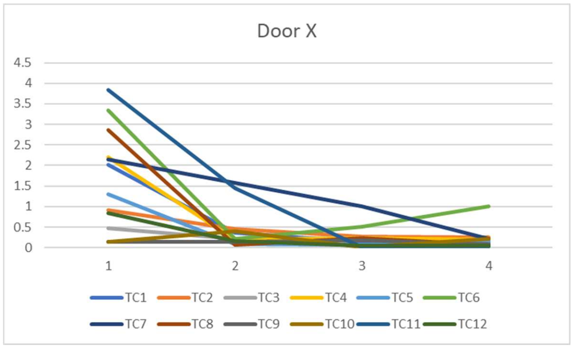


Figure 17: All Test Cases Door X Difference Over Readings Received

Figure 17 is a plotted representation of the data in Table 8. It displays the difference between the expected true value of door x position and the average door x position value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of door x location and door y location. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

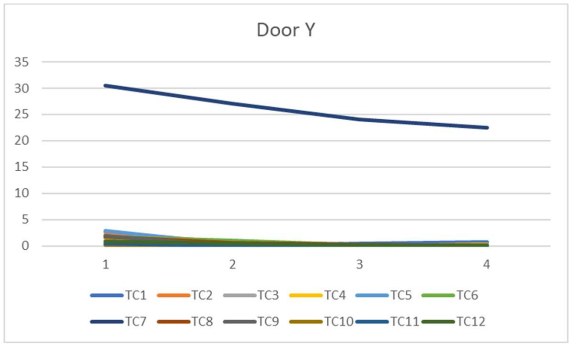


Figure 18: All Test Cases Door Y Difference Over Readings Received

Figure 18 is a plotted representation of the data in Table 9. It displays the difference between the expected true value of door y position and the average door y position value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of door x location and door y location. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

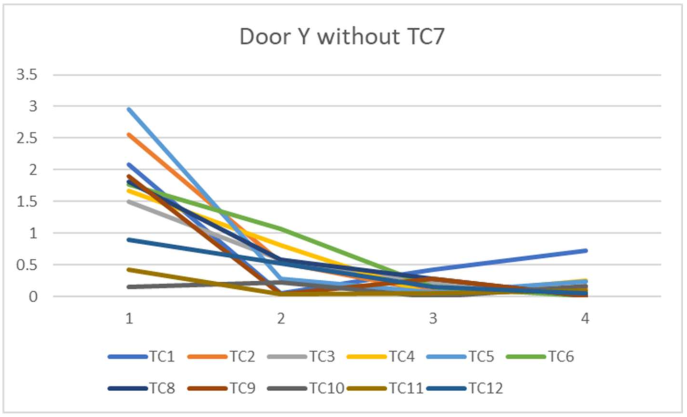


Figure 19: All Test Cases Except TC7 Door Y Difference Over Readings Received

Figure 19 displays the same data as described in Figure 18 without plotting TC7. This is displayed for readability's sake.

#	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
1	4.630983	3.432666	6.64148	5.526137	1.885769	4.496863	2.591595	36.33748	0.567727	4.642722	41.54407	2.792172
2	0.589635	1.460194	0.35399	0.05607	0.997649	0.259954	0.952559	12.37286	0.870232	0.537699	7.567705	0.63421
3	0.441	0.256	0.335	0.221	0.222	0.188	0.167	3.625	0.016	0.277	0.650	0.524

	739	339	224	49	676	644	424	74	203	542	131	086
4	0.334827	0.142357	0.073837	0.099319	0.082741	0.104155	0.271118	0.171858	0.026998	0.343683	287.9052	0.016161

Table 10: Global Angle Differences for Each Test Case Over Readings Received

Table 10 displays the difference between the expected true value of global angle and the average global angle value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the global angle. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

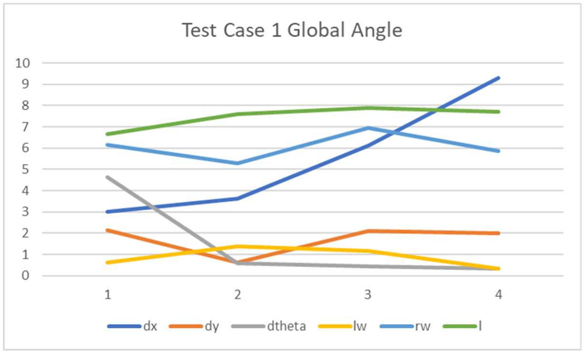


Figure 20: Test Case 1 Example of Attributes Over Readings Received Optimizing Global Angle

Figure 20 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for Test Case 1. This run was scored on solely optimization of the global angle. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

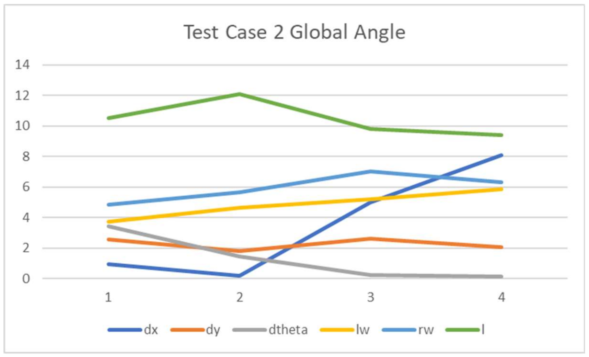


Figure 21: Test Case 2 Example of Attributes Over Readings Received Optimizing Global Angle

Figure 21 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for Test Case 2. This run was scored on solely optimization of the global angle. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

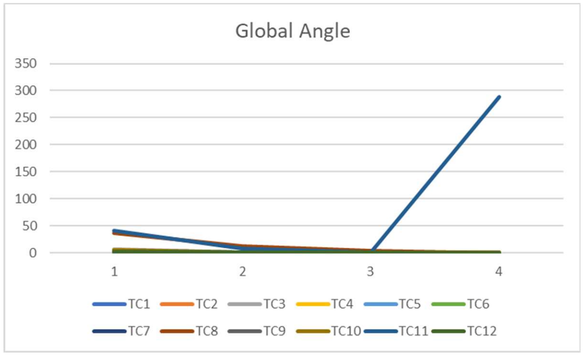


Figure 22: All Test Cases Global Angle Difference Over Readings Received

Figure 22 is a plotted representation of the data in Table 10. It displays the difference between the expected true value of the global angle and the average global angle received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the global angle. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

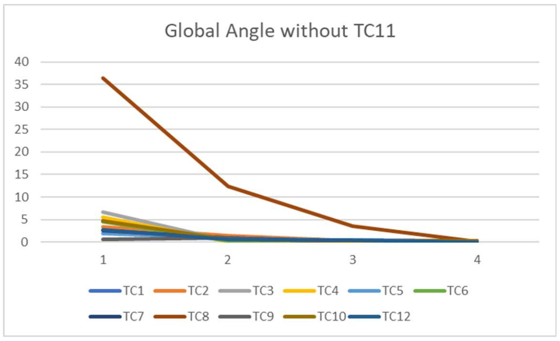


Figure 23: All Test Cases Except TC11 Global Angle Difference Over Readings Received

Figure 23 displays the same data as described in Figure 22 without plotting TC7. This is displayed for readability's sake.

#	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
1	111.7508	134.0195	132.8671	121.0911	107.837	72.99263	101.6572	183.3256	77.49792	11.58641	188.4441	72.49167
2	59.86491	126.1881	126.6359	138.306	95.67358	69.57913	83.50013	176.6059	71.96895	7.774883	191.6793	63.0451
3	64.72486	121.6668	110.9554	156.5255	111.1369	61.05881	63.20962	200.7385	65.63679	13.79889	200.1803	51.98099
4	66.35401	91.05703	78.68125	169.2598	105.7996	66.06629	59.31763	213.0205	77.84613	12.7987	231.4782	75.17156

Table 11: Global Angle Differences for Corner Distances Approach for Each Test Case Over Readings Received

Table 11 displays the difference between the expected true value of the global angle and the average global angle value received from the set of particles returned from the particle filter for each successive run on the particle filter. This was run focused on optimizing the global angle via the corner distances approach. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

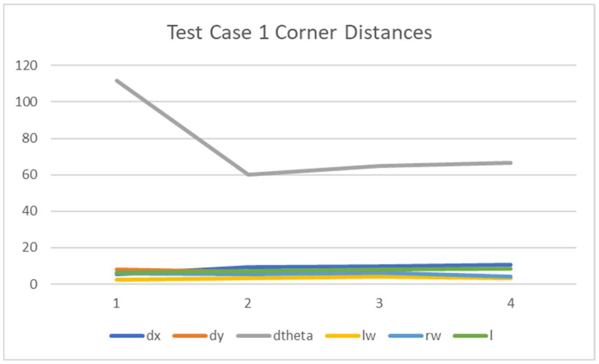


Figure 24: Test Case 1 Example of Attributes Over Readings Received Optimizing Global Angle Through Corner Distances Approach

Figure 24 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for Test Case 1. This run was scored on solely optimization of the global angle via the corner distances approach. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

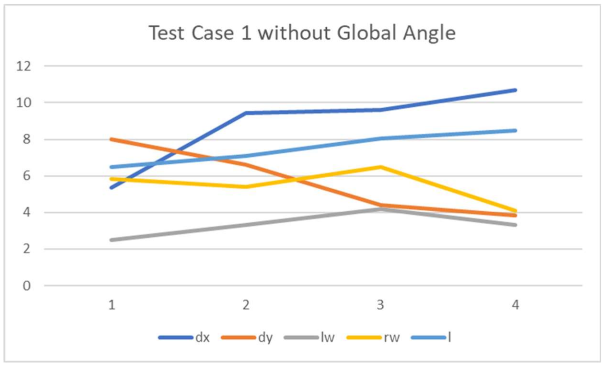


Figure 25: Test Case 1 Example of Attributes Except Global Angle Over Readings Received

Optimizing Global Angle Through Corner Distances Approach

Figure 25 displays the same data as described in Figure 24 without plotting the global angle. This is displayed for readability's sake.

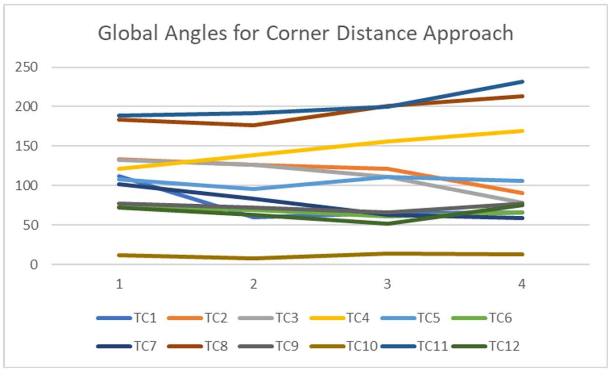


Figure 26: All Test Cases Global Angle Difference Over Readings Received Via Corner

Distance Approach

Figure 26 is a plotted representation of the data in Table 11. It displays the difference between the expected true value of the global angle and the average global angle received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the global angle via the corner distances approach. Each test case is represented by TC1 for the first test case, TC2 for the second test *case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.*

#	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
1	6.890353	5.851874	1.649742	1.852036	5.563687	4.473679	43.71646	1.969719	3.862531	2.65774	4.887342	0.910059
2	1.062284	6.44068	1.486313	7.73059	0.612503	3.958859	43.86336	2.633875	6.842879	1.218674	2.303175	0.507485
3	0.125392	8.183036	1.721222	7.854504	0.347145	0.495719	40.94059	0.177934	7.779531	0.707496	1.869274	0.070283
4	0.380685	8.454564	1.846151	8.12174	0.239911	0.230537	40.99535	0.046687	7.462921	1.914026	1.713987	0.361717

Table 12: Right Width Differences for Each Test Case Over Readings Received

Table 12 displays the difference between the expected true value of the right width and the average right width value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the right width. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

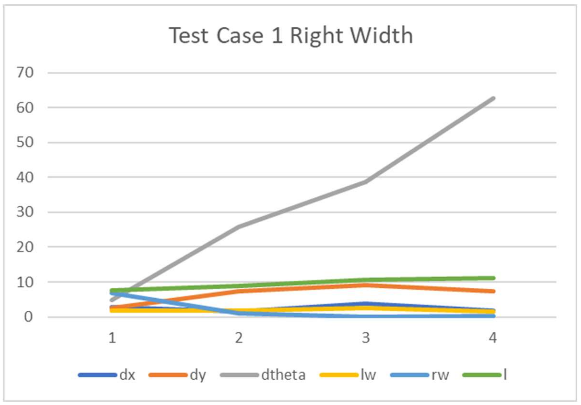


Figure 27: Test Case 1 Example of Attributes Over Readings Received Optimizing Right Width

Figure 27 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for Test Case 1. This run was scored on solely optimization of the right width. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

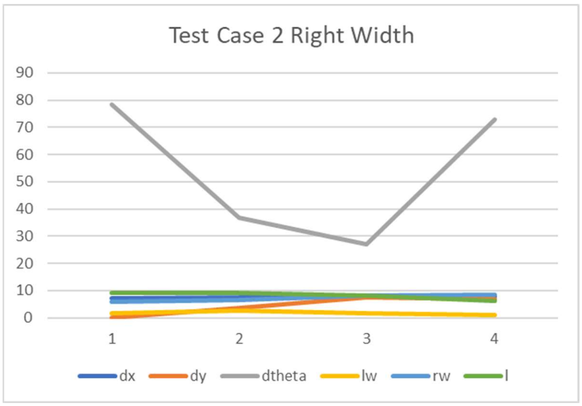


Figure 28: Test Case 2 Example of Attributes Over Readings Received Optimizing Right Width

Figure 27 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for

Test Case 2. This run was scored on solely optimization of the right width. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

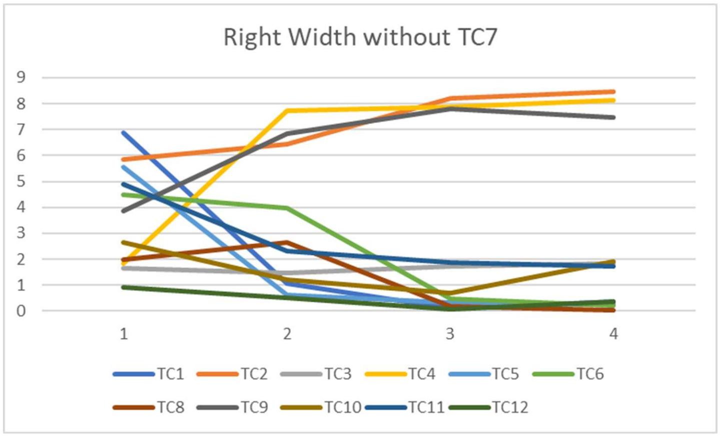


Figure 29: All Test Cases Right Width Difference Over Readings Received without TC7

Figure 29 is a plotted representation of the data in Table 12. It displays the difference between the expected true value of the right width and the average right width received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the right width. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

#	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10	TC11	TC12
1	4.084582	2.339732	7.429135	4.243045	5.912401	6.797942	1443.638	6.176986	1.95611	6.817146	2.455656	1.636053
2	2.121942	0.936957	7.546445	3.569828	5.256354	7.162841	1445.09	4.970234	2.126044	8.671791	2.942452	0.339338
3	2.677744	0.454717	2.097449	4.92963	0.531577	0.986213	1445.109	4.961304	6.671631	8.741652	2.068103	0.245666

4	2.607217	0.523057	1.430823	4.162245	0.538077	0.650592	1444.917	4.163706	6.931738	8.824859	0.244162	0.155648

Table 13: Left Width Differences for Each Test Cases Over Readings Received

Table 13 displays the difference between the expected true value of the left width and the average left width value received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the left width. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

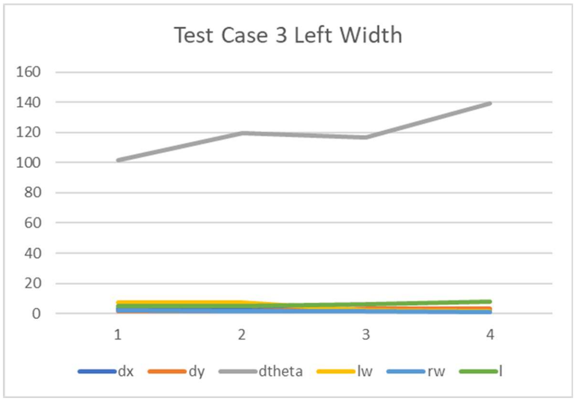


Figure 30: Test Case 3 Example of Attributes Over Readings Received Optimizing Left Width

Figure 30 displays the difference between the expected true value of each attribute and the average attribute value received from the set of particles returned from the particle filter for each successive run on the particle filter for Test Case 3. This run was scored on solely optimization of the left width. The lower the value in the plot, the closer the returned set attribute average is to the expected return value of that attribute.

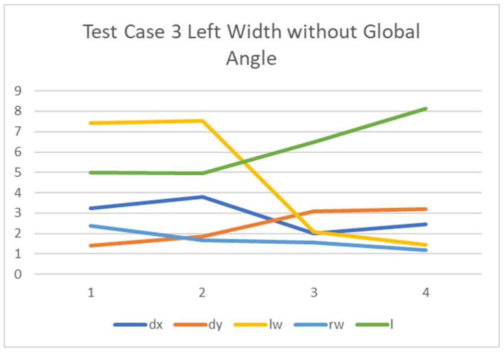


Figure 31: Test Case 3 Example of Attributes Excluding Global Angle Over Readings Received Optimizing Left Width

Figure 31 displays the same data as described in Figure 30 without plotting the global angle. This is displayed for readability's sake.

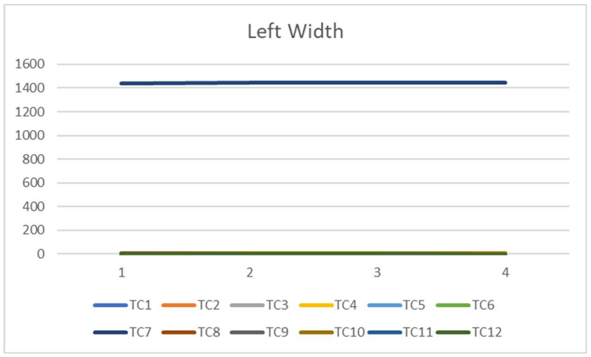


Figure 32: All Test Cases Left Width Difference Over Readings Received

Figure 32 is a plotted representation of the data in Table 13. It displays the difference between the expected true value of the left width and the average left width received from the set of particles returned from the particle filter for each successive run on the particle filter. This run was scored on solely optimization of the left width. Each test case is represented by TC1 for the first test case, TC2 for the second test case, etc. The lower the value in the table, the closer the returned set attribute average is to the expected return value of that attribute.

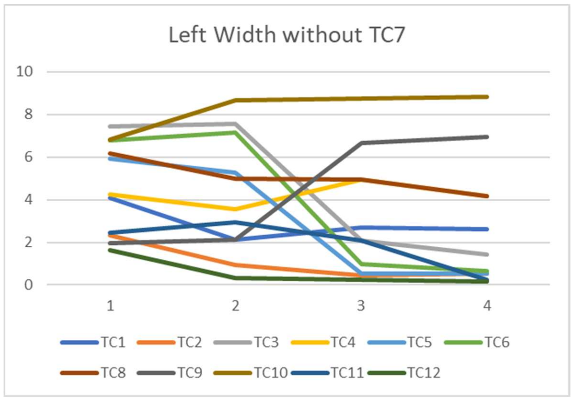


Figure 33: All Test Cases Except TC7 Left Width Difference Over Readings Received

Figure 33 displays the same data as described in Figure 32 without plotting TC7. This is displayed for readability's sake.

5. **Discussion**

5.1. Room Representation & Generation

Across the three primary methods for room representation and generation, each had their own advantages and disadvantages. The global coordinate system representation required no conversion from sensor readings, allowing these attributes to be compared directly to the sensor readings. However, geometric room properties required additional derivation to derive to apply various geometric scoring methods. Additionally, this representation makes a concerning seed for particle generation. If resampling these attributes with some mean and standard deviation of global coordinate values, it does not actively enforce traditional geometric properties of the room. This is concerning that the values produced over time could converge to a non-physically possible or non-box-shaped room. Additionally, this method of room representation requires 10 parameters to represent all the room data, whereas the other methods of room representation only require 6 or 7 parameters.

The geometric method utilizes traditional methods of geometrically describing rectangles. Due to this, there are limited concerns in representing attributes within the resampling step. Unfortunately, this method has one major downfall - its non-intuitive representations of the door parameter. Expressing the door in forms of door x location and door y location coordinates disassociates the door parameter from the geometric properties of the room, not actively describing the door relative to the room geometry. The angle-based door representation does represent the door relative to the room geometry, though it involves a complicated implementation and a non-intuitive representation of the door. This made testing for room convergence very inconvenient, as it was difficult to reasonably tell when the door was being estimated in its intended location. Additionally, there were concerns of decreasing accuracy for larger sizes of rooms when representing via angle. Though the distance door representation was not ever implemented, it was predicted to solve the issue of loss of accuracy over time. Besides that, it was still predicted to have every other complication as the angle method. These concerns spawned the idea for the door-centric room representation method.

Since the door is the location our robot enters through, not only are we consistently receiving that reading first and most confidently, but the robot also constructs its understanding of the room relative to the door. Using the door as our anchoring point in space allows us to have more confidence in our anchor point as well as constructing the room in a method relevant to tactical room entry. By designating the width to the right of the door and the width to the right of the door, it creates a convenient implementation of geometric properties relevant to tactical language, particularly in describing both the feed and weight of the room.

5.2. Elimination of Particles

The three approaches considered for the elimination of particles were the deterministic approach, the probabilistic draw approach, and the probabilistic individual assessment approach.

Though the deterministic approach was selected to collect results on the test cases, we had concerns with this implementation. By eliminating scores based on percentile, we are guaranteeing the elimination of the lowest scoring particles. When primarily determining scores based on only half of the total attributes of the room when using the first reading, we had concerns that guaranteeing the elimination of all lowest scoring particles could cause improper convergence while potentially eliminating solutions accurate to the unobservable attributes from the set of potential states.

The probabilistic draw approach is the only approach that does not do traditional elimination and backfilling of particles. Instead, it draws some number of particles with replacement from the distribution according to score and creates an entirely new distribution based on the attributes of the selected particles. In this sense, there are no particles that we keep from the previous iteration. This is a concern if there happens to be more than one promising representation of a room, a multi-modal distribution of particles drawn. Creating an entirely new set of particles based on the mean of the attribute values could remove the variability in the set completely. This was the primary concern with the probabilistic draw approach and why it was not chosen to be pursued.

The probabilistic assessment approach was the approach remaining with no immediate concerns. Unfortunately, due to time-constraints and technical complications, this approach was unable to be implemented in gathering results.

5.3. Scoring

In Section 6, the displayed results were collected with the intention of verifying the ability to converge each attribute to the intended true room value across various test cases. We can assess the success of the scoring of each attribute from this returned data.

5.3.1. Door X Position and Door Y Position

The score value for the door x position and the door y position were calculated together. As shown in Table 8, which displays door x optimization across test cases, eleven of the twelve test cases returned a value for the fourth reading that was lower than the first reading. In Table 9, which displays door y optimization across test cases, eleven of the twelve test cases returned a value for the fourth reading that was lower than the first reading. These results are visualized in Figures 17, 18, and 19. Overall, eleven of the twelve test cases returning a lower value is a positive result. This indicates that door x position and door y position are being accessed effectively. Where twenty-one of the twenty-four results across door x position and door y position returned a difference value of <0.27 by the fourth reading received.

To shed light on potential issues, in both cases, it was Test Case 10 which returned fourth reading results higher than the first reading results. However, it is important to note that the first reading results for Test Case 10 were also the most accurate first reading results across all test cases. Knowing this information, it may be of interest to continue investigating this test case to see if this is a trend across many runs. If we were to acknowledge differences in this test case compared to the others, Test Case 10 is the global angle is 180 degrees edge-case test case. The true values for (door x position, door y position) were (3, 15). These coordinate values are not unusual values across successful test cases.

In the door y position results, though Test Case 7 door y values decreased over readings received, it remained an outlier compared to the other test cases. The Test Case 7 edge case is the large-scale room. The true values for the (door x position, door y position) were (0, 50). After viewing these results of decreasing difference across measurements received while maintaining large inaccuracy to the true value for door position y, it was discovered that the particle generation method does not generate particles with door y position values greater than 25 in the initial set of particles. This ensures that there will be no particles representative of this door y position in the initial set. The particle generation function needs to be kept under surveillance to ensure all potentially needed particles are accounted for.

5.3.2. Global Angle

Table 10 displays the global angle optimization across test cases. In eleven of the twelve test cases, the value returned from the fourth readings was lower than the first reading. These results are visualized in Figure 22 and Figure 23. Overall, eleven of the twelve test cases returning a lower value is a positive result. This indicates that the global angle is being accessed effectively. Where eleven of the twelve test cases returned a difference value of <0.35 by the fourth reading received.

To shed light on potential issues, it was Test Case 11 which returned fourth reading results higher than the first reading results. It may be of interest to continue investigating this test case to see if this is a trend across many runs. If we were to acknowledge differences in this test case compared to the others, Test Case 11 is the global angle is 0 degrees test case. Test Case 8 is the only other test case at global angle 0 degrees. When viewing the successful Test Case 8 in Figure 23, there is a large amount of inaccuracy in the global angle at the first three readings compared to the other test cases. These results indicate that test cases where the true global angle value is close to zero are worthy of further evaluation.

5.3.3. Global Angle Corner Distances Approach

Table 11 displays the global angle optimization using the corner distances approach. In six of the twelve test cases, the value returned from the fourth readings was lower than the first reading. These results are visualized in Figure 26. Overall, six of the twelve test cases returning a lower value is a negative result. All but one test case return difference values of >59 after the fourth reading. This indicates that the implementation of this method of scoring was done incorrectly and is ineffective toward optimizing the global angle.

5.3.4. Right Width

Table 12 displays the right width optimization. In seven of the twelve test cases, the value returned from the fourth readings was lower than the first reading. These results are visualized in Figure 29. Overall, seven of the twelve test cases returning a lower value is a negative result. This indicates that the implementation of this method of scoring was done incorrectly and is ineffective toward optimizing the right width. However, the test cases that did converge, converged very accurately with a difference value of <0.4.

To shed light on potential issues, observe that Test Cases 2, 4, and 9 returned unusually high fourth reading results. These test case edge cases were l<w, l<<w, and angle = 90. The ideal right width values were 1, 9, and 11. These were not unusual values across all the test cases. It may be of interest to continue investigating this test case to see if this is a trend across many runs.

Additionally, Test Case 7 was highly inaccurate throughout its full run. This can be attributed to similar reasons as noted in Section 7.3.1, as the initial particle generation did not allow for the generation of particles with a right width of 50.

5.3.5. Left Width

Table 13 displays the left width optimization. In nine of the twelve test cases, the value returned from the fourth readings was lower than the first reading. These results are visualized in Figure 32 and Figure 33.

To shed light on potential issues, observe that Test Cases 9 and 10 returned unusually high fourth reading results. These test case edge cases were angle = 90 and angle = 180. The ideal left width values were 4 and 12. These were not unusual values across all the test cases. It may be of interest to continue investigating this test case to see if this is a trend across many runs.

Additionally, Test Case 7 was highly inaccurate throughout its full run. This can be attributed to similar reasons as noted in Section 7.3.1, as the initial particle generation did not allow for the generation of particles with a right width of 1450.

6. **Conclusion**

For each point of discussion – room representation and generation, elimination of particles, and scoring methods - various conclusions can be made.

The room representation and generation approach showing the most promise is the door-centric room representation. This is most relevant to tactical language descriptions and representative of room entry perspective. Additionally, it represents a room using the lowest number of parameters across all representation methods explored and has an intuitive generation process. This is compared to other methods like the geometric approach, which has non-intuitive door representation, and the coordinate approach, which has an inconvenient seed for room generation and needs a larger number of parameters to represent a room.

The particle elimination approach showing the most promise is the probabilistic individual assessment approach. However, the deterministic approach shows promise as well. The probabilistic individual assessment approach is selected to pursue further due to concerns with the alternate approaches. For the probabilistic draw approach, we fear complications with multimodal distribution of particles. For the deterministic approach, we fear ensuring the elimination of lowest scoring particles could lead to converging to local score maxima. A local score maximum would be a particle (or set of particles) that score high relative to particles with similar attributes, without being the highest scoring particle across the full set of potential states (i.e., this is not the correct solution).

Across methods of scoring for (door x location, door y location), global angle, left width, and right width, we can draw the following conclusions. The (door x location, door y location) and global angle parameters are being isolated correctly. However, the global angle optimization via alternative corner distances approach is ineffective. Since there is already an effective way of optimizing the global angle, this method is not necessary. The left width and right width parameters are being isolated incorrectly. This requires additional investigation to determine and rectify the source of the problem, as failing test cases do not have any immediately observable similarities.

Though some scoring methods for attributes were effective, there were specific test cases that require additional observance. Test Case 7 was the test case assessing a large-scale room. Unfortunately, our current method of generating an initial set of particles does not accommodate for this. This should be addressed in future work. For optimizing global angle, the current method shows outliers at global angle = 0 test cases differently than other true global angle value test cases. This needs to be investigated further to determine the cause of the outliers.

By the end of the project period, there exists a determined room representation technique, a room generation function for both initialization and resampling functionalities, some particle elimination implementations, isolated scoring functions for individual attributes, as well as a function to export run data to a csv file.

There were many improvements that could have been made to the execution of work within the project period. Primarily, it would have been beneficial to approach the representation of rooms from the relevant tactical language descriptors from the beginning of the project. This would have saved a lot of time dedicated to less-relevant traditional room representation approaches. Overall, however, the project would have greatly benefitted from introducing myself to the existing project code earlier on. In this project, there was a lot of research conducted in the first three to four weeks. Since many aspects of the project are relatively novel, a lot of this research ended up unnecessary in final implementations. Though collecting redundant information frequently becomes unavoidable in the research project process, as we do not always know what to avoid from the start, the time spent on researching redundant information could have been greatly decreased by earlier introduction to the existing codebase. This would have given a more solid understanding of the state of the project earlier and avoided a lot of fruitless efforts.

In this project, we were able to make substantial functioning additions to the codebase for feature prediction via particle filtering as well as make observations that will drive future work of the project.

7. **Future Work**

7.1. Likelihood of Hostile Agents or Civilians

As any agent, robotic or human, executes a tactical room search, there are many considerations that could be relevant to the search. Particularly, we may have interest in who may be located within the space and where. Depending on the context of the room search, there could be hostile agents or civilians within the space. It may be of further interest to the project to include some probability of hostile agents or civilians in specified rooms or areas. This could add a layer of usefulness to the robot as a member of a human team. This prediction could be made based off some provided context of the search. Or we could include additional understanding of room features and of geometric blind spots from the primary line of vision. These areas could have higher potential for hidden threats and can be noted as such from the system.

7.2. Recognition of Additional Relevant Features

Currently, the only features the system assumes to identify are corners and doors. The observations of these features are purely positional. If additional features could be recognized and additional observations of features could be made, there would be added usefulness to the system. Some potential additions are listed below.

7.2.1. Materials

Materials of doors or additional features of the room could be of large importance to a room entry team. Particularly, some materials can be penetrated while others are not. In a tactical room search situation, there will be potential for gunfire. If our robot could identify a likelihood of a material being bulletproof or flammable, this could impact how the team executes the search. This is something that could be considered in further developments of the project.

7.2.2. Other Room Obstacles

When entering a typical room, it is uncommon that the room is completely empty. It is possible that there may be obstacles in that room. This could impact the team's method of entry. If an obstacle is blocking their intended path, that search path must be adjusted. If the robot could observe and identify these obstacles within a room, it could be a more valuable member to human teams.

In addition to identifying these objects and their location within space, if bulletproof materials could also be identified, the robot could further assess areas of cover for its team. Incorporating identification of other room obstacles and materials can make the robot more useful for its team and can be considered as an addition to the project in future work.

7.2.3. Windows or Non-Door Methods of Exit/Entry

Currently, the only feature of the room that the robot identifies as a method of exit or entry is a door. There may be other methods of entering or exiting a space. These could be windows or small entryways. When in a tactical room search setting, it would be useful for the robot to identify such points of entry and their size.

7.3. Generation of Additional Room Types

In the current state of the project, we assume that the room is a *box-shaped* room. Though this is a common shape of room, there are many other shapes rooms can take. As represented in tactical language, there are also *L-shaped rooms*, *U-shaped rooms*, *T-shaped rooms*, and more. Future additions to the project will include room generator functions that can accommodate each additional room type, including options for irregular rooms as described by tactical language.

7.4. Fusion of Tactical Language Interpreter

Returning to the primary goal of the TFLIP project, the goal was to develop a robotic agent that can understand structured tactical language and utilize it for a tangible purpose. At the current stage of the project, there is no portion that understands structured tactical language. Instead, we have been assuming tactical language input. Separately, a TERRAA team member created a language parser. This parser takes in some free-text input and outputs a parsed and categorized representation of the text as it pertains to tactical language. The fusion of this tactical language parser with existing code as well as addition of associated room type generation functions would complete the tactical language interpretation portion of the project. This is necessary to meet the primary TFLIP objective and therefore an important part of future work.

7.5. Conversion of Room-Representing Data Types

The code currently utilizes the Network X Python graph package. This is used to represent a copy of the particles in the particle filter. Primarily, this graph package was used for its visualization tools. However, we could also represent this data through a dictionary. The transfer of particle information to dictionary key value pairs is an opportunity for future work.

7.6. Adjacent Room Representation

In a practical application, it is unlikely that we enter a building that consists solely of one room with single point entry. For this reason, it is important for us to be able to accommodate for connecting rooms and multiple doors. This could be using a graph to connect rooms and doors. The implementation of this will be left to future work.

7.7. Common Lines of Sight Represented in Order of Readings Received

As described in Section 5.1.2, common lines of sight to impact what readings we are most likely to see when entering a room. Taking this information into account could improve the accuracy of scoring by the robot as well as simulate a more accurate environment. A method to present reading information to the robot in a realistic way is left for future work. In addition, having this information assist in scoring particles is also left for future work.

7.8. Combining Scoring Attributes

As described in Section 5.3.2, The combining of scoring attributes into a single value representative score was unable to be completed during this project. Since there are various ways to combine these values, multiple methods must be implemented and evaluated in future work. This is essential for the particle filter to run successfully and to converge to an accurate estimate of the room.

References

1. Blair, J. P., Martaindale, M. H., & Sandel, W. L. (2019). Peek or Push: An Examination of Two Types of Room Clearing Tactics for Active Shooter Event Response. SAGE Open, 9(3). https://doi.org/10.1177/2158244019871052
2. Del Moral, P. (1996). "Non Linear Filtering: Interacting Particle Solution" *Markov Processes and Related Fields.* 2 (4): 555-580 https://people.bordeaux.inria.fr/pierre.delmoral/delmoral96nonlinear.pdf
3. Dirndorfer, T., Botsch, M., Knoll, A. (2011). Model-based analysis of sensor-noise in predictive passive safety algorithms. Proceedings of the 22nd International Technical Conference on the Enhanced Safety of Vehicles (ESV). https://mediatum.ub.tum.de/doc/1287148/1287148.pdf
4. Xiang, X. and Foo, S. (2021). Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (POMDP) problems: Part 1—Fundamentals and applications in games, robotics and natural language processing. (2021). *Machine Learning and Knowledge Extraction, 3*(3), 554. https://doi.org/10.3390/make3030029
5. Piirainen, Elisabeth, Filatkina, Natalia, Stumpf, Sören and Pfeiffer, Christian. *Formulaic Language and New Data: Theoretical and Methodological Implications*, Berlin, Boston: De Gruyter, 2020. https://doi.org/10.1515/9783110669824
6. McManamon, Paul. (2019). *LiDAR Technologies and Systems - 4.1.6.4 Data Processing and Dissemination.* SPIE. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt012EELE2/lidar-technologies-systems/data-processing-dissemination

7. Ready, J. (1997) Laser Applications in Spectroscopy (Ch. 20). Industrial Applications of Lasers (2ed), Academic Press, 1997. https://doi.org/10.1016/B978-012583961-7/50022-3.
8. Saeedi, S., Trentini, M., Seto, M., and Li, H. (2015), A hybrid approach for multiple-robot SLAM with particle filtering, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3421-3426, doi: 10.1109/IROS.2015.7353854. https://ieeexplore-ieee-org.ezpv7-web-p-u01.wpi.edu/document/7353854
9. Song, W., Yang, Y., Fu, M., Kornhauser, A., & Wang, M. (2018). Critical rays self-adaptive particle filtering SLAM. *Journal of Intelligent & Robotic Systems, 92*(1), 107-124. https://doi-org.ezpv7-web-p-u01.wpi.edu/10.1007/s10846-017-0742-z
10. Thrub, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*, *Intelligent Robotics and Autonomous Agents series*
11. Wang, P., Xu, P., Bonnifait, P., Jiang, J. (2018). Box Particle Filtering for SLAM with Bounded Errors. 15th International Conference on Control, Automation, Robotics and Vision (ICARCV 2018), Nov 2018, Singapore, Singapore. pp.1032-1038, ff10.1109/ICARCV.2018.8581234ff. ffhal-02060133f https://hal.archives-ouvertes.fr/hal-02060133/document